目　录

- psdk-mavlink-M4系列
  - 免责申明
  - 产品简介
    - 产品图片
    - 产品参数
  - 物品清单
  - 部件名称
  - 系统配置
  - 开发准备
  - QGC功能
  - Mavlink功能
    - PSDK-Payload（基础功能）
      - 机型信息
      - 相机视频流传输
      - 基础相机功能
        - 基础功能
        - 对焦功能
        - 测光功能
        - 变焦功能
        - 数据传输
      - 云台功能
      - GPS定位
      - 回放下载
      - 电源管理
      - 自定义HMS功能
      - 精准定位
    - E-port（高级功能）
      - 相机管理
      - 飞行控制
      - 码流 AI 识别
      - 运动规划

网络RTK
Mavros功能
参考链接

psdk-mavlink-M4系列

免责申明

本文所提及的内容关系到您的安全以及合法权益与责任。使用本产品之前，请仔细阅读本文以确保已对产品进行正确的设置。不遵循和不按照本文的说明与警告来操作可能会给您和周围的人带来伤害，损坏本产品或其它周围的物品。本文档及本产品所有相关的文档最终解释权归爱华芯（大连）科技有限公司所有。如有更新，恕不另行通知。
一旦使用本产品，即视为您已经仔细阅读本免责声明与警告、理解、认可和接受本声明全部条款和内容。您承诺对使用本产品以及可能带来的后果负全部责任。您承诺仅出于正当目的使用本产品，并且同意本条款以及爱华芯科技制定的任何相关条例、政策和指引。爱华芯科技对于直接或间接使用本产品而造成的损坏、伤害以及任何法律责任不予负责。用户应遵循包括但不限于本文提及的所有安全指引。
即使存在上述规定，消费者权益依然受当地法律法规所保障，并不受本免责声明影响。

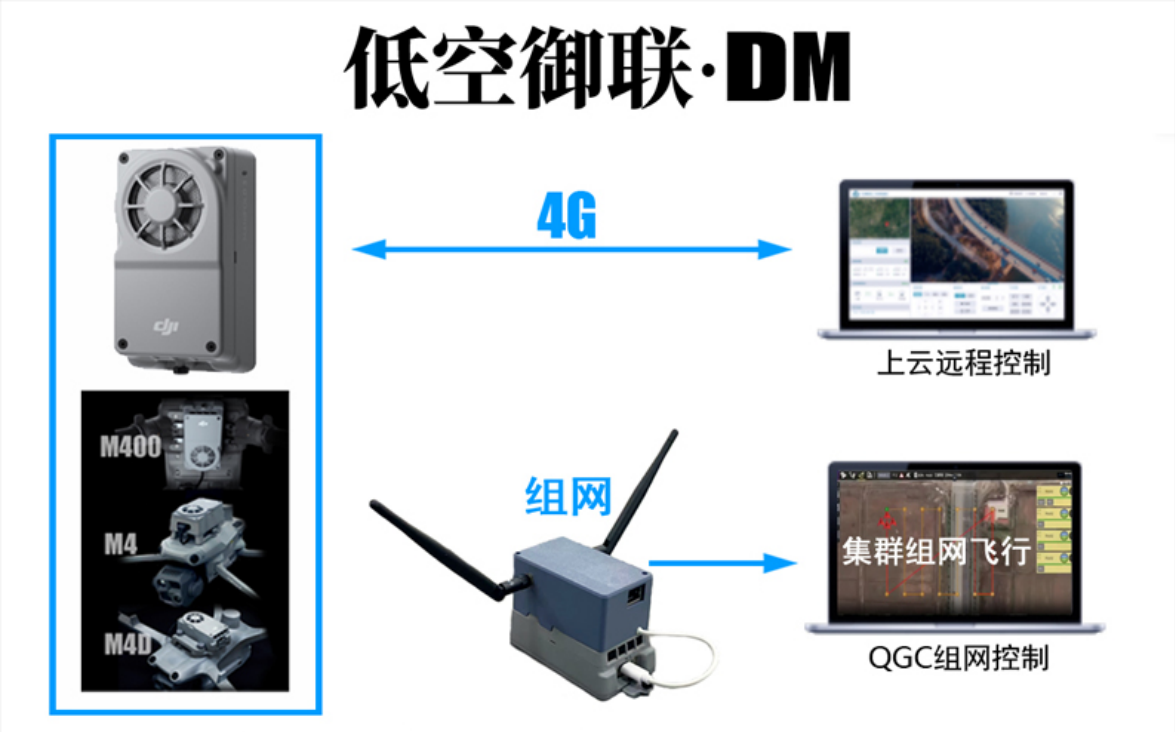
警告！

1. 务必使用爱华芯指定连接线，并严格按照各接口定义连接外部设备。
2. 严禁擅自拆解妙算3系列产品及其配件。
3. 防止水、油、沙等进入机身内部。
4. 选择合适的位置进行安装，确保散热良好。
5. 部件工作时会发热，请勿用手直接接触，否则可能造成烫伤。
6. 使用、储存及运输时，避免震动和撞击。

产品简介

产品图片


具身灵通·DM
M400
M4
M4D
MANIFOLD 3
dji
Not Ready
Position
69%
Yaw follow
P: 0.0 Az: 36.0
Gimbal Controls
Yaw Lock
Center
Tilt 90
Point Home
Retract
Settings
Fly
Plan
Takeoff
Return
Noble Merci
Dachonger Rd
Kefa Rd
Lixiang Ln
Kerun Rd
X256
Shahe Rd W
Shahe East Rd
Shahe Bridge
0.0 m
0.0 m/s
00:00:00
1.0 m
0.0 m/s
0.0 m


低空御联·DM
4G
上云远程控制
M400
M4
M4D
组网
集群组网飞行
QGC组网控制

产品参数

NO	类别	参数
1	CPU	Orin NX
2	AI性能	100 TOPS (INT 8)
3	显存	16GB 128 位 LPDDR5102.4GB/S
4	存储	256G SSD
5	防护等级	IP55
6	接口	EPORT V2/USB-C
7		可使用USB-C→HUB-LAN→5G网卡、组网模块等通信链路
8	工作温度	-20°C至50°C
9	尺寸	98mmx57mmx36 mm
10	重量	120±5 g

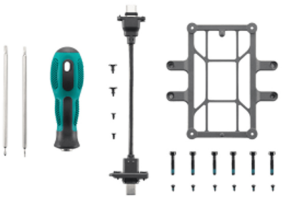
物品清单

妙算3-M400物品清单

妙算3 x 1



妙算3-M400工具包 x 1



TYPE-C分线器 x 1

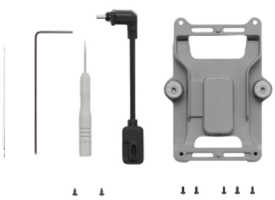


妙算3-M4系列物品清单

妙算3 x 1



妙算3-M4系列工具包 x 1



TYPE-C分线器 x 1

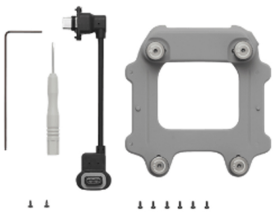


妙算3-M4D系列物品清单

妙算3 x 1



妙算3-M4D系列工具包 x 1

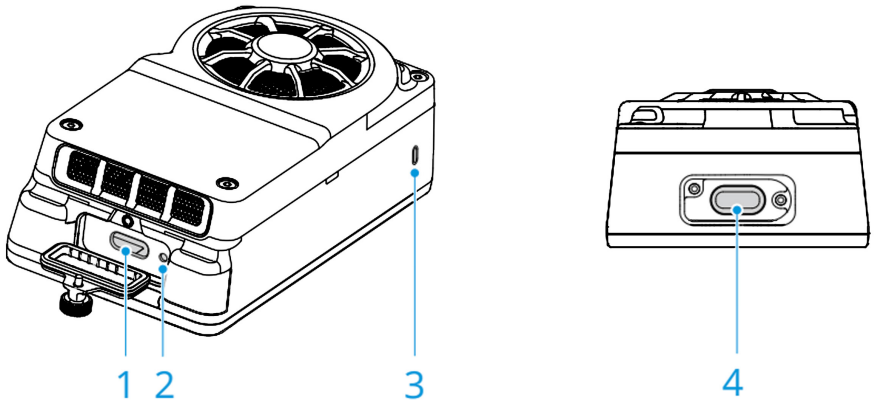


TYPE-C分线器 x 1



部件名称

妙算3部件名称



1. USB-C 接口

接入支持USB3.0的外部设备(如拓展坞、U盘等)，不支持直接连接电脑。

- 使用支持USB PD协议的充电器连接该接口为设备供电，如DJI100W桌面充电器或DI65W便携充电器。
- 接口未使用时，需盖好防水胶塞并拧紧手拧螺丝。

2. Reset 按键

长按10s后，恢复出厂设置，清除已安装应用及所有数据。

3. LED 指示灯

—	绿灯常亮	启动中
·····	绿灯慢闪	正常运行中/升级完成
·····	黄灯慢闪	系统过热，降低频率运行中
·····	黄灯快闪	升级中
·····	红灯慢闪	升级失败
·····	蓝灯快闪	恢复出厂设置

4. E-Port 接口

仅支持连接至适配的飞行器

系统配置

妙算 3 系统配置说明

本文档用于说明妙算 3 当前开发环境、系统版本、常用查询命令、应用安装目录和开发注意事项，便于项目部署、问题定位和后续维护。

1. 基础系统信息

项目	当前配置	说明
设备平台	DJI 妙算 3	面向 PSDK 应用开发与部署。
JetPack 版本	JetPack 5.1.3	当前暂不支持升级。
CUDA 版本	V11.4.315	可通过 `nvcc -V` 查看。
CMake 版本	3.16.3	系统默认 CMake 版本。
应用开发和安装目录	`/open_app`	用户开发应用和安装应用建议放在该目录。
系统可用空间	约 99 GB	实际空间以设备当前查询结果为准。

2. CUDA 版本查看

```
nvcc -V
```

当前环境 CUDA 版本为：

```
V11.4.315
```

3. 应用开发和安装目录

妙算 3 用户开发应用和安装应用的推荐目录为：

```
/open_app
```

建议项目部署时将可执行程序、配置文件、运行脚本和必要资源统一放在 `/open_app` 或其子目录下，便于后续维护和升级。

示例目录结构：

```
/open_app/
  mavlink_app/
    bin/
    config/
    logs/
    scripts/
```

4. 获取设备唯一编码

4.1 产品 Serial Number

```
cat /factory/device_id.txt
```

4.2 SOC Serial Number

```
cat /sys/firmware/devicetree/base/serial-number
```

说明：

- 产品 Serial Number 可用于设备识别、部署记录和售后追踪。
- SOC Serial Number 可用于底层硬件识别。
- 注意命令路径中 `devicetree/base` 之间只需要一个 `/`。

5. 系统时间与同步

妙算 3 已通过 PPS 硬件引脚与飞机系统完成毫秒级时间同步，应用层无需额外开发时间同步功能。

5.1 Bash 查看 UTC 时间

```
date -u '+%Y-%m-%d %H:%M:%S.%3N'
```

输出示例：

```
2026-07-08 03:20:15.123
```

5.2 C/C++ 程序中获取时间

C/C++ 程序中建议使用系统实时时钟：

```
CLOCK_REALTIME
```

或使用 C++ 标准库：

```
std::chrono::system_clock
```

说明：

- `CLOCK_REALTIME` 和 `std::chrono::system_clock` 获取的是系统实时时间。
- 在本平台上，系统时间已由飞机系统同步，适合用于日志时间戳、MAVLink 时间戳和业务事件记录。

6. 码流硬件编解码

妙算 3 已封装基于硬件编解码的接口，应用开发中无需手动实现底层硬件编解码流程。

6.1 获取硬件解码后的码流

```
DjiLiveview_StartImagestream
```

6.2 将码流硬件编码为 H264

```
DjiLiveview_EncodeAFrameToH264
```

说明：

- 推荐优先使用 PSDK 已封装接口获取、编码和处理视频流。
- 不建议在应用层重复实现底层硬件编解码逻辑。
- 若需要转发到 QGC、播放器或网络端，应重点确认编码格式、分辨率、帧率、RTP/UDP 地址和端口配置。

7. USB 转串口通信

妙算 3 支持接入常见 USB 转串口设备，例如：

- FT232
- CP2102

接入设备后，可通过以下命令查看串口设备节点：

```
ll /dev/ttyUSB*
```

如果系统未识别到设备，可检查：

- USB 转串口模块是否连接正常。
- 设备是否被系统识别。
- 用户权限是否允许访问 `/dev/ttyUSB*`。
- 串口波特率、数据位、停止位、校验位配置是否与外设一致。

8. 系统 OpenCV 版本查看

查看系统 OpenCV 相关安装包：

```
dpkg -l | grep -E 'libopencv|opencv' | head
```

说明：

- 系统 OpenCV 版本以设备实际查询结果为准。
- 如果项目需要指定 OpenCV 版本，应在 CMake 中明确查找和链接路径，避免误链接到不符合预期的系统库。

9. CMake 版本

当前系统 CMake 版本：

```
cmake version 3.16.3
```

查看命令：

```
cmake --version
```

开发注意事项：

- CMakeLists.txt 中不要使用高于 3.16.3 才支持的语法。
- 如果需要跨平台构建，应在 `cmake_minimum_required()` 中选择与妙算 3 环境兼容的版本。

示例：

```
cmake_minimum_required(VERSION 3.16)
```

10. 头文件依赖索引路径

在 C/C++ CMake 工程中，应让头文件依赖索引和搜索路径优先指向项目自己的目录，而不是系统目录，例如 `/usr/include` 。

推荐使用：

```
target_include_directories()
```

示例：

```
target_include_directories(my_target
    PRIVATE
        ${CMAKE_CURRENT_SOURCE_DIR}/include
        ${CMAKE_CURRENT_SOURCE_DIR}/src
)
```

不推荐在项目中全局滥用：

```
include_directories(...)
```

原因：

- 全局 include 路径容易影响其它 target。
- 容易误引用系统头文件或旧版本头文件。
- 不利于定位依赖关系。

11. 开发环境检查命令汇总

功能	命令
查看 CUDA 版本	`nvcc -V`
查看产品 Serial Number	`cat /factory/device_id.txt`
查看 SOC Serial Number	`cat /sys/firmware/devicetree/base/serial-number`
查看 UTC 时间	`date -u '+%Y-%m-%d %H:%M:%S.%3N'`
查看 USB 串口设备	`ll /dev/ttyUSB*`
查看 OpenCV 安装信息	`dpkg -l | grep -E 'libopencv|opencv' | head`
查看 CMake 版本	`cmake --version`

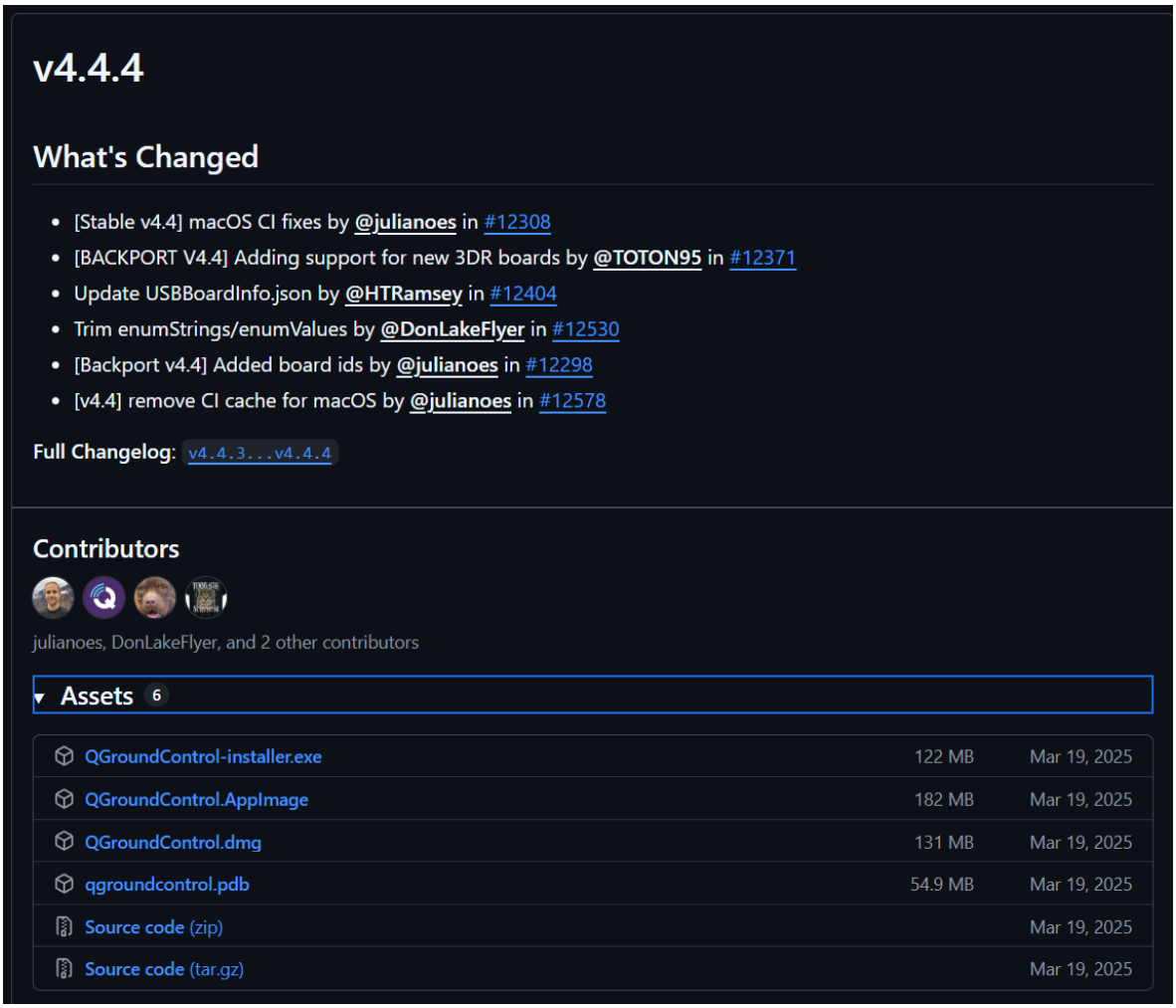
12. 开发建议

1. 应用程序、脚本、配置文件和日志建议统一放在 `/open_app` 或其子目录下。
2. 不要自行升级 JetPack，当前环境固定为 JetPack 5.1.3。
3. 涉及 CUDA、OpenCV、GStreamer、PSDK 的工程，应明确 CMake include 和 link 路径。
4. 视频流处理优先使用 PSDK 已封装的硬件编解码接口。
5. 与飞机系统时间相关的功能优先使用系统时间，不需要额外实现时间同步。
6. 串口外设接入后，先确认 `/dev/ttyUSB*` 是否出现，再调试协议层。
7. 交付部署时建议记录设备 Serial Number、应用版本、配置文件版本和启动命令。

开发准备

QGC4.4.4下载地址

https://github.com/mavlink/QGroundControl/releases



v4.4.4

What's Changed

- [Stable v4.4] macOS CI fixes by @julianoes in #12308
- [BACKPORT V4.4] Adding support for new 3DR boards by @TOTON95 in #12371
- Update USBBoardInfo.json by @HTRamsey in #12404
- Trim enumStrings/enumValues by @DonLakeFlyer in #12530
- [Backport v4.4] Added board ids by @julianoes in #12298
- [v4.4] remove CI cache for macOS by @julianoes in #12578

Full Changelog: v4.4.3...v4.4.4

Contributors

julianoes, DonLakeFlyer, and 2 other contributors

Assets 6

QGroundControl-installer.exe	122 MB	Mar 19, 2025
QGroundControl.AppImage	182 MB	Mar 19, 2025
QGroundControl.dmg	131 MB	Mar 19, 2025
qgroundcontrol.pdb	54.9 MB	Mar 19, 2025
Source code (zip)		Mar 19, 2025
Source code (tar.gz)		Mar 19, 2025

QGC功能

QGroundControl 用户手册

项目	说明
核心目的	将 DJI PSDK 订阅、相机、云台、任务和视频能力转换为 QGC 可识别的标准 MAVLink 数据与命令。
验证方式	通过 QGC 飞行界面、参数界面、Analyze Tools 和视频界面确认遥测、任务、相机与视频链路。

1. 系统概述

本项目的核心目标是实现DJI PSDK到标准MAVLink协议的转换，打通开源飞控生态壁垒。它不仅支持将Ardupilot、PX4和iNAV的现有成果无缝移植至DJI无人机，并重点解决DJI无人机基于ROS/ROS2及MAVROS中间件的高层控制与通信需求。
本项目在妙算 3 / DJI PSDK 环境中运行，将飞控订阅数据、相机状态、云台控制、任务控制、视频流和扩展状态转换为标准MAVLink消息，发送给 QGroundControl（以下简称GGC，其为开源的地面站软件，开发者可自行设计修改，但设计版权商业问题由开发者自行负责；本次使用QGC为V4.4.4，下载地址：https://github.com/mavlink/QGroundControl/releases ）。QGC 作为地面站，负责显示飞行器状态、地图航迹、航点任务、参数、相机控制和实时视频。

- 飞行器遥测：姿态、位置、GPS、速度、高度、电池、飞行模式、任务状态等。
- 任务链路：QGC 航点规划、任务上传、任务开始、暂停、继续和返航状态显示。
- 相机链路：拍照、录像、相机模式、存储信息、拍照状态和相机信息。
- 视频链路：PSDK Liveview / H264 视频经 UDP/RTP 推送至 QGC。
- 参数链路：QGC 参数列表读取、单个参数读取和参数设置。

主要 MAVLink 映射

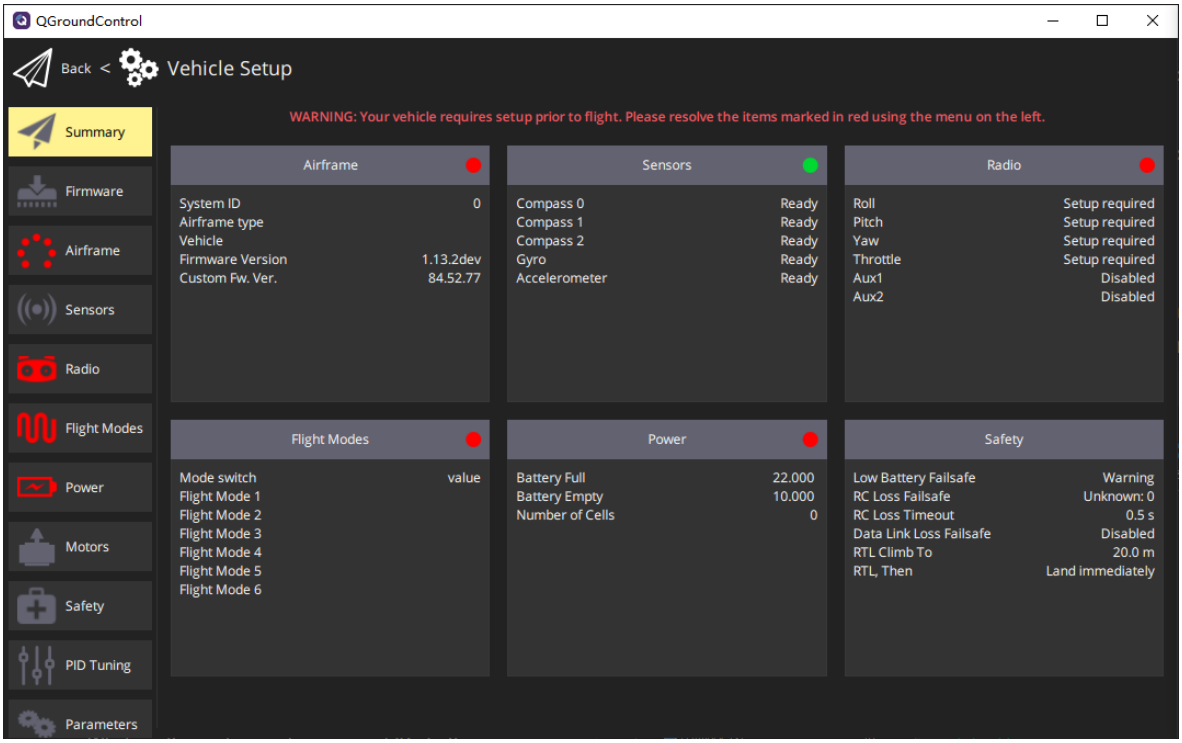
QGC 显示/功能	主要 MAVLink 消息或命令	PSDK 侧来源/动作
地图位置与航迹	GLOBAL_POSITION_INT, GPS_RAW_INT, LOCAL_POSITION_NED	飞控 GPS、融合位置、高度、速度订阅

QGC 显示/功能	主要 MAVLink 消息或命令	PSDK 侧来源/动作
姿态仪与航向	ATTITUDE, ATTITUDE_QUATERNION, VFR_HUD	飞控四元数、指南针、速度和高度数据
电池与系统状态	BATTERY_STATUS, SYS_STATUS	电池订阅与系统健康状态
任务规划与执行	MISSION_COUNT, MISSION_ITEM_INT, MISSION_ACK, MISSION_CURRENT	QGC 航点任务和 PSDK 任务控制逻辑
相机拍照/录像	MAV_CMD_IMAGE_START_CAPTURE, MAV_CMD_VIDEO_START_CAPTURE, CAMERA_CAPTURE_STATUS	DJI Camera Manager 拍照录像接口
视频流	VIDEO_STREAM_INFORMATION, RTP/H264 UDP stream	PSDK Liveview 转 H264/RTP/UDP
参数管理	PARAM_REQUEST_LIST, PARAM_VALUE, PARAM_SET	项目运行参数表

2. QGC 连接与基础状态

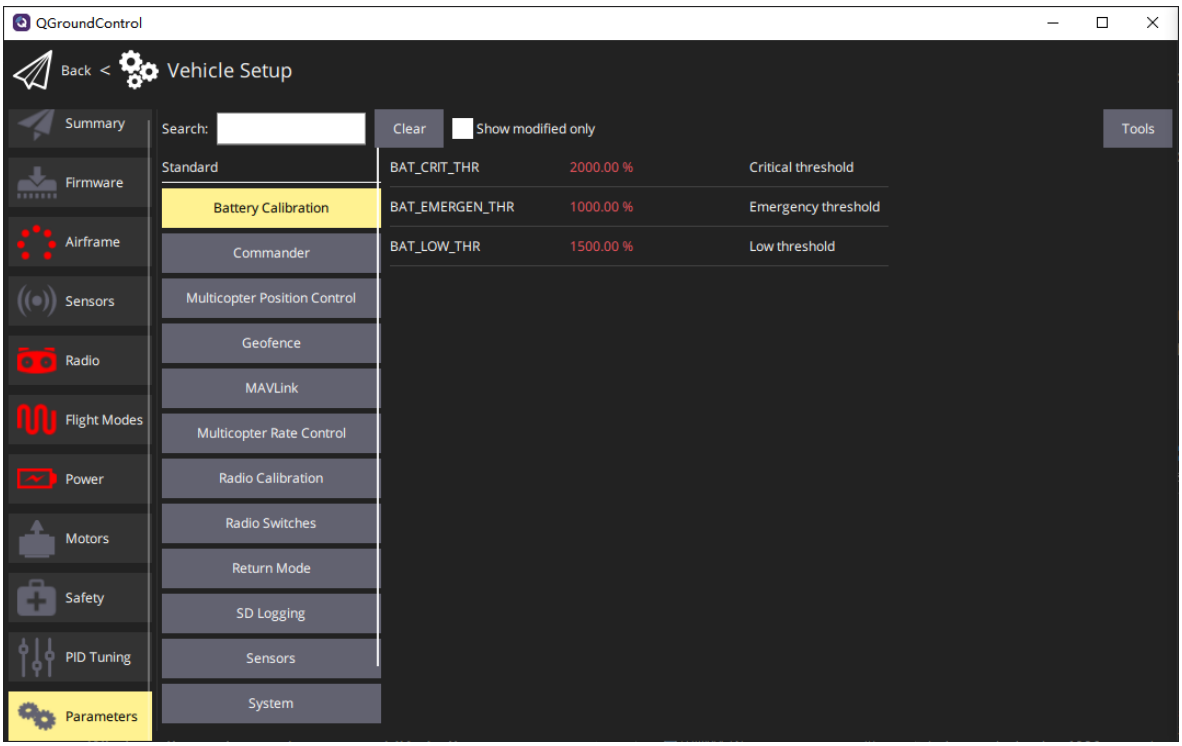
连接成功后，QGC 会显示飞行器系统状态、传感器状态、电源状态、飞行模式、GPS 状态和安全设置。

无人机概要页面



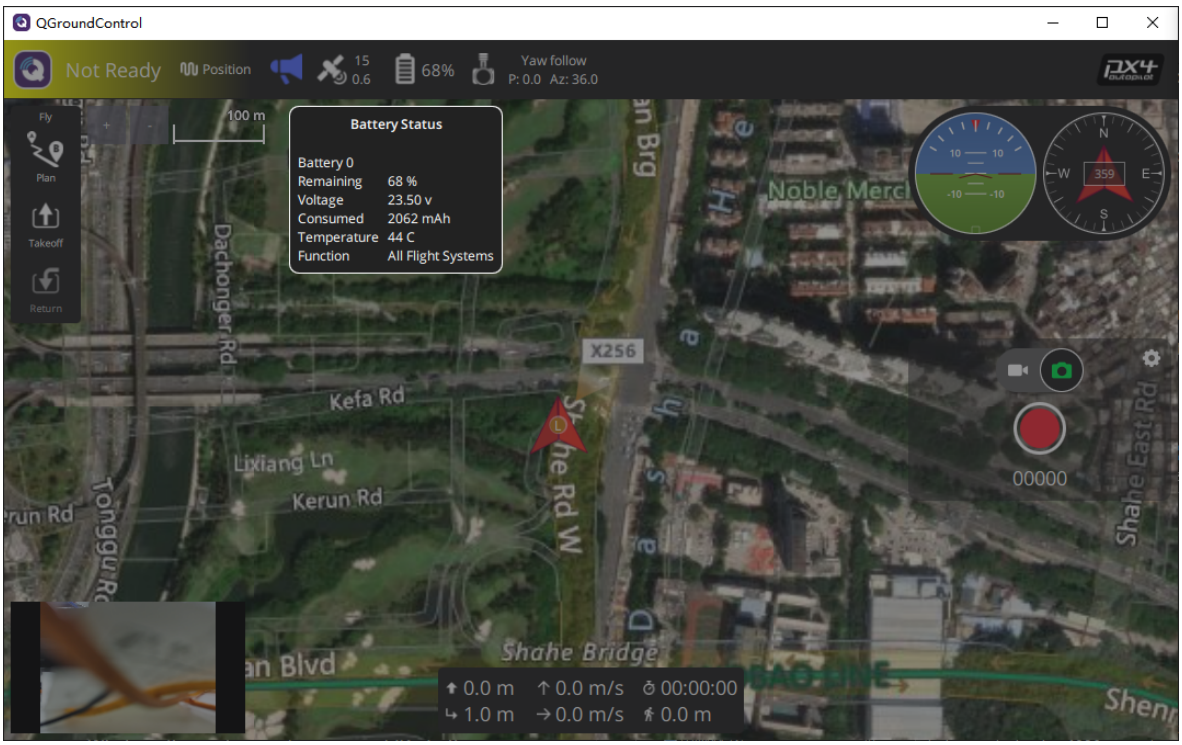
> 用于确认 QGC 已识别飞行器系统，重点查看系统 ID、传感器状态、电源状态、飞行模式和安全设置。

参数页面



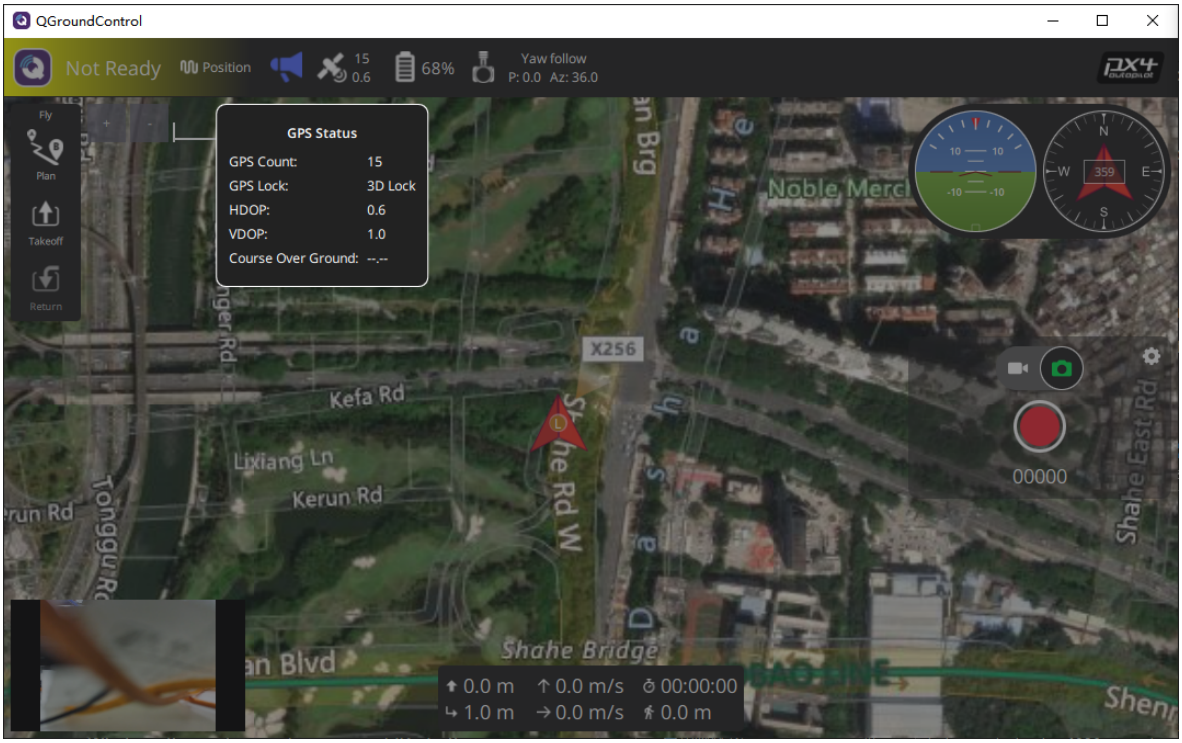
> 用于查看 QGC 参数树与参数值，验证 PARAM_REQUEST_LIST、PARAM_VALUE、PARAM_SET 等参数协议是否能正常响应。

电池状态弹窗



> 展示电池电压、电流、剩余电量和健康状态，对应 BATTERY_STATUS、SYS_STATUS 等状态消息。

GPS 状态弹窗

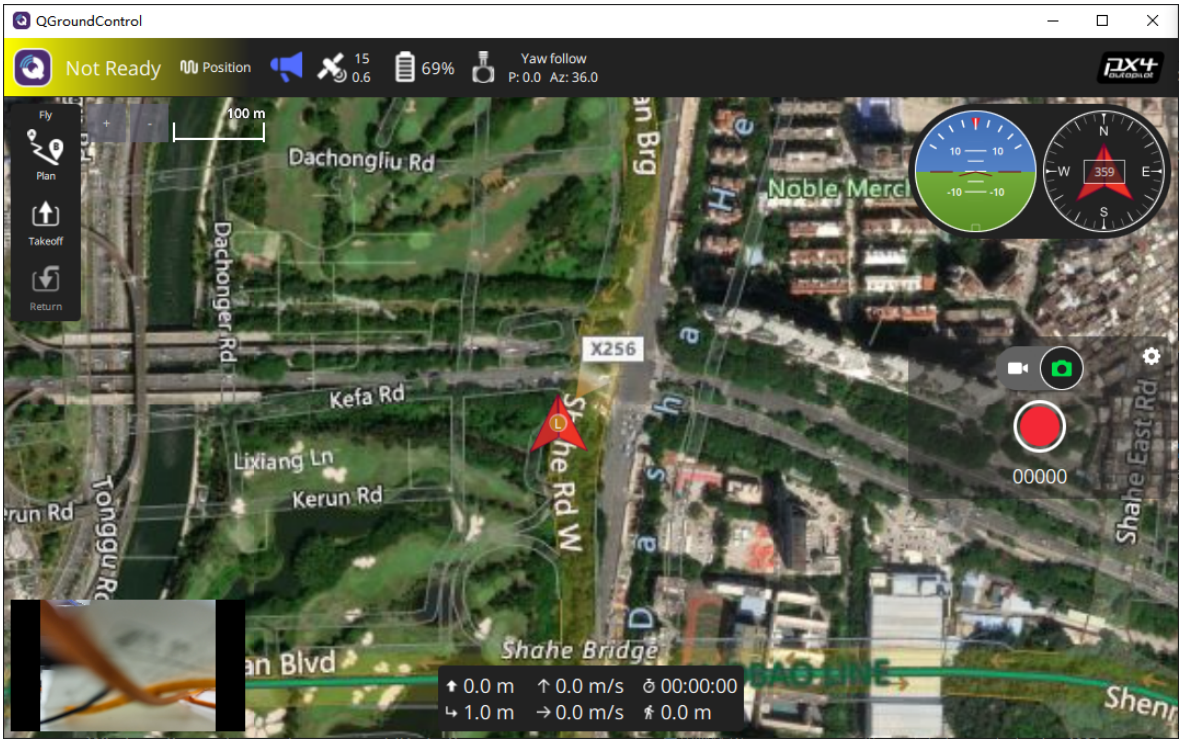


> 展示 GPS 锁定、卫星、HDOP/VDOP、RTK 或定位质量相关信息，对应 GPS_RAW_INT、GPS_STATUS 等消息。

3. 飞行主界面与遥测显示

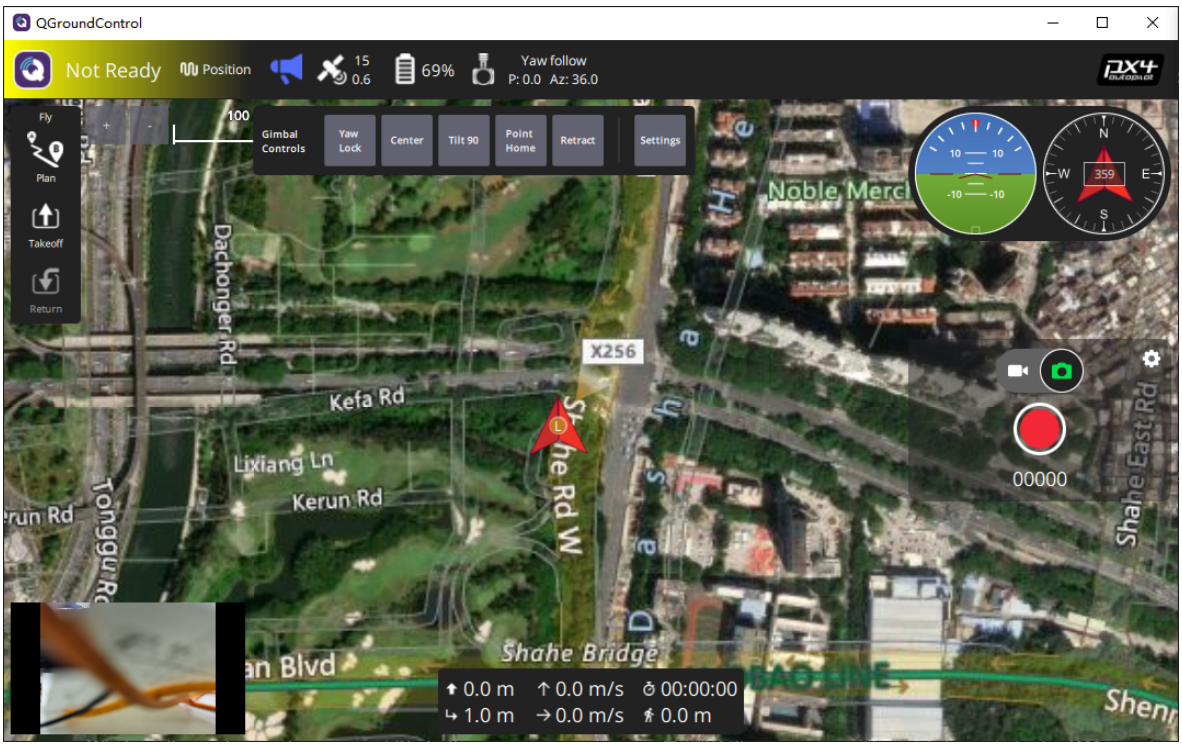
飞行主界面是用户最常用的操作页面。地图显示飞行器位置、航向、航迹和任务路线；右上角显示姿态仪与罗盘；底部显示高度、速度、飞行时间等状态；右侧提供相机拍照和录像控制。

飞行主界面



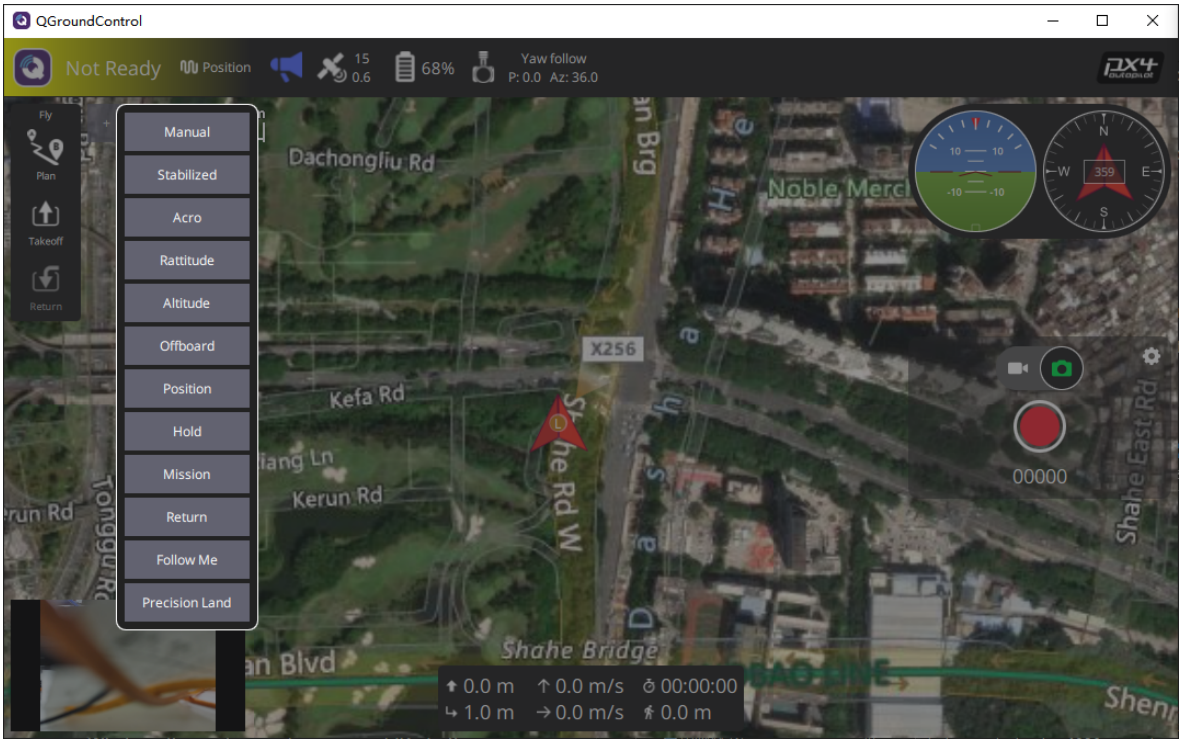
> 展示地图、姿态仪、航向、当前位置、电池、电台链路和相机控制入口，是用户日常查看飞行状态的主要页面。

飞行主界面 - 地图与状态联动



> 用于观察位置、姿态、电池、GPS、任务和相机状态在 QGC 中的综合呈现。

飞行操作菜单

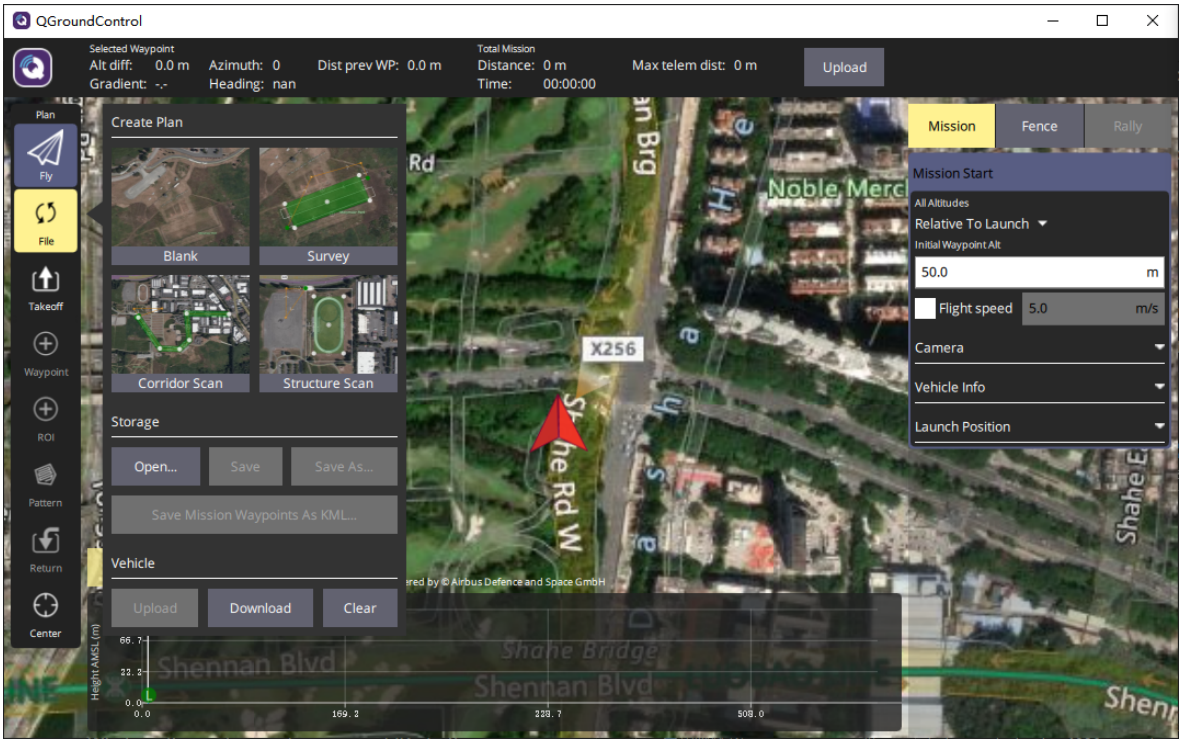


> 展示起飞、返航、暂停、继续、任务等操作入口，反映 QGC 对当前飞行模式和安全状态的判断。

4. 航点任务规划与执行

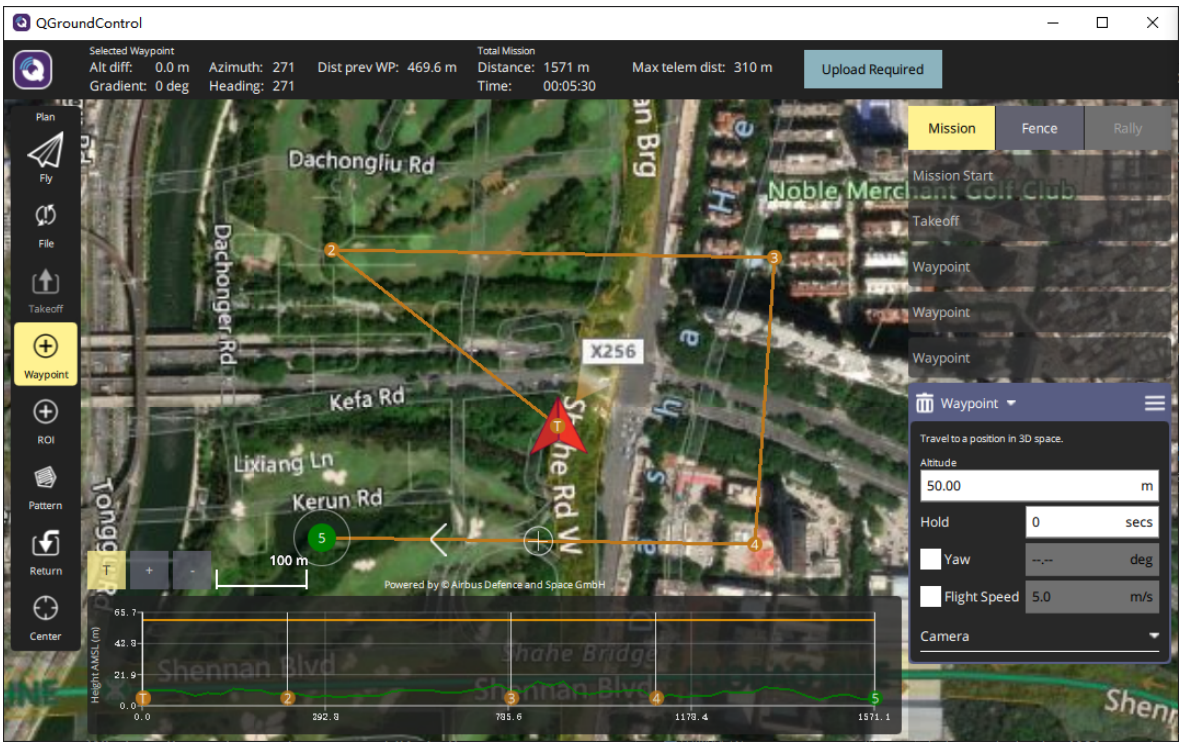
QGC 的 Plan 页面可用于创建航点任务。上传任务时，QGC 和 PSDK 转 MAVLink 模块通过标准 Mission 协议交互；执行任务时，地图会显示航线、目标航点、当前位置和任务状态。

航线规划 - 地图编辑



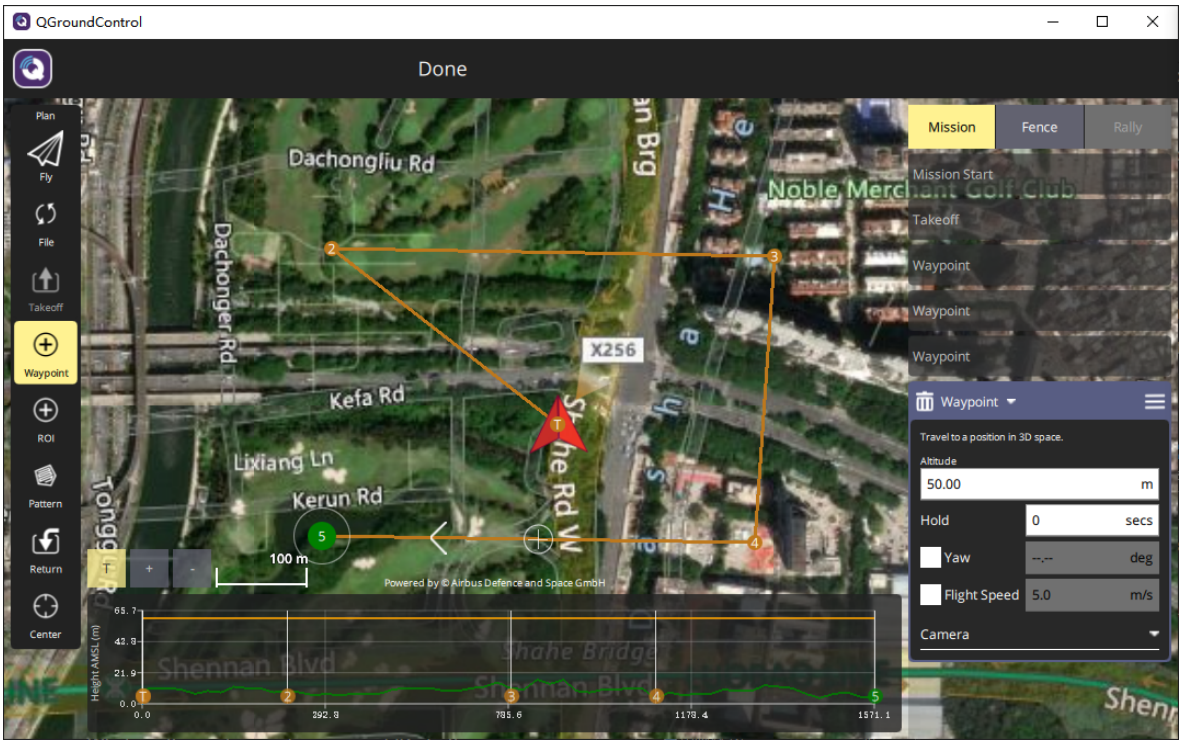
展示在地图上添加航点和规划航线的过程。

航线规划 - 任务编辑



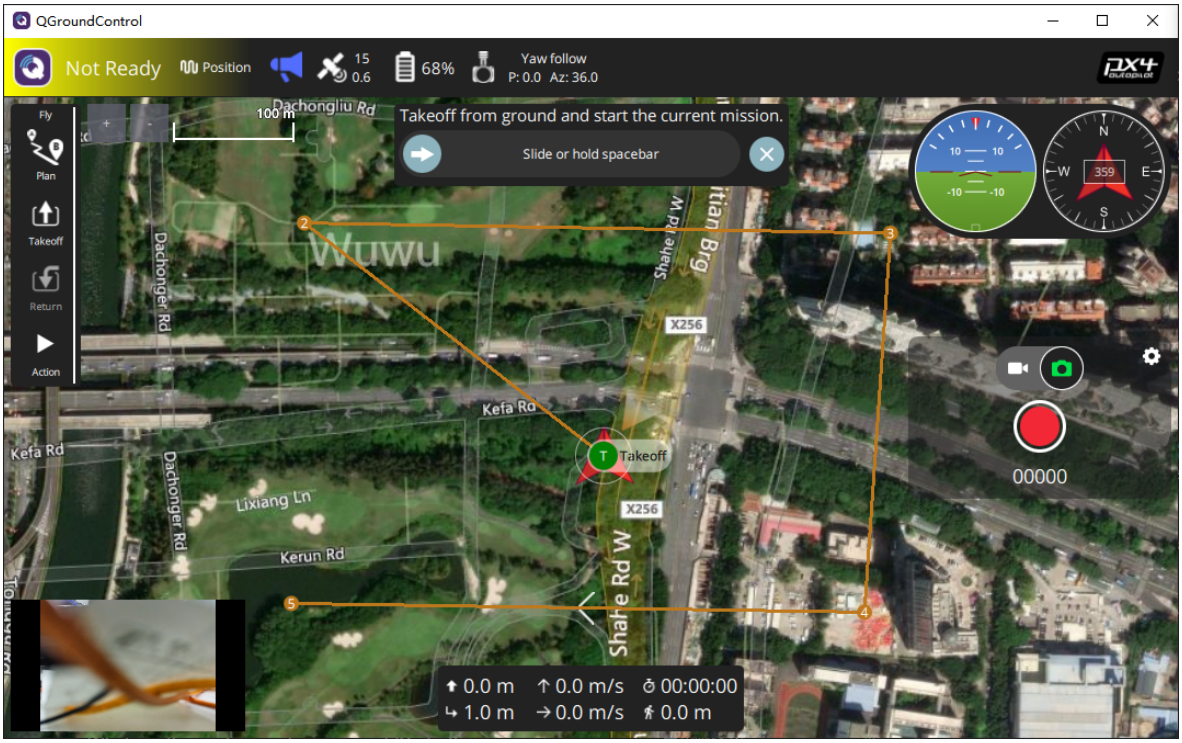
展示 Plan 页面中的航点编辑、任务高度、速度、航点参数和上传按钮。

航线规划 - 任务上传



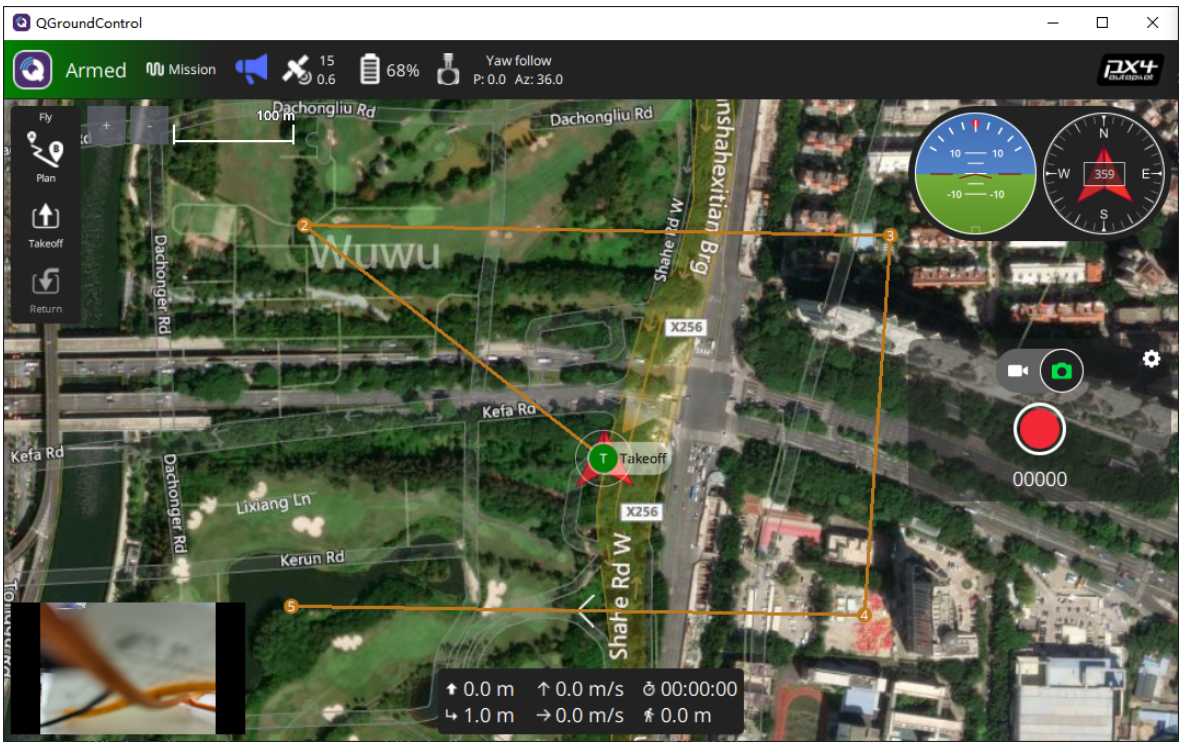
> 展示航点任务准备上传的状态，对应 MISSION_COUNT、MISSION_REQUEST_INT、MISSION_ITEM_INT、MISSION_ACK 流程。

任务上传提示



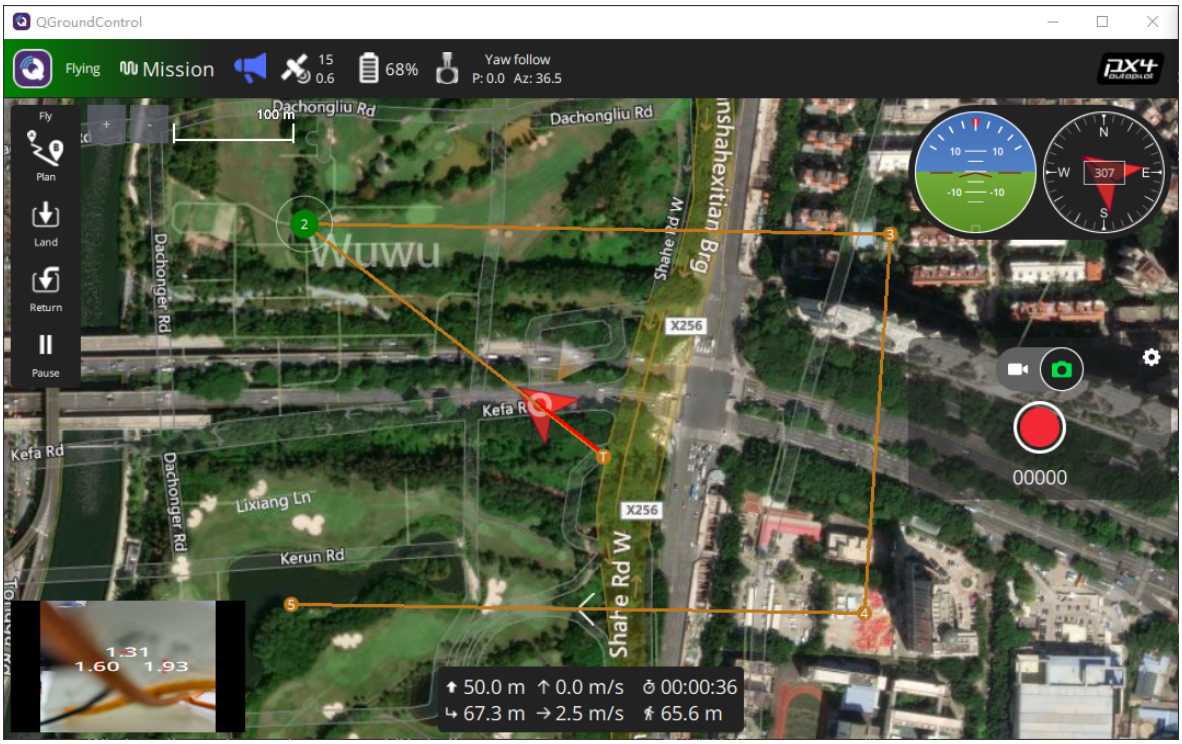
> 展示 QGC 任务上传完成或准备执行的提示信息。

任务待执行状态



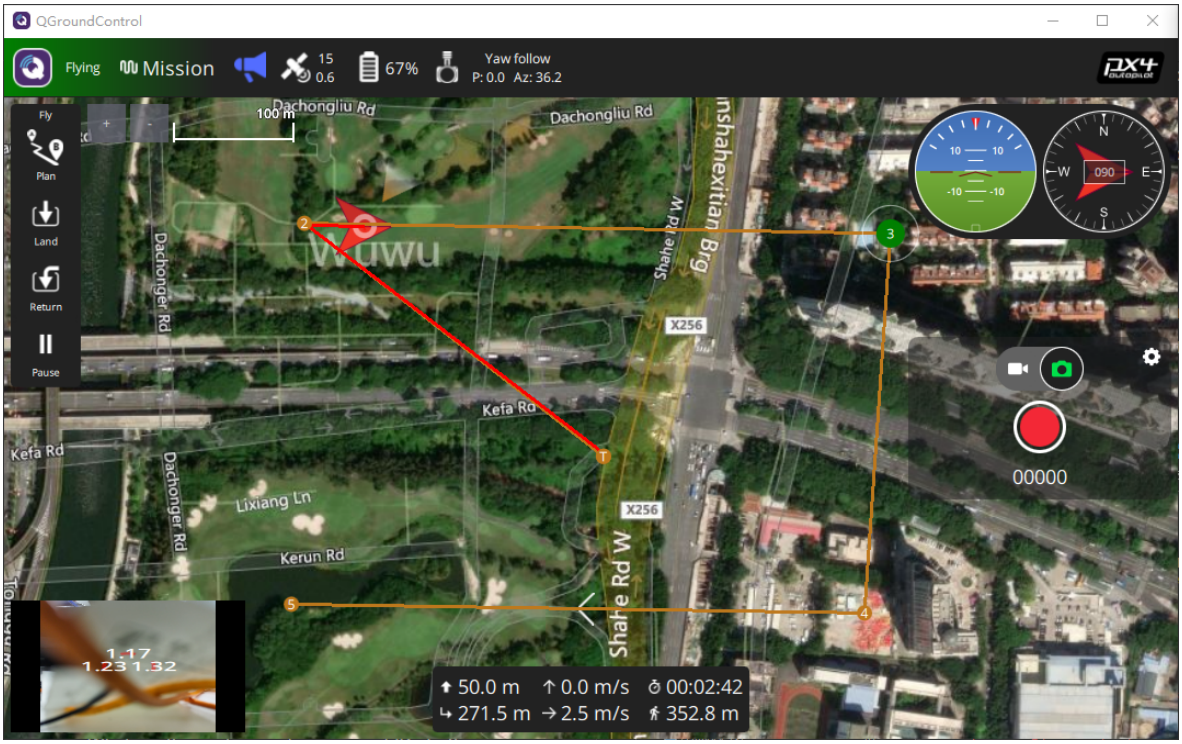
> 展示地图上的航线与飞行器当前位置，说明任务已在 QGC 中可视化。

任务执行过程 - 航向变化



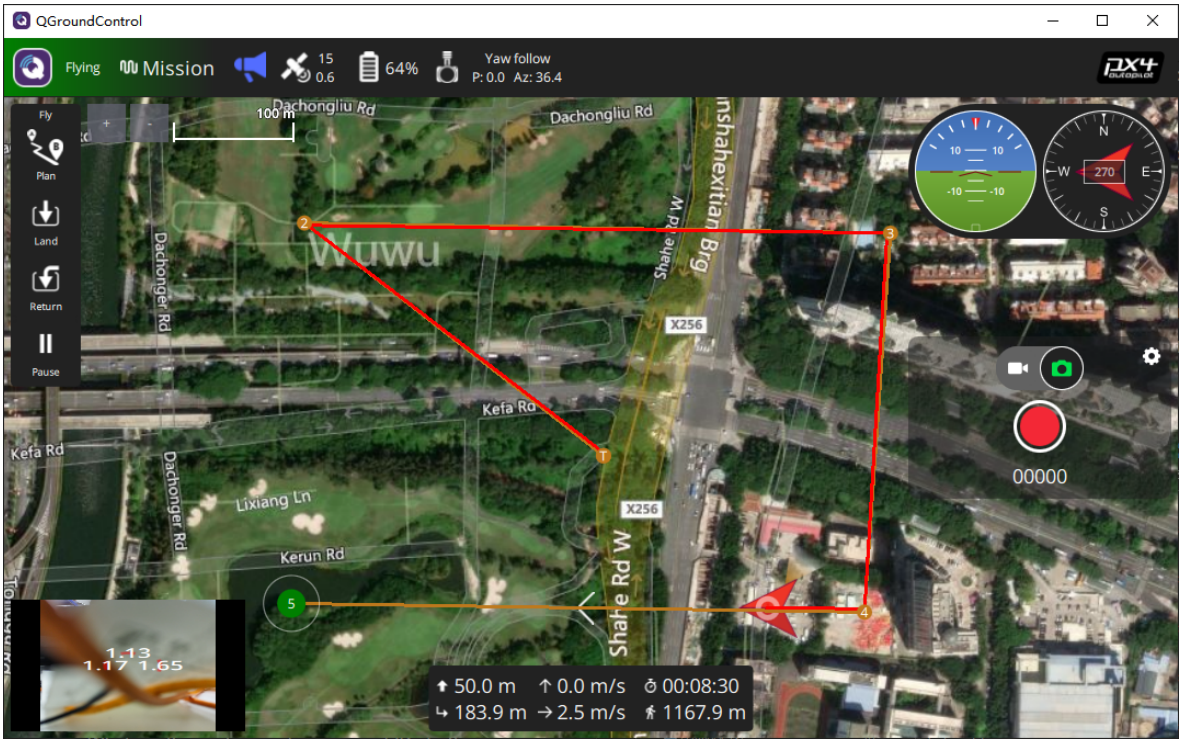
> 展示任务执行时飞行器沿航线移动，姿态、速度、高度和航线状态同步显示。

任务执行过程 - 航点跟踪



> 展示航点任务执行中的航迹线、目标航点和当前飞行器位置。

任务执行过程 - 航线段显示

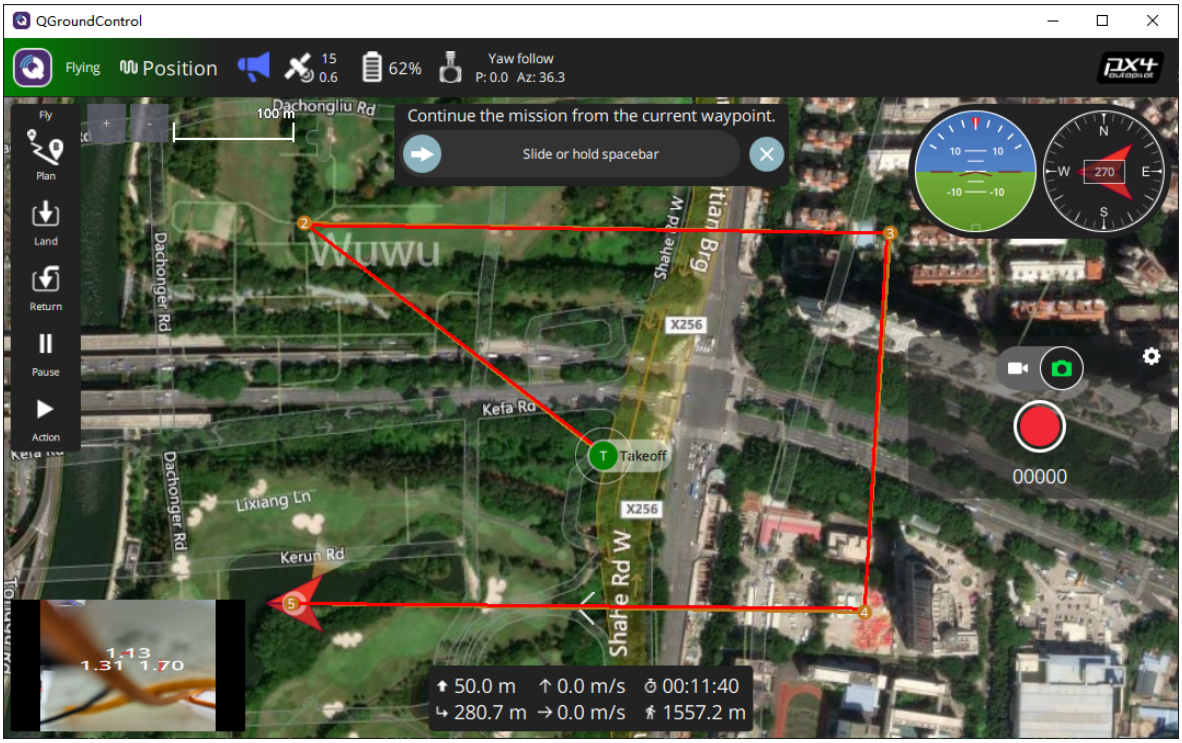


> 展示多段航线和任务边界，便于用户确认航线下发和任务状态。

5. 任务控制与安全状态

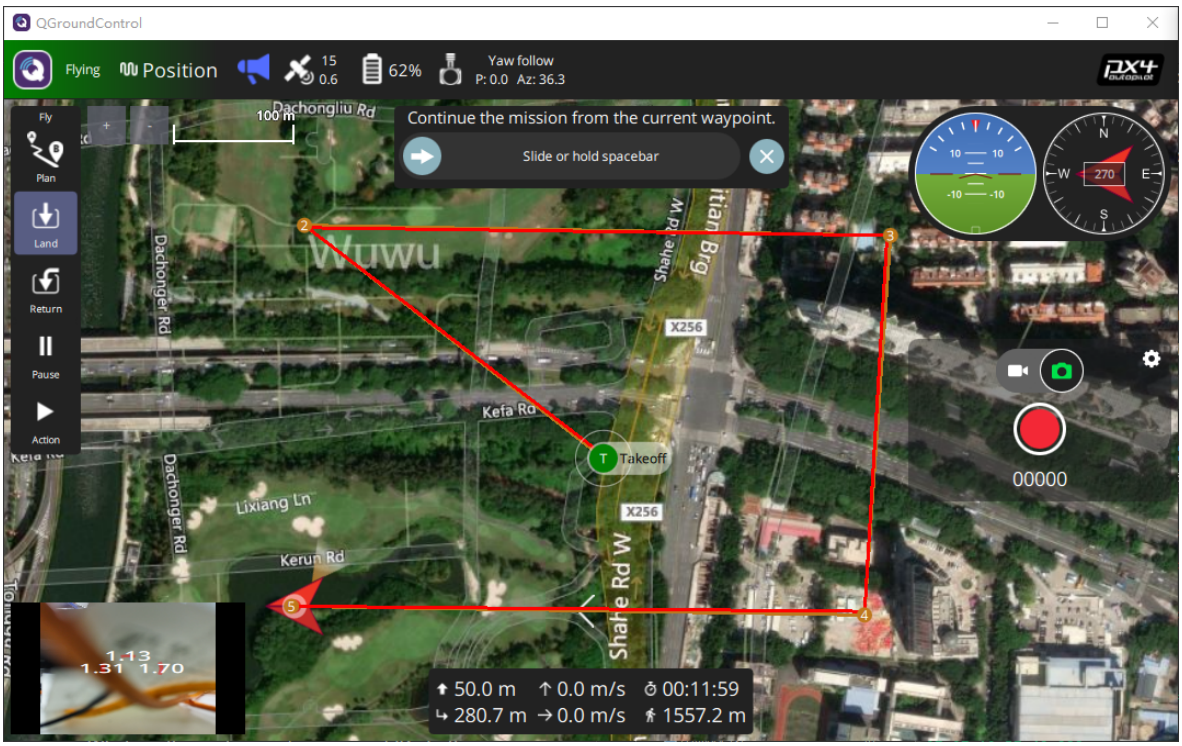
任务执行过程中，QGC 会显示任务状态、控制按钮和飞行器实时遥测。用户应关注任务控制命令是否有响应、航线是否保持显示、飞行器位置是否连续更新，以及返航/降落等安全状态是否能被 QGC 正确识别。

任务控制 - 开始/继续



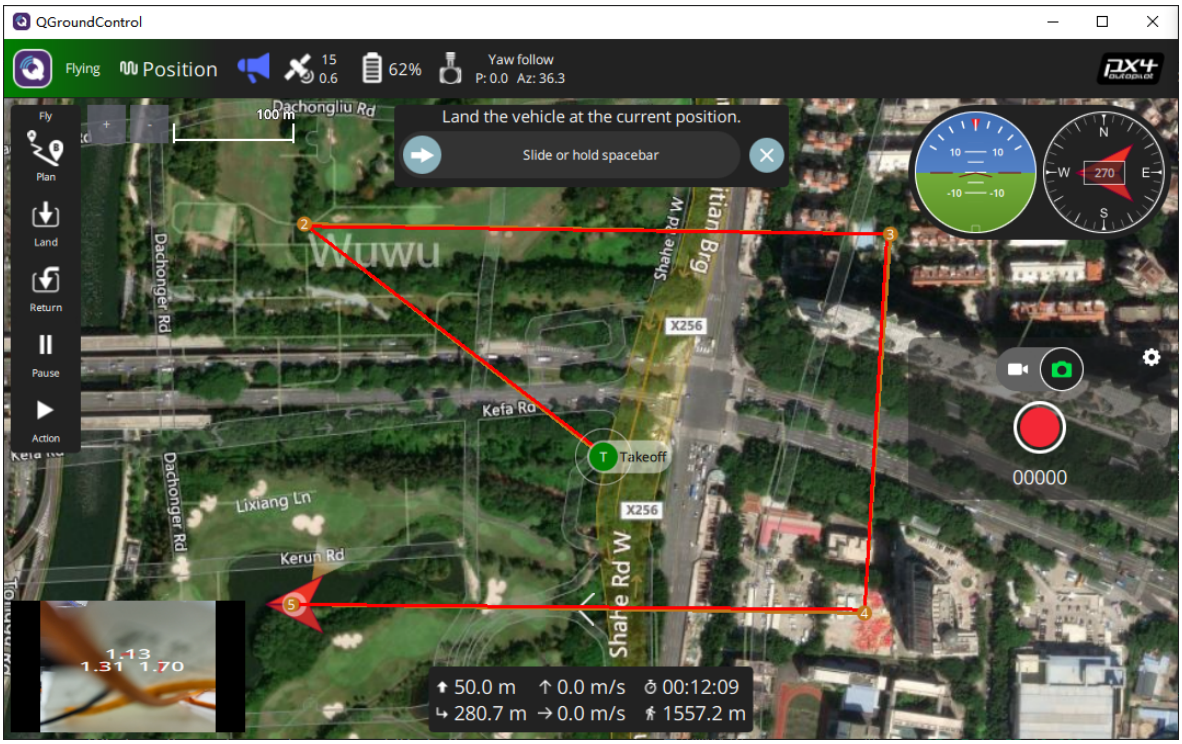
> 展示任务执行控制条，可用于开始、暂停、继续任务。

任务控制 - 暂停



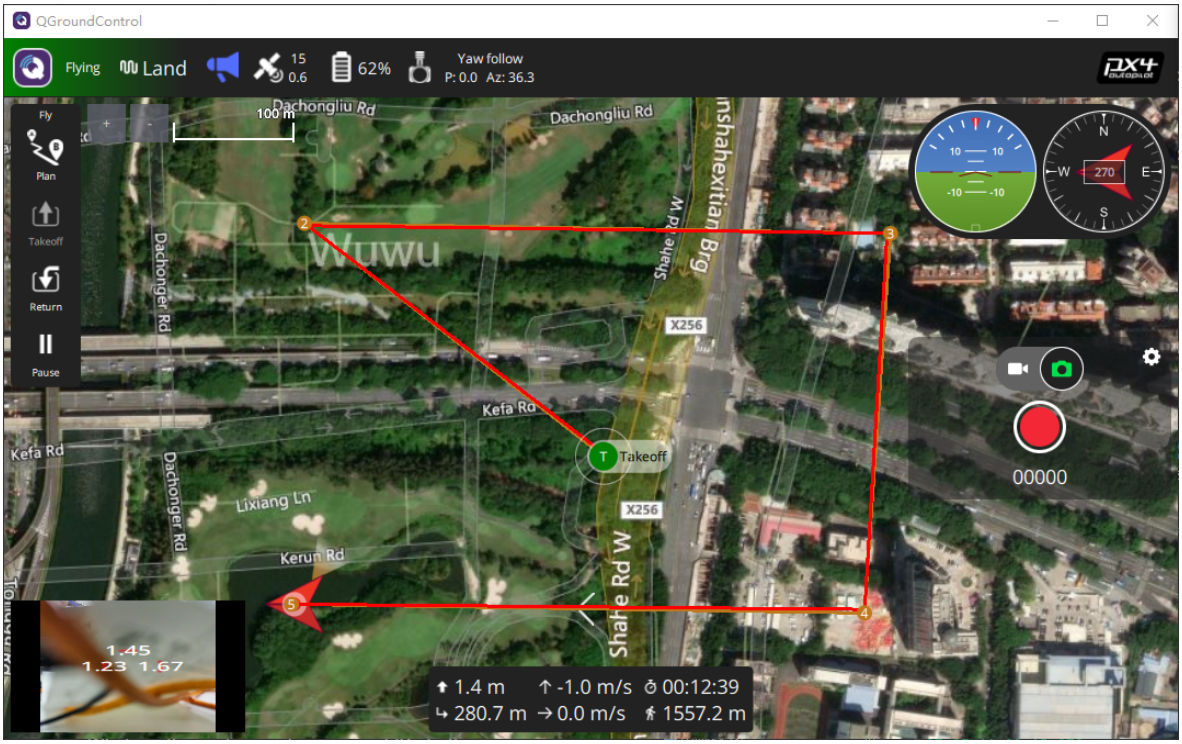
> 展示任务暂停或状态切换时的 QGC 提示。

任务控制 - 命令确认



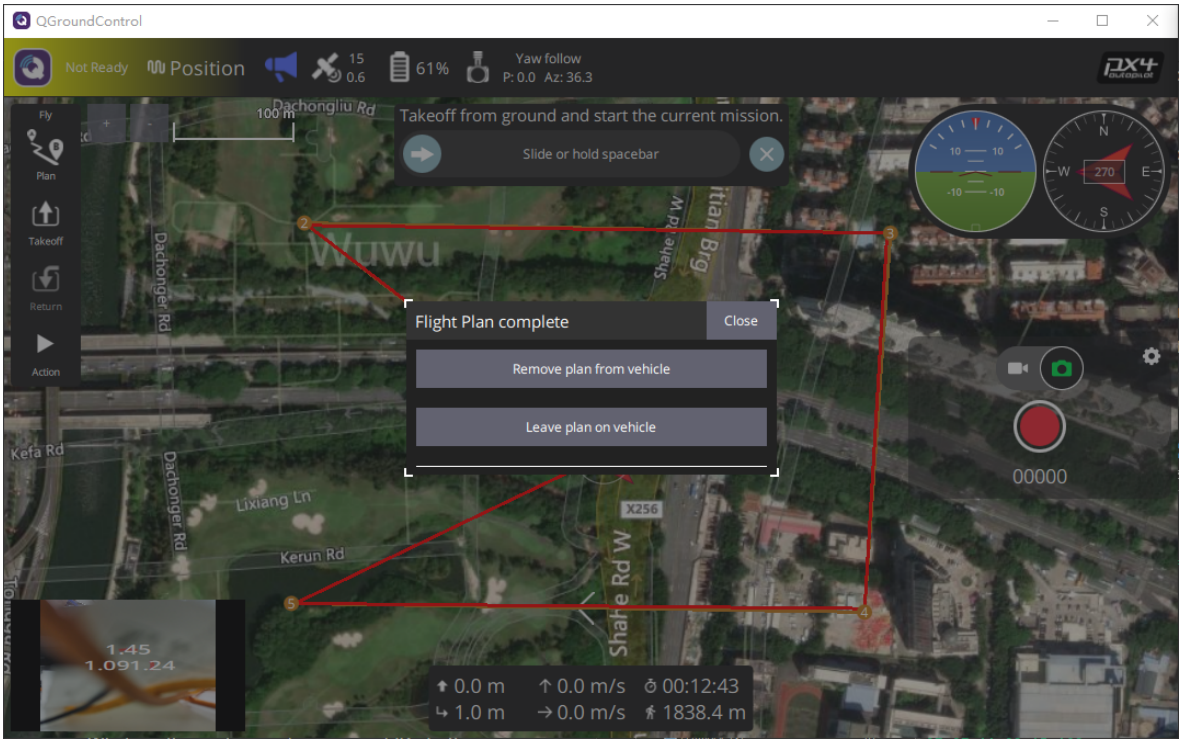
> 展示执行任务命令时 QGC 的命令提示和状态反馈。

任务执行与返航入口



> 展示任务飞行过程中的地图、航线和 Return 入口。

权限与状态提示

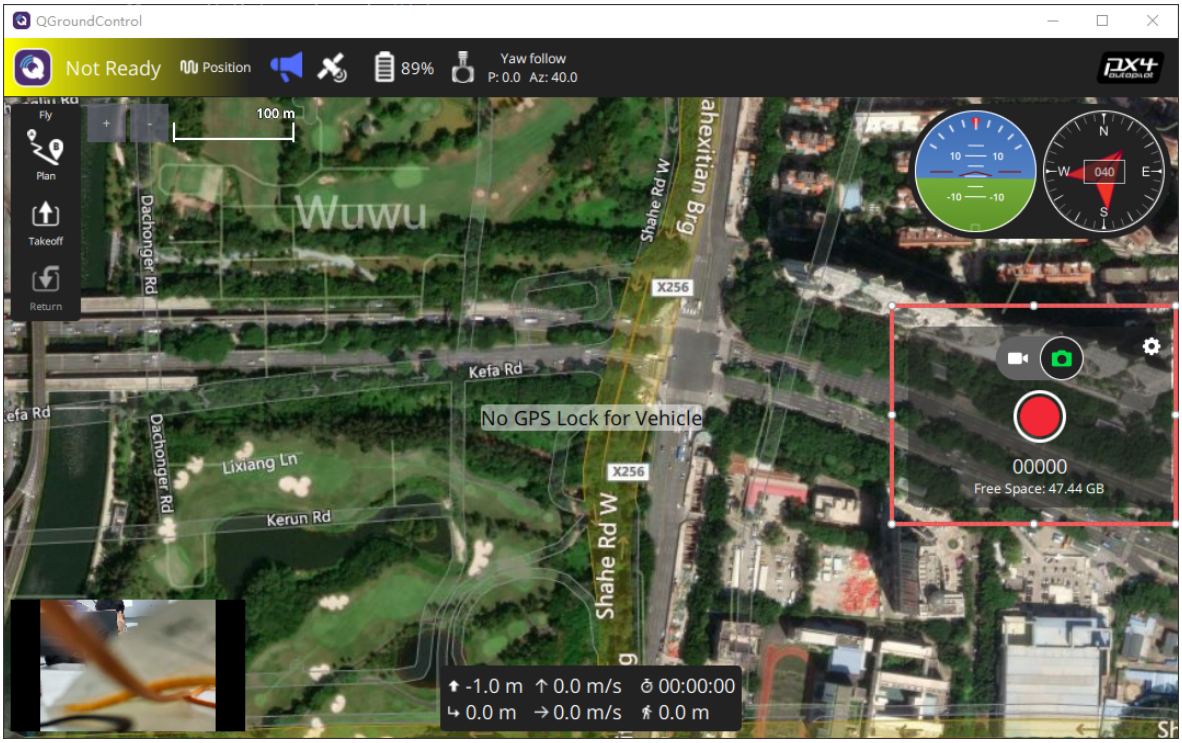


> 展示 QGC 或系统弹窗提示，需要结合 PSDK 控制权和设备状态确认。

6. 相机、录像与实时视频

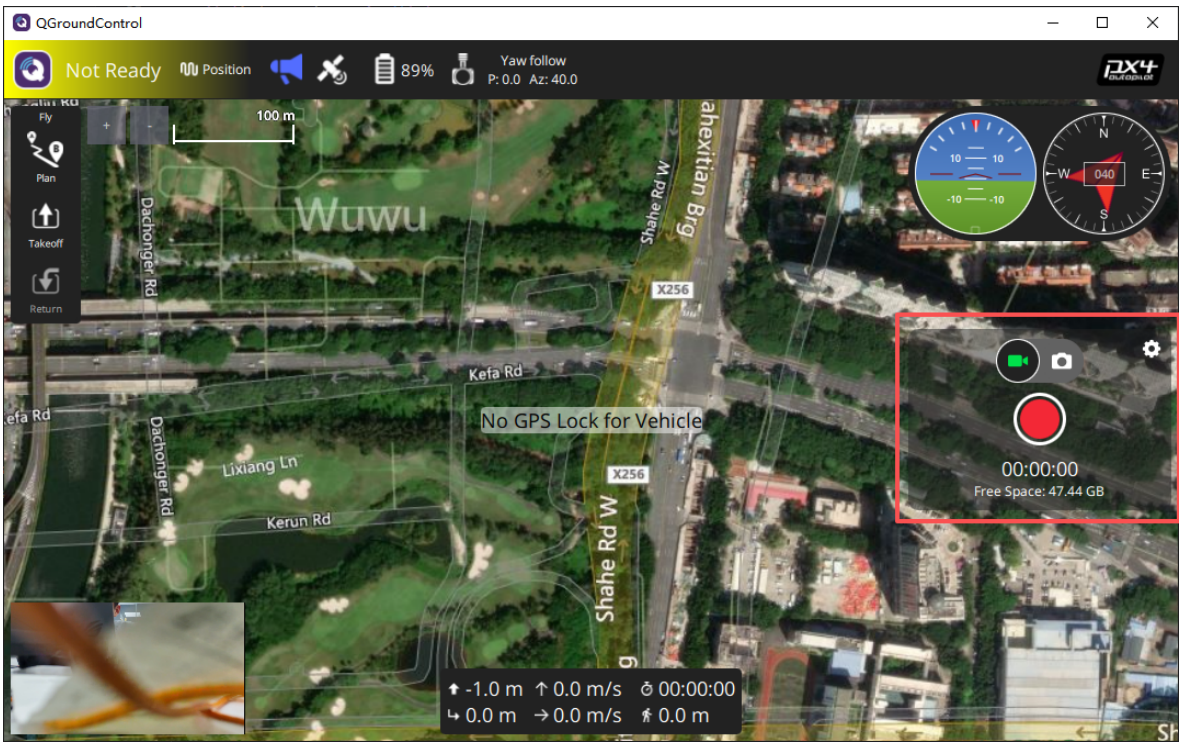
相机功能包括模式切换、单张拍照、开始录像、停止录像、拍照状态、录像状态和存储空间显示。实时视频通过 PSDK Liveview 获取并推送到 QGC，QGC 中可看到主画面和画中画地图。

相机模式切换 - 拍照



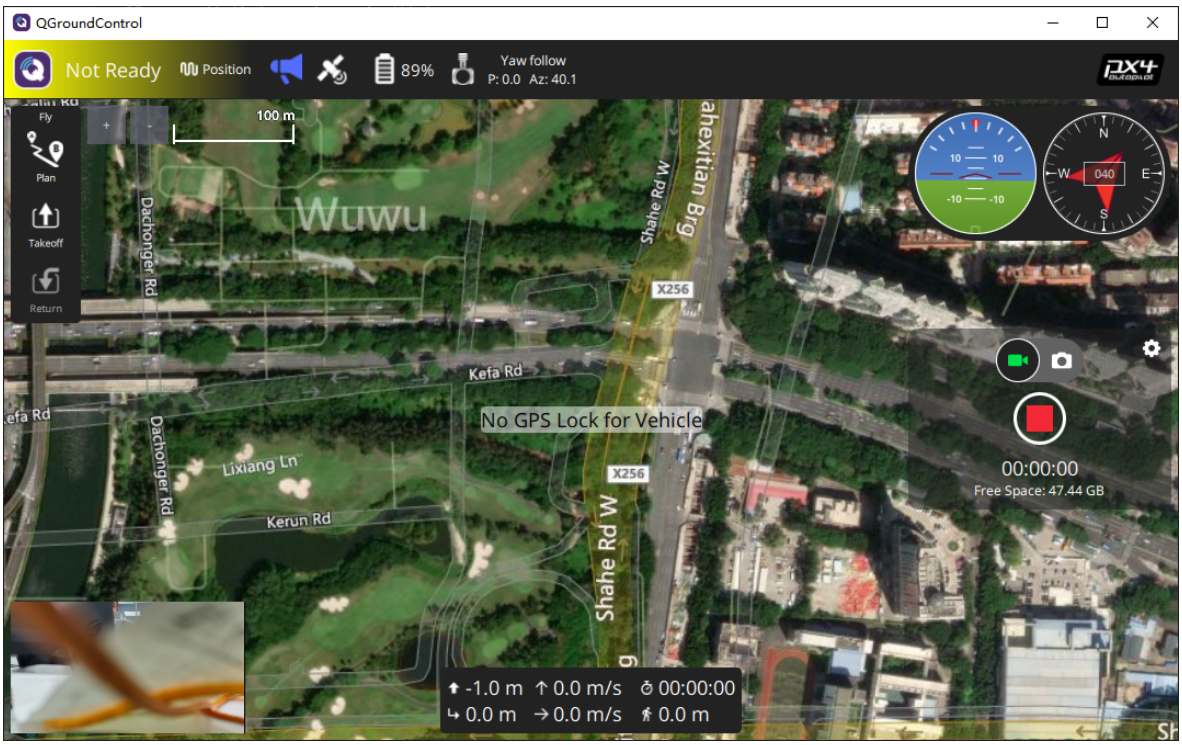
展示 QGC 相机面板切换到拍照模式后的界面状态，用户可通过拍照按钮触发单张拍摄。

相机模式切换 - 录像



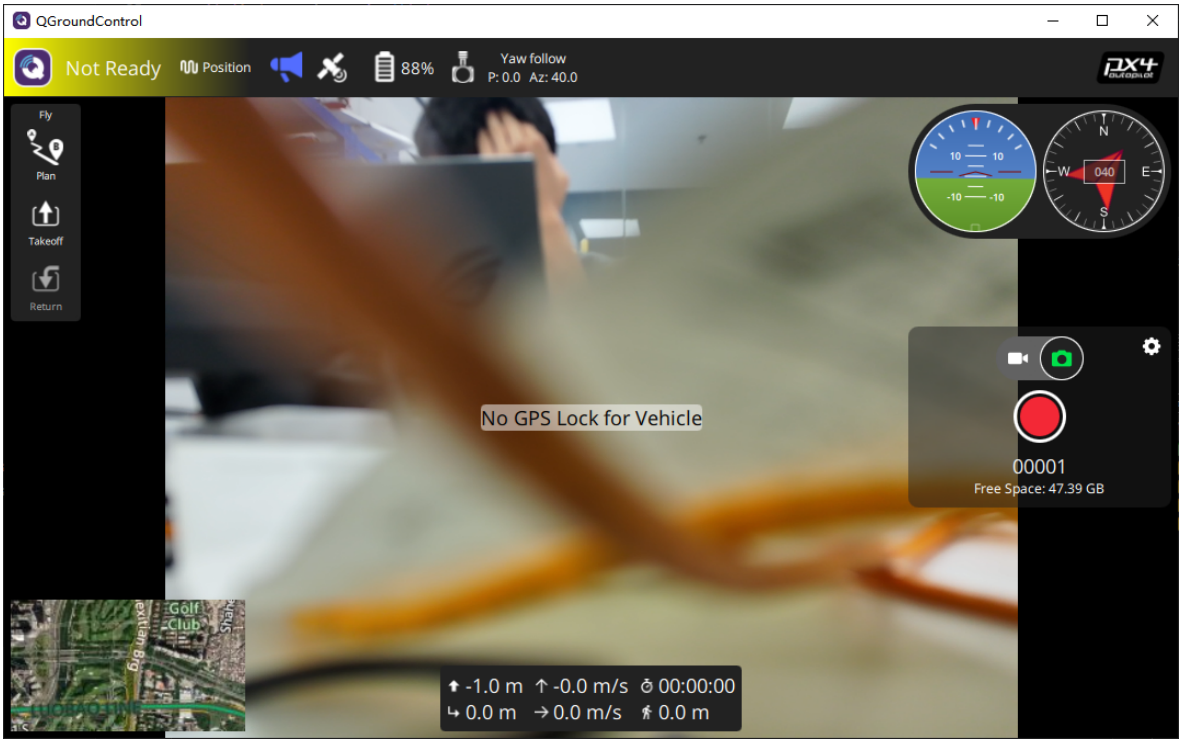
> 展示 QGC 相机面板切换到录像模式后的界面状态，用户可通过录像按钮开始或停止录像。

开始录像状态



> 展示 QGC 相机面板进入录像状态后的界面，用户可查看录像按钮状态和录像过程提示。

实时视频与相机控制

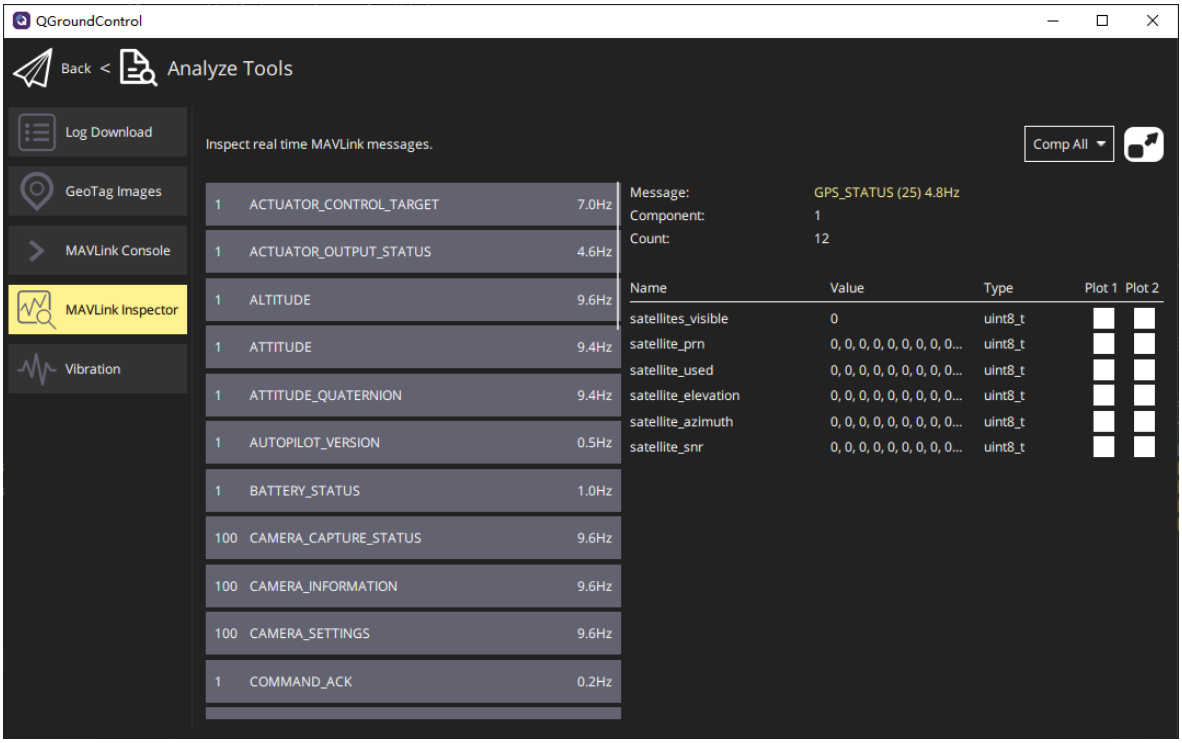


> 展示 QGC 视频画面、拍照/录像按钮、剩余存储空间和画中画地图，对应视频流、相机状态和存储信息。

7. MAVLink Inspector 验证

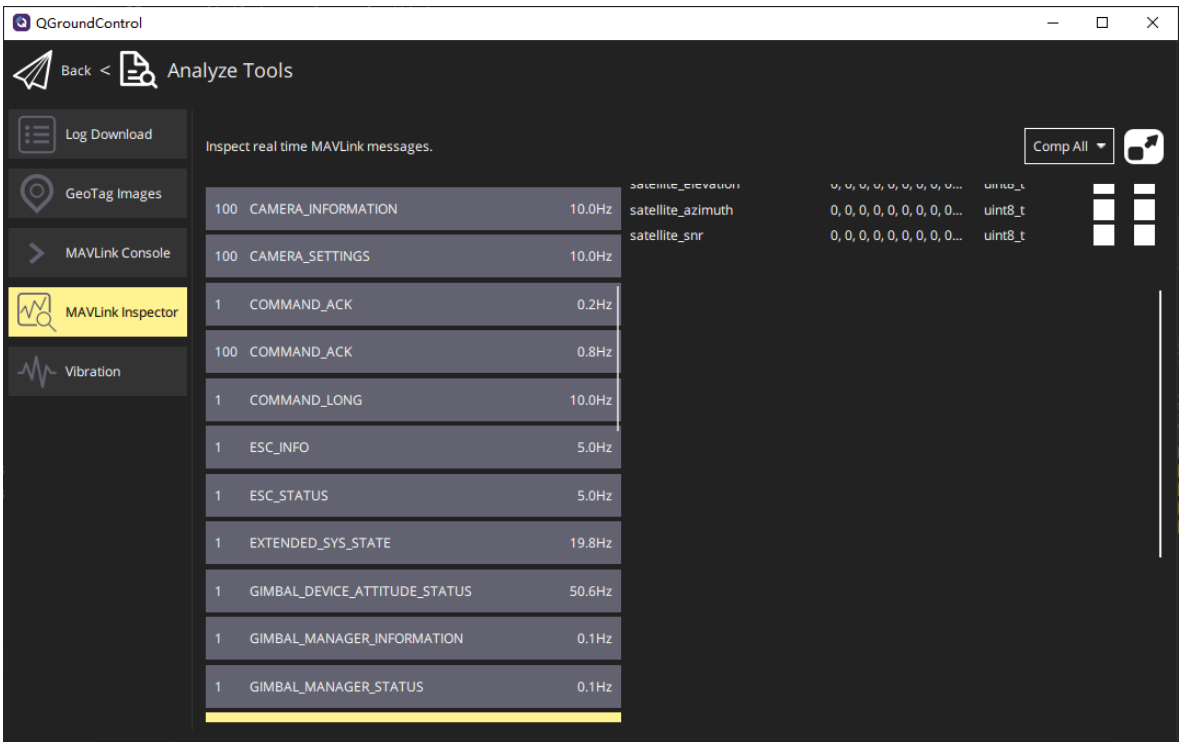
Analyze Tools 中的 MAVLink Inspector 是查看底层 MAVLink 数据的辅助工具。它可以看到每条 MAVLink 消息的 component、频率和字段值。若 QGC 页面没有显示预期数据，可在 Inspector 中检查对应消息是否存在、频率是否稳定、字段值是否合理。

MAVLink Inspector - 实时消息



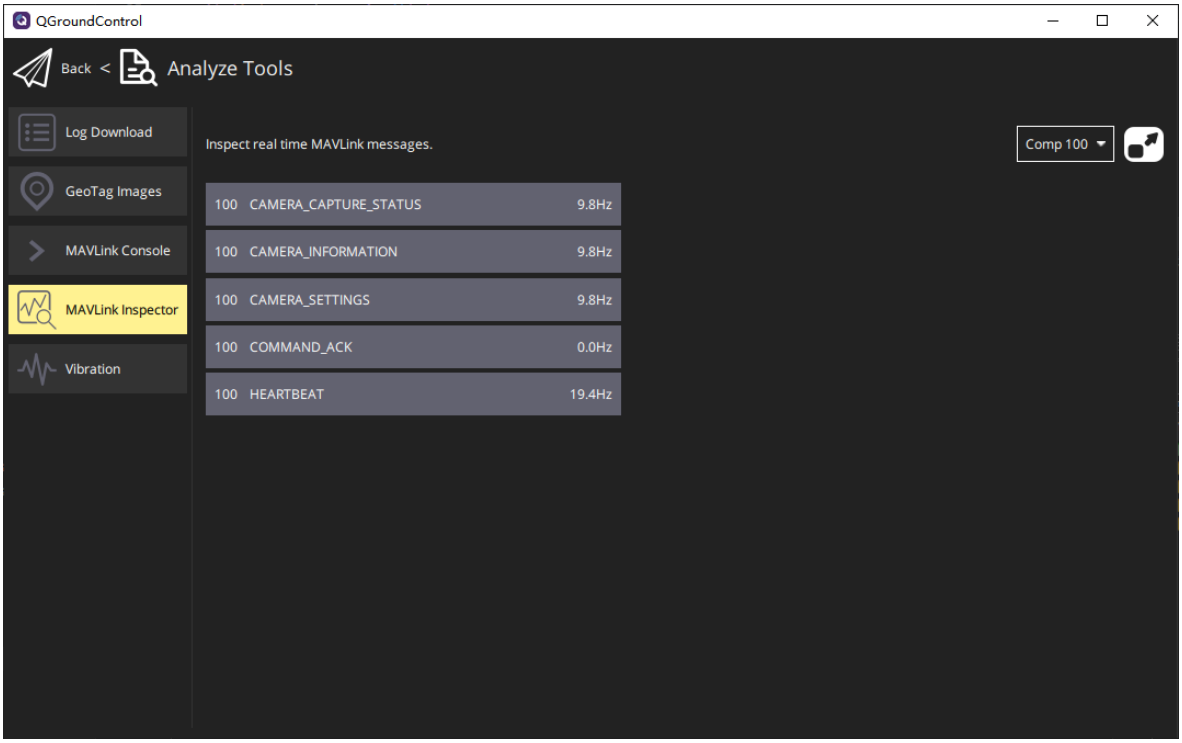
> 展示实时 MAVLink 消息列表、频率、component 和字段值，是验证 PSDK 转 MAVLink 数据是否正常的主要工具。

MAVLink Inspector - 状态消息



> 展示更多飞控状态类消息，便于检查姿态、GPS、电池、相机和任务消息频率。

MAVLink Inspector - 相机/云台消息



> 展示相机、云台或扩展消息列表，用于验证相机状态、拍照录像和云台控制相关 MAVLink 数据。

8. 日常使用检查

检查项	正常表现	检查位置
QGC 连接	顶部状态栏出现飞行器连接状态，地图和姿态数据刷新	Fly 页面
参数读取	Parameters 页面能显示参数列表，Analyze Tools 中存在 PARAM_VALUE	设置 / Parameters
遥测显示	位置、姿态、电池、高度、速度持续更新	Fly 页面 / MAVLink Inspector
航点任务	Plan 页面可上传，Fly 页面显示航线并执行任务	Plan / Fly 页面
拍照录像	按钮操作后 QGC 状态变化，存储空间正常显示	Fly 页面相机面板

检查项	正常表现	检查位置
视频流	实时画面持续显示，无长时间黑屏	Fly 页面视频窗口

注意事项

- 若 QGC 显示 Not Ready，不一定代表 PSDK 转 MAVLink 链路异常，需要结合 Inspector 检查具体消息是否正常。
- 若 GPS 未锁定，地图位置和任务执行状态会受影响，应优先确认 GPS_RAW_INT、GLOBAL_POSITION_INT 是否有稳定数据。
- 若视频黑屏但 MAVLink 数据正常，应检查 UDP 地址、端口、防火墙、QGC 视频设置和 PSDK Liveview 状态。
- 若参数列表为空，应确认项目运行参数表已加载，并确认 QGC 的参数请求能得到 PARAM_VALUE 响应。

附录：界面索引

界面名称	说明
无人机概要页面	用于确认 QGC 已识别飞行器系统，重点查看系统 ID、传感器状态、电源状态、飞行模式和安全设置。
参数页面	用于查看 QGC 参数树与参数值，验证 PARAM_REQUEST_LIST、PARAM_VALUE、PARAM_SET 等参数协议是否能正常响应。
飞行主界面 - 地图与状态联动	用于观察位置、姿态、电池、GPS、任务和相机状态在 QGC 中的综合呈现。
飞行主界面	展示地图、姿态仪、航向、当前位置、电池、电台链路和相机控制入口，是用户日常查看飞行状态的主要页面。
电池状态弹窗	展示电池电压、电流、剩余电量和健康状态，对应 BATTERY_STATUS、SYS_STATUS 等状态消息。
GPS 状态弹窗	展示 GPS 锁定、卫星、HDOP/VDOP、RTK 或定位质量相关信息，对应 GPS_RAW_INT、GPS_STATUS 等消息。

界面名称	说明
飞行操作菜单	展示起飞、返航、暂停、继续、任务等操作入口，反映 QGC 对当前飞行模式和安全状态的判断。
航线规划 - 地图编辑	展示在地图上添加航点和规划航线的过程。
航线规划 - 任务编辑	展示 Plan 页面中的航点编辑、任务高度、速度、航点参数和上传按钮。
航线规划 - 任务上传	展示航点任务准备上传的状态，对应 MISSION_COUNT、MISSION_REQUEST_INT、MISSION_ITEM_INT、MISSION_ACK 流程。
任务上传提示	展示 QGC 任务上传完成或准备执行的提示信息。
任务待执行状态	展示地图上的航线与飞行器当前位置，说明任务已在 QGC 中可视化。
任务执行过程 - 航向变化	展示任务执行时飞行器沿航线移动，姿态、速度、高度和航线状态同步显示。
任务执行过程 - 航点跟踪	展示航点任务执行中的航迹线、目标航点和当前飞行器位置。
任务执行过程 - 航线段显示	展示多段航线和任务边界，便于用户确认航线下发和任务状态。
任务控制 - 开始/继续	展示任务执行控制条，可用于开始、暂停、继续任务。
任务控制 - 暂停	展示任务暂停或状态切换时的 QGC 提示。
任务控制 - 命令确认	展示执行任务命令时 QGC 的命令提示和状态反馈。
任务执行与返航入口	展示任务飞行过程中的地图、航线和 Return 入口。
权限与状态提示	展示 QGC 或系统弹窗提示，需要结合 PSDK 控制权和设备状态确认。
相机模式切换 - 拍照	展示 QGC 相机面板切换到拍照模式后的界面状态，用户可通过拍照按钮触发单张拍摄。
相机模式切换 - 录像	展示 QGC 相机面板切换到录像模式后的界面状态，用户可通过录像按钮开始或停止录像。
开始录像状态	展示 QGC 相机面板进入录像状态后的界面，

界面名称	说明
	用户可查看录像按钮状态和录像过程提示。
MAVLink Inspector - 实时消息	展示实时 MAVLink 消息列表、频率、component 和字段值，是验证 PSDK 转 MAVLink 数据是否正常的主要工具。
MAVLink Inspector - 状态消息	展示更多飞控状态类消息，便于检查姿态、GPS、电池、相机和任务消息频率。
MAVLink Inspector - 相机/云台消息	展示相机、云台或扩展消息列表，用于验证相机状态、拍照录像和云台控制相关 MAVLink 数据。
实时视频与相机控制	展示 QGC 视频画面、拍照/录像按钮、剩余存储空间和画中画地图，对应视频流、相机状态和存储信息。

Mavlink功能

PSDK-Payload（基础功能）

机型信息

mavlink id 0 HEARTBEAT

返回字段名称	类型	单位	描
custom_mode	uint32_t	PX4 custom_mode 编码格式	无
type	uint8_t	MAV_TYPE 见下表	飞
autopilot	uint8_t	MAV_AUTOPILOT 见下表	无
base_mode	uint8_t	MAV_MODE_FLAG 见下表	无
system_status	uint8_t	MAV_STATE 见下表	无
mavlink_version	uint8_t		m

PX4 custom_mode

低16位保留，16~24位为主模式，24~32位为子模式

飞行模式	值
POSCTL	0x00030000

TAKEOFF	0x02040000
HOLD	0x03040000
MISSION	0x04040000
RTL	0x05040000
LAND	0x06040000

MAV_TYPE

MAVLINK 组件类型在 HEARTBEAT 消息中报告。飞控必须报告所安装飞行器的类型。

Value	Field Name	Description
0	MAV_TYPE_GENERIC	通用微型飞行
1	MAV_TYPE_FIXED_WING	固定翼
2	MAV_TYPE_QUADROTOR	四旋翼
3	MAV_TYPE_COAXIAL	共轴直升机
4	MAV_TYPE_HELICOPTER	带尾桨的普通
5	MAV_TYPE_ANTENNA_TRACKER	地面天线跟踪
6	MAV_TYPE_GCS	操作员控制台

MAV_AUTOPILOT

微型飞行器/自动驾驶仪类型。标志着各个模型的具体特征。

Value	Field Name	Description
0	MAV_AUTOPILOT_GENERIC	通用自动驾
1	MAV_AUTOPILOT_RESERVED	预留给未来
12	MAV_AUTOPILOT_PX4	PX4 自动驾

MAV_MODE_FLAG

无人机基础状态的位掩码。

Value	Field Name	Descriptio
0b00000001	MAV_MODE_FLAG_CUSTOM_MODE_ENABLED	系统特定 使用该标 其他所有
0b00000100	MAV_MODE_FLAG_AUTO_ENABLED	启用自动 无人机自
0b00010000	MAV_MODE_FLAG_STABILIZE_ENABLED	飞控自动 但需要接
0b00100000	MAV_MODE_FLAG_HIL_ENABLED	硬件进入 所有电机 但内部软
0b01000000	MAV_MODE_FLAG_MANUAL_INPUT_ENABLED	遥控输入

0b10000000	MAV_MODE_FLAG_SAFETY_ARMED	MAV 安全 电机已启

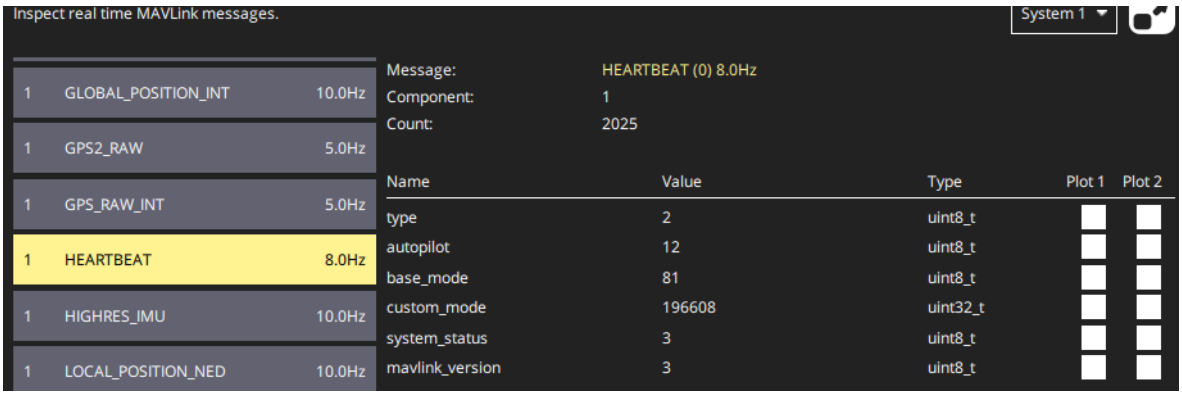
MAV_STATE

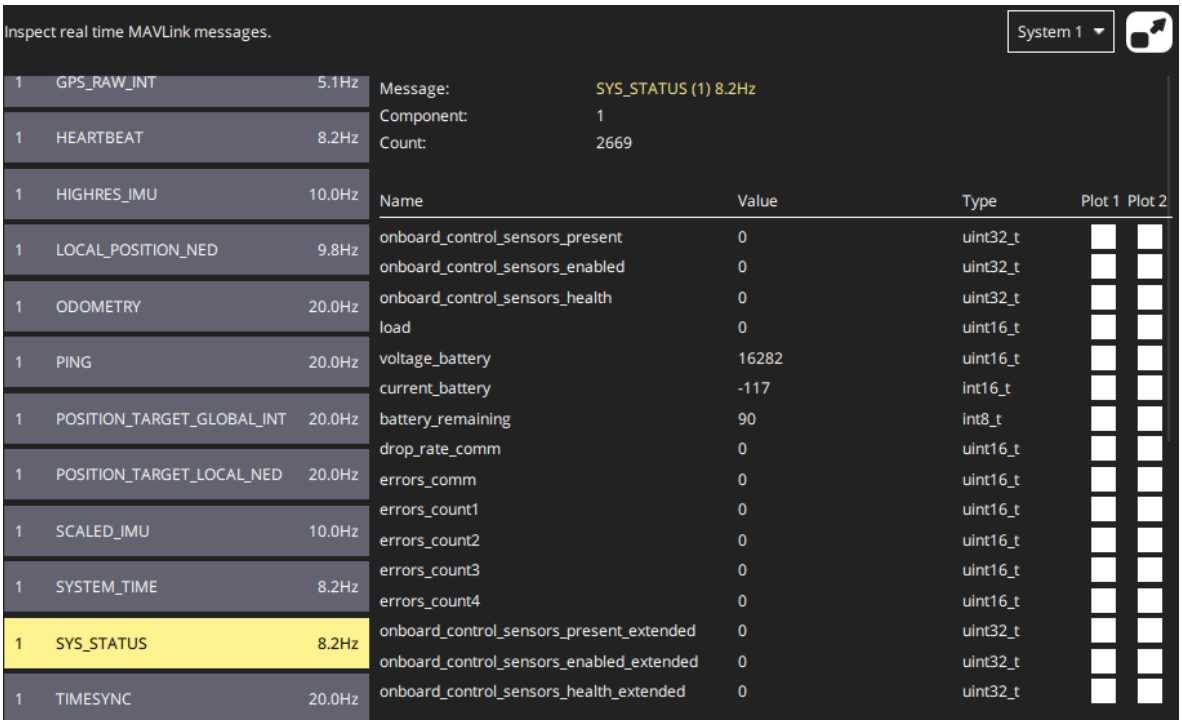
Value	Field Name	Description
3	MAV_STATE_STANDBY	当无人机处
4	MAV_STATE_ACTIVE	当无人机处

mavlink id 1 SYS_STATUS

返回字段名称	类型	单位
voltage_battery	uint16_t	mV
current_battery	int16_t	mA
battery_remaining	int8_t	%
onboard_control_sensors_enabled_extended	uint32_t	

结果（使用QGC查看）





获取移动设备App的基础信息

MOBILE_APP_INFO

mavlink消息：

接收PARAM_EXT_REQUEST_READ
发送PARAM_EXT_VALUE
参数：
T_DjiMobileAppInfo

字段	类型	说明
appLanguage	E_DjiMobileAppLanguage	APP 系统语言

字段	类型	说明
appScreenType	E_DjiMobileAppScreenType	APP 屏幕尺寸类型

APP 系统语言类型

E_DjiMobileAppLanguage

名称	数值	说明
DJI_MOBILE_APP_LANGUAGE_UNKNOWN	255	未知语言
DJI_MOBILE_APP_LANGUAGE_ENGLISH	0	英语
DJI_MOBILE_APP_LANGUAGE_CHINESE	1	中文
DJI_MOBILE_APP_LANGUAGE_JAPANESE	2	日语
DJI_MOBILE_APP_LANGUAGE_FRENCH	3	法语

移动 APP 屏幕尺寸类型

E_DjiMobileAppScreenType

名称	数值	说明
DJI_MOBILE_APP_SCREEN_TYPE_UNKNOWN	255	未知屏幕类型
DJI_MOBILE_APP_SCREEN_TYPE_BIG_SCREEN	0	大屏设备（屏幕尺寸 ≥ 6 英寸）
DJI_MOBILE_APP_SCREEN_TYPE_LITTLE_SCREEN	1	小屏设备（屏幕尺寸 < 6 英寸）

示例

```
import time
import sys
import struct
import ctypes
from pymavlink import mavutil

def connect_to_vehicle():
    """建立 UDP 监听连接（默认监听所有接口的 14550 端口）"""
```

```
    connection_string = "udpin:0.0.0.0:14550"
    print(f"Listening for MAVLink on: {connection_string}")
    vehicle = mavutil.mavlink_connection(connection_string)
    print("Waiting for HEARTBEAT (timeout: 10s)...")
    try:
        vehicle.wait_heartbeat(timeout=10)
        print(f"  Heartbeat received from system {vehicle.target_sy
stem}, component {vehicle.target_component}")
        return vehicle
    except Exception as e:
        print(f"  Connection failed: {str(e)}")
        print("Ensure your vehicle is sending MAVLink messages to 1
4550 port")
        print("Example QGroundControl setup: Settings > General > M
AVLink > Listening Port = 14550")
        sys.exit(1)

def format_param_id(param_id):
    """格式化16字节参数ID（符合MAVLink协议要求）"""
    if len(param_id) > 16:
        raise ValueError("param_id cannot exceed 16 characters")

    param_id_bytes = bytearray(16)
    for i in range(min(len(param_id), 16)):
        param_id_bytes[i] = ord(param_id[i])

    # 长度<16时添加NULL终止符
    if len(param_id) < 16:
        param_id_bytes[len(param_id)] = 0

    return param_id_bytes

# ============ 结构体打包函数 ============

def pack_uint8_xy(x, y):
    """打包 {uint8_t x; uint8_t y} 结构体 (2字节)"""
    return struct.pack('<BB', x, y)

def pack_float32_xy(x, y):
    """打包 {float32 x; float32 y} 结构体 (8字节)"""
    return struct.pack('<ff', x, y)

def pack_float32_lxlyrxry(LX, LY, RX, RY):
    """打包 {float32 LX; float32 LY; float32 RX; float32 RY;} 结构体
(16字节)"""
```

```
    return struct.pack('<ffff', LX, LY, RX, RY)

def pack_ering_mode(size, eringMode, minValue, maxValue):
    """打包 {uint8_t size; uint8_t eringMode[16]; uint32_t minValue; uint32_t maxValue;} 结构体 (25字节)

    Args:
        size: uint8_t, eringMode数组的有效长度
        eringMode: list of int (0-255), 16个uint8_t值
        minValue: uint32_t, 最小值
        maxValue: uint32_t, 最大值

    Returns:
        bytes: 打包后的二进制数据
    """
    # 确保eringMode是16字节
    eringMode_bytes = bytearray(16)
    for i in range(min(len(eringMode), 16)):
        eringMode_bytes[i] = eringMode[i] & 0xFF

    return struct.pack('<B16sII', size, eringMode_bytes, minValue, maxValue)

def format_param_value(param_value):
    """格式化128字节参数值（接受二进制输入）"""
    if isinstance(param_value, str):
        # 如果是十六进制字符串（如 "48656c6c6f"），先转换为字节
        if param_value.startswith('0x') or all(c in '0123456789abcdefABCDEF' for c in param_value.replace(' ', '')):
            hex_str = param_value.replace('0x', '').replace(' ', '')
            if len(hex_str) % 2 == 0:
                param_value = bytes.fromhex(hex_str)
            else:
                raise ValueError("Invalid hex string: must have even number of characters")
        else:
            # 普通字符串转字节
            param_value = param_value.encode('utf-8')

    if len(param_value) > 128:
        raise ValueError("param_value cannot exceed 128 bytes")

    param_value_bytes = bytearray(128)
    param_value_bytes[:len(param_value)] = param_value
```

```
    return param_value_bytes

def hex_dump(data, bytes_per_line=16):
    """格式化二进制数据为 hex + ASCII 展示（类 hexdump 风格）

    Returns:
        str: 格式化后的多行字符串
    """
    lines = []
    for i in range(0, len(data), bytes_per_line):
        chunk = data[i:i + bytes_per_line]
        hex_part = ' '.join(f'{b:02X}' for b in chunk)
        ascii_part = ''.join(chr(b) if 32 <= b < 127 else '.' for b
in chunk)
        lines.append(f"  {i:04X}  {hex_part:<{bytes_per_line*3}}
{ascii_part}")
    return '\n'.join(lines)

def send_param_ext_request_read(vehicle, param_id=None, param_index
=-1):
    """发送参数读取请求"""
    param_id_bytes = format_param_id(param_id) if param_id else byt
earray(16)

    vehicle.mav.param_ext_request_read_send(
        vehicle.target_system,
        vehicle.target_component,
        param_id_bytes,
        param_index
    )

    target = f"index {param_index}" if param_index >= 0 else f"ID
'{param_id}'"
    print(f"  Sent PARAM_EXT_REQUEST_READ for {target}")

def send_param_ext_request_list(vehicle):
    """发送请求所有参数列表"""
    vehicle.mav.param_ext_request_list_send(
        vehicle.target_system,
        vehicle.target_component
    )
    print(f"  Sent PARAM_EXT_REQUEST_LIST to system {vehicle.target
_system}, component {vehicle.target_component}")
```

```
def send_param_ext_set(vehicle, param_id, param_value, param_type):
    """发送参数设置请求"""
    vehicle.mav.param_ext_set_send(
        vehicle.target_system,
        vehicle.target_component,
        format_param_id(param_id),
        format_param_value(param_value),
        param_type
    )
    print(f"  Sent PARAM_EXT_SET for '{param_id}' = '{param_value}'
(type: {param_type})")

def receive_param_ext_value(vehicle, timeout=5.0):
    """接收参数值响应"""
    print(f"  Waiting for PARAM_EXT_VALUE ({timeout}s timeout)...")
    msg = vehicle.recv_match(type='PARAM_EXT_VALUE', blocking=True,
timeout=timeout)

    if not msg:
        print("  Timeout: No PARAM_EXT_VALUE received")
        return None

    # 解析参数ID
    param_id = msg.param_id

    # 解析参数值（直接获取128字节原始数据）
    param_value_raw = msg.param_value
    # byte_pointer = (ctypes.c_ubyte * 128).from_buffer_copy(msg.pa
ram_value)
    if isinstance(param_value_raw, (bytes, bytearray)):
        param_value_bytes = bytes(param_value_raw)
    # param_value_bytes = bytes(byte_pointer)
    elif isinstance(param_value_raw, str):
        # 替换无法编码的字符（如 \ufffd）为 0x00，还原为128字节
        param_value_bytes = param_value_raw.encode('latin1', errors
='replace')
    else:
        param_value_bytes = bytes(param_value_raw)

    # 参数类型映射
    type_names = {
        0: "INT8", 1: "UINT8", 2: "INT16", 3: "UINT16", 4: "INT32",
        5: "UINT32", 6: "INT64", 7: "UINT64", 8: "FLOAT", 9: "FLOA
T",
        10: "CHAR", 11: "CHAR", 12: "UINT8_ENUM", 13: "STRING"
```

```
    }
    param_type = type_names.get(msg.param_type, f"UNKNOWN({msg.para
m_type})")

    print("\n  Received PARAM_EXT_VALUE:")
    print(f"    ID: {param_id}")
    print(f"    Type: {param_type}  |  Data: {len(param_value_byte
s)} bytes")
    print(f"    Index/Total: {msg.param_index}/{msg.param_count}")
    print(hex_dump(param_value_bytes))

    return {
        'id': param_id,
        'value': param_value_bytes,
        'type': param_type,
        'index': msg.param_index,
        'count': msg.param_count
    }


def receive_all_param_ext_values(vehicle, timeout=10.0):
    """接收所有参数值，直到收到 param_count 条消息后退出循环"""
    received_params = []
    expected_count = None
    print(f"  Waiting for all PARAM_EXT_VALUE messages...")

    while True:
        msg = vehicle.recv_match(type='PARAM_EXT_VALUE', blocking=T
rue, timeout=timeout)

        if not msg:
            print(f"  Timeout: Received {len(received_params)} para
meters (expected {expected_count if expected_count else 'unknow
n'})")
            break

        param_id = msg.param_id
        param_value_raw = msg.param_value

        # 直接获取128字节原始数据
        if isinstance(param_value_raw, (bytes, bytearray)):
            param_value_bytes = bytes(param_value_raw)
        elif isinstance(param_value_raw, str):
            # 替换无法编码的字符（如 \ufffd）为 0x00
            param_value_bytes = param_value_raw.encode('latin1', er
```

```
rors='replace')
            else:
                param_value_bytes = bytes(param_value_raw)

            # 参数类型映射
            type_names = {
                0: "INT8", 1: "UINT8", 2: "INT16", 3: "UINT16", 4: "INT
32",
                5: "UINT32", 6: "INT64", 7: "UINT64", 8: "FLOAT", 9: "D
OUBLE",
                10: "CHAR", 11: "BOOL", 12: "UINT8_ENUM", 13: "STRING"
            }
            param_type = type_names.get(msg.param_type, f"UNKNOWN({msg.
param_type})")

            # 首次收到消息时记录总参数数量
            if expected_count is None:
                expected_count = msg.param_count
                print(f"  Total parameters to receive: {expected_coun
t}")

            print(f"  [{msg.param_index}/{expected_count}] {param_id}
[{param_type}][{len(param_value_bytes)}B]")
            print(hex_dump(param_value_bytes))

            received_params.append({
                'id': param_id,
                'value': param_value_bytes,
                'type': param_type,
                'index': msg.param_index,
                'count': msg.param_count
            })

            # 当已接收数量达到预期总数时退出循环
            if len(received_params) >= expected_count:
                print(f"\n  Successfully received all {expected_count}
parameters")
                break

    return received_params

def receive_param_ext_ack(vehicle, timeout=5.0):
    """接收参数设置确认"""
    print(f"  Waiting for PARAM_EXT_ACK ({timeout}s timeout)...")
    msg = vehicle.recv_match(type='PARAM_EXT_ACK', blocking=True, t
```

```
imeout=timeout)

    if not msg:
        print("  Timeout: No PARAM_EXT_ACK received")
        return None

    param_id = msg.param_id
    param_value = msg.param_value

    # 结果代码映射
    result_codes = {
        0: "  PARAM_ACK_ACCEPTED",
        1: "⚠ PARAM_ACK_VALUE_UNSUPPORTED",
        2: "  PARAM_ACK_FAILED",
        3: "  PARAM_ACK_IN_PROGRESS"
    }
    result = result_codes.get(msg.param_result, f"  UNKNOWN({msg.pa
ram_result})")

    print("\n  Received PARAM_EXT_ACK:")
    print(f"    ID: {param_id}")
    print(f"    Value: {param_value}")
    print(f"    Result: {result}")

    return {
        'id': param_id,
        'value': param_value,
        'result': result,
        'result_code': msg.param_result
    }

def main():
    print("  Starting MAVLink PARAM_EXT Tool (UDP Listening Mode)")
    print("-----------------------------------------------------")
    print("This script listens for MAVLink on ALL interfaces:1455
0")
    print("Ensure your vehicle is configured to send to this port")
    print("-----------------------------------------------------
\n")

    vehicle = connect_to_vehicle()

    # 操作选择
    print("\nAvailable operations:")
    print("1. Read parameter value (by ID or index)")
```

```
    print("2. Set parameter value (raw/hex/string)")
    print("3. Request all parameters (PARAM_EXT_REQUEST_LIST)")
    print("4. Set struct: uint8_xy {x, y}")
    print("5. Set struct: float32_xy {x, y}")
    print("6. Set struct: float32_lxlyrxry {LX, LY, RX, RY}")
    print("7. Set struct: ering_mode {size, eringMode[16], minValu
e, maxValue}")
    choice = input("\nSelect operation [1-7]: ").strip()

    if choice == '1':
        # mode = input("Read by [id/index]: ").lower()
        mode = 'id'
        if mode == 'id':
            param_id = input("Enter parameter ID: ").strip()
            send_param_ext_request_read(vehicle, param_id=param_id)
            receive_param_ext_value(vehicle)
        elif mode == 'index':
            param_index = int(input("Enter parameter index: "))
            send_param_ext_request_read(vehicle, param_index=param_
index)
            receive_param_ext_value(vehicle)
        else:
            print("Invalid mode. Use 'id' or 'index'")

    elif choice == '2':
        param_id = input("Enter parameter ID to set: ").strip()

        print("\nEnter parameter value (supports):")
        print("  - Plain string (e.g., hello)")
        print("  - Hex string (e.g., 48656c6c6f or 0x48656c6c6f)")
        print("  - Raw bytes (e.g., b'\\x48\\x65\\x6c\\x6c\\x6f')")
        param_value_input = input("Enter value: ").strip()

        # 解析输入值
        if param_value_input.startswith("b'") or param_value_input.
startswith('b"'):
            # Python字节字面量格式
            param_value = eval(param_value_input)
        elif param_value_input.startswith('0x') or all(c in '012345
6789abcdefABCDEF ' for c in param_value_input):
            # 十六进制格式
            hex_str = param_value_input.replace('0x', '').replace('
', '')
            try:
                param_value = bytes.fromhex(hex_str)
```

```
            except ValueError:
                param_value = param_value_input.encode('utf-8')
        else:
            # 默认作为字符串处理
            param_value = param_value_input.encode('utf-8')

        print("\nParameter types:")
        print("  8 = FLOAT (most common)")
        print("  13 = STRING")
        print("  10 = CHAR (raw bytes)")
        print("  Other types require specific binary formats")
        param_type = int(input("Enter type number [8]: ") or 8)

        send_param_ext_set(vehicle, param_id, param_value, param_ty
pe)
        receive_param_ext_ack(vehicle)

    elif choice == '4':
        """设置 uint8_xy 结构体 {uint8_t x; uint8_t y}"""
        param_id = input("Enter parameter ID: ").strip()
        x = int(input("Enter x (0-255): "))
        y = int(input("Enter y (0-255): "))
        param_value = pack_uint8_xy(x, y)
        print(f"  Packed: {param_value.hex()} ({len(param_value)} b
ytes)")
        send_param_ext_set(vehicle, param_id, param_value, 10)  # C
HAR type
        receive_param_ext_ack(vehicle)

    elif choice == '5':
        """设置 float32_xy 结构体 {float32 x; float32 y}"""
        param_id = input("Enter parameter ID: ").strip()
        x = float(input("Enter x: "))
        y = float(input("Enter y: "))
        param_value = pack_float32_xy(x, y)
        print(f"  Packed: {param_value.hex()} ({len(param_value)} b
ytes)")
        send_param_ext_set(vehicle, param_id, param_value, 10)  # C
HAR type
        receive_param_ext_ack(vehicle)

    elif choice == '6':
        """设置 float32_lxlyrxry 结构体 {float32 LX; float32 LY; floa
t32 RX; float32 RY;}"""
        param_id = input("Enter parameter ID: ").strip()
```

```
        LX = float(input("Enter LX: "))
        LY = float(input("Enter LY: "))
        RX = float(input("Enter RX: "))
        RY = float(input("Enter RY: "))
        param_value = pack_float32_lxlyrxry(LX, LY, RX, RY)
        print(f"  Packed: {param_value.hex()} ({len(param_value)} bytes)")
        send_param_ext_set(vehicle, param_id, param_value, 10)  # CHAR type
        receive_param_ext_ack(vehicle)

    elif choice == '7':
        """设置 ering_mode 结构体 {uint8_t size; uint8_t eringMode[16]; uint32_t minValue; uint32_t maxValue;}"""
        param_id = input("Enter parameter ID: ").strip()
        size = int(input("Enter size (0-16): "))
        eringMode = []
        print("Enter 16 eringMode values (0-255), one per line:")
        for i in range(16):
            val = int(input(f"  eringMode[{i}]: "))
            eringMode.append(val)
        minValue = int(input("Enter minValue: "))
        maxValue = int(input("Enter maxValue: "))
        param_value = pack_ering_mode(size, eringMode, minValue, maxValue)
        print(f"  Packed: {param_value.hex()} ({len(param_value)} bytes)")
        send_param_ext_set(vehicle, param_id, param_value, 10)  # CHAR type
        receive_param_ext_ack(vehicle)

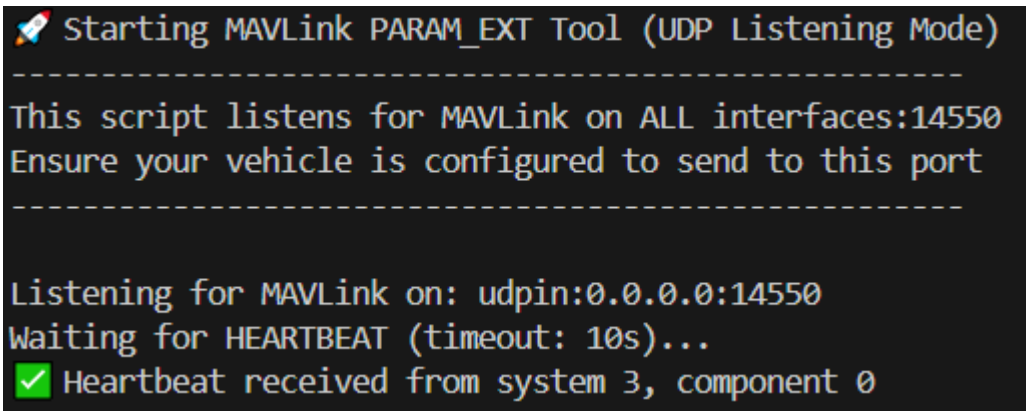
    elif choice == '3':
        print("\n  Requesting all parameters...")
        send_param_ext_request_list(vehicle)
        all_params = receive_all_param_ext_values(vehicle)
        print(f"\n  Total parameters received: {len(all_params)}")

    else:
        print("Invalid choice. Select 1, 2, or 3.")

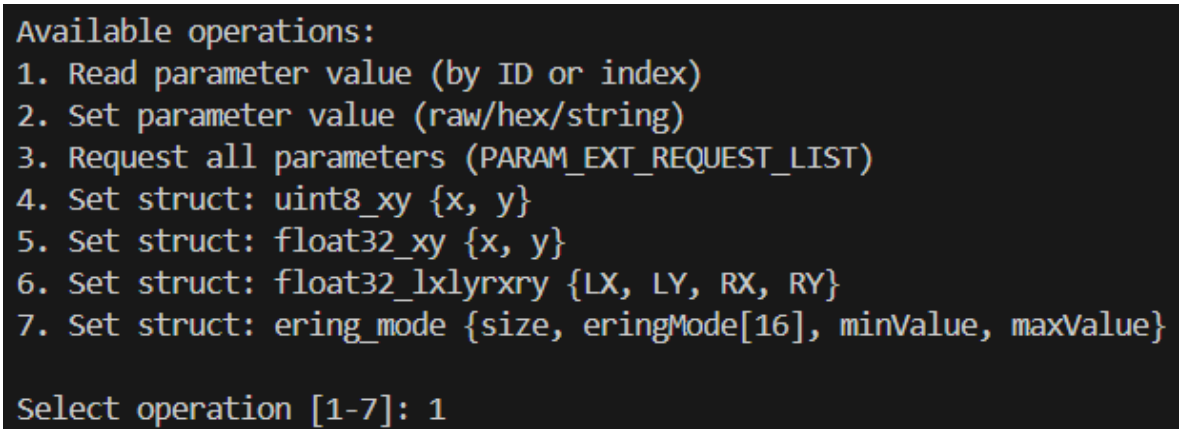

if __name__ == "__main__":
    main()
```

示例使用说明

运行后，成功建立连接



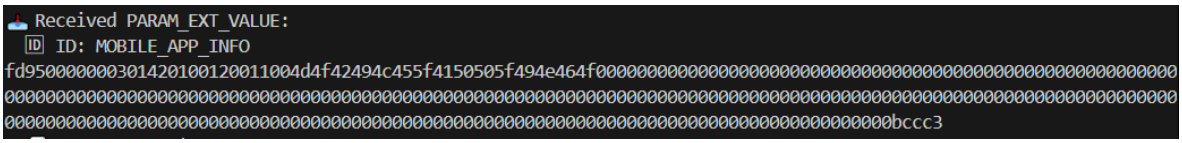
输入“1”选择设置参数



输入对应参数“MOBILE_APP_INFO”



成功返回完整消息


📥 Received PARAM_EXT_VALUE:
ID: MOBILE_APP_INFO
fd950000003014201001200110004d4f42494c455f4150505f494e464f0000000000000000000000000000000000000000000000000000000000000000
0000000000000000000000000000000000000000000000000000000000000

名称	枚举值	说明
DJI_ENHANCEED_TRANSMISSION_STATE_DISABLED	0	Mobile app 图传设置：关闭增强图传（Enhanced transmission disabled）
DJI_ENHANCEED_TRANSMISSION_STATE_ENABLED	3	Mobile app 图传设置：开启增强图传（Enhanced transmission enabled）

示例

```
import time
import sys
from pymavlink import mavutil

def connect_to_vehicle():
    """建立 UDP 监听连接（默认监听所有接口的 14550 端口）"""
    connection_string = "udpin:0.0.0.0:14550"
    print(f"Listening for MAVLink on: {connection_string}")
    vehicle = mavutil.mavlink_connection(connection_string)
    print("Waiting for HEARTBEAT (timeout: 10s)...")
    try:
        vehicle.wait_heartbeat(timeout=10)
        print(f"  Heartbeat received from system {vehicle.target_system}, component {vehicle.target_component}")
        return vehicle
    except Exception as e:
        print(f"  Connection failed: {str(e)}")
        print("Ensure your vehicle is sending MAVLink messages to 14550 port")
        print("Example QGroundControl setup: Settings > General > MAVLink > Listening Port = 14550")
        sys.exit(1)

def format_param_id(param_id):
    """格式化16字节参数ID（符合MAVLink协议要求）"""
    if len(param_id) > 16:
```

```
        raise ValueError("param_id cannot exceed 16 characters")

    param_id_bytes = bytearray(16)
    for i in range(min(len(param_id), 16)):
        param_id_bytes[i] = ord(param_id[i])

    # 长度<16时添加NULL终止符
    if len(param_id) < 16:
        param_id_bytes[len(param_id)] = 0

    return param_id_bytes

def send_param_set(vehicle, param_id, param_value, param_type):
    """发送参数设置请求"""
    vehicle.mav.param_set_send(
        vehicle.target_system,
        vehicle.target_component,
        format_param_id(param_id),
        param_value,
        param_type
    )
    print(f"  Sent PARAM_SET for '{param_id}' = {param_value} (typ
e: {param_type})")

def send_param_request_read(vehicle, param_id=None, param_index=-
1):
    """发送参数读取请求"""
    param_id_bytes = format_param_id(param_id) if param_id else byt
earray(16)

    vehicle.mav.param_request_read_send(
        vehicle.target_system,
        vehicle.target_component,
        param_id_bytes,
        param_index
    )

    target = f"index {param_index}" if param_index >= 0 else f"ID
'{param_id}'"
    print(f"  Sent PARAM_REQUEST_READ for {target}")

def send_param_request_list(vehicle):
    """发送请求所有参数列表"""
    vehicle.mav.param_request_list_send(
        vehicle.target_system,
```

```
        vehicle.target_component
    )
    print(f"  Sent PARAM_REQUEST_LIST to system {vehicle.target_sys
tem}, component {vehicle.target_component}")

def receive_param_value(vehicle, timeout=5.0):
    """接收参数值响应"""
    print(f"  Waiting for PARAM_VALUE ({timeout}s timeout)...")
    msg = vehicle.recv_match(type='PARAM_VALUE', blocking=True, tim
eout=timeout)

    if not msg:
        print("  Timeout: No PARAM_VALUE received")
        return None

    # 解析参数ID
    param_id = msg.param_id

    # 解析参数值
    param_value = msg.param_value

    # 参数类型映射
    type_names = {
        0: "INT8", 1: "UINT8", 2: "INT16", 3: "UINT16", 4: "INT32",
        5: "UINT32", 6: "INT64", 7: "UINT64", 8: "FLOAT", 9: "DOUBL
E"
    }
    param_type = type_names.get(msg.param_type, f"UNKNOWN({msg.para
m_type})")

    print("\n  Received PARAM_VALUE:")
    print(f"    ID: {param_id}")
    print(f"    Value: {param_value}")
    print(f"    Type: {param_type}")
    print(f"    Index/Total: {msg.param_index}/{msg.param_count}")

    return {
        'id': param_id,
        'value': param_value,
        'type': param_type,
        'index': msg.param_index,
        'count': msg.param_count
    }

def receive_all_param_values(vehicle, timeout=10.0):
```

```
    """接收所有参数值，直到收到 param_count 条消息后退出循环"""
    received_params = []
    expected_count = None
    print(f"  Waiting for all PARAM_VALUE messages...")

    while True:
        msg = vehicle.recv_match(type='PARAM_VALUE', blocking=True, timeout=timeout)

        if not msg:
            print(f"  Timeout: Received {len(received_params)} parameters (expected {expected_count if expected_count else 'unknown'})")
            break

        param_id = msg.param_id
        param_value = msg.param_value

        # 参数类型映射
        type_names = {
            0: "INT8", 1: "UINT8", 2: "INT16", 3: "UINT16", 4: "INT32",
            5: "UINT32", 6: "INT64", 7: "UINT64", 8: "FLOAT", 9: "DOUBLE"
        }
        param_type = type_names.get(msg.param_type, f"UNKNOWN({msg.param_type})")

        # 首次收到消息时记录总参数数量
        if expected_count is None:
            expected_count = msg.param_count
            print(f"  Total parameters to receive: {expected_count}")

        print(f"  [{msg.param_index}/{expected_count}] {param_id} = {param_value} ({param_type})")

        received_params.append({
            'id': param_id,
            'value': param_value,
            'type': param_type,
            'index': msg.param_index,
            'count': msg.param_count
        })
```

```
        # 当已接收数量达到预期总数时退出循环
        if len(received_params) >= expected_count:
            print(f"\n  Successfully received all {expected_count}
parameters")
            break

    return received_params

def main():
    print("  Starting MAVLink PARAM_SET Tool (UDP Listening Mode)")
    print("-----------------------------------------------------")
    print("This script listens for MAVLink on ALL interfaces:1455
0")
    print("Ensure your vehicle is configured to send to this port")
    print("-----------------------------------------------------
\n")

    vehicle = connect_to_vehicle()

    # 操作选择
    print("\nAvailable operations:")
    print("1. Set parameter value")
    print("2. Read parameter value (by ID or index)")
    print("3. Request all parameters (PARAM_REQUEST_LIST)")
    choice = input("\nSelect operation [1-3]: ").strip()

    if choice == '1':
        param_id = input("Enter parameter ID to set: ").strip()
        param_value = float(input("Enter new value: ").strip())

        print("\nParameter types:")
        print("  8 = FLOAT (most common)")
        print("  9 = DOUBLE")
        print("  4 = INT32")
        print("  1 = UINT8")
        param_type = int(input("Enter type number [8]: ").strip() o
r 8)

        send_param_set(vehicle, param_id, param_value, param_type)
        receive_param_value(vehicle)

    elif choice == '2':
        # mode = input("Read by [id/index]: ").lower()
        mode = 'id'
        if mode == 'id':
```

```
            param_id = input("Enter parameter ID: ").strip()
            send_param_request_read(vehicle, param_id=param_id)
            receive_param_value(vehicle)
        elif mode == 'index':
            param_index = int(input("Enter parameter index: "))
            send_param_request_read(vehicle, param_index=param_inde
x)
            receive_param_value(vehicle)
        else:
            print("Invalid mode. Use 'id' or 'index'")

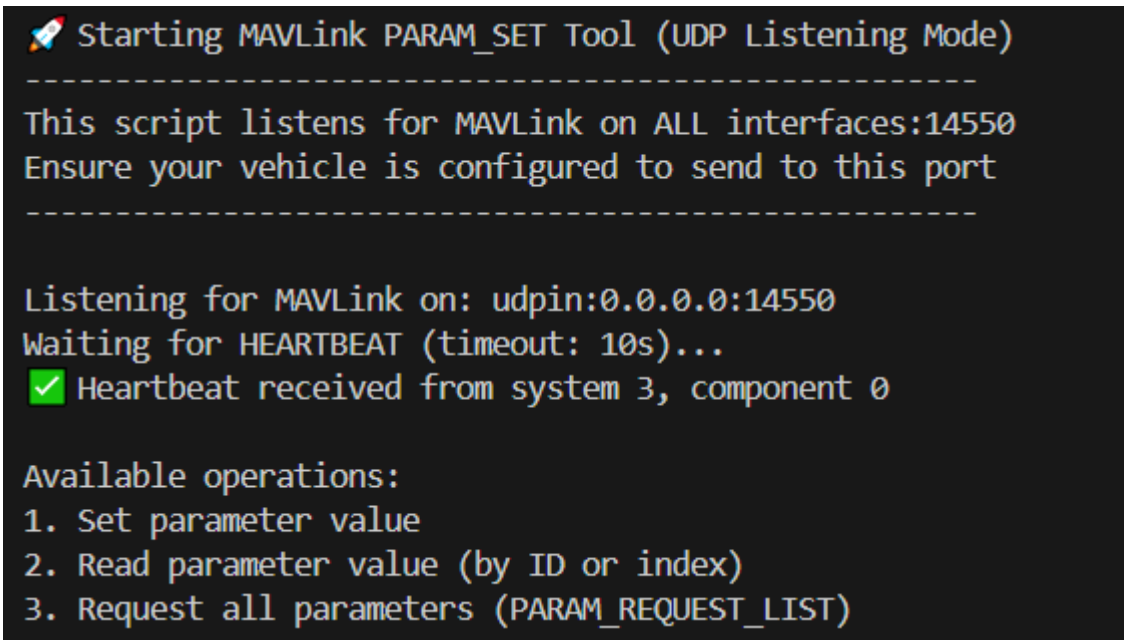
    elif choice == '3':
        print("\n  Requesting all parameters...")
        send_param_request_list(vehicle)
        all_params = receive_all_param_values(vehicle)
        print(f"\n  Total parameters received: {len(all_params)}")

    else:
        print("Invalid choice. Select 1, 2, or 3.")


if __name__ == "__main__":
    main()
```

示例使用说明

运行后，成功建立连接



输入“2”选择读取参数

```
Select operation [1-3]: 2
```

输入对应参数“ENH_TRANS_STATE”

```
Enter parameter ID: ENH_TRANS_STATE
```

成功返回该参数对应的数值

```
Received PARAM_VALUE:
  ID: ENH_TRANS_STATE
  Value: 0.0
```

相机视频流传输

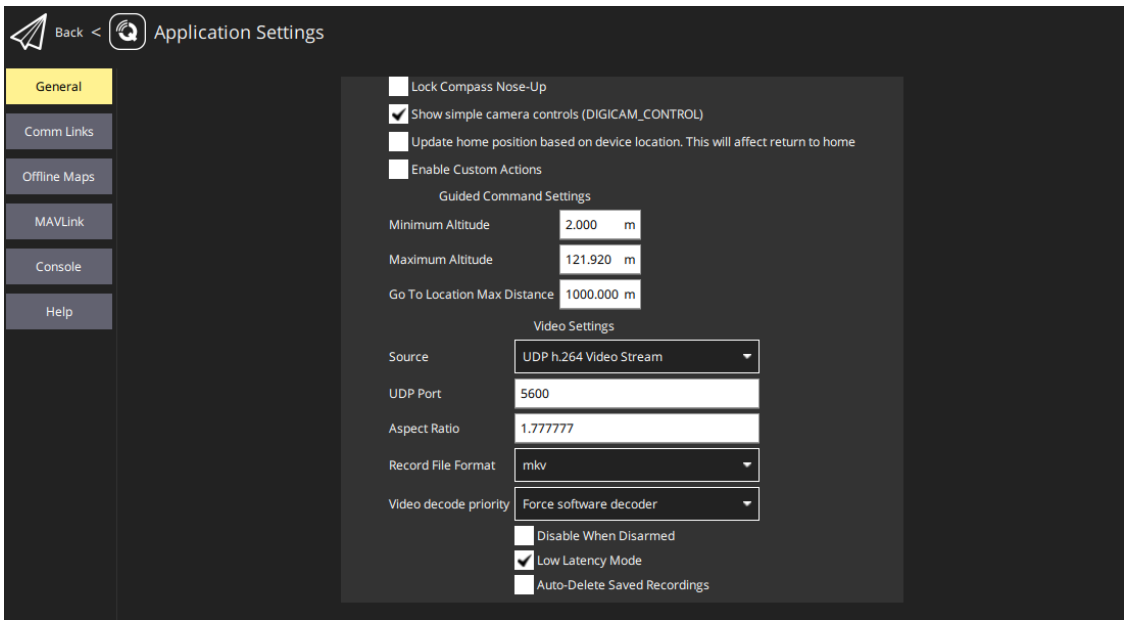
Mavlink 视频流配置与使用指南

配置文件说明 (`mavlink_config.json`)

- **video_ip**：视频流接收方 IP 地址
- **main_port**：主摄像机视频流端口
- **down_port**：下视摄像机视频端口
- **视频格式**：**h264 视频流**

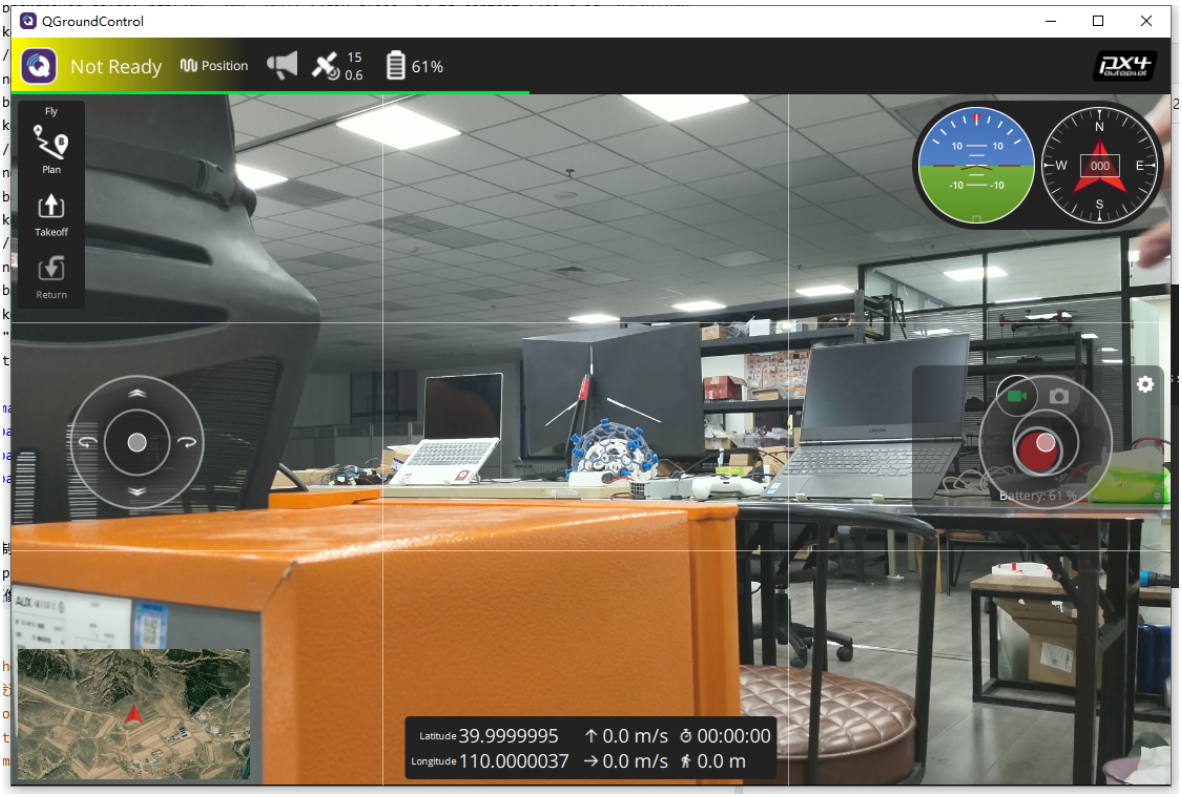
操作步骤

1. 填写psdk/mavlink_config.json文件中的配置信息。
2. **运行 mavlink_demo**
3. **打开 QGC 并连接到无人机**
4. **设置 Video Settings**
   按下图所示配置，以接收主摄像机视频流数据。其中 **UDP Port** 填写 `mavlink_config.json` 中设置的视频流传输端口。



效果预览

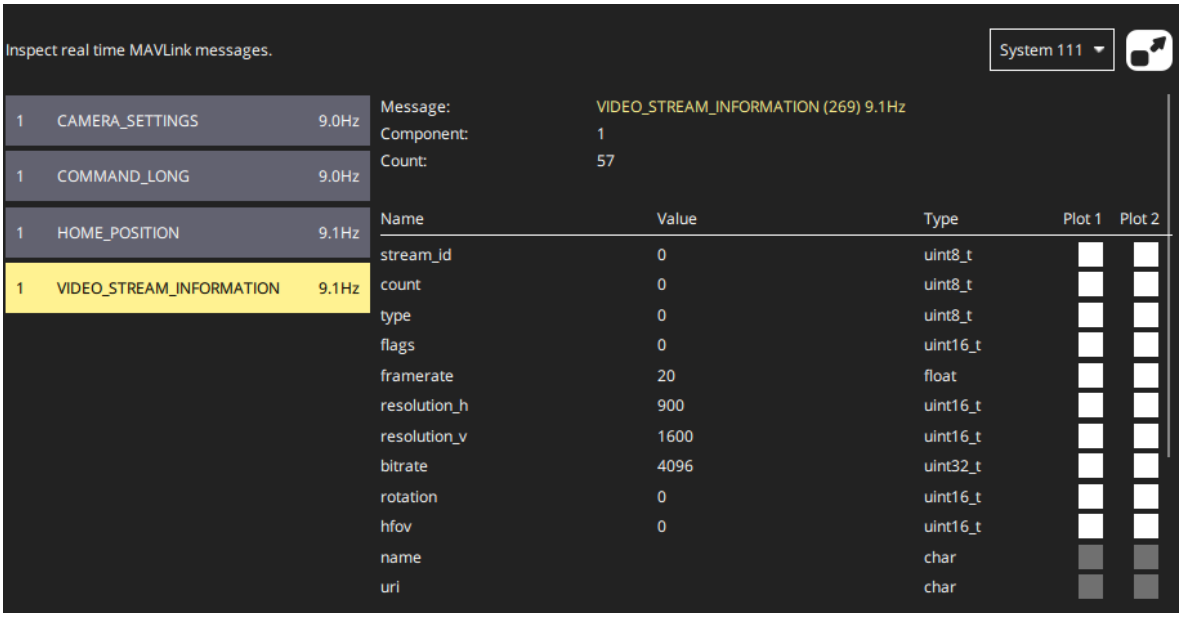
主摄像机视频流效果如下图所示：



mavlink id 269 VIDEO_STREAM_INFORMATION

关于视频流的信息。

返回字段名称	类型	单位	描
bitrate		bits/s	码
framerate		Hz	帧
resolution_v	int16_t	pix	视
resolution_h	int16_t	pix	视



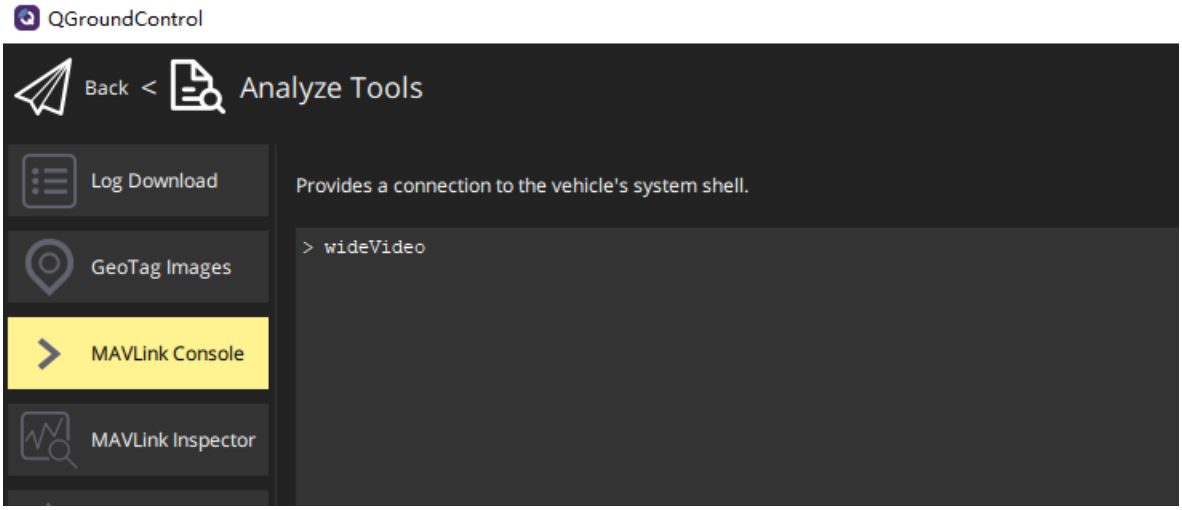
MAV_CMD_VIDEO_START_STREAMING 2502

开始视频流式播放

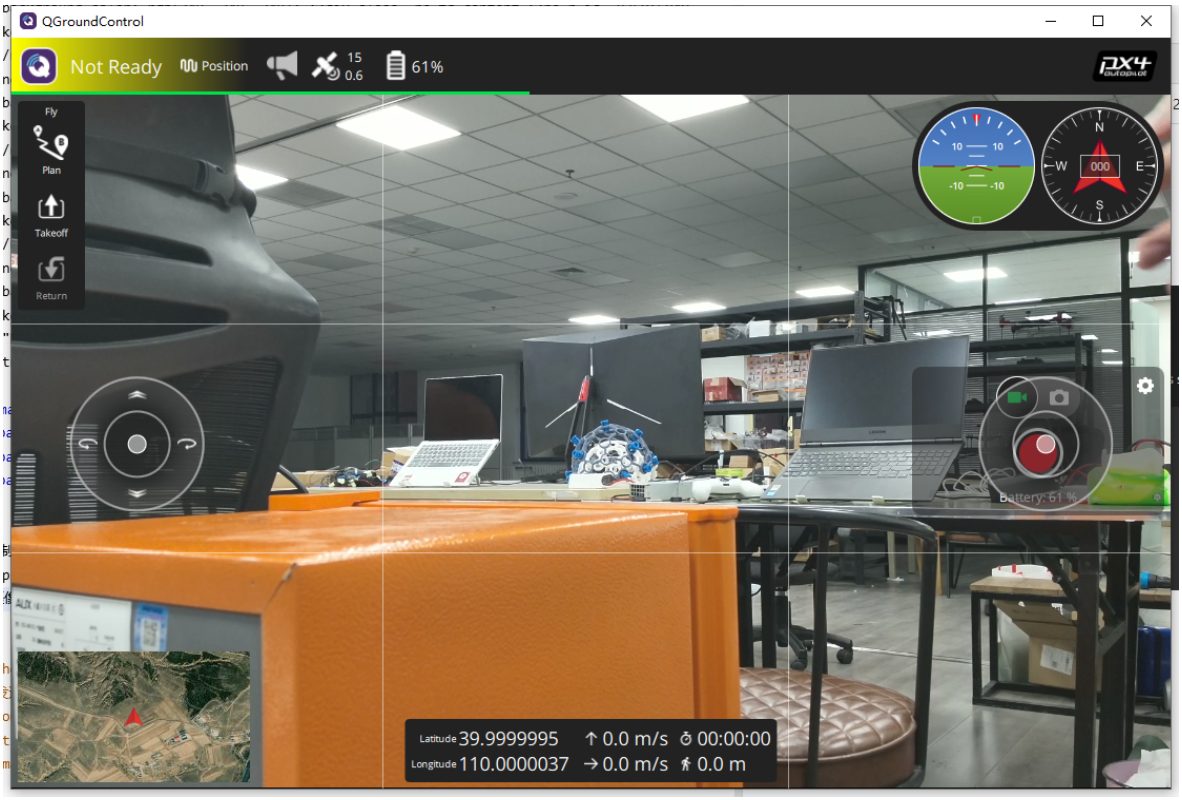
Param (:Label)	描述	值
param1:(Stream ID)	要开启的视频流ID	1：主摄像头，2：下视摄像头，3：红外摄像头

返回消息COMMAND_ACK
返回值 command 2502 执行命令id
返回值 result 0或4 0：执行成功，4：执行失败

于QGC控制台输入wideVideo

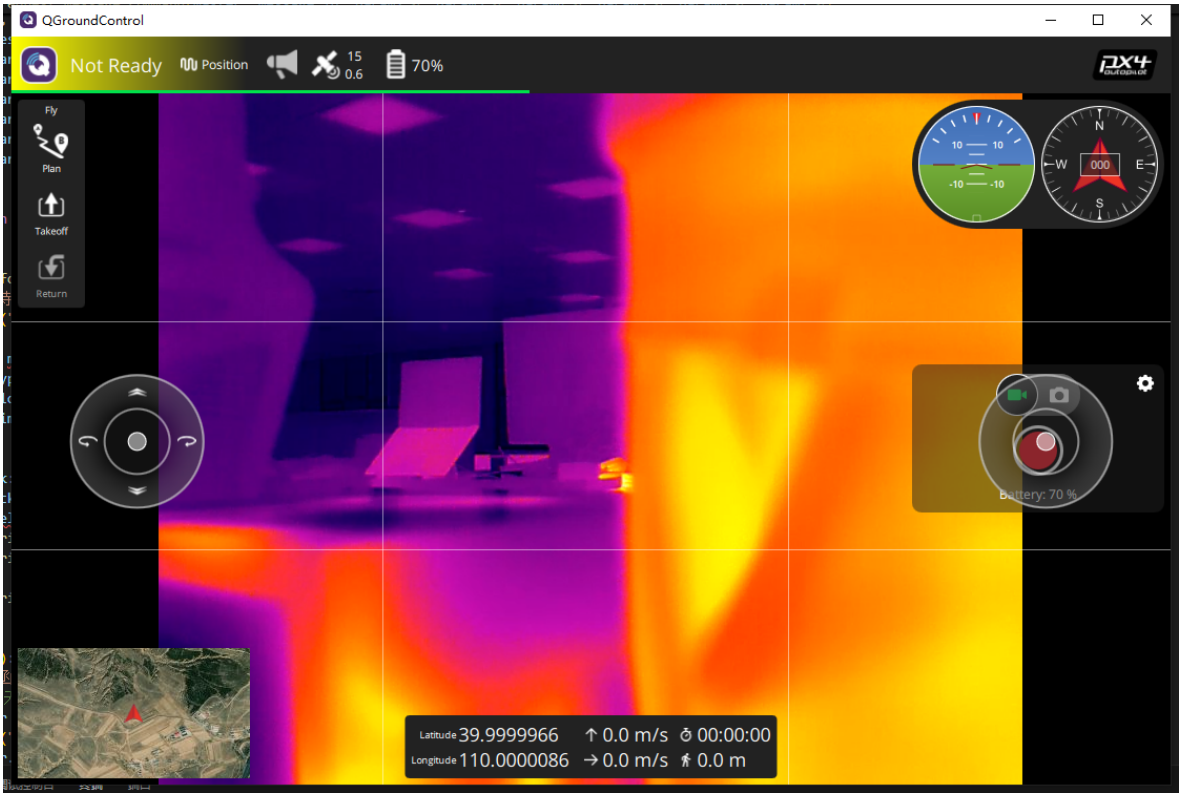


打开主摄像机视频流

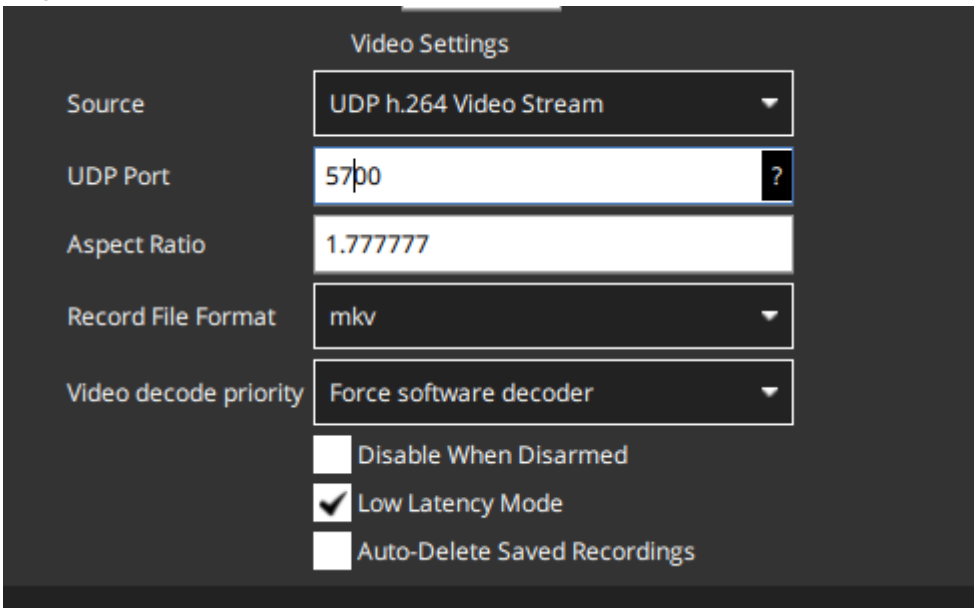


于QGC控制台输入IRVideo

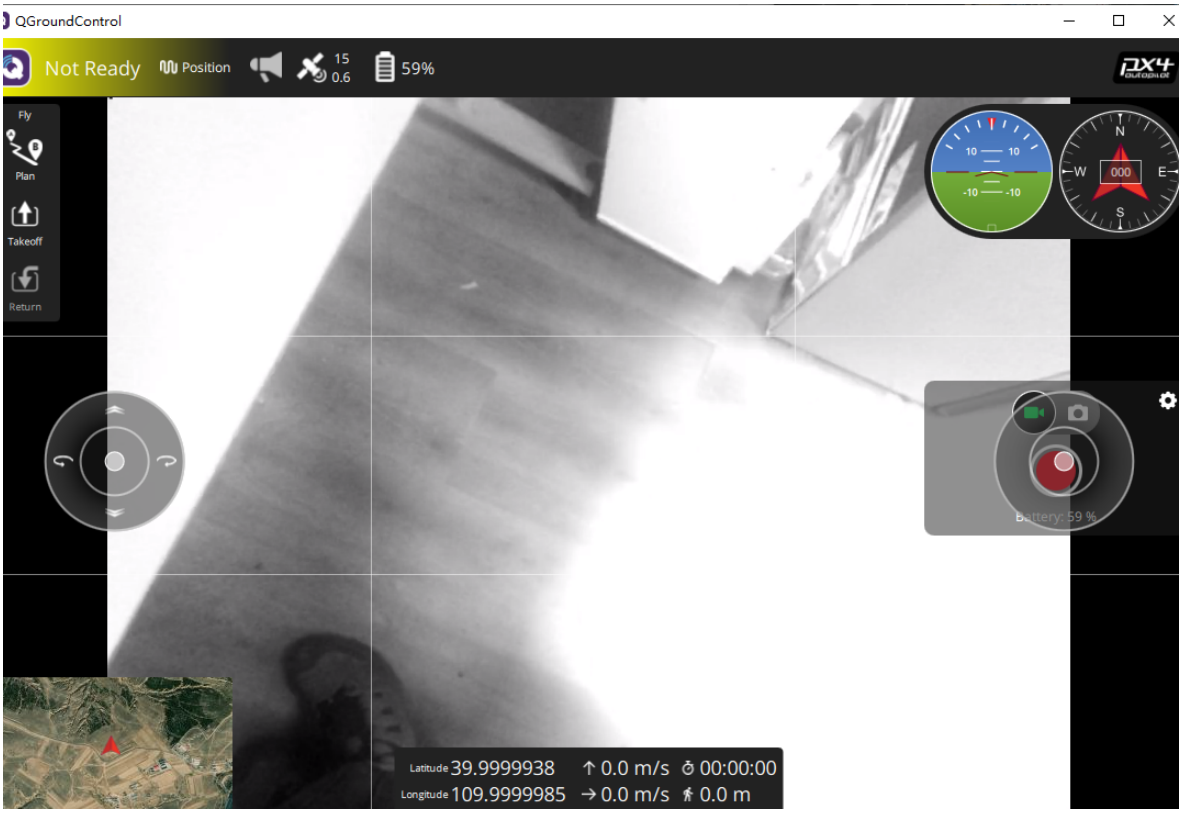
打开主摄像机红外视频流



于QGC控制台输入DownVideo，并将视频流接收端口改为5700



打开下视摄像机视频流



结果（使用QGC查看）

基础相机功能

基础功能

获取相机类型

CAMERA_TYPE

mavlink消息：

接收PARAM_REQUEST_READ
发送PARAM_VALUE

参数：E_DjiCameraType

名称	枚举值	说明
DJI_CAMERA_TYPE_UNKNOWN	0	未知相机类型
DJI_CAMERA_TYPE_Z30	20	Z30 相机
DJI_CAMERA_TYPE_XT2	26	XT2 相机
DJI_CAMERA_TYPE_PSDK	31	基于 PSDK 的第三方相机
DJI_CAMERA_TYPE_XTS	41	XTS 相机
DJI_CAMERA_TYPE_H20	42	H20 相机
DJI_CAMERA_TYPE_H20T	43	H20T 相机
DJI_CAMERA_TYPE_P1	50	P1 相机
DJI_CAMERA_TYPE_L1	51	L1 相机
DJI_CAMERA_TYPE_M30	52	M30 相机
DJI_CAMERA_TYPE_M30T	53	M30T 相机
DJI_CAMERA_TYPE_H20N	61	H20N 相机
DJI_CAMERA_TYPE_M3E	66	M3E 相机
DJI_CAMERA_TYPE_M3T	67	M3T 相机
DJI_CAMERA_TYPE_M3TA	68	M3TA 相机

名称	枚举值	说明
DJI_CAMERA_TYPE_M3D	80	Matrice 3D 相机
DJI_CAMERA_TYPE_M3TD	81	Matrice 3TD 相机
DJI_CAMERA_TYPE_H30	82	H30 相机
DJI_CAMERA_TYPE_H30T	83	H30T 相机
DJI_CAMERA_TYPE_L2	84	L2 相机
DJI_CAMERA_TYPE_M4T	89	M4T 相机
DJI_CAMERA_TYPE_M4TD	90	M4TD 相机
DJI_CAMERA_TYPE_M4D	91	M4D 相机
DJI_CAMERA_TYPE_L3	117	L3 相机
DJI_CAMERA_TYPE_M4E	891	M4E 相机

示例

```
import time
import sys
from pymavlink import mavutil

def connect_to_vehicle():
    """建立 UDP 监听连接（默认监听所有接口的 14550 端口）"""
    connection_string = "udpin:0.0.0.0:14550"
    print(f"Listening for MAVLink on: {connection_string}")
    vehicle = mavutil.mavlink_connection(connection_string)
    print("Waiting for HEARTBEAT (timeout: 10s)...")
    try:
        vehicle.wait_heartbeat(timeout=10)
        print(f"  Heartbeat received from system {vehicle.target_system}, component {vehicle.target_component}")
        return vehicle
    except Exception as e:
        print(f"  Connection failed: {str(e)}")
        print("Ensure your vehicle is sending MAVLink messages to 14550 port")
        print("Example QGroundControl setup: Settings > General > MAVLink > Listening Port = 14550")
```

```
        sys.exit(1)

def format_param_id(param_id):
    """格式化16字节参数ID（符合MAVLink协议要求）"""
    if len(param_id) > 16:
        raise ValueError("param_id cannot exceed 16 characters")

    param_id_bytes = bytearray(16)
    for i in range(min(len(param_id), 16)):
        param_id_bytes[i] = ord(param_id[i])

    # 长度<16时添加NULL终止符
    if len(param_id) < 16:
        param_id_bytes[len(param_id)] = 0

    return param_id_bytes

def send_param_set(vehicle, param_id, param_value, param_type):
    """发送参数设置请求"""
    vehicle.mav.param_set_send(
        vehicle.target_system,
        vehicle.target_component,
        format_param_id(param_id),
        param_value,
        param_type
    )
    print(f"  Sent PARAM_SET for '{param_id}' = {param_value} (typ
e: {param_type})")

def send_param_request_read(vehicle, param_id=None, param_index=-
1):
    """发送参数读取请求"""
    param_id_bytes = format_param_id(param_id) if param_id else byt
earray(16)

    vehicle.mav.param_request_read_send(
        vehicle.target_system,
        vehicle.target_component,
        param_id_bytes,
        param_index
    )

    target = f"index {param_index}" if param_index >= 0 else f"ID
'{param_id}'"
    print(f"  Sent PARAM_REQUEST_READ for {target}")
```

```
def send_param_request_list(vehicle):
    """发送请求所有参数列表"""
    vehicle.mav.param_request_list_send(
        vehicle.target_system,
        vehicle.target_component
    )
    print(f"  Sent PARAM_REQUEST_LIST to system {vehicle.target_sys
tem}, component {vehicle.target_component}")

def receive_param_value(vehicle, timeout=5.0):
    """接收参数值响应"""
    print(f"  Waiting for PARAM_VALUE ({timeout}s timeout)...")
    msg = vehicle.recv_match(type='PARAM_VALUE', blocking=True, tim
eout=timeout)

    if not msg:
        print("  Timeout: No PARAM_VALUE received")
        return None

    # 解析参数ID
    param_id = msg.param_id

    # 解析参数值
    param_value = msg.param_value

    # 参数类型映射
    type_names = {
        0: "INT8", 1: "UINT8", 2: "INT16", 3: "UINT16", 4: "INT32",
        5: "UINT32", 6: "INT64", 7: "UINT64", 8: "FLOAT", 9: "DOUBL
E"
    }
    param_type = type_names.get(msg.param_type, f"UNKNOWN({msg.para
m_type})")

    print("\n  Received PARAM_VALUE:")
    print(f"    ID: {param_id}")
    print(f"    Value: {param_value}")
    print(f"    Type: {param_type}")
    print(f"    Index/Total: {msg.param_index}/{msg.param_count}")

    return {
        'id': param_id,
        'value': param_value,
        'type': param_type,
```

```
        'index': msg.param_index,
        'count': msg.param_count
    }

def receive_all_param_values(vehicle, timeout=10.0):
    """接收所有参数值，直到收到 param_count 条消息后退出循环"""
    received_params = []
    expected_count = None
    print(f"  Waiting for all PARAM_VALUE messages...")

    while True:
        msg = vehicle.recv_match(type='PARAM_VALUE', blocking=True,
timeout=timeout)

        if not msg:
            print(f"  Timeout: Received {len(received_params)} para
meters (expected {expected_count if expected_count else 'unknow
n'})")
            break

        param_id = msg.param_id
        param_value = msg.param_value

        # 参数类型映射
        type_names = {
            0: "INT8", 1: "UINT8", 2: "INT16", 3: "UINT16", 4: "INT
32",
            5: "UINT32", 6: "INT64", 7: "UINT64", 8: "FLOAT", 9: "D
OUBLE"
        }
        param_type = type_names.get(msg.param_type, f"UNKNOWN({msg.
param_type})")

        # 首次收到消息时记录总参数数量
        if expected_count is None:
            expected_count = msg.param_count
            print(f"  Total parameters to receive: {expected_coun
t}")

        print(f"  [{msg.param_index}/{expected_count}] {param_id} =
{param_value} ({param_type})")

        received_params.append({
            'id': param_id,
            'value': param_value,
```

```
            'type': param_type,
            'index': msg.param_index,
            'count': msg.param_count
        })

        # 当已接收数量达到预期总数时退出循环
        if len(received_params) >= expected_count:
            print(f"\n  Successfully received all {expected_count} parameters")
            break

    return received_params

def main():
    print("  Starting MAVLink PARAM_SET Tool (UDP Listening Mode)")
    print("------------------------------------------------------")
    print("This script listens for MAVLink on ALL interfaces:14550")
    print("Ensure your vehicle is configured to send to this port")
    print("------------------------------------------------------\n")

    vehicle = connect_to_vehicle()

    # 操作选择
    print("\nAvailable operations:")
    print("1. Set parameter value")
    print("2. Read parameter value (by ID or index)")
    print("3. Request all parameters (PARAM_REQUEST_LIST)")
    choice = input("\nSelect operation [1-3]: ").strip()

    if choice == '1':
        param_id = input("Enter parameter ID to set: ").strip()
        param_value = float(input("Enter new value: ").strip())

        print("\nParameter types:")
        print("  8 = FLOAT (most common)")
        print("  9 = DOUBLE")
        print("  4 = INT32")
        print("  1 = UINT8")
        param_type = int(input("Enter type number [8]: ").strip() or 8)

        send_param_set(vehicle, param_id, param_value, param_type)
        receive_param_value(vehicle)
```

```
    elif choice == '2':
        # mode = input("Read by [id/index]: ").lower()
        mode = 'id'
        if mode == 'id':
            param_id = input("Enter parameter ID: ").strip()
            send_param_request_read(vehicle, param_id=param_id)
            receive_param_value(vehicle)
        elif mode == 'index':
            param_index = int(input("Enter parameter index: "))
            send_param_request_read(vehicle, param_index=param_inde
x)
            receive_param_value(vehicle)
        else:
            print("Invalid mode. Use 'id' or 'index'")

    elif choice == '3':
        print("\n  Requesting all parameters...")
        send_param_request_list(vehicle)
        all_params = receive_all_param_values(vehicle)
        print(f"\n  Total parameters received: {len(all_params)}")

    else:
        print("Invalid choice. Select 1, 2, or 3.")


if __name__ == "__main__":
    main()
```

示例使用说明

运行后，成功建立连接

```
🚀 Starting MAVLink PARAM_SET Tool (UDP Listening Mode)
--------------------------------------------------
This script listens for MAVLink on ALL interfaces:14550
Ensure your vehicle is configured to send to this port
--------------------------------------------------

Listening for MAVLink on: udpin:0.0.0.0:14550
Waiting for HEARTBEAT (timeout: 10s)...
✅ Heartbeat received from system 3, component 0

Available operations:
1. Set parameter value
2. Read parameter value (by ID or index)
3. Request all parameters (PARAM_REQUEST_LIST)
```

输入“2”选择读取参数

```
Select operation [1-3]: 2
```

输入对应参数“CAMERA_TYPE”

```
Enter parameter ID: CAMERA_TYPE
```

成功返回该参数对应的数值

```
📥 Received PARAM_VALUE:
  🆔 ID: CAMERA_TYPE
  🔢 Value: 90.0
```

获取相机固件版本

FIRMWARE_VERSION

mavlink消息：

接收PARAM_REQUEST_READ
发送PARAM_VALUE

参数：T_DjiCameraManagerFirmwareVersion

字段名	类型	说明
firmware_version	foat	固件版本号

示例

```
import time
import sys
from pymavlink import mavutil

def connect_to_vehicle():
    """建立 UDP 监听连接（默认监听所有接口的 14550 端口）"""
    connection_string = "udpin:0.0.0.0:14550"
    print(f"Listening for MAVLink on: {connection_string}")
    vehicle = mavutil.mavlink_connection(connection_string)
    print("Waiting for HEARTBEAT (timeout: 10s)...")
    try:
        vehicle.wait_heartbeat(timeout=10)
        print(f"  Heartbeat received from system {vehicle.target_sy
stem}, component {vehicle.target_component}")
        return vehicle
    except Exception as e:
        print(f"  Connection failed: {str(e)}")
        print("Ensure your vehicle is sending MAVLink messages to 1
4550 port")
        print("Example QGroundControl setup: Settings > General > M
AVLink > Listening Port = 14550")
        sys.exit(1)

def format_param_id(param_id):
    """格式化16字节参数ID（符合MAVLink协议要求）"""
    if len(param_id) > 16:
        raise ValueError("param_id cannot exceed 16 characters")

    param_id_bytes = bytearray(16)
    for i in range(min(len(param_id), 16)):
        param_id_bytes[i] = ord(param_id[i])

    # 长度<16时添加NULL终止符
    if len(param_id) < 16:
        param_id_bytes[len(param_id)] = 0

    return param_id_bytes

def send_param_set(vehicle, param_id, param_value, param_type):
    """发送参数设置请求"""
    vehicle.mav.param_set_send(
        vehicle.target_system,
        vehicle.target_component,
```

```
        format_param_id(param_id),
        param_value,
        param_type
    )
    print(f"  Sent PARAM_SET for '{param_id}' = {param_value} (typ
e: {param_type})")

def send_param_request_read(vehicle, param_id=None, param_index=-
1):
    """发送参数读取请求"""
    param_id_bytes = format_param_id(param_id) if param_id else byt
earray(16)

    vehicle.mav.param_request_read_send(
        vehicle.target_system,
        vehicle.target_component,
        param_id_bytes,
        param_index
    )

    target = f"index {param_index}" if param_index >= 0 else f"ID
'{param_id}'"
    print(f"  Sent PARAM_REQUEST_READ for {target}")

def send_param_request_list(vehicle):
    """发送请求所有参数列表"""
    vehicle.mav.param_request_list_send(
        vehicle.target_system,
        vehicle.target_component
    )
    print(f"  Sent PARAM_REQUEST_LIST to system {vehicle.target_sys
tem}, component {vehicle.target_component}")

def receive_param_value(vehicle, timeout=5.0):
    """接收参数值响应"""
    print(f"  Waiting for PARAM_VALUE ({timeout}s timeout)...")
    msg = vehicle.recv_match(type='PARAM_VALUE', blocking=True, tim
eout=timeout)

    if not msg:
        print("  Timeout: No PARAM_VALUE received")
        return None

    # 解析参数ID
    param_id = msg.param_id
```

```
    # 解析参数值
    param_value = msg.param_value

    # 参数类型映射
    type_names = {
        0: "INT8", 1: "UINT8", 2: "INT16", 3: "UINT16", 4: "INT32",
        5: "UINT32", 6: "INT64", 7: "UINT64", 8: "FLOAT", 9: "DOUBL
E"
    }
    param_type = type_names.get(msg.param_type, f"UNKNOWN({msg.para
m_type})")

    print("\n  Received PARAM_VALUE:")
    print(f"    ID: {param_id}")
    print(f"    Value: {param_value}")
    print(f"    Type: {param_type}")
    print(f"    Index/Total: {msg.param_index}/{msg.param_count}")

    return {
        'id': param_id,
        'value': param_value,
        'type': param_type,
        'index': msg.param_index,
        'count': msg.param_count
    }

def receive_all_param_values(vehicle, timeout=10.0):
    """接收所有参数值，直到收到 param_count 条消息后退出循环"""
    received_params = []
    expected_count = None
    print(f"  Waiting for all PARAM_VALUE messages...")

    while True:
        msg = vehicle.recv_match(type='PARAM_VALUE', blocking=True,
timeout=timeout)

        if not msg:
            print(f"  Timeout: Received {len(received_params)} para
meters (expected {expected_count if expected_count else 'unknow
n'})")
            break

        param_id = msg.param_id
        param_value = msg.param_value
```

```
        # 参数类型映射
        type_names = {
            0: "INT8", 1: "UINT8", 2: "INT16", 3: "UINT16", 4: "INT
32",
            5: "UINT32", 6: "INT64", 7: "UINT64", 8: "FLOAT", 9: "D
OUBLE"
        }
        param_type = type_names.get(msg.param_type, f"UNKNOWN({msg.
param_type})")

        # 首次收到消息时记录总参数数量
        if expected_count is None:
            expected_count = msg.param_count
            print(f"  Total parameters to receive: {expected_coun
t}")

        print(f"  [{msg.param_index}/{expected_count}] {param_id} =
{param_value} ({param_type})")

        received_params.append({
            'id': param_id,
            'value': param_value,
            'type': param_type,
            'index': msg.param_index,
            'count': msg.param_count
        })

        # 当已接收数量达到预期总数时退出循环
        if len(received_params) >= expected_count:
            print(f"\n  Successfully received all {expected_count}
parameters")
            break

    return received_params

def main():
    print("  Starting MAVLink PARAM_SET Tool (UDP Listening Mode)")
    print("------------------------------------------------------")
    print("This script listens for MAVLink on ALL interfaces:1455
0")
    print("Ensure your vehicle is configured to send to this port")
    print("------------------------------------------------------
\n")
```

```
    vehicle = connect_to_vehicle()

    # 操作选择
    print("\nAvailable operations:")
    print("1. Set parameter value")
    print("2. Read parameter value (by ID or index)")
    print("3. Request all parameters (PARAM_REQUEST_LIST)")
    choice = input("\nSelect operation [1-3]: ").strip()

    if choice == '1':
        param_id = input("Enter parameter ID to set: ").strip()
        param_value = float(input("Enter new value: ").strip())

        print("\nParameter types:")
        print("  8 = FLOAT (most common)")
        print("  9 = DOUBLE")
        print("  4 = INT32")
        print("  1 = UINT8")
        param_type = int(input("Enter type number [8]: ").strip() or 8)

        send_param_set(vehicle, param_id, param_value, param_type)
        receive_param_value(vehicle)

    elif choice == '2':
        # mode = input("Read by [id/index]: ").lower()
        mode = 'id'
        if mode == 'id':
            param_id = input("Enter parameter ID: ").strip()
            send_param_request_read(vehicle, param_id=param_id)
            receive_param_value(vehicle)
        elif mode == 'index':
            param_index = int(input("Enter parameter index: "))
            send_param_request_read(vehicle, param_index=param_index)
            receive_param_value(vehicle)
        else:
            print("Invalid mode. Use 'id' or 'index'")

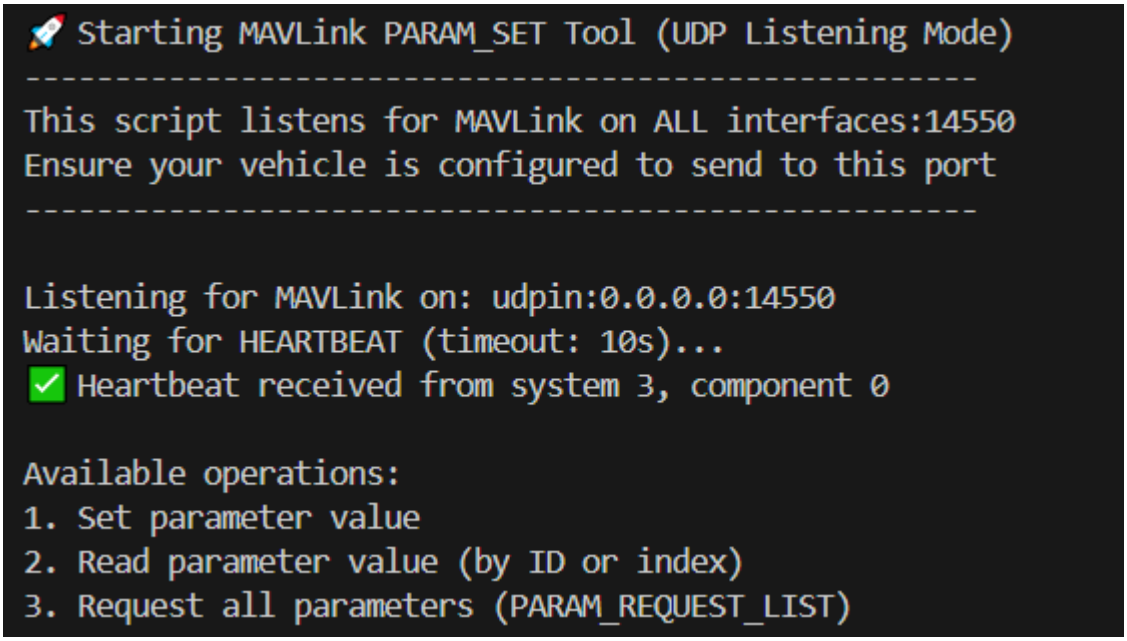
    elif choice == '3':
        print("\n  Requesting all parameters...")
        send_param_request_list(vehicle)
        all_params = receive_all_param_values(vehicle)
        print(f"\n  Total parameters received: {len(all_params)}")
```

```
    else:
        print("Invalid choice. Select 1, 2, or 3.")

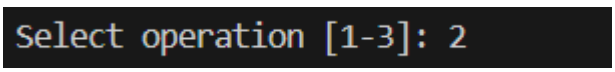

if __name__ == "__main__":
    main()
```

示例使用说明

运行后，成功建立连接



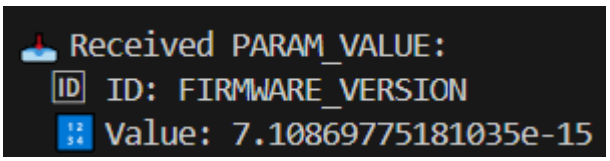
输入“2”选择读取参数



输入对应参数“FIRMWARE_VERSION”



成功返回该参数对应的数值

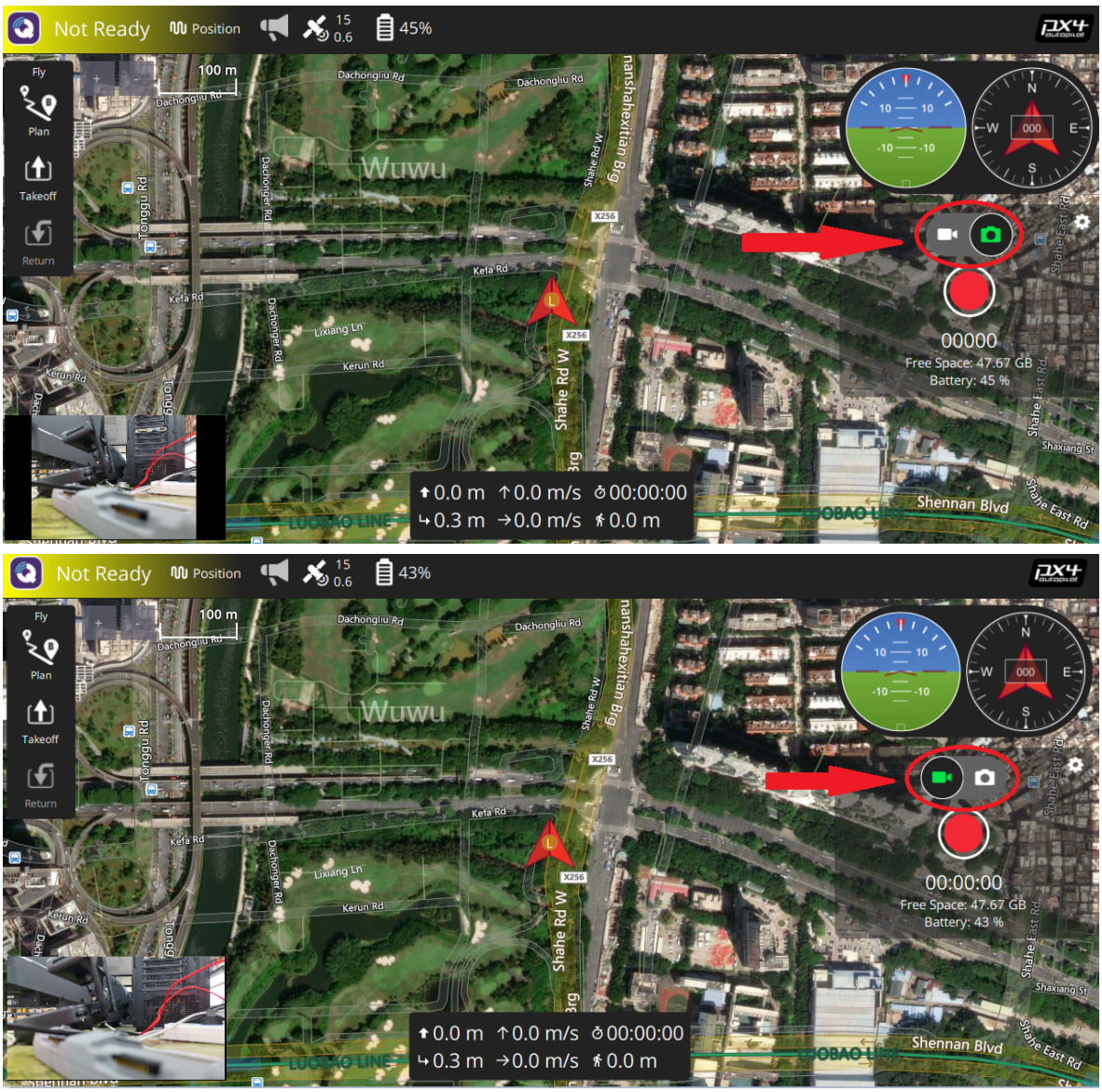


切换相机工作模式

CAMERA_MODE

设置

通过QGC图形界面切换相机工作模式



获取

mavlink消息

接收PARAM_REQUEST_READ
发送PARAM_VALUE

参数：

名称	枚举值	说明
DJI_CAMERA_MANAGER_SHOOT_PHOTO_MODE_SINGLE	0x01	单拍模式（Photo ）
DJI_CAMERA_MANAGER_SHOOT_PHOTO_MODE_HDR	0x02	HDR 拍照

名称	枚举值	说明
DJI_CAMERA_MANAGER_SHOOT_PHOTO_MODE_BURST	0x04	连拍模式（
DJI_CAMERA_MANAGER_SHOOT_PHOTO_MODE_AEB	0x05	AEB 自动曝光包
DJI_CAMERA_MANAGER_SHOOT_PHOTO_MODE_INTERVAL	0x06	定时拍照模 （Interval
DJI_CAMERA_MANAGER_SHOOT_PHOTO_MODE_UNKNOWN	0xFF	未知拍照模

示例

```
import time
import sys
from pymavlink import mavutil

def connect_to_vehicle():
    """建立 UDP 监听连接（默认监听所有接口的 14550 端口）"""
    connection_string = "udpin:0.0.0.0:14550"
    print(f"Listening for MAVLink on: {connection_string}")
    vehicle = mavutil.mavlink_connection(connection_string)
    print("Waiting for HEARTBEAT (timeout: 10s)...")
    try:
        vehicle.wait_heartbeat(timeout=10)
        print(f"  Heartbeat received from system {vehicle.target_sy
stem}, component {vehicle.target_component}")
        return vehicle
    except Exception as e:
        print(f"  Connection failed: {str(e)}")
        print("Ensure your vehicle is sending MAVLink messages to 1
4550 port")
        print("Example QGroundControl setup: Settings > General > M
AVLink > Listening Port = 14550")
        sys.exit(1)

def format_param_id(param_id):
    """格式化16字节参数ID（符合MAVLink协议要求）"""
    if len(param_id) > 16:
        raise ValueError("param_id cannot exceed 16 characters")

    param_id_bytes = bytearray(16)
    for i in range(min(len(param_id), 16)):
```

```
            param_id_bytes[i] = ord(param_id[i])

    # 长度<16时添加NULL终止符
    if len(param_id) < 16:
            param_id_bytes[len(param_id)] = 0

    return param_id_bytes

def send_param_set(vehicle, param_id, param_value, param_type):
    """发送参数设置请求"""
    vehicle.mav.param_set_send(
            vehicle.target_system,
            vehicle.target_component,
            format_param_id(param_id),
            param_value,
            param_type
    )
    print(f"  Sent PARAM_SET for '{param_id}' = {param_value} (typ
e: {param_type})")

def send_param_request_read(vehicle, param_id=None, param_index=-
1):
    """发送参数读取请求"""
    param_id_bytes = format_param_id(param_id) if param_id else byt
earray(16)

    vehicle.mav.param_request_read_send(
            vehicle.target_system,
            vehicle.target_component,
            param_id_bytes,
            param_index
    )

    target = f"index {param_index}" if param_index >= 0 else f"ID
'{param_id}'"
    print(f"  Sent PARAM_REQUEST_READ for {target}")

def send_param_request_list(vehicle):
    """发送请求所有参数列表"""
    vehicle.mav.param_request_list_send(
            vehicle.target_system,
            vehicle.target_component
    )
    print(f"  Sent PARAM_REQUEST_LIST to system {vehicle.target_sys
tem}, component {vehicle.target_component}")
```

```
def receive_param_value(vehicle, timeout=5.0):
    """接收参数值响应"""
    print(f"  Waiting for PARAM_VALUE ({timeout}s timeout)...")
    msg = vehicle.recv_match(type='PARAM_VALUE', blocking=True, tim
eout=timeout)

    if not msg:
        print("  Timeout: No PARAM_VALUE received")
        return None

    # 解析参数ID
    param_id = msg.param_id

    # 解析参数值
    param_value = msg.param_value

    # 参数类型映射
    type_names = {
        0: "INT8", 1: "UINT8", 2: "INT16", 3: "UINT16", 4: "INT32",
        5: "UINT32", 6: "INT64", 7: "UINT64", 8: "FLOAT", 9: "DOUBL
E"
    }
    param_type = type_names.get(msg.param_type, f"UNKNOWN({msg.para
m_type})")

    print("\n  Received PARAM_VALUE:")
    print(f"    ID: {param_id}")
    print(f"    Value: {param_value}")
    print(f"    Type: {param_type}")
    print(f"    Index/Total: {msg.param_index}/{msg.param_count}")

    return {
        'id': param_id,
        'value': param_value,
        'type': param_type,
        'index': msg.param_index,
        'count': msg.param_count
    }

def receive_all_param_values(vehicle, timeout=10.0):
    """接收所有参数值，直到收到 param_count 条消息后退出循环"""
    received_params = []
    expected_count = None
    print(f"  Waiting for all PARAM_VALUE messages...")
```

```
    while True:
        msg = vehicle.recv_match(type='PARAM_VALUE', blocking=True, timeout=timeout)

        if not msg:
            print(f"  Timeout: Received {len(received_params)} parameters (expected {expected_count if expected_count else 'unknown'})")
            break

        param_id = msg.param_id
        param_value = msg.param_value

        # 参数类型映射
        type_names = {
            0: "INT8", 1: "UINT8", 2: "INT16", 3: "UINT16", 4: "INT32",
            5: "UINT32", 6: "INT64", 7: "UINT64", 8: "FLOAT", 9: "DOUBLE"
        }
        param_type = type_names.get(msg.param_type, f"UNKNOWN({msg.param_type})")

        # 首次收到消息时记录总参数数量
        if expected_count is None:
            expected_count = msg.param_count
            print(f"  Total parameters to receive: {expected_count}")

        print(f"  [{msg.param_index}/{expected_count}] {param_id} = {param_value} ({param_type})")

        received_params.append({
            'id': param_id,
            'value': param_value,
            'type': param_type,
            'index': msg.param_index,
            'count': msg.param_count
        })

        # 当已接收数量达到预期总数时退出循环
        if len(received_params) >= expected_count:
            print(f"\n  Successfully received all {expected_count} parameters")
```

```
            break

    return received_params

def main():
    print("  Starting MAVLink PARAM_SET Tool (UDP Listening Mode)")
    print("------------------------------------------------------")
    print("This script listens for MAVLink on ALL interfaces:1455
0")
    print("Ensure your vehicle is configured to send to this port")
    print("------------------------------------------------------
\n")

    vehicle = connect_to_vehicle()

    # 操作选择
    print("\nAvailable operations:")
    print("1. Set parameter value")
    print("2. Read parameter value (by ID or index)")
    print("3. Request all parameters (PARAM_REQUEST_LIST)")
    choice = input("\nSelect operation [1-3]: ").strip()

    if choice == '1':
        param_id = input("Enter parameter ID to set: ").strip()
        param_value = float(input("Enter new value: ").strip())

        print("\nParameter types:")
        print("  8 = FLOAT (most common)")
        print("  9 = DOUBLE")
        print("  4 = INT32")
        print("  1 = UINT8")
        param_type = int(input("Enter type number [8]: ").strip() o
r 8)

        send_param_set(vehicle, param_id, param_value, param_type)
        receive_param_value(vehicle)

    elif choice == '2':
        # mode = input("Read by [id/index]: ").lower()
        mode = 'id'
        if mode == 'id':
            param_id = input("Enter parameter ID: ").strip()
            send_param_request_read(vehicle, param_id=param_id)
            receive_param_value(vehicle)
        elif mode == 'index':
```

```
            param_index = int(input("Enter parameter index: "))
            send_param_request_read(vehicle, param_index=param_inde
x)
            receive_param_value(vehicle)
        else:
            print("Invalid mode. Use 'id' or 'index'")

    elif choice == '3':
        print("\n  Requesting all parameters...")
        send_param_request_list(vehicle)
        all_params = receive_all_param_values(vehicle)
        print(f"\n  Total parameters received: {len(all_params)}")

    else:
        print("Invalid choice. Select 1, 2, or 3.")


if __name__ == "__main__":
    main()
```

示例使用说明

运行后，成功建立连接

```
🚀 Starting MAVLink PARAM_SET Tool (UDP Listening Mode)
------------------------------------------------------
This script listens for MAVLink on ALL interfaces:14550
Ensure your vehicle is configured to send to this port
------------------------------------------------------

Listening for MAVLink on: udpin:0.0.0.0:14550
Waiting for HEARTBEAT (timeout: 10s)...
✅ Heartbeat received from system 3, component 0

Available operations:
1. Set parameter value
2. Read parameter value (by ID or index)
3. Request all parameters (PARAM_REQUEST_LIST)
```

输入“2”选择读取参数

```
Select operation [1-3]: 2
```

输入对应参数“CAMERA_MODE”

```
Enter parameter ID: CAMERA_MODE
```

成功返回该参数对应的数值

```
📤 Sent PARAM_REQUEST_READ for ID 'CAMERA_MODE'
⏳ Waiting for PARAM_VALUE (5.0s timeout)...

📥 Received PARAM_VALUE:
  🆔 ID: CAMERA_MODE
  🔢 Value: 1.0
```

获取间隔拍摄的剩余时间

INT_SHOOT_REM

mavlink消息：

接收PARAM_REQUEST_READ
发送PARAM_VALUE

参数：

字段名	数据类型	含义	单位
remainTime	uint8_t	剩余时间	s

示例

```
import time
import sys
from pymavlink import mavutil

def connect_to_vehicle():
    """建立 UDP 监听连接（默认监听所有接口的 14550 端口）"""
    connection_string = "udpin:0.0.0.0:14550"
    print(f"Listening for MAVLink on: {connection_string}")
    vehicle = mavutil.mavlink_connection(connection_string)
    print("Waiting for HEARTBEAT (timeout: 10s)...")
    try:
        vehicle.wait_heartbeat(timeout=10)
        print(f"  Heartbeat received from system {vehicle.target_system}, component {vehicle.target_component}")
        return vehicle
    except Exception as e:
```

```
        print(f"  Connection failed: {str(e)}")
        print("Ensure your vehicle is sending MAVLink messages to 1
4550 port")
        print("Example QGroundControl setup: Settings > General > M
AVLink > Listening Port = 14550")
        sys.exit(1)

def format_param_id(param_id):
    """格式化16字节参数ID（符合MAVLink协议要求）"""
    if len(param_id) > 16:
        raise ValueError("param_id cannot exceed 16 characters")

    param_id_bytes = bytearray(16)
    for i in range(min(len(param_id), 16)):
        param_id_bytes[i] = ord(param_id[i])

    # 长度<16时添加NULL终止符
    if len(param_id) < 16:
        param_id_bytes[len(param_id)] = 0

    return param_id_bytes

def send_param_set(vehicle, param_id, param_value, param_type):
    """发送参数设置请求"""
    vehicle.mav.param_set_send(
        vehicle.target_system,
        vehicle.target_component,
        format_param_id(param_id),
        param_value,
        param_type
    )
    print(f"  Sent PARAM_SET for '{param_id}' = {param_value} (typ
e: {param_type})")

def send_param_request_read(vehicle, param_id=None, param_index=-
1):
    """发送参数读取请求"""
    param_id_bytes = format_param_id(param_id) if param_id else byt
earray(16)

    vehicle.mav.param_request_read_send(
        vehicle.target_system,
        vehicle.target_component,
        param_id_bytes,
        param_index
```

```
    )

    target = f"index {param_index}" if param_index >= 0 else f"ID
'{param_id}'"
    print(f"  Sent PARAM_REQUEST_READ for {target}")

def send_param_request_list(vehicle):
    """发送请求所有参数列表"""
    vehicle.mav.param_request_list_send(
        vehicle.target_system,
        vehicle.target_component
    )
    print(f"  Sent PARAM_REQUEST_LIST to system {vehicle.target_sys
tem}, component {vehicle.target_component}")

def receive_param_value(vehicle, timeout=5.0):
    """接收参数值响应"""
    print(f"  Waiting for PARAM_VALUE ({timeout}s timeout)...")
    msg = vehicle.recv_match(type='PARAM_VALUE', blocking=True, tim
eout=timeout)

    if not msg:
        print("  Timeout: No PARAM_VALUE received")
        return None

    # 解析参数ID
    param_id = msg.param_id

    # 解析参数值
    param_value = msg.param_value

    # 参数类型映射
    type_names = {
        0: "INT8", 1: "UINT8", 2: "INT16", 3: "UINT16", 4: "INT32",
        5: "UINT32", 6: "INT64", 7: "UINT64", 8: "FLOAT", 9: "DOUBL
E"
    }
    param_type = type_names.get(msg.param_type, f"UNKNOWN({msg.para
m_type})")

    print("\n  Received PARAM_VALUE:")
    print(f"    ID: {param_id}")
    print(f"    Value: {param_value}")
    print(f"    Type: {param_type}")
    print(f"    Index/Total: {msg.param_index}/{msg.param_count}")
```

```
    return {
        'id': param_id,
        'value': param_value,
        'type': param_type,
        'index': msg.param_index,
        'count': msg.param_count
    }

def receive_all_param_values(vehicle, timeout=10.0):
    """接收所有参数值，直到收到 param_count 条消息后退出循环"""
    received_params = []
    expected_count = None
    print(f"  Waiting for all PARAM_VALUE messages...")

    while True:
        msg = vehicle.recv_match(type='PARAM_VALUE', blocking=True,
timeout=timeout)

        if not msg:
            print(f"  Timeout: Received {len(received_params)} para
meters (expected {expected_count if expected_count else 'unknow
n'})")
            break

        param_id = msg.param_id
        param_value = msg.param_value

        # 参数类型映射
        type_names = {
            0: "INT8", 1: "UINT8", 2: "INT16", 3: "UINT16", 4: "INT
32",
            5: "UINT32", 6: "INT64", 7: "UINT64", 8: "FLOAT", 9: "D
OUBLE"
        }
        param_type = type_names.get(msg.param_type, f"UNKNOWN({msg.
param_type})")

        # 首次收到消息时记录总参数数量
        if expected_count is None:
            expected_count = msg.param_count
            print(f"  Total parameters to receive: {expected_coun
t}")

        print(f"  [{msg.param_index}/{expected_count}] {param_id} =
```

```
{param_value} ({param_type})")

        received_params.append({
            'id': param_id,
            'value': param_value,
            'type': param_type,
            'index': msg.param_index,
            'count': msg.param_count
        })

        # 当已接收数量达到预期总数时退出循环
        if len(received_params) >= expected_count:
            print(f"\n  Successfully received all {expected_count}
parameters")
            break

    return received_params

def main():
    print("  Starting MAVLink PARAM_SET Tool (UDP Listening Mode)")
    print("------------------------------------------------------")
    print("This script listens for MAVLink on ALL interfaces:1455
0")
    print("Ensure your vehicle is configured to send to this port")
    print("------------------------------------------------------
\n")

    vehicle = connect_to_vehicle()

    # 操作选择
    print("\nAvailable operations:")
    print("1. Set parameter value")
    print("2. Read parameter value (by ID or index)")
    print("3. Request all parameters (PARAM_REQUEST_LIST)")
    choice = input("\nSelect operation [1-3]: ").strip()

    if choice == '1':
        param_id = input("Enter parameter ID to set: ").strip()
        param_value = float(input("Enter new value: ").strip())

        print("\nParameter types:")
        print("  8 = FLOAT (most common)")
        print("  9 = DOUBLE")
        print("  4 = INT32")
        print("  1 = UINT8")
```

```
        param_type = int(input("Enter type number [8]: ").strip() o
r 8)

        send_param_set(vehicle, param_id, param_value, param_type)
        receive_param_value(vehicle)

    elif choice == '2':
        # mode = input("Read by [id/index]: ").lower()
        mode = 'id'
        if mode == 'id':
            param_id = input("Enter parameter ID: ").strip()
            send_param_request_read(vehicle, param_id=param_id)
            receive_param_value(vehicle)
        elif mode == 'index':
            param_index = int(input("Enter parameter index: "))
            send_param_request_read(vehicle, param_index=param_inde
x)
            receive_param_value(vehicle)
        else:
            print("Invalid mode. Use 'id' or 'index'")

    elif choice == '3':
        print("\n  Requesting all parameters...")
        send_param_request_list(vehicle)
        all_params = receive_all_param_values(vehicle)
        print(f"\n  Total parameters received: {len(all_params)}")

    else:
        print("Invalid choice. Select 1, 2, or 3.")


if __name__ == "__main__":
    main()
```

示例使用说明

运行后，成功建立连接

```
🚀 Starting MAVLink PARAM_SET Tool (UDP Listening Mode)
--------------------------------------------------
This script listens for MAVLink on ALL interfaces:14550
Ensure your vehicle is configured to send to this port
--------------------------------------------------

Listening for MAVLink on: udpin:0.0.0.0:14550
Waiting for HEARTBEAT (timeout: 10s)...
✅ Heartbeat received from system 3, component 0

Available operations:
1. Set parameter value
2. Read parameter value (by ID or index)
3. Request all parameters (PARAM_REQUEST_LIST)
```

输入“2”选择读取参数

```
Select operation [1-3]: 2
```

输入对应参数“INT_SHOOT_REM”

```
Enter parameter ID: INT_SHOOT_REM
```

成功返回该参数对应的数值

```
📥 Received PARAM_VALUE:
   🆔 ID: INT_SHOOT_REM
   🔢 Value: 0.0
```

对焦功能

相机对焦模式

FOCUS_MODE

设置

mavlink消息：

接收PARAM_SET
发送PARAM_VALUE

获取

mavlink消息

接收PARAM_REQUEST_READ
发送PARAM_VALUE

参数：

名称	枚举值	说明
DJI_CAMERA_MANAGER_FOCUS_MODE_MANUAL	0	手动对焦模式
DJI_CAMERA_MANAGER_FOCUS_MODE_AUTO	1	自动对焦模式
DJI_CAMERA_MANAGER_FOCUS_MODE_AFC	2	连续自动对焦模式（Continuous AF）
DJI_CAMERA_MANAGER_FOCUS_MODE_UNKNOWN	0xFF	未知对焦模式

示例

```
import time
import sys
from pymavlink import mavutil

def connect_to_vehicle():
    """建立 UDP 监听连接（默认监听所有接口的 14550 端口）"""
```

```
    connection_string = "udpin:0.0.0.0:14550"
    print(f"Listening for MAVLink on: {connection_string}")
    vehicle = mavutil.mavlink_connection(connection_string)
    print("Waiting for HEARTBEAT (timeout: 10s)...")
    try:
        vehicle.wait_heartbeat(timeout=10)
        print(f"  Heartbeat received from system {vehicle.target_sy
stem}, component {vehicle.target_component}")
        return vehicle
    except Exception as e:
        print(f"  Connection failed: {str(e)}")
        print("Ensure your vehicle is sending MAVLink messages to 1
4550 port")
        print("Example QGroundControl setup: Settings > General > M
AVLink > Listening Port = 14550")
        sys.exit(1)

def format_param_id(param_id):
    """格式化16字节参数ID（符合MAVLink协议要求）"""
    if len(param_id) > 16:
        raise ValueError("param_id cannot exceed 16 characters")

    param_id_bytes = bytearray(16)
    for i in range(min(len(param_id), 16)):
        param_id_bytes[i] = ord(param_id[i])

    # 长度<16时添加NULL终止符
    if len(param_id) < 16:
        param_id_bytes[len(param_id)] = 0

    return param_id_bytes

def send_param_set(vehicle, param_id, param_value, param_type):
    """发送参数设置请求"""
    vehicle.mav.param_set_send(
        vehicle.target_system,
        vehicle.target_component,
        format_param_id(param_id),
        param_value,
        param_type
    )
    print(f"  Sent PARAM_SET for '{param_id}' = {param_value} (typ
e: {param_type})")

def send_param_request_read(vehicle, param_id=None, param_index=-
```

```
1):
    """发送参数读取请求"""
    param_id_bytes = format_param_id(param_id) if param_id else byt
earray(16)

    vehicle.mav.param_request_read_send(
        vehicle.target_system,
        vehicle.target_component,
        param_id_bytes,
        param_index
    )

    target = f"index {param_index}" if param_index >= 0 else f"ID
'{param_id}'"
    print(f"  Sent PARAM_REQUEST_READ for {target}")

def send_param_request_list(vehicle):
    """发送请求所有参数列表"""
    vehicle.mav.param_request_list_send(
        vehicle.target_system,
        vehicle.target_component
    )
    print(f"  Sent PARAM_REQUEST_LIST to system {vehicle.target_sys
tem}, component {vehicle.target_component}")

def receive_param_value(vehicle, timeout=5.0):
    """接收参数值响应"""
    print(f"  Waiting for PARAM_VALUE ({timeout}s timeout)...")
    msg = vehicle.recv_match(type='PARAM_VALUE', blocking=True, tim
eout=timeout)

    if not msg:
        print("  Timeout: No PARAM_VALUE received")
        return None

    # 解析参数ID
    param_id = msg.param_id

    # 解析参数值
    param_value = msg.param_value

    # 参数类型映射
    type_names = {
        0: "INT8", 1: "UINT8", 2: "INT16", 3: "UINT16", 4: "INT32",
        5: "UINT32", 6: "INT64", 7: "UINT64", 8: "FLOAT", 9: "DOUBL
```

```
E"
    }
    param_type = type_names.get(msg.param_type, f"UNKNOWN({msg.para
m_type})")

    print("\n  Received PARAM_VALUE:")
    print(f"    ID: {param_id}")
    print(f"    Value: {param_value}")
    print(f"    Type: {param_type}")
    print(f"    Index/Total: {msg.param_index}/{msg.param_count}")

    return {
        'id': param_id,
        'value': param_value,
        'type': param_type,
        'index': msg.param_index,
        'count': msg.param_count
    }

def receive_all_param_values(vehicle, timeout=10.0):
    """接收所有参数值，直到收到 param_count 条消息后退出循环"""
    received_params = []
    expected_count = None
    print(f"  Waiting for all PARAM_VALUE messages...")

    while True:
        msg = vehicle.recv_match(type='PARAM_VALUE', blocking=True,
timeout=timeout)

        if not msg:
            print(f"  Timeout: Received {len(received_params)} para
meters (expected {expected_count if expected_count else 'unknow
n'})")
            break

        param_id = msg.param_id
        param_value = msg.param_value

        # 参数类型映射
        type_names = {
            0: "INT8", 1: "UINT8", 2: "INT16", 3: "UINT16", 4: "INT
32",
            5: "UINT32", 6: "INT64", 7: "UINT64", 8: "FLOAT", 9: "D
OUBLE"
        }
```

```
            param_type = type_names.get(msg.param_type, f"UNKNOWN({msg.
param_type})")

            # 首次收到消息时记录总参数数量
            if expected_count is None:
                expected_count = msg.param_count
                print(f"  Total parameters to receive: {expected_coun
t}")

            print(f"  [{msg.param_index}/{expected_count}] {param_id} =
{param_value} ({param_type})")

            received_params.append({
                'id': param_id,
                'value': param_value,
                'type': param_type,
                'index': msg.param_index,
                'count': msg.param_count
            })

            # 当已接收数量达到预期总数时退出循环
            if len(received_params) >= expected_count:
                print(f"\n  Successfully received all {expected_count}
parameters")
                break

    return received_params

def main():
    print("  Starting MAVLink PARAM_SET Tool (UDP Listening Mode)")
    print("------------------------------------------------------")
    print("This script listens for MAVLink on ALL interfaces:1455
0")
    print("Ensure your vehicle is configured to send to this port")
    print("------------------------------------------------------
\n")

    vehicle = connect_to_vehicle()

    # 操作选择
    print("\nAvailable operations:")
    print("1. Set parameter value")
    print("2. Read parameter value (by ID or index)")
    print("3. Request all parameters (PARAM_REQUEST_LIST)")
    choice = input("\nSelect operation [1-3]: ").strip()
```

```
    if choice == '1':
        param_id = input("Enter parameter ID to set: ").strip()
        param_value = float(input("Enter new value: ").strip())

        print("\nParameter types:")
        print("  8 = FLOAT (most common)")
        print("  9 = DOUBLE")
        print("  4 = INT32")
        print("  1 = UINT8")
        param_type = int(input("Enter type number [8]: ").strip() or 8)

        send_param_set(vehicle, param_id, param_value, param_type)
        receive_param_value(vehicle)

    elif choice == '2':
        # mode = input("Read by [id/index]: ").lower()
        mode = 'id'
        if mode == 'id':
            param_id = input("Enter parameter ID: ").strip()
            send_param_request_read(vehicle, param_id=param_id)
            receive_param_value(vehicle)
        elif mode == 'index':
            param_index = int(input("Enter parameter index: "))
            send_param_request_read(vehicle, param_index=param_index)
            receive_param_value(vehicle)
        else:
            print("Invalid mode. Use 'id' or 'index'")

    elif choice == '3':
        print("\n  Requesting all parameters...")
        send_param_request_list(vehicle)
        all_params = receive_all_param_values(vehicle)
        print(f"\n  Total parameters received: {len(all_params)}")

    else:
        print("Invalid choice. Select 1, 2, or 3.")


if __name__ == "__main__":
    main()
```

示例使用说明

获取

运行后，成功建立连接

```
🚀 Starting MAVLink PARAM_SET Tool (UDP Listening Mode)
------------------------------------------------------
This script listens for MAVLink on ALL interfaces:14550
Ensure your vehicle is configured to send to this port
------------------------------------------------------

Listening for MAVLink on: udpin:0.0.0.0:14550
Waiting for HEARTBEAT (timeout: 10s)...
✅ Heartbeat received from system 3, component 0

Available operations:
1. Set parameter value
2. Read parameter value (by ID or index)
3. Request all parameters (PARAM_REQUEST_LIST)
```

输入“2”选择读取参数

```
Select operation [1-3]: 2
```

输入对应参数“FOCUS_MODE”

```
Enter parameter ID: FOCUS_MODE
```

成功返回该参数对应的数值

```
📥 Received PARAM_VALUE:
  🆔 ID: FOCUS_MODE
  🔢 Value: 2.0
```

设置

运行后，成功建立连接

```
🚀 Starting MAVLink PARAM_SET Tool (UDP Listening Mode)
--------------------------------------------------
This script listens for MAVLink on ALL interfaces:14550
Ensure your vehicle is configured to send to this port
--------------------------------------------------

Listening for MAVLink on: udpin:0.0.0.0:14550
Waiting for HEARTBEAT (timeout: 10s)...
✅ Heartbeat received from system 3, component 0

Available operations:
1. Set parameter value
2. Read parameter value (by ID or index)
3. Request all parameters (PARAM_REQUEST_LIST)
```

输入“1”选择设置参数

```
Select operation [1-3]: 1
```

输入对应参数“FOCUS_MODE”

```
Enter parameter ID to set: FOCUS_MODE
```

输入要设置的数值

```
Enter new value: 1
```

选择参数类型“1”

```
Parameter types:
  9 = DOUBLE
  4 = INT32
  1 = UINT8
Enter type number [8]: 1
```

成功返回设置值

```
📩 Received PARAM_VALUE:
  🆔 ID: FOCUS_MODE
  🔢 Value: 1.0
```

相机对焦点

FOCUS_TARGET

设置

mavlink消息：

接收PARAM_EXT_SET
发送PARAM_EXT_ACK

获取

mavlink消息：

接收PARAM_EXT_REQUEST_READ
发送PARAM_EXT_VALUE

参数：T_DjiCameraPointInScreen

字段名	类型	说明
focusX	dji_f32_t	指定对焦区域的水平坐标，范围：0 ~ 1；(0.0, 0.0) 表示屏幕左上角
focusY	dji_f32_t	指定对焦区域的垂直坐标，范围：0 ~ 1

示例

```
import time
import sys
import struct
import ctypes
from pymavlink import mavutil

def connect_to_vehicle():
    """建立 UDP 监听连接（默认监听所有接口的 14550 端口）"""
    connection_string = "udpin:0.0.0.0:14550"
    print(f"Listening for MAVLink on: {connection_string}")
    vehicle = mavutil.mavlink_connection(connection_string)
    print("Waiting for HEARTBEAT (timeout: 10s)...")
    try:
        vehicle.wait_heartbeat(timeout=10)
        print(f"  Heartbeat received from system {vehicle.target_sy
stem}, component {vehicle.target_component}")
        return vehicle
    except Exception as e:
        print(f"  Connection failed: {str(e)}")
        print("Ensure your vehicle is sending MAVLink messages to 1
```

```
4550 port")
        print("Example QGroundControl setup: Settings > General > M
AVLink > Listening Port = 14550")
        sys.exit(1)

def format_param_id(param_id):
    """格式化16字节参数ID（符合MAVLink协议要求）"""
    if len(param_id) > 16:
        raise ValueError("param_id cannot exceed 16 characters")

    param_id_bytes = bytearray(16)
    for i in range(min(len(param_id), 16)):
        param_id_bytes[i] = ord(param_id[i])

    # 长度<16时添加NULL终止符
    if len(param_id) < 16:
        param_id_bytes[len(param_id)] = 0

    return param_id_bytes

# ============ 结构体打包函数 ============

def pack_uint8_xy(x, y):
    """打包 {uint8_t x; uint8_t y} 结构体 (2字节)"""
    return struct.pack('<BB', x, y)

def pack_float32_xy(x, y):
    """打包 {float32 x; float32 y} 结构体 (8字节)"""
    return struct.pack('<ff', x, y)

def pack_float32_lxlyrxry(LX, LY, RX, RY):
    """打包 {float32 LX; float32 LY; float32 RX; float32 RY;} 结构体
(16字节)"""
    return struct.pack('<ffff', LX, LY, RX, RY)

def pack_ering_mode(size, eringMode, minValue, maxValue):
    """打包 {uint8_t size; uint8_t eringMode[16]; uint32_t minValue;
uint32_t maxValue;} 结构体 (25字节)

    Args:
        size: uint8_t, eringMode数组的有效长度
        eringMode: list of int (0-255), 16个uint8_t值
        minValue: uint32_t, 最小值
        maxValue: uint32_t, 最大值
```

```
    Returns:
        bytes: 打包后的二进制数据
    """
    # 确保eringMode是16字节
    eringMode_bytes = bytearray(16)
    for i in range(min(len(eringMode), 16)):
        eringMode_bytes[i] = eringMode[i] & 0xFF

    return struct.pack('<B16sII', size, eringMode_bytes, minValue,
maxValue)

def format_param_value(param_value):
    """格式化128字节参数值（接受二进制输入）"""
    if isinstance(param_value, str):
        # 如果是十六进制字符串（如 "48656c6c6f"），先转换为字节
        if param_value.startswith('0x') or all(c in '0123456789abcd
efABCDEF' for c in param_value.replace(' ', '')):
            hex_str = param_value.replace('0x', '').replace(' ',
'')
            if len(hex_str) % 2 == 0:
                param_value = bytes.fromhex(hex_str)
            else:
                raise ValueError("Invalid hex string: must have eve
n number of characters")
        else:
            # 普通字符串转字节
            param_value = param_value.encode('utf-8')

    if len(param_value) > 128:
        raise ValueError("param_value cannot exceed 128 bytes")

    param_value_bytes = bytearray(128)
    param_value_bytes[:len(param_value)] = param_value

    return param_value_bytes

def hex_dump(data, bytes_per_line=16):
    """格式化二进制数据为 hex + ASCII 展示（类 hexdump 风格）

    Returns:
        str: 格式化后的多行字符串
    """
    lines = []
    for i in range(0, len(data), bytes_per_line):
        chunk = data[i:i + bytes_per_line]
```

```
        hex_part = ' '.join(f'{b:02X}' for b in chunk)
        ascii_part = ''.join(chr(b) if 32 <= b < 127 else '.' for b
in chunk)
        lines.append(f"  {i:04X}  {hex_part:<{bytes_per_line*3}}
{ascii_part}")
    return '\n'.join(lines)

def send_param_ext_request_read(vehicle, param_id=None, param_index
=-1):
    """发送参数读取请求"""
    param_id_bytes = format_param_id(param_id) if param_id else byt
earray(16)

    vehicle.mav.param_ext_request_read_send(
        vehicle.target_system,
        vehicle.target_component,
        param_id_bytes,
        param_index
    )

    target = f"index {param_index}" if param_index >= 0 else f"ID
'{param_id}'"
    print(f"  Sent PARAM_EXT_REQUEST_READ for {target}")

def send_param_ext_request_list(vehicle):
    """发送请求所有参数列表"""
    vehicle.mav.param_ext_request_list_send(
        vehicle.target_system,
        vehicle.target_component
    )
    print(f"  Sent PARAM_EXT_REQUEST_LIST to system {vehicle.target
_system}, component {vehicle.target_component}")

def send_param_ext_set(vehicle, param_id, param_value, param_type):
    """发送参数设置请求"""
    vehicle.mav.param_ext_set_send(
        vehicle.target_system,
        vehicle.target_component,
        format_param_id(param_id),
        format_param_value(param_value),
        param_type
    )
    print(f"  Sent PARAM_EXT_SET for '{param_id}' = '{param_value}'
(type: {param_type})")
```

```
def receive_param_ext_value(vehicle, timeout=5.0):
    """接收参数值响应"""
    print(f"  Waiting for PARAM_EXT_VALUE ({timeout}s timeout)...")
    msg = vehicle.recv_match(type='PARAM_EXT_VALUE', blocking=True,
timeout=timeout)

    if not msg:
        print("  Timeout: No PARAM_EXT_VALUE received")
        return None

    # 解析参数ID
    param_id = msg.param_id

    # 解析参数值（直接获取128字节原始数据）
    param_value_raw = msg.param_value
    # byte_pointer = (ctypes.c_ubyte * 128).from_buffer_copy(msg.pa
ram_value)
    if isinstance(param_value_raw, (bytes, bytearray)):
        param_value_bytes = bytes(param_value_raw)
    # param_value_bytes = bytes(byte_pointer)
    elif isinstance(param_value_raw, str):
        # 替换无法编码的字符（如 \ufffd）为 0x00，还原为128字节
        param_value_bytes = param_value_raw.encode('latin1', errors
='replace')
    else:
        param_value_bytes = bytes(param_value_raw)

    # 参数类型映射
    type_names = {
        0: "INT8", 1: "UINT8", 2: "INT16", 3: "UINT16", 4: "INT32",
        5: "UINT32", 6: "INT64", 7: "UINT64", 8: "FLOAT", 9: "FLOA
T",
        10: "CHAR", 11: "CHAR", 12: "UINT8_ENUM", 13: "STRING"
    }
    param_type = type_names.get(msg.param_type, f"UNKNOWN({msg.para
m_type})")

    print("\n  Received PARAM_EXT_VALUE:")
    print(f"    ID: {param_id}")
    print(f"    Type: {param_type}  |  Data: {len(param_value_byte
s)} bytes")
    print(f"    Index/Total: {msg.param_index}/{msg.param_count}")
    print(hex_dump(param_value_bytes))

    return {
```

```
        'id': param_id,
        'value': param_value_bytes,
        'type': param_type,
        'index': msg.param_index,
        'count': msg.param_count
    }


def receive_all_param_ext_values(vehicle, timeout=10.0):
    """接收所有参数值，直到收到 param_count 条消息后退出循环"""
    received_params = []
    expected_count = None
    print(f"  Waiting for all PARAM_EXT_VALUE messages...")

    while True:
        msg = vehicle.recv_match(type='PARAM_EXT_VALUE', blocking=True, timeout=timeout)

        if not msg:
            print(f"  Timeout: Received {len(received_params)} parameters (expected {expected_count if expected_count else 'unknown'})")
            break

        param_id = msg.param_id
        param_value_raw = msg.param_value

        # 直接获取128字节原始数据
        if isinstance(param_value_raw, (bytes, bytearray)):
            param_value_bytes = bytes(param_value_raw)
        elif isinstance(param_value_raw, str):
            # 替换无法编码的字符（如 \ufffd）为 0x00
            param_value_bytes = param_value_raw.encode('latin1', errors='replace')
        else:
            param_value_bytes = bytes(param_value_raw)

        # 参数类型映射
        type_names = {
            0: "INT8", 1: "UINT8", 2: "INT16", 3: "UINT16", 4: "INT32",
            5: "UINT32", 6: "INT64", 7: "UINT64", 8: "FLOAT", 9: "DOUBLE",
            10: "CHAR", 11: "BOOL", 12: "UINT8_ENUM", 13: "STRING"
        }
```

```
        param_type = type_names.get(msg.param_type, f"UNKNOWN({msg.
param_type})")

        # 首次收到消息时记录总参数数量
        if expected_count is None:
            expected_count = msg.param_count
            print(f"  Total parameters to receive: {expected_coun
t}")

        print(f"  [{msg.param_index}/{expected_count}] {param_id}
[{param_type}][{len(param_value_bytes)}B]")
        print(hex_dump(param_value_bytes))

        received_params.append({
            'id': param_id,
            'value': param_value_bytes,
            'type': param_type,
            'index': msg.param_index,
            'count': msg.param_count
        })

        # 当已接收数量达到预期总数时退出循环
        if len(received_params) >= expected_count:
            print(f"\n  Successfully received all {expected_count}
parameters")
            break

    return received_params

def receive_param_ext_ack(vehicle, timeout=5.0):
    """接收参数设置确认"""
    print(f"  Waiting for PARAM_EXT_ACK ({timeout}s timeout)...")
    msg = vehicle.recv_match(type='PARAM_EXT_ACK', blocking=True, t
imeout=timeout)

    if not msg:
        print("  Timeout: No PARAM_EXT_ACK received")
        return None

    param_id = msg.param_id
    param_value = msg.param_value

    # 结果代码映射
    result_codes = {
        0: "  PARAM_ACK_ACCEPTED",
```

```
            1: "⚠ PARAM_ACK_VALUE_UNSUPPORTED",
            2: "  PARAM_ACK_FAILED",
            3: "  PARAM_ACK_IN_PROGRESS"
        }
        result = result_codes.get(msg.param_result, f"  UNKNOWN({msg.pa
ram_result})")

        print("\n  Received PARAM_EXT_ACK:")
        print(f"    ID: {param_id}")
        print(f"    Value: {param_value}")
        print(f"    Result: {result}")

        return {
            'id': param_id,
            'value': param_value,
            'result': result,
            'result_code': msg.param_result
        }

def main():
    print("  Starting MAVLink PARAM_EXT Tool (UDP Listening Mode)")
    print("-----------------------------------------------------")
    print("This script listens for MAVLink on ALL interfaces:1455
0")
    print("Ensure your vehicle is configured to send to this port")
    print("-----------------------------------------------------
\n")

    vehicle = connect_to_vehicle()

    # 操作选择
    print("\nAvailable operations:")
    print("1. Read parameter value (by ID or index)")
    print("2. Set parameter value (raw/hex/string)")
    print("3. Request all parameters (PARAM_EXT_REQUEST_LIST)")
    print("4. Set struct: uint8_xy {x, y}")
    print("5. Set struct: float32_xy {x, y}")
    print("6. Set struct: float32_lxlyrxry {LX, LY, RX, RY}")
    print("7. Set struct: ering_mode {size, eringMode[16], minValu
e, maxValue}")
    choice = input("\nSelect operation [1-7]: ").strip()

    if choice == '1':
        # mode = input("Read by [id/index]: ").lower()
        mode = 'id'
```

```
        if mode == 'id':
            param_id = input("Enter parameter ID: ").strip()
            send_param_ext_request_read(vehicle, param_id=param_id)
            receive_param_ext_value(vehicle)
        elif mode == 'index':
            param_index = int(input("Enter parameter index: "))
            send_param_ext_request_read(vehicle, param_index=param_
index)
            receive_param_ext_value(vehicle)
        else:
            print("Invalid mode. Use 'id' or 'index'")

    elif choice == '2':
        param_id = input("Enter parameter ID to set: ").strip()

        print("\nEnter parameter value (supports):")
        print("  - Plain string (e.g., hello)")
        print("  - Hex string (e.g., 48656c6c6f or 0x48656c6c6f)")
        print("  - Raw bytes (e.g., b'\\x48\\x65\\x6c\\x6c\\x6f')")
        param_value_input = input("Enter value: ").strip()

        # 解析输入值
        if param_value_input.startswith("b'") or param_value_input.
startswith('b"'):
            # Python字节字面量格式
            param_value = eval(param_value_input)
        elif param_value_input.startswith('0x') or all(c in '012345
6789abcdefABCDEF ' for c in param_value_input):
            # 十六进制格式
            hex_str = param_value_input.replace('0x', '').replace('
', '')
            try:
                param_value = bytes.fromhex(hex_str)
            except ValueError:
                param_value = param_value_input.encode('utf-8')
        else:
            # 默认作为字符串处理
            param_value = param_value_input.encode('utf-8')

        print("\nParameter types:")
        print("  8 = FLOAT (most common)")
        print("  13 = STRING")
        print("  10 = CHAR (raw bytes)")
        print("  Other types require specific binary formats")
        param_type = int(input("Enter type number [8]: ") or 8)
```

```
        send_param_ext_set(vehicle, param_id, param_value, param_ty
pe)
        receive_param_ext_ack(vehicle)

    elif choice == '4':
        """设置 uint8_xy 结构体 {uint8_t x; uint8_t y}"""
        param_id = input("Enter parameter ID: ").strip()
        x = int(input("Enter x (0-255): "))
        y = int(input("Enter y (0-255): "))
        param_value = pack_uint8_xy(x, y)
        print(f"  Packed: {param_value.hex()} ({len(param_value)} b
ytes)")
        send_param_ext_set(vehicle, param_id, param_value, 10)  # C
HAR type
        receive_param_ext_ack(vehicle)

    elif choice == '5':
        """设置 float32_xy 结构体 {float32 x; float32 y}"""
        param_id = input("Enter parameter ID: ").strip()
        x = float(input("Enter x: "))
        y = float(input("Enter y: "))
        param_value = pack_float32_xy(x, y)
        print(f"  Packed: {param_value.hex()} ({len(param_value)} b
ytes)")
        send_param_ext_set(vehicle, param_id, param_value, 10)  # C
HAR type
        receive_param_ext_ack(vehicle)

    elif choice == '6':
        """设置 float32_lxlyrxry 结构体 {float32 LX; float32 LY; floa
t32 RX; float32 RY;}"""
        param_id = input("Enter parameter ID: ").strip()
        LX = float(input("Enter LX: "))
        LY = float(input("Enter LY: "))
        RX = float(input("Enter RX: "))
        RY = float(input("Enter RY: "))
        param_value = pack_float32_lxlyrxry(LX, LY, RX, RY)
        print(f"  Packed: {param_value.hex()} ({len(param_value)} b
ytes)")
        send_param_ext_set(vehicle, param_id, param_value, 10)  # C
HAR type
        receive_param_ext_ack(vehicle)

    elif choice == '7':
```

```
        """设置 ering_mode 结构体 {uint8_t size; uint8_t eringMode[16]; uint32_t minValue; uint32_t maxValue;}"""
        param_id = input("Enter parameter ID: ").strip()
        size = int(input("Enter size (0-16): "))
        eringMode = []
        print("Enter 16 eringMode values (0-255), one per line:")
        for i in range(16):
            val = int(input(f"  eringMode[{i}]: "))
            eringMode.append(val)
        minValue = int(input("Enter minValue: "))
        maxValue = int(input("Enter maxValue: "))
        param_value = pack_ering_mode(size, eringMode, minValue, maxValue)
        print(f"  Packed: {param_value.hex()} ({len(param_value)} bytes)")
        send_param_ext_set(vehicle, param_id, param_value, 10)  # CHAR type
        receive_param_ext_ack(vehicle)

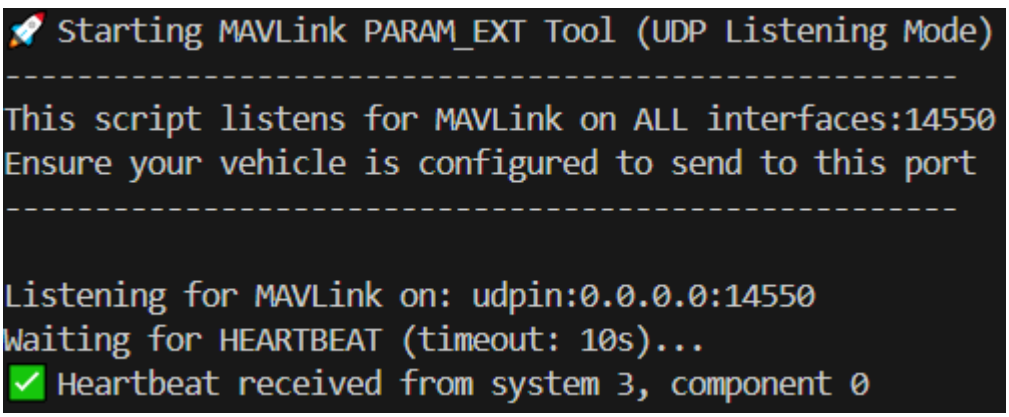
    elif choice == '3':
        print("\n  Requesting all parameters...")
        send_param_ext_request_list(vehicle)
        all_params = receive_all_param_ext_values(vehicle)
        print(f"\n  Total parameters received: {len(all_params)}")

    else:
        print("Invalid choice. Select 1, 2, or 3.")


if __name__ == "__main__":
    main()
```

示例使用说明

运行后，成功建立连接



输入“1”选择读取参数

```
Available operations:
1. Read parameter value (by ID or index)
2. Set parameter value (raw/hex/string)
3. Request all parameters (PARAM_EXT_REQUEST_LIST)
4. Set struct: uint8_xy {x, y}
5. Set struct: float32_xy {x, y}
6. Set struct: float32_lxlyrxry {LX, LY, RX, RY}
7. Set struct: ering_mode {size, eringMode[16], minValue, maxValue}

Select operation [1-7]: 1
```

输入对应参数“FOCUS_TARGET”

```
Enter parameter ID: FOCUS_TARGET
```

成功返回该参数对应的数值

```
Received PARAM_EXT_VALUE:
  ID: FOCUS_TARGET
fd950000000301420100120003004 64f4355535f5441524745540000000000ffff3e00ffff3e0000000000
```

测光功能

获取测光模式范围

METER_MOD_RANGE

mavlink消息：

接收PARAM_EXT_REQUEST_READ
发送PARAM_EXT_VALUE

参数：

字段	类型	说明
size	uint8_t	有效列表长度（最大16）
minValue	uint32_t	数值范围最小值
maxValue	uint32_t	数值范围最大值

union（同一时刻仅生效一个字段）

字段	类型	说明
photoStorageFormat	E_DjiCameraManagerPhotoStorageFormat[16]	照片存储格式枚
videoStorageFormat	E_DjiCameraManagerVideoStorageFormat[16]	视频存储格式枚
photoRatioFormat	E_DjiCameraManagerPhotoRatio[16]	照片比例枚举列
streamSource	E_DjiCameraManagerStreamSource[16]	视频流源枚举列
streamStorage	E_DjiCameraManagerStreamStorage[16]	视频存储介质枚
nightSceneMode	E_DjiCameraManagerNightSceneMode[16]	夜景模式枚举列
meteringMode	E_DjiCameraManagerMeteringMode[16]	测光模式枚举列

示例

```
import time
import sys
import struct
import ctypes
from pymavlink import mavutil

def connect_to_vehicle():
    """建立 UDP 监听连接（默认监听所有接口的 14550 端口）"""
    connection_string = "udpin:0.0.0.0:14550"
    print(f"Listening for MAVLink on: {connection_string}")
    vehicle = mavutil.mavlink_connection(connection_string)
    print("Waiting for HEARTBEAT (timeout: 10s)...")
    try:
        vehicle.wait_heartbeat(timeout=10)
        print(f"  Heartbeat received from system {vehicle.target_system}, component {vehicle.target_component}")
        return vehicle
    except Exception as e:
        print(f"  Connection failed: {str(e)}")
        print("Ensure your vehicle is sending MAVLink messages to 14550 port")
        print("Example QGroundControl setup: Settings > General > MAVLink > Listening Port = 14550")
        sys.exit(1)

def format_param_id(param_id):
    """格式化16字节参数ID（符合MAVLink协议要求）"""
    if len(param_id) > 16:
        raise ValueError("param_id cannot exceed 16 characters")

    param_id_bytes = bytearray(16)
    for i in range(min(len(param_id), 16)):
        param_id_bytes[i] = ord(param_id[i])

    # 长度<16时添加NULL终止符
    if len(param_id) < 16:
        param_id_bytes[len(param_id)] = 0

    return param_id_bytes

# ============ 结构体打包函数 ============

def pack_uint8_xy(x, y):
    """打包 {uint8_t x; uint8_t y} 结构体 (2字节)"""
    return struct.pack('<BB', x, y)
```

```
def pack_float32_xy(x, y):
    """打包 {float32 x; float32 y} 结构体 (8字节)"""
    return struct.pack('<ff', x, y)

def pack_float32_lxlyrxry(LX, LY, RX, RY):
    """打包 {float32 LX; float32 LY; float32 RX; float32 RY;} 结构体 (16字节)"""
    return struct.pack('<ffff', LX, LY, RX, RY)

def pack_ering_mode(size, eringMode, minValue, maxValue):
    """打包 {uint8_t size; uint8_t eringMode[16]; uint32_t minValue; uint32_t maxValue;} 结构体 (25字节)

    Args:
        size: uint8_t, eringMode数组的有效长度
        eringMode: list of int (0-255), 16个uint8_t值
        minValue: uint32_t, 最小值
        maxValue: uint32_t, 最大值

    Returns:
        bytes: 打包后的二进制数据
    """
    # 确保eringMode是16字节
    eringMode_bytes = bytearray(16)
    for i in range(min(len(eringMode), 16)):
        eringMode_bytes[i] = eringMode[i] & 0xFF

    return struct.pack('<B16sII', size, eringMode_bytes, minValue, maxValue)

def format_param_value(param_value):
    """格式化128字节参数值（接受二进制输入）"""
    if isinstance(param_value, str):
        # 如果是十六进制字符串（如 "48656c6c6f"），先转换为字节
        if param_value.startswith('0x') or all(c in '0123456789abcdefABCDEF' for c in param_value.replace(' ', '')):
            hex_str = param_value.replace('0x', '').replace(' ', '')
            if len(hex_str) % 2 == 0:
                param_value = bytes.fromhex(hex_str)
            else:
                raise ValueError("Invalid hex string: must have even number of characters")
        else:
```

```
            # 普通字符串转字节
            param_value = param_value.encode('utf-8')

    if len(param_value) > 128:
        raise ValueError("param_value cannot exceed 128 bytes")

    param_value_bytes = bytearray(128)
    param_value_bytes[:len(param_value)] = param_value

    return param_value_bytes

def hex_dump(data, bytes_per_line=16):
    """格式化二进制数据为 hex + ASCII 展示（类 hexdump 风格）

    Returns:
        str: 格式化后的多行字符串
    """
    lines = []
    for i in range(0, len(data), bytes_per_line):
        chunk = data[i:i + bytes_per_line]
        hex_part = ' '.join(f'{b:02X}' for b in chunk)
        ascii_part = ''.join(chr(b) if 32 <= b < 127 else '.' for b
in chunk)
        lines.append(f"  {i:04X}  {hex_part:<{bytes_per_line*3}}
{ascii_part}")
    return '\n'.join(lines)

def send_param_ext_request_read(vehicle, param_id=None, param_index
=-1):
    """发送参数读取请求"""
    param_id_bytes = format_param_id(param_id) if param_id else byt
earray(16)

    vehicle.mav.param_ext_request_read_send(
        vehicle.target_system,
        vehicle.target_component,
        param_id_bytes,
        param_index
    )

    target = f"index {param_index}" if param_index >= 0 else f"ID
'{param_id}'"
    print(f"  Sent PARAM_EXT_REQUEST_READ for {target}")

def send_param_ext_request_list(vehicle):
```

```
    """发送请求所有参数列表"""
    vehicle.mav.param_ext_request_list_send(
        vehicle.target_system,
        vehicle.target_component
    )
    print(f"  Sent PARAM_EXT_REQUEST_LIST to system {vehicle.target
_system}, component {vehicle.target_component}")

def send_param_ext_set(vehicle, param_id, param_value, param_type):
    """发送参数设置请求"""
    vehicle.mav.param_ext_set_send(
        vehicle.target_system,
        vehicle.target_component,
        format_param_id(param_id),
        format_param_value(param_value),
        param_type
    )
    print(f"  Sent PARAM_EXT_SET for '{param_id}' = '{param_value}'
(type: {param_type})")

def receive_param_ext_value(vehicle, timeout=5.0):
    """接收参数值响应"""
    print(f"  Waiting for PARAM_EXT_VALUE ({timeout}s timeout)...")
    msg = vehicle.recv_match(type='PARAM_EXT_VALUE', blocking=True,
timeout=timeout)

    if not msg:
        print("  Timeout: No PARAM_EXT_VALUE received")
        return None

    # 解析参数ID
    param_id = msg.param_id

    # 解析参数值（直接获取128字节原始数据）
    param_value_raw = msg.param_value
    # byte_pointer = (ctypes.c_ubyte * 128).from_buffer_copy(msg.pa
ram_value)
    if isinstance(param_value_raw, (bytes, bytearray)):
        param_value_bytes = bytes(param_value_raw)
    # param_value_bytes = bytes(byte_pointer)
    elif isinstance(param_value_raw, str):
        # 替换无法编码的字符（如 \ufffd）为 0x00，还原为128字节
        param_value_bytes = param_value_raw.encode('latin1', errors
='replace')
    else:
```

```
        param_value_bytes = bytes(param_value_raw)

    # 参数类型映射
    type_names = {
        0: "INT8", 1: "UINT8", 2: "INT16", 3: "UINT16", 4: "INT32",
        5: "UINT32", 6: "INT64", 7: "UINT64", 8: "FLOAT", 9: "FLOA
T",
        10: "CHAR", 11: "CHAR", 12: "UINT8_ENUM", 13: "STRING"
    }
    param_type = type_names.get(msg.param_type, f"UNKNOWN({msg.para
m_type})")

    print("\n  Received PARAM_EXT_VALUE:")
    print(f"    ID: {param_id}")
    print(f"    Type: {param_type}  |  Data: {len(param_value_byte
s)} bytes")
    print(f"    Index/Total: {msg.param_index}/{msg.param_count}")
    print(hex_dump(param_value_bytes))

    return {
        'id': param_id,
        'value': param_value_bytes,
        'type': param_type,
        'index': msg.param_index,
        'count': msg.param_count
    }


def receive_all_param_ext_values(vehicle, timeout=10.0):
    """接收所有参数值，直到收到 param_count 条消息后退出循环"""
    received_params = []
    expected_count = None
    print(f"  Waiting for all PARAM_EXT_VALUE messages...")

    while True:
        msg = vehicle.recv_match(type='PARAM_EXT_VALUE', blocking=T
rue, timeout=timeout)

        if not msg:
            print(f"  Timeout: Received {len(received_params)} para
meters (expected {expected_count if expected_count else 'unknow
n'})")
            break

        param_id = msg.param_id
```

```
        param_value_raw = msg.param_value

        # 直接获取128字节原始数据
        if isinstance(param_value_raw, (bytes, bytearray)):
            param_value_bytes = bytes(param_value_raw)
        elif isinstance(param_value_raw, str):
            # 替换无法编码的字符（如 \ufffd）为 0x00
            param_value_bytes = param_value_raw.encode('latin1', er
rors='replace')
        else:
            param_value_bytes = bytes(param_value_raw)

        # 参数类型映射
        type_names = {
            0: "INT8", 1: "UINT8", 2: "INT16", 3: "UINT16", 4: "INT
32",
            5: "UINT32", 6: "INT64", 7: "UINT64", 8: "FLOAT", 9: "D
OUBLE",
            10: "CHAR", 11: "BOOL", 12: "UINT8_ENUM", 13: "STRING"
        }
        param_type = type_names.get(msg.param_type, f"UNKNOWN({msg.
param_type})")

        # 首次收到消息时记录总参数数量
        if expected_count is None:
            expected_count = msg.param_count
            print(f"  Total parameters to receive: {expected_coun
t}")

        print(f"  [{msg.param_index}/{expected_count}] {param_id}
[{param_type}][{len(param_value_bytes)}B]")
        print(hex_dump(param_value_bytes))

        received_params.append({
            'id': param_id,
            'value': param_value_bytes,
            'type': param_type,
            'index': msg.param_index,
            'count': msg.param_count
        })

        # 当已接收数量达到预期总数时退出循环
        if len(received_params) >= expected_count:
            print(f"\n  Successfully received all {expected_count}
parameters")
```

```
            break

    return received_params

def receive_param_ext_ack(vehicle, timeout=5.0):
    """接收参数设置确认"""
    print(f"  Waiting for PARAM_EXT_ACK ({timeout}s timeout)...")
    msg = vehicle.recv_match(type='PARAM_EXT_ACK', blocking=True, t
imeout=timeout)

    if not msg:
        print("  Timeout: No PARAM_EXT_ACK received")
        return None

    param_id = msg.param_id
    param_value = msg.param_value

    # 结果代码映射
    result_codes = {
        0: "  PARAM_ACK_ACCEPTED",
        1: "⚠ PARAM_ACK_VALUE_UNSUPPORTED",
        2: "  PARAM_ACK_FAILED",
        3: "  PARAM_ACK_IN_PROGRESS"
    }
    result = result_codes.get(msg.param_result, f"  UNKNOWN({msg.pa
ram_result})")

    print("\n  Received PARAM_EXT_ACK:")
    print(f"    ID: {param_id}")
    print(f"    Value: {param_value}")
    print(f"    Result: {result}")

    return {
        'id': param_id,
        'value': param_value,
        'result': result,
        'result_code': msg.param_result
    }

def main():
    print("  Starting MAVLink PARAM_EXT Tool (UDP Listening Mode)")
    print("-----------------------------------------------------")
    print("This script listens for MAVLink on ALL interfaces:1455
0")
    print("Ensure your vehicle is configured to send to this port")
```

```
    print("-----------------------------------------------------
\n")

    vehicle = connect_to_vehicle()

    # 操作选择
    print("\nAvailable operations:")
    print("1. Read parameter value (by ID or index)")
    print("2. Set parameter value (raw/hex/string)")
    print("3. Request all parameters (PARAM_EXT_REQUEST_LIST)")
    print("4. Set struct: uint8_xy {x, y}")
    print("5. Set struct: float32_xy {x, y}")
    print("6. Set struct: float32_lxlyrxry {LX, LY, RX, RY}")
    print("7. Set struct: ering_mode {size, eringMode[16], minValu
e, maxValue}")
    choice = input("\nSelect operation [1-7]: ").strip()

    if choice == '1':
        # mode = input("Read by [id/index]: ").lower()
        mode = 'id'
        if mode == 'id':
            param_id = input("Enter parameter ID: ").strip()
            send_param_ext_request_read(vehicle, param_id=param_id)
            receive_param_ext_value(vehicle)
        elif mode == 'index':
            param_index = int(input("Enter parameter index: "))
            send_param_ext_request_read(vehicle, param_index=param_
index)
            receive_param_ext_value(vehicle)
        else:
            print("Invalid mode. Use 'id' or 'index'")

    elif choice == '2':
        param_id = input("Enter parameter ID to set: ").strip()

        print("\nEnter parameter value (supports):")
        print("  - Plain string (e.g., hello)")
        print("  - Hex string (e.g., 48656c6c6f or 0x48656c6c6f)")
        print("  - Raw bytes (e.g., b'\\x48\\x65\\x6c\\x6c\\x6f')")
        param_value_input = input("Enter value: ").strip()

        # 解析输入值
        if param_value_input.startswith("b'") or param_value_input.
startswith('b"'):
            # Python字节字面量格式
```

```
            param_value = eval(param_value_input)
        elif param_value_input.startswith('0x') or all(c in '012345
6789abcdefABCDEF ' for c in param_value_input):
            # 十六进制格式
            hex_str = param_value_input.replace('0x', '').replace('
', '')
            try:
                param_value = bytes.fromhex(hex_str)
            except ValueError:
                param_value = param_value_input.encode('utf-8')
        else:
            # 默认作为字符串处理
            param_value = param_value_input.encode('utf-8')

        print("\nParameter types:")
        print("  8 = FLOAT (most common)")
        print("  13 = STRING")
        print("  10 = CHAR (raw bytes)")
        print("  Other types require specific binary formats")
        param_type = int(input("Enter type number [8]: ") or 8)

        send_param_ext_set(vehicle, param_id, param_value, param_ty
pe)
        receive_param_ext_ack(vehicle)

    elif choice == '4':
        """设置 uint8_xy 结构体 {uint8_t x; uint8_t y}"""
        param_id = input("Enter parameter ID: ").strip()
        x = int(input("Enter x (0-255): "))
        y = int(input("Enter y (0-255): "))
        param_value = pack_uint8_xy(x, y)
        print(f"  Packed: {param_value.hex()} ({len(param_value)} b
ytes)")
        send_param_ext_set(vehicle, param_id, param_value, 10)  # C
HAR type
        receive_param_ext_ack(vehicle)

    elif choice == '5':
        """设置 float32_xy 结构体 {float32 x; float32 y}"""
        param_id = input("Enter parameter ID: ").strip()
        x = float(input("Enter x: "))
        y = float(input("Enter y: "))
        param_value = pack_float32_xy(x, y)
        print(f"  Packed: {param_value.hex()} ({len(param_value)} b
ytes)")
```

```
        send_param_ext_set(vehicle, param_id, param_value, 10)  # C
HAR type
        receive_param_ext_ack(vehicle)

    elif choice == '6':
        """设置 float32_lxlyrxry 结构体 {float32 LX; float32 LY; floa
t32 RX; float32 RY;}"""
        param_id = input("Enter parameter ID: ").strip()
        LX = float(input("Enter LX: "))
        LY = float(input("Enter LY: "))
        RX = float(input("Enter RX: "))
        RY = float(input("Enter RY: "))
        param_value = pack_float32_lxlyrxry(LX, LY, RX, RY)
        print(f"  Packed: {param_value.hex()} ({len(param_value)} b
ytes)")
        send_param_ext_set(vehicle, param_id, param_value, 10)  # C
HAR type
        receive_param_ext_ack(vehicle)

    elif choice == '7':
        """设置 ering_mode 结构体 {uint8_t size; uint8_t eringMode[1
6]; uint32_t minValue; uint32_t maxValue;}"""
        param_id = input("Enter parameter ID: ").strip()
        size = int(input("Enter size (0-16): "))
        eringMode = []
        print("Enter 16 eringMode values (0-255), one per line:")
        for i in range(16):
            val = int(input(f"  eringMode[{i}]: "))
            eringMode.append(val)
        minValue = int(input("Enter minValue: "))
        maxValue = int(input("Enter maxValue: "))
        param_value = pack_ering_mode(size, eringMode, minValue, ma
xValue)
        print(f"  Packed: {param_value.hex()} ({len(param_value)} b
ytes)")
        send_param_ext_set(vehicle, param_id, param_value, 10)  # C
HAR type
        receive_param_ext_ack(vehicle)

    elif choice == '3':
        print("\n  Requesting all parameters...")
        send_param_ext_request_list(vehicle)
        all_params = receive_all_param_ext_values(vehicle)
        print(f"\n  Total parameters received: {len(all_params)}")
```

```
    else:
        print("Invalid choice. Select 1, 2, or 3.")


if __name__ == "__main__":
    main()
```

示例使用说明

运行后，成功建立连接

```
🚀 Starting MAVLink PARAM_EXT Tool (UDP Listening Mode)
------------------------------------------------------
This script listens for MAVLink on ALL interfaces:14550
Ensure your vehicle is configured to send to this port
------------------------------------------------------

Listening for MAVLink on: udpin:0.0.0.0:14550
Waiting for HEARTBEAT (timeout: 10s)...
✅ Heartbeat received from system 3, component 0
```

输入“1”选择设置参数

```
Available operations:
1. Read parameter value (by ID or index)
2. Set parameter value (raw/hex/string)
3. Request all parameters (PARAM_EXT_REQUEST_LIST)
4. Set struct: uint8_xy {x, y}
5. Set struct: float32_xy {x, y}
6. Set struct: float32_lxlyrxry {LX, LY, RX, RY}
7. Set struct: ering_mode {size, eringMode[16], minValue, maxValue}

Select operation [1-7]: 1
```

输入对应参数“METER_MOD_RANGE”

```
Enter parameter ID: METER_MOD_RANGE
```

成功返回完整消息

```
📥 Received PARAM_EXT_VALUE:
  🆔 ID: METER_MOD_RANGE
fd950000000301420100120004004d455445525f4d4f445f52414e47450003bafd67000000000100000002000000d8fff
fff80ffffff0000000000000000078000000050000000682187caaaaa0000600b005cffff0000800e005cffff000050bafd
67ffff0000f0e145ca00000000000000000000000000000000000000000000000000000000000000000000000000000000
00000000000000000000000000000bd87c
```

相机测光模式

METER_MODE

设置

mavlink消息：

接收PARAM_SET
发送PARAM_VALUE

获取

mavlink消息

接收PARAM_REQUEST_READ
发送PARAM_VALUE

参数：

名称	枚举值	描述
DJI_CAMERA_MANAGER_METERING_MODE_CENTRAL	0	中央测光（Cente Metering）， 以画面中央区域为
DJI_CAMERA_MANAGER_METERING_MODE_AVERAGE	1	平均测光（Avera 对整个画面区域进
DJI_CAMERA_MANAGER_METERING_MODE_SPOT	2	点测光（Spot Me 仅对选定的小区域

示例

```
import time
import sys
from pymavlink import mavutil

def connect_to_vehicle():
    """建立 UDP 监听连接（默认监听所有接口的 14550 端口）"""
    connection_string = "udpin:0.0.0.0:14550"
    print(f"Listening for MAVLink on: {connection_string}")
    vehicle = mavutil.mavlink_connection(connection_string)
    print("Waiting for HEARTBEAT (timeout: 10s)...")
    try:
        vehicle.wait_heartbeat(timeout=10)
        print(f"  Heartbeat received from system {vehicle.target_sy
```

```
stem}, component {vehicle.target_component}")
        return vehicle
    except Exception as e:
        print(f"  Connection failed: {str(e)}")
        print("Ensure your vehicle is sending MAVLink messages to 1
4550 port")
        print("Example QGroundControl setup: Settings > General > M
AVLink > Listening Port = 14550")
        sys.exit(1)

def format_param_id(param_id):
    """格式化16字节参数ID（符合MAVLink协议要求）"""
    if len(param_id) > 16:
        raise ValueError("param_id cannot exceed 16 characters")

    param_id_bytes = bytearray(16)
    for i in range(min(len(param_id), 16)):
        param_id_bytes[i] = ord(param_id[i])

    # 长度<16时添加NULL终止符
    if len(param_id) < 16:
        param_id_bytes[len(param_id)] = 0

    return param_id_bytes

def send_param_set(vehicle, param_id, param_value, param_type):
    """发送参数设置请求"""
    vehicle.mav.param_set_send(
        vehicle.target_system,
        vehicle.target_component,
        format_param_id(param_id),
        param_value,
        param_type
    )
    print(f"  Sent PARAM_SET for '{param_id}' = {param_value} (typ
e: {param_type})")

def send_param_request_read(vehicle, param_id=None, param_index=-
1):
    """发送参数读取请求"""
    param_id_bytes = format_param_id(param_id) if param_id else byt
earray(16)

    vehicle.mav.param_request_read_send(
        vehicle.target_system,
```

```
        vehicle.target_component,
        param_id_bytes,
        param_index
    )

    target = f"index {param_index}" if param_index >= 0 else f"ID
'{param_id}'"
    print(f"  Sent PARAM_REQUEST_READ for {target}")

def send_param_request_list(vehicle):
    """发送请求所有参数列表"""
    vehicle.mav.param_request_list_send(
        vehicle.target_system,
        vehicle.target_component
    )
    print(f"  Sent PARAM_REQUEST_LIST to system {vehicle.target_sys
tem}, component {vehicle.target_component}")

def receive_param_value(vehicle, timeout=5.0):
    """接收参数值响应"""
    print(f"  Waiting for PARAM_VALUE ({timeout}s timeout)...")
    msg = vehicle.recv_match(type='PARAM_VALUE', blocking=True, tim
eout=timeout)

    if not msg:
        print("  Timeout: No PARAM_VALUE received")
        return None

    # 解析参数ID
    param_id = msg.param_id

    # 解析参数值
    param_value = msg.param_value

    # 参数类型映射
    type_names = {
        0: "INT8", 1: "UINT8", 2: "INT16", 3: "UINT16", 4: "INT32",
        5: "UINT32", 6: "INT64", 7: "UINT64", 8: "FLOAT", 9: "DOUBL
E"
    }
    param_type = type_names.get(msg.param_type, f"UNKNOWN({msg.para
m_type})")

    print("\n  Received PARAM_VALUE:")
    print(f"    ID: {param_id}")
```

```
    print(f"    Value: {param_value}")
    print(f"    Type: {param_type}")
    print(f"    Index/Total: {msg.param_index}/{msg.param_count}")

    return {
        'id': param_id,
        'value': param_value,
        'type': param_type,
        'index': msg.param_index,
        'count': msg.param_count
    }

def receive_all_param_values(vehicle, timeout=10.0):
    """接收所有参数值，直到收到 param_count 条消息后退出循环"""
    received_params = []
    expected_count = None
    print(f"  Waiting for all PARAM_VALUE messages...")

    while True:
        msg = vehicle.recv_match(type='PARAM_VALUE', blocking=True,
timeout=timeout)

        if not msg:
            print(f"  Timeout: Received {len(received_params)} para
meters (expected {expected_count if expected_count else 'unknow
n'})")
            break

        param_id = msg.param_id
        param_value = msg.param_value

        # 参数类型映射
        type_names = {
            0: "INT8", 1: "UINT8", 2: "INT16", 3: "UINT16", 4: "INT
32",
            5: "UINT32", 6: "INT64", 7: "UINT64", 8: "FLOAT", 9: "D
OUBLE"
        }
        param_type = type_names.get(msg.param_type, f"UNKNOWN({msg.
param_type})")

        # 首次收到消息时记录总参数数量
        if expected_count is None:
            expected_count = msg.param_count
            print(f"  Total parameters to receive: {expected_coun
```

```
t}")

        print(f"  [{msg.param_index}/{expected_count}] {param_id} =
{param_value} ({param_type})")

        received_params.append({
            'id': param_id,
            'value': param_value,
            'type': param_type,
            'index': msg.param_index,
            'count': msg.param_count
        })

        # 当已接收数量达到预期总数时退出循环
        if len(received_params) >= expected_count:
            print(f"\n  Successfully received all {expected_count}
parameters")
            break

    return received_params

def main():
    print("  Starting MAVLink PARAM_SET Tool (UDP Listening Mode)")
    print("------------------------------------------------------")
    print("This script listens for MAVLink on ALL interfaces:1455
0")
    print("Ensure your vehicle is configured to send to this port")
    print("------------------------------------------------------
\n")

    vehicle = connect_to_vehicle()

    # 操作选择
    print("\nAvailable operations:")
    print("1. Set parameter value")
    print("2. Read parameter value (by ID or index)")
    print("3. Request all parameters (PARAM_REQUEST_LIST)")
    choice = input("\nSelect operation [1-3]: ").strip()

    if choice == '1':
        param_id = input("Enter parameter ID to set: ").strip()
        param_value = float(input("Enter new value: ").strip())

        print("\nParameter types:")
        print("  8 = FLOAT (most common)")
```

```
        print("  9 = DOUBLE")
        print("  4 = INT32")
        print("  1 = UINT8")
        param_type = int(input("Enter type number [8]: ").strip() o
r 8)

        send_param_set(vehicle, param_id, param_value, param_type)
        receive_param_value(vehicle)

    elif choice == '2':
        # mode = input("Read by [id/index]: ").lower()
        mode = 'id'
        if mode == 'id':
            param_id = input("Enter parameter ID: ").strip()
            send_param_request_read(vehicle, param_id=param_id)
            receive_param_value(vehicle)
        elif mode == 'index':
            param_index = int(input("Enter parameter index: "))
            send_param_request_read(vehicle, param_index=param_inde
x)
            receive_param_value(vehicle)
        else:
            print("Invalid mode. Use 'id' or 'index'")

    elif choice == '3':
        print("\n  Requesting all parameters...")
        send_param_request_list(vehicle)
        all_params = receive_all_param_values(vehicle)
        print(f"\n  Total parameters received: {len(all_params)}")

    else:
        print("Invalid choice. Select 1, 2, or 3.")


if __name__ == "__main__":
    main()
```

示例使用说明

获取

运行后，成功建立连接

```
🚀 Starting MAVLink PARAM_SET Tool (UDP Listening Mode)
--------------------------------------------------
This script listens for MAVLink on ALL interfaces:14550
Ensure your vehicle is configured to send to this port
--------------------------------------------------

Listening for MAVLink on: udpin:0.0.0.0:14550
Waiting for HEARTBEAT (timeout: 10s)...
✅ Heartbeat received from system 3, component 0

Available operations:
1. Set parameter value
2. Read parameter value (by ID or index)
3. Request all parameters (PARAM_REQUEST_LIST)
```

输入“2”选择读取参数

```
Select operation [1-3]: 2
```

输入对应参数“METER_MODE”

```
Enter parameter ID: METER_MODE
```

成功返回该参数对应的数值

```
📥 Received PARAM_VALUE:
  🆔 ID: METER_MODE
  🔢 Value: 1.0
```

设置

运行后，成功建立连接

```
🚀 Starting MAVLink PARAM_SET Tool (UDP Listening Mode)
--------------------------------------------------
This script listens for MAVLink on ALL interfaces:14550
Ensure your vehicle is configured to send to this port
--------------------------------------------------

Listening for MAVLink on: udpin:0.0.0.0:14550
Waiting for HEARTBEAT (timeout: 10s)...
✅ Heartbeat received from system 3, component 0

Available operations:
1. Set parameter value
2. Read parameter value (by ID or index)
3. Request all parameters (PARAM_REQUEST_LIST)
```

输入“1”选择设置参数

```
Available operations:
1. Set parameter value
2. Read parameter value (by ID or index)
3. Request all parameters (PARAM_REQUEST_LIST)

Select operation [1-3]: 1
```

输入对应参数“METER_MODE”

```
Enter parameter ID to set: METER_MODE
```

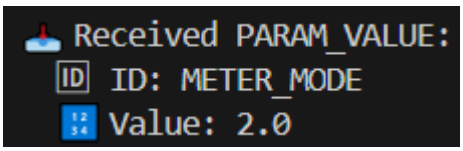
输入要设置的数值

```
Enter new value: 2
```

选择参数类型“1”

```
Parameter types:
  8 = FLOAT (most common)
  9 = DOUBLE
  4 = INT32
  1 = UINT8
Enter type number [8]: 1
```

成功返回设置值



变焦功能

开始相机的持续光学变焦

MAV_DJI_CMD_START_OPT_ZOOM

参数：

参数	类型	说明
zoomDirection	E_DjiCameraZoomDirection	变焦方向
zoomSpeed	E_DjiCameraZoomSpeed	变焦速度

zoomDirection

名称	枚举值	说明
DJI_CAMERA_ZOOM_DIRECTION_OUT	0	缩小变焦（Zoom Out）， 降低变焦倍率
DJI_CAMERA_ZOOM_DIRECTION_IN	1	放大变焦（Zoom In）， 提高变焦倍率

zoomSpeed

名称	枚举值	说明
DJI_CAMERA_ZOOM_SPEED_SLOWEST	72	最慢变焦速度
DJI_CAMERA_ZOOM_SPEED_SLOW	73	慢速变焦
DJI_CAMERA_ZOOM_SPEED_MODERATELY_SLOW	74	略慢于正常速度
DJI_CAMERA_ZOOM_SPEED_NORMAL	75	正常变焦速度
DJI_CAMERA_ZOOM_SPEED_MODERATELY_FAST	76	略快于正常速度
DJI_CAMERA_ZOOM_SPEED_FAST	77	快速变焦
DJI_CAMERA_ZOOM_SPEED_FASTEST	78	最快变焦速度

示例

```
### 模板
import time
from pymavlink import mavutil
from enum import IntEnum

# MAV_CMD 执行结果映射
MAV_RESULT_NAMES = {
    mavutil.mavlink.MAV_RESULT_ACCEPTED: "ACCEPTED",
    mavutil.mavlink.MAV_RESULT_TEMPORARILY_REJECTED: "TEMPORARILY_REJECTED",
    mavutil.mavlink.MAV_RESULT_DENIED: "DENIED",
    mavutil.mavlink.MAV_RESULT_UNSUPPORTED: "UNSUPPORTED",
    mavutil.mavlink.MAV_RESULT_FAILED: "FAILED",
    mavutil.mavlink.MAV_RESULT_IN_PROGRESS: "IN_PROGRESS",
}


def recv_command_ack(timeout=5):
    """接收并显示 command_ack 消息

    Args:
        timeout: 超时时间（秒）

    Returns:
        ack 消息对象，None 表示超时
    """
    print(f"等待 command_ack 响应 (超时 {timeout}s)...")

    ack = master.recv_match(type='COMMAND_ACK', blocking=True, timeout=timeout)

    if ack is None:
        print("错误: 命令响应超时")
        return None

    # 获取命令名称
    cmd_name = getattr(ack, 'command_name', None)
    if not cmd_name:
        # 尝试通过枚举查找命令名称
        try:
            cmd_name = mavutil.mavlink.enums.get('MAV_CMD', {}).get(ack.command, {}).name
        except:
```

```
            cmd_name = str(ack.command)

    # 获取结果名称
    result_name = MAV_RESULT_NAMES.get(ack.result, f"UNKNOWN({ack.r
esult})")

    print("=" * 50)
    print(f"命令: {cmd_name} ({ack.command})")
    print(f"结果: {result_name}")
    print(f"进度: {ack.progress * 100:.0f}%" if ack.progress > 0 els
e "")
    # print(f"参数2: {ack.result_param2}")
    print(f"系统ID: {ack.target_system}")
    print(f"组件ID: {ack.target_component}")
    print("=" * 50)

    return ack


class MavDjiCmd(IntEnum):
    MAV_CMD_NAV_LAND = 21
    MAV_DJI_CMD_TRIGGER_FFC=40
    MAV_DJI_CMD_START_RECORD_POINT_CLOUD=41
    MAV_DJI_CMD_STOP_RECORD_POINT_CLOUD=42
    MAV_DJI_CMD_RESTORE_FACTORY_SETTINGS=43
    MAV_DJI_CMD_APPLY_HIGH_POWER_SYNC=44
    MAV_DJI_CMD_APPLY_HIGH_POWER_SYNCV2=45
    MAV_DJI_CMD_START_CONFIRM_LANDING=46
    MAV_DJI_CMD_START_FORCE_LANDING=47
    MAV_DJI_CMD_EXECUTE_EMERGENCY_BRAKE_ACTION=48
    MAV_DJI_CMD_CANCEL_EMERGENCY_BRAKE_ACTION=49
    MAV_DJI_CMD_START_SLOW_ROTATE_MOTOR=50
    MAV_DJI_CMD_STOP_SLOW_ROTATE_MOTOR=51
    MAV_DJI_CMD_ARREST_FLYING=52
    MAV_DJI_CMD_CANCEL_ARREST_FLYING=53
    MAV_DJI_CMD_OUTPUT_HIGH_POWER=54
    TUNNEL_PROTOCOL_ID_INJECT_HMS_ERROR=32770

timestamp = int(time.time() * 1000)
millis = int(time.time() * 1000)

master = mavutil.mavlink_connection('udpin:0.0.0.0:14550')
print("waiting for heartbeat")
master.wait_heartbeat()
print("heartbeat received")
```

```
boot_time = time.time()

def send_command(command, param1=0, param2=0, param3=0, param4=0, p
aram5=0, param6=0, param7=0, wait_ack=True, timeout=5):
    """发送 command_long 并可选接收 command_ack

    Args:
        command: 命令ID (如 MavDjiCmd.TRIGGER_FFC)
        param1~param7: 命令参数
        wait_ack: 是否等待并显示 ACK
        timeout: ACK 超时时间(秒)

    Returns:
        ack 消息对象,None 表示超时或未等待
    """
    print(f"发送命令: {command} ({int(command)})")
    master.mav.command_long_send(
        master.target_system,
        master.target_component,
        int(command),
        0,  # confirmation
        param1, param2, param3, param4, param5, param6, param7
    )

    if wait_ack:
        return recv_command_ack(timeout=timeout)
    return None


def ready():
    send_command(55,0,1)

ready()
```

示例使用说明

运行后，成功建立连接并发送指令

```
waiting for heartbeat
heartbeat received
发送命令: 55 (55)
等待 command_ack 响应 (超时 5s)...
================================================
命令: 55 (55)
结果: ACCEPTED
```

停止相机的持续光学变焦

MAV_DJI_CMD_STOP_OPT_ZOOM

示例

```
### 模板
import time
from pymavlink import mavutil
from enum import IntEnum

# MAV_CMD 执行结果映射
MAV_RESULT_NAMES = {
    mavutil.mavlink.MAV_RESULT_ACCEPTED: "ACCEPTED",
    mavutil.mavlink.MAV_RESULT_TEMPORARILY_REJECTED: "TEMPORARILY_R
EJECTED",
    mavutil.mavlink.MAV_RESULT_DENIED: "DENIED",
    mavutil.mavlink.MAV_RESULT_UNSUPPORTED: "UNSUPPORTED",
    mavutil.mavlink.MAV_RESULT_FAILED: "FAILED",
    mavutil.mavlink.MAV_RESULT_IN_PROGRESS: "IN_PROGRESS",
}


def recv_command_ack(timeout=5):
    """接收并显示 command_ack 消息

    Args:
        timeout: 超时时间（秒）

    Returns:
        ack 消息对象，None 表示超时
    """
    print(f"等待 command_ack 响应 (超时 {timeout}s)...")

    ack = master.recv_match(type='COMMAND_ACK', blocking=True, time
out=timeout)

    if ack is None:
        print("错误: 命令响应超时")
        return None

    # 获取命令名称
    cmd_name = getattr(ack, 'command_name', None)
    if not cmd_name:
        # 尝试通过枚举查找命令名称
```

```
        try:
            cmd_name = mavutil.mavlink.enums.get('MAV_CMD', {}).get
(ack.command, {}).name
        except:
            cmd_name = str(ack.command)

    # 获取结果名称
    result_name = MAV_RESULT_NAMES.get(ack.result, f"UNKNOWN({ack.r
esult})")

    print("=" * 50)
    print(f"命令: {cmd_name} ({ack.command})")
    print(f"结果: {result_name}")
    print(f"进度: {ack.progress * 100:.0f}%" if ack.progress > 0 els
e "")
    # print(f"参数2: {ack.result_param2}")
    print(f"系统ID: {ack.target_system}")
    print(f"组件ID: {ack.target_component}")
    print("=" * 50)

    return ack


class MavDjiCmd(IntEnum):
    MAV_CMD_NAV_LAND = 21
    MAV_DJI_CMD_TRIGGER_FFC=40
    MAV_DJI_CMD_START_RECORD_POINT_CLOUD=41
    MAV_DJI_CMD_STOP_RECORD_POINT_CLOUD=42
    MAV_DJI_CMD_RESTORE_FACTORY_SETTINGS=43
    MAV_DJI_CMD_APPLY_HIGH_POWER_SYNC=44
    MAV_DJI_CMD_APPLY_HIGH_POWER_SYNCV2=45
    MAV_DJI_CMD_START_CONFIRM_LANDING=46
    MAV_DJI_CMD_START_FORCE_LANDING=47
    MAV_DJI_CMD_EXECUTE_EMERGENCY_BRAKE_ACTION=48
    MAV_DJI_CMD_CANCEL_EMERGENCY_BRAKE_ACTION=49
    MAV_DJI_CMD_START_SLOW_ROTATE_MOTOR=50
    MAV_DJI_CMD_STOP_SLOW_ROTATE_MOTOR=51
    MAV_DJI_CMD_ARREST_FLYING=52
    MAV_DJI_CMD_CANCEL_ARREST_FLYING=53
    MAV_DJI_CMD_OUTPUT_HIGH_POWER=54
    TUNNEL_PROTOCOL_ID_INJECT_HMS_ERROR=32770

timestamp = int(time.time() * 1000)
millis = int(time.time() * 1000)
```

```
master = mavutil.mavlink_connection('udpin:0.0.0.0:14550')
print("waiting for heartbeat")
master.wait_heartbeat()
print("heartbeat received")
boot_time = time.time()

def send_command(command, param1=0, param2=0, param3=0, param4=0, p
aram5=0, param6=0, param7=0, wait_ack=True, timeout=5):
    """发送 command_long 并可选接收 command_ack

    Args:
        command: 命令ID (如 MavDjiCmd.TRIGGER_FFC)
        param1~param7: 命令参数
        wait_ack: 是否等待并显示 ACK
        timeout: ACK 超时时间(秒)

    Returns:
        ack 消息对象,None 表示超时或未等待
    """
    print(f"发送命令: {command} ({int(command)})")
    master.mav.command_long_send(
        master.target_system,
        master.target_component,
        int(command),
        0,  # confirmation
        param1, param2, param3, param4, param5, param6, param7
    )

    if wait_ack:
        return recv_command_ack(timeout=timeout)
    return None


def ready():
    send_command(56,0,1)

ready()
```

示例使用说明

运行后，成功建立连接并发送指令

```
waiting for heartbeat
heartbeat received
发送命令: 56 (56)
等待 command_ack 响应 (超时 5s)...
================================================
命令: 56 (56)
结果: ACCEPTED
```

启用或禁用相机的点击变焦功能

TAP_ZOOM_ENABLED

设置

mavlink消息：

接收PARAM_SET
发送PARAM_VALUE

获取

mavlink消息

接收PARAM_REQUEST_READ
发送PARAM_VALUE

参数：

类型	枚举值	说明
bool	0	禁用 Tap-Zoom 功能
bool	1	启用 Tap-Zoom 功能

示例

```
import time
import sys
from pymavlink import mavutil

def connect_to_vehicle():
    """建立 UDP 监听连接（默认监听所有接口的 14550 端口）"""
    connection_string = "udpin:0.0.0.0:14550"
```

```
    print(f"Listening for MAVLink on: {connection_string}")
    vehicle = mavutil.mavlink_connection(connection_string)
    print("Waiting for HEARTBEAT (timeout: 10s)...")
    try:
        vehicle.wait_heartbeat(timeout=10)
        print(f"  Heartbeat received from system {vehicle.target_sy
stem}, component {vehicle.target_component}")
        return vehicle
    except Exception as e:
        print(f"  Connection failed: {str(e)}")
        print("Ensure your vehicle is sending MAVLink messages to 1
4550 port")
        print("Example QGroundControl setup: Settings > General > M
AVLink > Listening Port = 14550")
        sys.exit(1)

def format_param_id(param_id):
    """格式化16字节参数ID（符合MAVLink协议要求）"""
    if len(param_id) > 16:
        raise ValueError("param_id cannot exceed 16 characters")

    param_id_bytes = bytearray(16)
    for i in range(min(len(param_id), 16)):
        param_id_bytes[i] = ord(param_id[i])

    # 长度<16时添加NULL终止符
    if len(param_id) < 16:
        param_id_bytes[len(param_id)] = 0

    return param_id_bytes

def send_param_set(vehicle, param_id, param_value, param_type):
    """发送参数设置请求"""
    vehicle.mav.param_set_send(
        vehicle.target_system,
        vehicle.target_component,
        format_param_id(param_id),
        param_value,
        param_type
    )
    print(f"  Sent PARAM_SET for '{param_id}' = {param_value} (typ
e: {param_type})")

def send_param_request_read(vehicle, param_id=None, param_index=-
1):
```

```
    """发送参数读取请求"""
    param_id_bytes = format_param_id(param_id) if param_id else byt
earray(16)

    vehicle.mav.param_request_read_send(
        vehicle.target_system,
        vehicle.target_component,
        param_id_bytes,
        param_index
    )

    target = f"index {param_index}" if param_index >= 0 else f"ID
'{param_id}'"
    print(f"  Sent PARAM_REQUEST_READ for {target}")

def send_param_request_list(vehicle):
    """发送请求所有参数列表"""
    vehicle.mav.param_request_list_send(
        vehicle.target_system,
        vehicle.target_component
    )
    print(f"  Sent PARAM_REQUEST_LIST to system {vehicle.target_sys
tem}, component {vehicle.target_component}")

def receive_param_value(vehicle, timeout=5.0):
    """接收参数值响应"""
    print(f"  Waiting for PARAM_VALUE ({timeout}s timeout)...")
    msg = vehicle.recv_match(type='PARAM_VALUE', blocking=True, tim
eout=timeout)

    if not msg:
        print("  Timeout: No PARAM_VALUE received")
        return None

    # 解析参数ID
    param_id = msg.param_id

    # 解析参数值
    param_value = msg.param_value

    # 参数类型映射
    type_names = {
        0: "INT8", 1: "UINT8", 2: "INT16", 3: "UINT16", 4: "INT32",
        5: "UINT32", 6: "INT64", 7: "UINT64", 8: "FLOAT", 9: "DOUBL
E"
```

```
    }
    param_type = type_names.get(msg.param_type, f"UNKNOWN({msg.para
m_type})")

    print("\n  Received PARAM_VALUE:")
    print(f"    ID: {param_id}")
    print(f"    Value: {param_value}")
    print(f"    Type: {param_type}")
    print(f"    Index/Total: {msg.param_index}/{msg.param_count}")

    return {
        'id': param_id,
        'value': param_value,
        'type': param_type,
        'index': msg.param_index,
        'count': msg.param_count
    }

def receive_all_param_values(vehicle, timeout=10.0):
    """接收所有参数值，直到收到 param_count 条消息后退出循环"""
    received_params = []
    expected_count = None
    print(f"  Waiting for all PARAM_VALUE messages...")

    while True:
        msg = vehicle.recv_match(type='PARAM_VALUE', blocking=True,
timeout=timeout)

        if not msg:
            print(f"  Timeout: Received {len(received_params)} para
meters (expected {expected_count if expected_count else 'unknow
n'})")
            break

        param_id = msg.param_id
        param_value = msg.param_value

        # 参数类型映射
        type_names = {
            0: "INT8", 1: "UINT8", 2: "INT16", 3: "UINT16", 4: "INT
32",
            5: "UINT32", 6: "INT64", 7: "UINT64", 8: "FLOAT", 9: "D
OUBLE"
        }
        param_type = type_names.get(msg.param_type, f"UNKNOWN({msg.
```

```
param_type})")

        # 首次收到消息时记录总参数数量
        if expected_count is None:
            expected_count = msg.param_count
            print(f"  Total parameters to receive: {expected_coun
t}")

        print(f"  [{msg.param_index}/{expected_count}] {param_id} =
{param_value} ({param_type})")

        received_params.append({
            'id': param_id,
            'value': param_value,
            'type': param_type,
            'index': msg.param_index,
            'count': msg.param_count
        })

        # 当已接收数量达到预期总数时退出循环
        if len(received_params) >= expected_count:
            print(f"\n  Successfully received all {expected_count}
parameters")
            break

    return received_params

def main():
    print("  Starting MAVLink PARAM_SET Tool (UDP Listening Mode)")
    print("------------------------------------------------------")
    print("This script listens for MAVLink on ALL interfaces:1455
0")
    print("Ensure your vehicle is configured to send to this port")
    print("------------------------------------------------------
\n")

    vehicle = connect_to_vehicle()

    # 操作选择
    print("\nAvailable operations:")
    print("1. Set parameter value")
    print("2. Read parameter value (by ID or index)")
    print("3. Request all parameters (PARAM_REQUEST_LIST)")
    choice = input("\nSelect operation [1-3]: ").strip()
```

```
    if choice == '1':
        param_id = input("Enter parameter ID to set: ").strip()
        param_value = float(input("Enter new value: ").strip())

        print("\nParameter types:")
        print("  8 = FLOAT (most common)")
        print("  9 = DOUBLE")
        print("  4 = INT32")
        print("  1 = UINT8")
        param_type = int(input("Enter type number [8]: ").strip() o
r 8)

        send_param_set(vehicle, param_id, param_value, param_type)
        receive_param_value(vehicle)

    elif choice == '2':
        # mode = input("Read by [id/index]: ").lower()
        mode = 'id'
        if mode == 'id':
            param_id = input("Enter parameter ID: ").strip()
            send_param_request_read(vehicle, param_id=param_id)
            receive_param_value(vehicle)
        elif mode == 'index':
            param_index = int(input("Enter parameter index: "))
            send_param_request_read(vehicle, param_index=param_inde
x)
            receive_param_value(vehicle)
        else:
            print("Invalid mode. Use 'id' or 'index'")

    elif choice == '3':
        print("\n  Requesting all parameters...")
        send_param_request_list(vehicle)
        all_params = receive_all_param_values(vehicle)
        print(f"\n  Total parameters received: {len(all_params)}")

    else:
        print("Invalid choice. Select 1, 2, or 3.")


if __name__ == "__main__":
    main()
```

示例使用说明

获取

运行后，成功建立连接

```
🚀 Starting MAVLink PARAM_SET Tool (UDP Listening Mode)
------------------------------------------------------
This script listens for MAVLink on ALL interfaces:14550
Ensure your vehicle is configured to send to this port
------------------------------------------------------

Listening for MAVLink on: udpin:0.0.0.0:14550
Waiting for HEARTBEAT (timeout: 10s)...
✅ Heartbeat received from system 3, component 0

Available operations:
1. Set parameter value
2. Read parameter value (by ID or index)
3. Request all parameters (PARAM_REQUEST_LIST)
```

输入“2”选择读取参数

```
Select operation [1-3]: 2
```

输入对应参数“TAP_ZOOM_ENABLED”

```
Enter parameter ID: TAP_ZOOM_ENABLED
```

成功返回该参数对应的数值

```
📥 Received PARAM_VALUE:
  🆔 ID: TAP_ZOOM_ENABLED
  🔢 Value: 1.0
```

设置

运行后，成功建立连接

```
🚀 Starting MAVLink PARAM_SET Tool (UDP Listening Mode)
--------------------------------------------------
This script listens for MAVLink on ALL interfaces:14550
Ensure your vehicle is configured to send to this port
--------------------------------------------------

Listening for MAVLink on: udpin:0.0.0.0:14550
Waiting for HEARTBEAT (timeout: 10s)...
✅ Heartbeat received from system 3, component 0

Available operations:
1. Set parameter value
2. Read parameter value (by ID or index)
3. Request all parameters (PARAM_REQUEST_LIST)
```

输入“1”选择设置参数

```
Available operations:
1. Set parameter value
2. Read parameter value (by ID or index)
3. Request all parameters (PARAM_REQUEST_LIST)

Select operation [1-3]: 1
```

输入对应参数“TAP_ZOOM_ENABLED”

```
Enter parameter ID to set: TAP_ZOOM_ENABLED
```

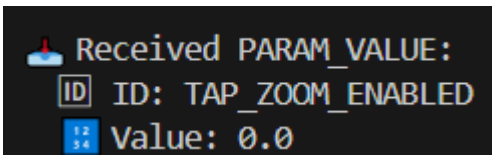
输入要设置的数值

```
Enter new value: 0
```

选择参数类型“1”

```
Parameter types:
  8 = FLOAT (most common)
  9 = DOUBLE
  4 = INT32
  1 = UINT8
Enter type number [8]: 1
```

成功返回设置值



点击变焦倍数

TAP_ZOOM_MULT

设置

mavlink消息：

接收PARAM_SET
发送PARAM_VALUE

获取

mavlink消息

接收PARAM_REQUEST_READ
发送PARAM_VALUE

参数：

参数	类型	取值范围	说明
tapZoomMultiplier	uint8_t	[1, 5]	Tap-Zoom倍率系数。最终变焦倍率 = 当前Zoom Scale × 该系数；当值为1时，Tap-Zoom不改变变焦倍率

示例

```
import time
import sys
from pymavlink import mavutil

def connect_to_vehicle():
    """建立 UDP 监听连接（默认监听所有接口的 14550 端口）"""
    connection_string = "udpin:0.0.0.0:14550"
    print(f"Listening for MAVLink on: {connection_string}")
    vehicle = mavutil.mavlink_connection(connection_string)
    print("Waiting for HEARTBEAT (timeout: 10s)...")
    try:
```

```
        vehicle.wait_heartbeat(timeout=10)
        print(f"  Heartbeat received from system {vehicle.target_system}, component {vehicle.target_component}")
        return vehicle
    except Exception as e:
        print(f"  Connection failed: {str(e)}")
        print("Ensure your vehicle is sending MAVLink messages to 14550 port")
        print("Example QGroundControl setup: Settings > General > MAVLink > Listening Port = 14550")
        sys.exit(1)

def format_param_id(param_id):
    """格式化16字节参数ID（符合MAVLink协议要求）"""
    if len(param_id) > 16:
        raise ValueError("param_id cannot exceed 16 characters")

    param_id_bytes = bytearray(16)
    for i in range(min(len(param_id), 16)):
        param_id_bytes[i] = ord(param_id[i])

    # 长度<16时添加NULL终止符
    if len(param_id) < 16:
        param_id_bytes[len(param_id)] = 0

    return param_id_bytes

def send_param_set(vehicle, param_id, param_value, param_type):
    """发送参数设置请求"""
    vehicle.mav.param_set_send(
        vehicle.target_system,
        vehicle.target_component,
        format_param_id(param_id),
        param_value,
        param_type
    )
    print(f"  Sent PARAM_SET for '{param_id}' = {param_value} (type: {param_type})")

def send_param_request_read(vehicle, param_id=None, param_index=-1):
    """发送参数读取请求"""
    param_id_bytes = format_param_id(param_id) if param_id else bytearray(16)
```

```
    vehicle.mav.param_request_read_send(
        vehicle.target_system,
        vehicle.target_component,
        param_id_bytes,
        param_index
    )

    target = f"index {param_index}" if param_index >= 0 else f"ID
'{param_id}'"
    print(f"  Sent PARAM_REQUEST_READ for {target}")

def send_param_request_list(vehicle):
    """发送请求所有参数列表"""
    vehicle.mav.param_request_list_send(
        vehicle.target_system,
        vehicle.target_component
    )
    print(f"  Sent PARAM_REQUEST_LIST to system {vehicle.target_sys
tem}, component {vehicle.target_component}")

def receive_param_value(vehicle, timeout=5.0):
    """接收参数值响应"""
    print(f"  Waiting for PARAM_VALUE ({timeout}s timeout)...")
    msg = vehicle.recv_match(type='PARAM_VALUE', blocking=True, tim
eout=timeout)

    if not msg:
        print("  Timeout: No PARAM_VALUE received")
        return None

    # 解析参数ID
    param_id = msg.param_id

    # 解析参数值
    param_value = msg.param_value

    # 参数类型映射
    type_names = {
        0: "INT8", 1: "UINT8", 2: "INT16", 3: "UINT16", 4: "INT32",
        5: "UINT32", 6: "INT64", 7: "UINT64", 8: "FLOAT", 9: "DOUBL
E"
    }
    param_type = type_names.get(msg.param_type, f"UNKNOWN({msg.para
m_type})")
```

```
    print("\n  Received PARAM_VALUE:")
    print(f"    ID: {param_id}")
    print(f"    Value: {param_value}")
    print(f"    Type: {param_type}")
    print(f"    Index/Total: {msg.param_index}/{msg.param_count}")

    return {
        'id': param_id,
        'value': param_value,
        'type': param_type,
        'index': msg.param_index,
        'count': msg.param_count
    }

def receive_all_param_values(vehicle, timeout=10.0):
    """接收所有参数值，直到收到 param_count 条消息后退出循环"""
    received_params = []
    expected_count = None
    print(f"  Waiting for all PARAM_VALUE messages...")

    while True:
        msg = vehicle.recv_match(type='PARAM_VALUE', blocking=True,
timeout=timeout)

        if not msg:
            print(f"  Timeout: Received {len(received_params)} para
meters (expected {expected_count if expected_count else 'unknow
n'})")
            break

        param_id = msg.param_id
        param_value = msg.param_value

        # 参数类型映射
        type_names = {
            0: "INT8", 1: "UINT8", 2: "INT16", 3: "UINT16", 4: "INT
32",
            5: "UINT32", 6: "INT64", 7: "UINT64", 8: "FLOAT", 9: "D
OUBLE"
        }
        param_type = type_names.get(msg.param_type, f"UNKNOWN({msg.
param_type})")

        # 首次收到消息时记录总参数数量
        if expected_count is None:
```

```
            expected_count = msg.param_count
            print(f"  Total parameters to receive: {expected_coun
t}")

        print(f"  [{msg.param_index}/{expected_count}] {param_id} =
{param_value} ({param_type})")

        received_params.append({
            'id': param_id,
            'value': param_value,
            'type': param_type,
            'index': msg.param_index,
            'count': msg.param_count
        })

        # 当已接收数量达到预期总数时退出循环
        if len(received_params) >= expected_count:
            print(f"\n  Successfully received all {expected_count}
parameters")
            break

    return received_params

def main():
    print("  Starting MAVLink PARAM_SET Tool (UDP Listening Mode)")
    print("------------------------------------------------------")
    print("This script listens for MAVLink on ALL interfaces:1455
0")
    print("Ensure your vehicle is configured to send to this port")
    print("------------------------------------------------------
\n")

    vehicle = connect_to_vehicle()

    # 操作选择
    print("\nAvailable operations:")
    print("1. Set parameter value")
    print("2. Read parameter value (by ID or index)")
    print("3. Request all parameters (PARAM_REQUEST_LIST)")
    choice = input("\nSelect operation [1-3]: ").strip()

    if choice == '1':
        param_id = input("Enter parameter ID to set: ").strip()
        param_value = float(input("Enter new value: ").strip())
```

```
        print("\nParameter types:")
        print("  8 = FLOAT (most common)")
        print("  9 = DOUBLE")
        print("  4 = INT32")
        print("  1 = UINT8")
        param_type = int(input("Enter type number [8]: ").strip() o
r 8)

        send_param_set(vehicle, param_id, param_value, param_type)
        receive_param_value(vehicle)

    elif choice == '2':
        # mode = input("Read by [id/index]: ").lower()
        mode = 'id'
        if mode == 'id':
            param_id = input("Enter parameter ID: ").strip()
            send_param_request_read(vehicle, param_id=param_id)
            receive_param_value(vehicle)
        elif mode == 'index':
            param_index = int(input("Enter parameter index: "))
            send_param_request_read(vehicle, param_index=param_inde
x)
            receive_param_value(vehicle)
        else:
            print("Invalid mode. Use 'id' or 'index'")

    elif choice == '3':
        print("\n  Requesting all parameters...")
        send_param_request_list(vehicle)
        all_params = receive_all_param_values(vehicle)
        print(f"\n  Total parameters received: {len(all_params)}")

    else:
        print("Invalid choice. Select 1, 2, or 3.")


if __name__ == "__main__":
    main()
```

示例使用说明

获取

运行后，成功建立连接

```
🚀 Starting MAVLink PARAM_SET Tool (UDP Listening Mode)
--------------------------------------------------
This script listens for MAVLink on ALL interfaces:14550
Ensure your vehicle is configured to send to this port
--------------------------------------------------

Listening for MAVLink on: udpin:0.0.0.0:14550
Waiting for HEARTBEAT (timeout: 10s)...
✅ Heartbeat received from system 3, component 0

Available operations:
1. Set parameter value
2. Read parameter value (by ID or index)
3. Request all parameters (PARAM_REQUEST_LIST)
```

输入“2”选择读取参数

```
Select operation [1-3]: 2
```

输入对应参数“TAP_ZOOM_MULT”

```
Enter parameter ID: TAP_ZOOM_MULT
```

成功返回该参数对应的数值

```
📥 Received PARAM_VALUE:
  🆔 ID: TAP_ZOOM_MULT
  🔢 Value: 1.0
```

设置

运行后，成功建立连接

```
🚀 Starting MAVLink PARAM_SET Tool (UDP Listening Mode)
--------------------------------------------------
This script listens for MAVLink on ALL interfaces:14550
Ensure your vehicle is configured to send to this port
--------------------------------------------------

Listening for MAVLink on: udpin:0.0.0.0:14550
Waiting for HEARTBEAT (timeout: 10s)...
✅ Heartbeat received from system 3, component 0

Available operations:
1. Set parameter value
2. Read parameter value (by ID or index)
3. Request all parameters (PARAM_REQUEST_LIST)
```

输入“1”选择设置参数

```
Available operations:
1. Set parameter value
2. Read parameter value (by ID or index)
3. Request all parameters (PARAM_REQUEST_LIST)

Select operation [1-3]: 1
```

输入对应参数“TAP_ZOOM_MULT”

```
Enter parameter ID to set: TAP_ZOOM_MULT
```

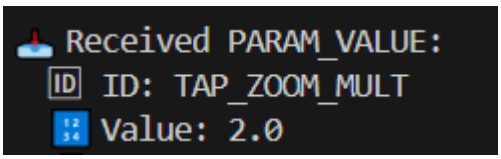
输入要设置的数值

```
Enter new value: 2
```

选择参数类型“1”

```
Parameter types:
  8 = FLOAT (most common)
  9 = DOUBLE
  4 = INT32
  1 = UINT8
Enter type number [8]: 1
```

成功返回设置值



获取相机焦距环的值范围

FOC_RING_RNG

设置

mavlink消息：

接收PARAM_EXT_SET
发送PARAM_EXT_ACK

获取

mavlink消息：

接收PARAM_EXT_REQUEST_READ
发送PARAM_EXT_VALUE

参数：

字段	类型	说明
size	uint8_t	有效数组长度（最大16）
value	union	可选取值集合（同一时刻仅使用其中一种类型）
minValue	uint32_t	数值范围最小值（适用于数值型参数）
maxValue	uint32_t	数值范围最大值（适用于数值型参数）

value（联合体说明）

该字段为联合体，同一时间仅有一个数组有效：

成员	类型	说明
photoStorageFormat	E_DjiCameraManagerPhotoStorageFormat[16]	照片存储格式枚
videoStorageFormat	E_DjiCameraManagerVideoStorageFormat[16]	视频存储格式枚

成员	类型	说明
photoRatioFormat	E_DjiCameraManagerPhotoRatio[16]	照片比例列表
streamSource	E_DjiCameraManagerStreamSource[16]	视频流源列表
streamStorage	E_DjiCameraManagerStreamStorage[16]	视频存储介质列
nightSceneMode	E_DjiCameraManagerNightSceneMode[16]	夜景模式列表
meteringMode	E_DjiCameraManagerMeteringMode[16]	测光模式列表

示例

```
import time
import sys
import struct
import ctypes
from pymavlink import mavutil

def connect_to_vehicle():
    """建立 UDP 监听连接（默认监听所有接口的 14550 端口）"""
    connection_string = "udpin:0.0.0.0:14550"
    print(f"Listening for MAVLink on: {connection_string}")
    vehicle = mavutil.mavlink_connection(connection_string)
    print("Waiting for HEARTBEAT (timeout: 10s)...")
    try:
        vehicle.wait_heartbeat(timeout=10)
        print(f"  Heartbeat received from system {vehicle.target_system}, component {vehicle.target_component}")
        return vehicle
    except Exception as e:
        print(f"  Connection failed: {str(e)}")
        print("Ensure your vehicle is sending MAVLink messages to 14550 port")
        print("Example QGroundControl setup: Settings > General > MAVLink > Listening Port = 14550")
        sys.exit(1)

def format_param_id(param_id):
    """格式化16字节参数ID（符合MAVLink协议要求）"""
    if len(param_id) > 16:
        raise ValueError("param_id cannot exceed 16 characters")
```

```
    param_id_bytes = bytearray(16)
    for i in range(min(len(param_id), 16)):
        param_id_bytes[i] = ord(param_id[i])

    # 长度<16时添加NULL终止符
    if len(param_id) < 16:
        param_id_bytes[len(param_id)] = 0

    return param_id_bytes

# ============ 结构体打包函数 ============

def pack_uint8_xy(x, y):
    """打包 {uint8_t x; uint8_t y} 结构体 (2字节)"""
    return struct.pack('<BB', x, y)

def pack_float32_xy(x, y):
    """打包 {float32 x; float32 y} 结构体 (8字节)"""
    return struct.pack('<ff', x, y)

def pack_float32_lxlyrxry(LX, LY, RX, RY):
    """打包 {float32 LX; float32 LY; float32 RX; float32 RY;} 结构体
(16字节)"""
    return struct.pack('<ffff', LX, LY, RX, RY)

def pack_ering_mode(size, eringMode, minValue, maxValue):
    """打包 {uint8_t size; uint8_t eringMode[16]; uint32_t minValue;
uint32_t maxValue;} 结构体 (25字节)

    Args:
        size: uint8_t, eringMode数组的有效长度
        eringMode: list of int (0-255), 16个uint8_t值
        minValue: uint32_t, 最小值
        maxValue: uint32_t, 最大值

    Returns:
        bytes: 打包后的二进制数据
    """
    # 确保eringMode是16字节
    eringMode_bytes = bytearray(16)
    for i in range(min(len(eringMode), 16)):
        eringMode_bytes[i] = eringMode[i] & 0xFF

    return struct.pack('<B16sII', size, eringMode_bytes, minValue,
maxValue)
```

```
def format_param_value(param_value):
    """格式化128字节参数值（接受二进制输入）"""
    if isinstance(param_value, str):
        # 如果是十六进制字符串（如 "48656c6c6f"），先转换为字节
        if param_value.startswith('0x') or all(c in '0123456789abcd
efABCDEF' for c in param_value.replace(' ', '')):
            hex_str = param_value.replace('0x', '').replace(' ',
'')
            if len(hex_str) % 2 == 0:
                param_value = bytes.fromhex(hex_str)
            else:
                raise ValueError("Invalid hex string: must have eve
n number of characters")
        else:
            # 普通字符串转字节
            param_value = param_value.encode('utf-8')

    if len(param_value) > 128:
        raise ValueError("param_value cannot exceed 128 bytes")

    param_value_bytes = bytearray(128)
    param_value_bytes[:len(param_value)] = param_value

    return param_value_bytes

def hex_dump(data, bytes_per_line=16):
    """格式化二进制数据为 hex + ASCII 展示（类 hexdump 风格）

    Returns:
        str: 格式化后的多行字符串
    """
    lines = []
    for i in range(0, len(data), bytes_per_line):
        chunk = data[i:i + bytes_per_line]
        hex_part = ' '.join(f'{b:02X}' for b in chunk)
        ascii_part = ''.join(chr(b) if 32 <= b < 127 else '.' for b
in chunk)
        lines.append(f"  {i:04X}  {hex_part:<{bytes_per_line*3}}
{ascii_part}")
    return '\n'.join(lines)

def send_param_ext_request_read(vehicle, param_id=None, param_index
=-1):
    """发送参数读取请求"""
```

```
    param_id_bytes = format_param_id(param_id) if param_id else byt
earray(16)

    vehicle.mav.param_ext_request_read_send(
        vehicle.target_system,
        vehicle.target_component,
        param_id_bytes,
        param_index
    )

    target = f"index {param_index}" if param_index >= 0 else f"ID
'{param_id}'"
    print(f"  Sent PARAM_EXT_REQUEST_READ for {target}")

def send_param_ext_request_list(vehicle):
    """发送请求所有参数列表"""
    vehicle.mav.param_ext_request_list_send(
        vehicle.target_system,
        vehicle.target_component
    )
    print(f"  Sent PARAM_EXT_REQUEST_LIST to system {vehicle.target
_system}, component {vehicle.target_component}")

def send_param_ext_set(vehicle, param_id, param_value, param_type):
    """发送参数设置请求"""
    vehicle.mav.param_ext_set_send(
        vehicle.target_system,
        vehicle.target_component,
        format_param_id(param_id),
        format_param_value(param_value),
        param_type
    )
    print(f"  Sent PARAM_EXT_SET for '{param_id}' = '{param_value}'
(type: {param_type})")

def receive_param_ext_value(vehicle, timeout=5.0):
    """接收参数值响应"""
    print(f"  Waiting for PARAM_EXT_VALUE ({timeout}s timeout)...")
    msg = vehicle.recv_match(type='PARAM_EXT_VALUE', blocking=True,
timeout=timeout)

    if not msg:
        print("  Timeout: No PARAM_EXT_VALUE received")
        return None
```

```
    # 解析参数ID
    param_id = msg.param_id

    # 解析参数值（直接获取128字节原始数据）
    param_value_raw = msg.param_value
    # byte_pointer = (ctypes.c_ubyte * 128).from_buffer_copy(msg.pa
ram_value)
    if isinstance(param_value_raw, (bytes, bytearray)):
        param_value_bytes = bytes(param_value_raw)
    # param_value_bytes = bytes(byte_pointer)
    elif isinstance(param_value_raw, str):
        # 替换无法编码的字符（如 \ufffd）为 0x00，还原为128字节
        param_value_bytes = param_value_raw.encode('latin1', errors
='replace')
    else:
        param_value_bytes = bytes(param_value_raw)

    # 参数类型映射
    type_names = {
        0: "INT8", 1: "UINT8", 2: "INT16", 3: "UINT16", 4: "INT32",
        5: "UINT32", 6: "INT64", 7: "UINT64", 8: "FLOAT", 9: "FLOA
T",
        10: "CHAR", 11: "CHAR", 12: "UINT8_ENUM", 13: "STRING"
    }
    param_type = type_names.get(msg.param_type, f"UNKNOWN({msg.para
m_type})")

    print("\n  Received PARAM_EXT_VALUE:")
    print(f"    ID: {param_id}")
    print(f"    Type: {param_type}  |  Data: {len(param_value_byte
s)} bytes")
    print(f"    Index/Total: {msg.param_index}/{msg.param_count}")
    print(hex_dump(param_value_bytes))

    return {
        'id': param_id,
        'value': param_value_bytes,
        'type': param_type,
        'index': msg.param_index,
        'count': msg.param_count
    }


def receive_all_param_ext_values(vehicle, timeout=10.0):
    """接收所有参数值，直到收到 param_count 条消息后退出循环"""
```

```
    received_params = []
    expected_count = None
    print(f"  Waiting for all PARAM_EXT_VALUE messages...")

    while True:
        msg = vehicle.recv_match(type='PARAM_EXT_VALUE', blocking=True, timeout=timeout)

        if not msg:
            print(f"  Timeout: Received {len(received_params)} parameters (expected {expected_count if expected_count else 'unknown'})")
            break

        param_id = msg.param_id
        param_value_raw = msg.param_value

        # 直接获取128字节原始数据
        if isinstance(param_value_raw, (bytes, bytearray)):
            param_value_bytes = bytes(param_value_raw)
        elif isinstance(param_value_raw, str):
            # 替换无法编码的字符（如 \ufffd）为 0x00
            param_value_bytes = param_value_raw.encode('latin1', errors='replace')
        else:
            param_value_bytes = bytes(param_value_raw)

        # 参数类型映射
        type_names = {
            0: "INT8", 1: "UINT8", 2: "INT16", 3: "UINT16", 4: "INT32",
            5: "UINT32", 6: "INT64", 7: "UINT64", 8: "FLOAT", 9: "DOUBLE",
            10: "CHAR", 11: "BOOL", 12: "UINT8_ENUM", 13: "STRING"
        }
        param_type = type_names.get(msg.param_type, f"UNKNOWN({msg.param_type})")

        # 首次收到消息时记录总参数数量
        if expected_count is None:
            expected_count = msg.param_count
            print(f"  Total parameters to receive: {expected_count}")

        print(f"  [{msg.param_index}/{expected_count}] {param_id}
```

```
[{param_type}][{len(param_value_bytes)}B]")
        print(hex_dump(param_value_bytes))

        received_params.append({
            'id': param_id,
            'value': param_value_bytes,
            'type': param_type,
            'index': msg.param_index,
            'count': msg.param_count
        })

        # 当已接收数量达到预期总数时退出循环
        if len(received_params) >= expected_count:
            print(f"\n  Successfully received all {expected_count}
parameters")
            break

    return received_params

def receive_param_ext_ack(vehicle, timeout=5.0):
    """接收参数设置确认"""
    print(f"  Waiting for PARAM_EXT_ACK ({timeout}s timeout)...")
    msg = vehicle.recv_match(type='PARAM_EXT_ACK', blocking=True, t
imeout=timeout)

    if not msg:
        print("  Timeout: No PARAM_EXT_ACK received")
        return None

    param_id = msg.param_id
    param_value = msg.param_value

    # 结果代码映射
    result_codes = {
        0: "  PARAM_ACK_ACCEPTED",
        1: "⚠ PARAM_ACK_VALUE_UNSUPPORTED",
        2: "  PARAM_ACK_FAILED",
        3: "  PARAM_ACK_IN_PROGRESS"
    }
    result = result_codes.get(msg.param_result, f"  UNKNOWN({msg.pa
ram_result})")

    print("\n  Received PARAM_EXT_ACK:")
    print(f"    ID: {param_id}")
    print(f"    Value: {param_value}")
```

```
    print(f"    Result: {result}")

    return {
        'id': param_id,
        'value': param_value,
        'result': result,
        'result_code': msg.param_result
    }

def main():
    print("  Starting MAVLink PARAM_EXT Tool (UDP Listening Mode)")
    print("------------------------------------------------------")
    print("This script listens for MAVLink on ALL interfaces:1455
0")
    print("Ensure your vehicle is configured to send to this port")
    print("------------------------------------------------------
\n")

    vehicle = connect_to_vehicle()

    # 操作选择
    print("\nAvailable operations:")
    print("1. Read parameter value (by ID or index)")
    print("2. Set parameter value (raw/hex/string)")
    print("3. Request all parameters (PARAM_EXT_REQUEST_LIST)")
    print("4. Set struct: uint8_xy {x, y}")
    print("5. Set struct: float32_xy {x, y}")
    print("6. Set struct: float32_lxlyrxry {LX, LY, RX, RY}")
    print("7. Set struct: ering_mode {size, eringMode[16], minValu
e, maxValue}")
    choice = input("\nSelect operation [1-7]: ").strip()

    if choice == '1':
        # mode = input("Read by [id/index]: ").lower()
        mode = 'id'
        if mode == 'id':
            param_id = input("Enter parameter ID: ").strip()
            send_param_ext_request_read(vehicle, param_id=param_id)
            receive_param_ext_value(vehicle)
        elif mode == 'index':
            param_index = int(input("Enter parameter index: "))
            send_param_ext_request_read(vehicle, param_index=param_
index)
            receive_param_ext_value(vehicle)
        else:
```

```
            print("Invalid mode. Use 'id' or 'index'")

    elif choice == '2':
        param_id = input("Enter parameter ID to set: ").strip()

        print("\nEnter parameter value (supports):")
        print("  - Plain string (e.g., hello)")
        print("  - Hex string (e.g., 48656c6c6f or 0x48656c6c6f)")
        print("  - Raw bytes (e.g., b'\\x48\\x65\\x6c\\x6c\\x6f')")
        param_value_input = input("Enter value: ").strip()

        # 解析输入值
        if param_value_input.startswith("b'") or param_value_input.
startswith('b"'):
            # Python字节字面量格式
            param_value = eval(param_value_input)
        elif param_value_input.startswith('0x') or all(c in '012345
6789abcdefABCDEF ' for c in param_value_input):
            # 十六进制格式
            hex_str = param_value_input.replace('0x', '').replace('
', '')
            try:
                param_value = bytes.fromhex(hex_str)
            except ValueError:
                param_value = param_value_input.encode('utf-8')
        else:
            # 默认作为字符串处理
            param_value = param_value_input.encode('utf-8')

        print("\nParameter types:")
        print("  8 = FLOAT (most common)")
        print("  13 = STRING")
        print("  10 = CHAR (raw bytes)")
        print("  Other types require specific binary formats")
        param_type = int(input("Enter type number [8]: ") or 8)

        send_param_ext_set(vehicle, param_id, param_value, param_ty
pe)
        receive_param_ext_ack(vehicle)

    elif choice == '4':
        """设置 uint8_xy 结构体 {uint8_t x; uint8_t y}"""
        param_id = input("Enter parameter ID: ").strip()
        x = int(input("Enter x (0-255): "))
        y = int(input("Enter y (0-255): "))
```

```
        param_value = pack_uint8_xy(x, y)
        print(f"  Packed: {param_value.hex()} ({len(param_value)} bytes)")
        send_param_ext_set(vehicle, param_id, param_value, 10)  # CHAR type
        receive_param_ext_ack(vehicle)

    elif choice == '5':
        """设置 float32_xy 结构体 {float32 x; float32 y}"""
        param_id = input("Enter parameter ID: ").strip()
        x = float(input("Enter x: "))
        y = float(input("Enter y: "))
        param_value = pack_float32_xy(x, y)
        print(f"  Packed: {param_value.hex()} ({len(param_value)} bytes)")
        send_param_ext_set(vehicle, param_id, param_value, 10)  # CHAR type
        receive_param_ext_ack(vehicle)

    elif choice == '6':
        """设置 float32_lxlyrxry 结构体 {float32 LX; float32 LY; float32 RX; float32 RY;}"""
        param_id = input("Enter parameter ID: ").strip()
        LX = float(input("Enter LX: "))
        LY = float(input("Enter LY: "))
        RX = float(input("Enter RX: "))
        RY = float(input("Enter RY: "))
        param_value = pack_float32_lxlyrxry(LX, LY, RX, RY)
        print(f"  Packed: {param_value.hex()} ({len(param_value)} bytes)")
        send_param_ext_set(vehicle, param_id, param_value, 10)  # CHAR type
        receive_param_ext_ack(vehicle)

    elif choice == '7':
        """设置 ering_mode 结构体 {uint8_t size; uint8_t eringMode[16]; uint32_t minValue; uint32_t maxValue;}"""
        param_id = input("Enter parameter ID: ").strip()
        size = int(input("Enter size (0-16): "))
        eringMode = []
        print("Enter 16 eringMode values (0-255), one per line:")
        for i in range(16):
            val = int(input(f"  eringMode[{i}]: "))
            eringMode.append(val)
        minValue = int(input("Enter minValue: "))
```

```
        maxValue = int(input("Enter maxValue: "))
        param_value = pack_ering_mode(size, eringMode, minValue, ma
xValue)
        print(f"  Packed: {param_value.hex()} ({len(param_value)} b
ytes)")
        send_param_ext_set(vehicle, param_id, param_value, 10)  # C
HAR type
        receive_param_ext_ack(vehicle)

    elif choice == '3':
        print("\n  Requesting all parameters...")
        send_param_ext_request_list(vehicle)
        all_params = receive_all_param_ext_values(vehicle)
        print(f"\n  Total parameters received: {len(all_params)}")

    else:
        print("Invalid choice. Select 1, 2, or 3.")


if __name__ == "__main__":
    main()
```

运行后，成功建立连接

```
🚀 Starting MAVLink PARAM_EXT Tool (UDP Listening Mode)
------------------------------------------------------
This script listens for MAVLink on ALL interfaces:14550
Ensure your vehicle is configured to send to this port
------------------------------------------------------

Listening for MAVLink on: udpin:0.0.0.0:14550
Waiting for HEARTBEAT (timeout: 10s)...
✅ Heartbeat received from system 3, component 0
```

输入“1”选择设置参数

```
Available operations:
1. Read parameter value (by ID or index)
2. Set parameter value (raw/hex/string)
3. Request all parameters (PARAM_EXT_REQUEST_LIST)
4. Set struct: uint8_xy {x, y}
5. Set struct: float32_xy {x, y}
6. Set struct: float32_lxlyrxry {LX, LY, RX, RY}
7. Set struct: ering_mode {size, eringMode[16], minValue, maxValue}

Select operation [1-7]: 1
```

输入对应参数“FOC_RING_RNG”

```
Enter parameter ID: FOC_RING_RNG
```

成功返回完整消息

```
📩 Received PARAM_EXT_VALUE:
  🆔 ID: FOC_RING_RNG
fd950000000301420100120008004 64f435f52494e475f524e470000000002aa7d3fffff000040aa7d3fffff0000d8ffffff80ffffff0000000
0000000007800000005000000 6821bdb7aaaa0000800e0034ffff0000f00c0034ffff000050aa7d3f0000000062000000000000000000000000
0000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000b9e9c
```

相机焦距环

FOCUS_RING_VALUE

设置

mavlink消息：

接收PARAM_SET
发送PARAM_VALUE

获取

mavlink消息

接收PARAM_REQUEST_READ
发送PARAM_VALUE

参数：

字段	类型	说明
value	uint16_t	对焦环数值

示例

```
import time
import sys
from pymavlink import mavutil

def connect_to_vehicle():
    """建立 UDP 监听连接（默认监听所有接口的 14550 端口）"""
    connection_string = "udpin:0.0.0.0:14550"
    print(f"Listening for MAVLink on: {connection_string}")
    vehicle = mavutil.mavlink_connection(connection_string)
    print("Waiting for HEARTBEAT (timeout: 10s)...")
    try:
        vehicle.wait_heartbeat(timeout=10)
        print(f"  Heartbeat received from system {vehicle.target_sy
stem}, component {vehicle.target_component}")
        return vehicle
    except Exception as e:
        print(f"  Connection failed: {str(e)}")
        print("Ensure your vehicle is sending MAVLink messages to 1
4550 port")
        print("Example QGroundControl setup: Settings > General > M
AVLink > Listening Port = 14550")
        sys.exit(1)

def format_param_id(param_id):
    """格式化16字节参数ID（符合MAVLink协议要求）"""
    if len(param_id) > 16:
        raise ValueError("param_id cannot exceed 16 characters")

    param_id_bytes = bytearray(16)
    for i in range(min(len(param_id), 16)):
        param_id_bytes[i] = ord(param_id[i])

    # 长度<16时添加NULL终止符
    if len(param_id) < 16:
        param_id_bytes[len(param_id)] = 0

    return param_id_bytes

def send_param_set(vehicle, param_id, param_value, param_type):
    """发送参数设置请求"""
    vehicle.mav.param_set_send(
        vehicle.target_system,
        vehicle.target_component,
```

```
        format_param_id(param_id),
        param_value,
        param_type
    )
    print(f"  Sent PARAM_SET for '{param_id}' = {param_value} (typ
e: {param_type})")

def send_param_request_read(vehicle, param_id=None, param_index=-
1):
    """发送参数读取请求"""
    param_id_bytes = format_param_id(param_id) if param_id else byt
earray(16)

    vehicle.mav.param_request_read_send(
        vehicle.target_system,
        vehicle.target_component,
        param_id_bytes,
        param_index
    )

    target = f"index {param_index}" if param_index >= 0 else f"ID
'{param_id}'"
    print(f"  Sent PARAM_REQUEST_READ for {target}")

def send_param_request_list(vehicle):
    """发送请求所有参数列表"""
    vehicle.mav.param_request_list_send(
        vehicle.target_system,
        vehicle.target_component
    )
    print(f"  Sent PARAM_REQUEST_LIST to system {vehicle.target_sys
tem}, component {vehicle.target_component}")

def receive_param_value(vehicle, timeout=5.0):
    """接收参数值响应"""
    print(f"  Waiting for PARAM_VALUE ({timeout}s timeout)...")
    msg = vehicle.recv_match(type='PARAM_VALUE', blocking=True, tim
eout=timeout)

    if not msg:
        print("  Timeout: No PARAM_VALUE received")
        return None

    # 解析参数ID
    param_id = msg.param_id
```

```
    # 解析参数值
    param_value = msg.param_value

    # 参数类型映射
    type_names = {
        0: "INT8", 1: "UINT8", 2: "INT16", 3: "UINT16", 4: "INT32",
        5: "UINT32", 6: "INT64", 7: "UINT64", 8: "FLOAT", 9: "DOUBL
E"
    }
    param_type = type_names.get(msg.param_type, f"UNKNOWN({msg.para
m_type})")

    print("\n  Received PARAM_VALUE:")
    print(f"    ID: {param_id}")
    print(f"    Value: {param_value}")
    print(f"    Type: {param_type}")
    print(f"    Index/Total: {msg.param_index}/{msg.param_count}")

    return {
        'id': param_id,
        'value': param_value,
        'type': param_type,
        'index': msg.param_index,
        'count': msg.param_count
    }

def receive_all_param_values(vehicle, timeout=10.0):
    """接收所有参数值，直到收到 param_count 条消息后退出循环"""
    received_params = []
    expected_count = None
    print(f"  Waiting for all PARAM_VALUE messages...")

    while True:
        msg = vehicle.recv_match(type='PARAM_VALUE', blocking=True,
timeout=timeout)

        if not msg:
            print(f"  Timeout: Received {len(received_params)} para
meters (expected {expected_count if expected_count else 'unknow
n'})")
            break

        param_id = msg.param_id
        param_value = msg.param_value
```

```
        # 参数类型映射
        type_names = {
            0: "INT8", 1: "UINT8", 2: "INT16", 3: "UINT16", 4: "INT
32",
            5: "UINT32", 6: "INT64", 7: "UINT64", 8: "FLOAT", 9: "D
OUBLE"
        }
        param_type = type_names.get(msg.param_type, f"UNKNOWN({msg.
param_type})")

        # 首次收到消息时记录总参数数量
        if expected_count is None:
            expected_count = msg.param_count
            print(f"  Total parameters to receive: {expected_coun
t}")

        print(f"  [{msg.param_index}/{expected_count}] {param_id} =
{param_value} ({param_type})")

        received_params.append({
            'id': param_id,
            'value': param_value,
            'type': param_type,
            'index': msg.param_index,
            'count': msg.param_count
        })

        # 当已接收数量达到预期总数时退出循环
        if len(received_params) >= expected_count:
            print(f"\n  Successfully received all {expected_count}
parameters")
            break

    return received_params

def main():
    print("  Starting MAVLink PARAM_SET Tool (UDP Listening Mode)")
    print("-----------------------------------------------------")
    print("This script listens for MAVLink on ALL interfaces:1455
0")
    print("Ensure your vehicle is configured to send to this port")
    print("-----------------------------------------------------
\n")
```

```
    vehicle = connect_to_vehicle()

    # 操作选择
    print("\nAvailable operations:")
    print("1. Set parameter value")
    print("2. Read parameter value (by ID or index)")
    print("3. Request all parameters (PARAM_REQUEST_LIST)")
    choice = input("\nSelect operation [1-3]: ").strip()

    if choice == '1':
        param_id = input("Enter parameter ID to set: ").strip()
        param_value = float(input("Enter new value: ").strip())

        print("\nParameter types:")
        print("  8 = FLOAT (most common)")
        print("  9 = DOUBLE")
        print("  4 = INT32")
        print("  1 = UINT8")
        param_type = int(input("Enter type number [8]: ").strip() or 8)

        send_param_set(vehicle, param_id, param_value, param_type)
        receive_param_value(vehicle)

    elif choice == '2':
        # mode = input("Read by [id/index]: ").lower()
        mode = 'id'
        if mode == 'id':
            param_id = input("Enter parameter ID: ").strip()
            send_param_request_read(vehicle, param_id=param_id)
            receive_param_value(vehicle)
        elif mode == 'index':
            param_index = int(input("Enter parameter index: "))
            send_param_request_read(vehicle, param_index=param_index)
            receive_param_value(vehicle)
        else:
            print("Invalid mode. Use 'id' or 'index'")

    elif choice == '3':
        print("\n  Requesting all parameters...")
        send_param_request_list(vehicle)
        all_params = receive_all_param_values(vehicle)
        print(f"\n  Total parameters received: {len(all_params)}")
```

```
    else:
        print("Invalid choice. Select 1, 2, or 3.")


if __name__ == "__main__":
    main()
```

示例使用说明

获取

运行后，成功建立连接

```
🚀 Starting MAVLink PARAM_SET Tool (UDP Listening Mode)
------------------------------------------------------
This script listens for MAVLink on ALL interfaces:14550
Ensure your vehicle is configured to send to this port
------------------------------------------------------

Listening for MAVLink on: udpin:0.0.0.0:14550
Waiting for HEARTBEAT (timeout: 10s)...
✅ Heartbeat received from system 3, component 0

Available operations:
1. Set parameter value
2. Read parameter value (by ID or index)
3. Request all parameters (PARAM_REQUEST_LIST)
```

输入“2”选择读取参数

```
Select operation [1-3]: 2
```

输入对应参数“FOCUS_RING_VALUE”

```
Enter parameter ID: FOCUS_RING_VALUE
```

成功返回该参数对应的数值

```
📥 Received PARAM_VALUE:
  🆔 ID: FOCUS_RING_VALUE
  🔢 Value: 59.0
```

设置

运行后，成功建立连接

```
🚀 Starting MAVLink PARAM_SET Tool (UDP Listening Mode)
--------------------------------------------------
This script listens for MAVLink on ALL interfaces:14550
Ensure your vehicle is configured to send to this port
--------------------------------------------------

Listening for MAVLink on: udpin:0.0.0.0:14550
Waiting for HEARTBEAT (timeout: 10s)...
✅ Heartbeat received from system 3, component 0

Available operations:
1. Set parameter value
2. Read parameter value (by ID or index)
3. Request all parameters (PARAM_REQUEST_LIST)
```

输入“1”选择设置参数

```
Available operations:
1. Set parameter value
2. Read parameter value (by ID or index)
3. Request all parameters (PARAM_REQUEST_LIST)

Select operation [1-3]: 1
```

输入对应参数“FOCUS_RING_VALUE”

```
Enter parameter ID to set: FOCUS_RING_VALUE
```

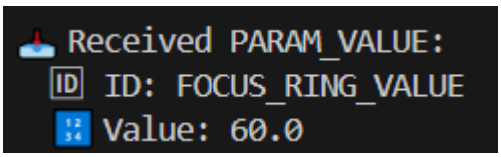
输入要设置的数值

```
Enter new value: 60
```

选择参数类型“1”

```
Parameter types:
  8 = FLOAT (most common)
  9 = DOUBLE
  4 = INT32
  1 = UINT8
Enter type number [8]: 1
```

成功返回设置值



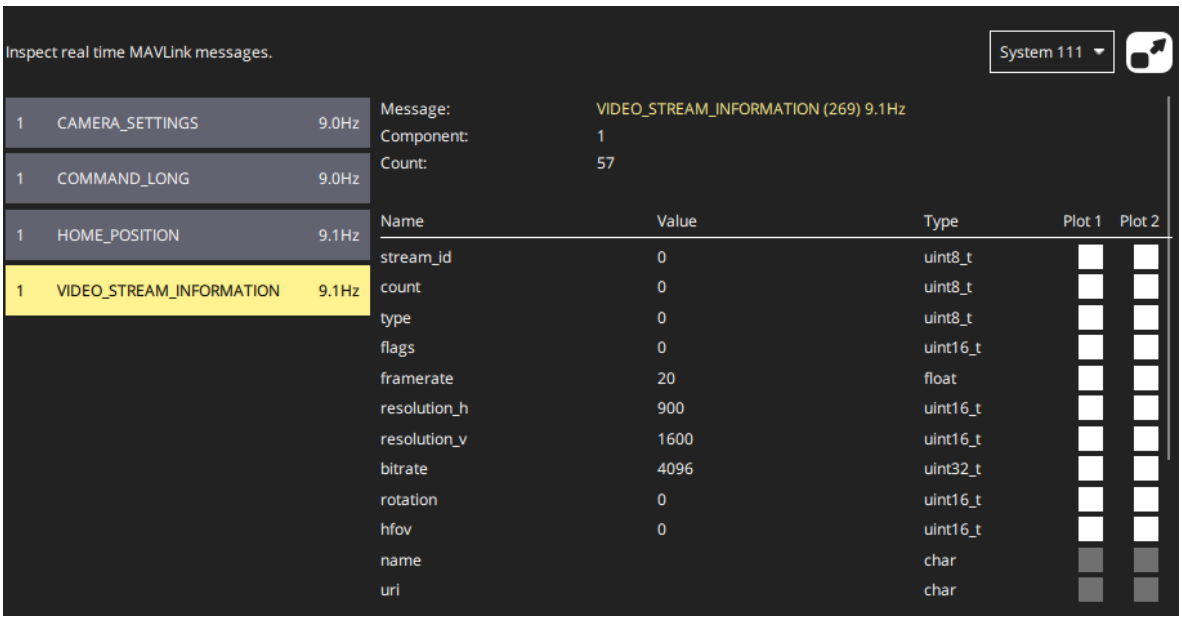
数据传输

视频流信息

mavlink id 269 VIDEO_STREAM_INFORMATION

关于视频流的信息。

返回字段名称	类型	单位	描
bitrate		bits/s	码
framerate		Hz	帧
resolution_v	int16_t	pix	视
resolution_h	int16_t	pix	视



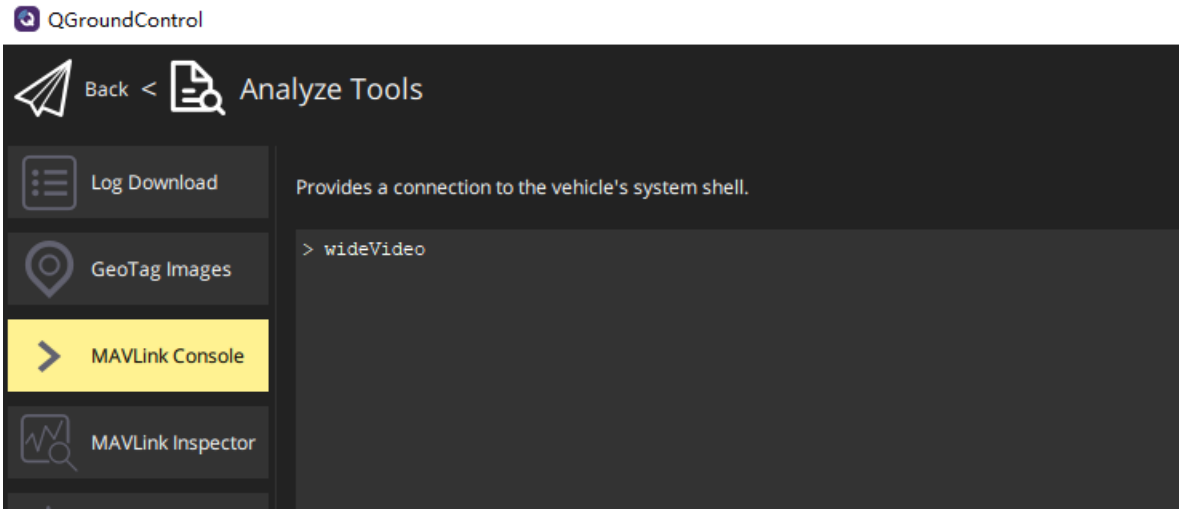
MAV_CMD_VIDEO_START_STREAMING 2502

开始视频流式播放

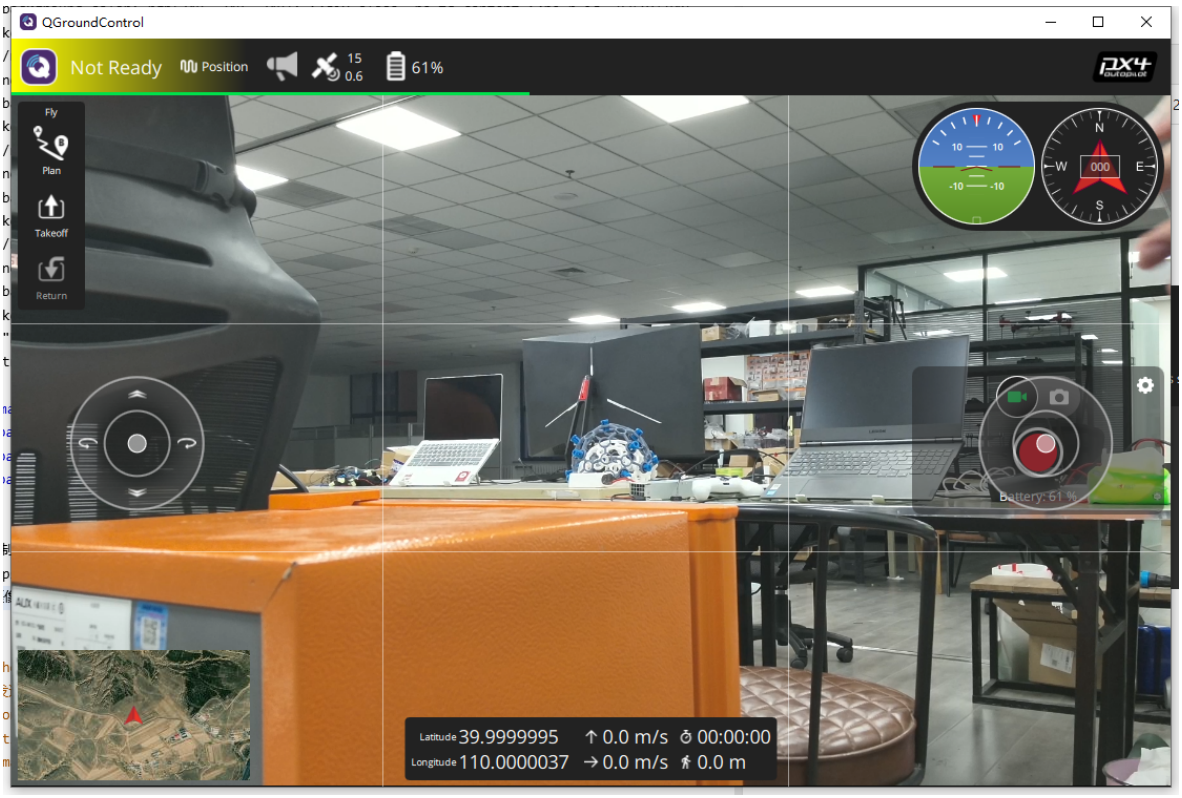
Param (:Label)	描述	值
param1:(Stream ID)	要开启的视频流ID	1：主摄像头，2：下视摄像头，3：红外摄像头

返回消息COMMAND_ACK
返回值 command 2502 执行命令id
返回值 result 0或4 0：执行成功，4：执行失败

于QGC控制台输入wideVideo

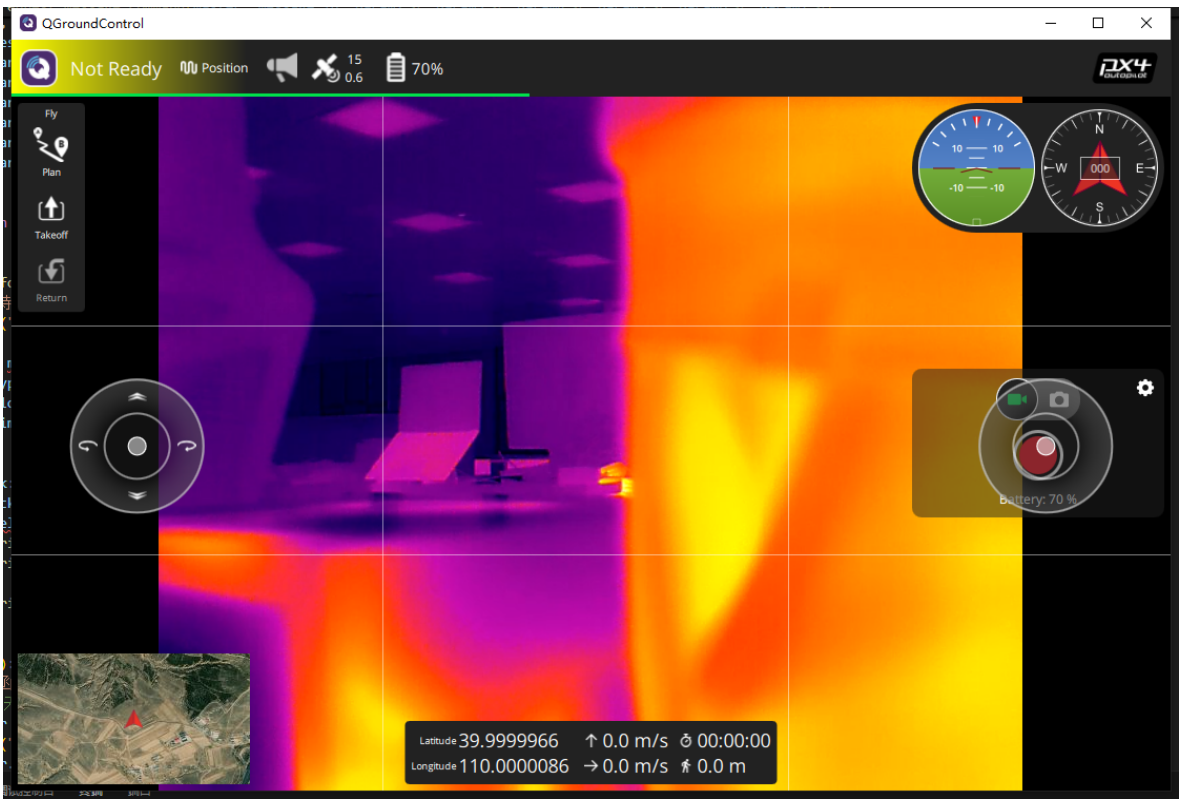


打开主摄像机视频流

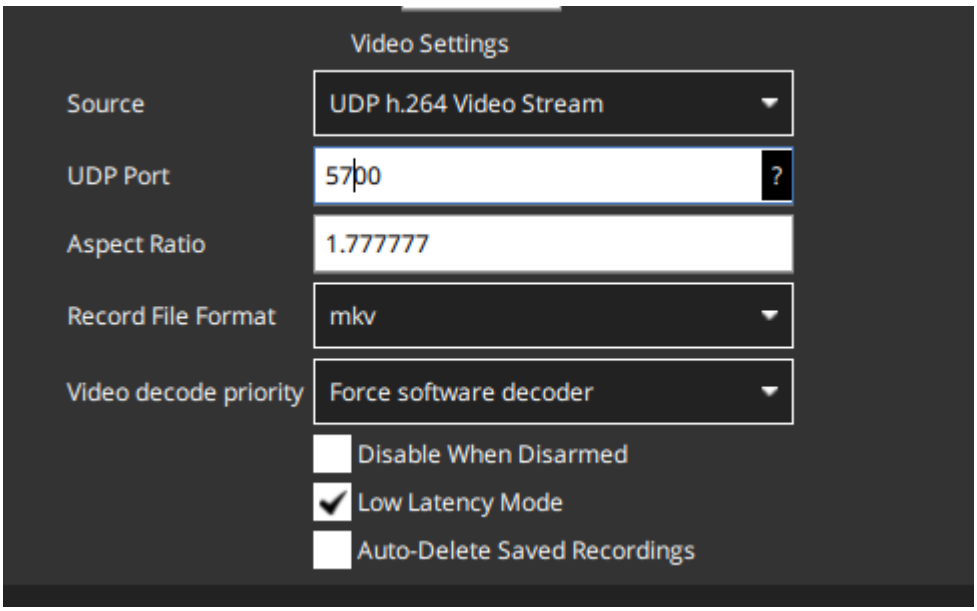


于QGC控制台输入IRVideo

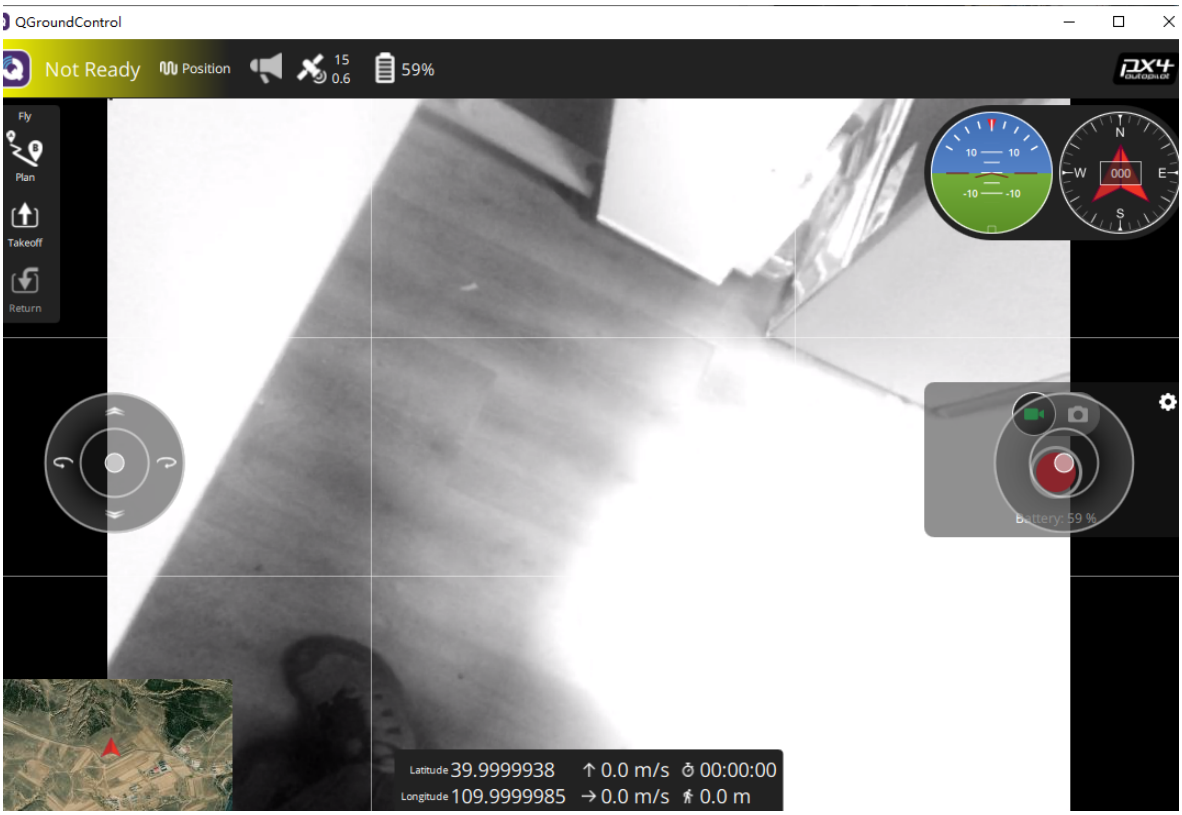
打开主摄像机红外视频流



于QGC控制台输入DownVideo，并将视频流接收端口改为5700



打开下视摄像机视频流



移动 App 图传设置中的“增强传输”状态

ENH_TRANS_STATE

设置

mavlink消息：

接收PARAM_SET
发送PARAM_VALUE

获取

mavlink消息

接收PARAM_REQUEST_READ
发送PARAM_VALUE

参数：

名称	枚举值	描述
DJI_ENHANCEED_TRANSMISSION_STATE_DISABLED	0	增强图传功能已禁用
DJI_ENHANCEED_TRANSMISSION_STATE_ENABLED	3	增强图传功能已启用

示例

```
import time
import sys
from pymavlink import mavutil

def connect_to_vehicle():
    """建立 UDP 监听连接（默认监听所有接口的 14550 端口）"""
    connection_string = "udpin:0.0.0.0:14550"
    print(f"Listening for MAVLink on: {connection_string}")
    vehicle = mavutil.mavlink_connection(connection_string)
    print("Waiting for HEARTBEAT (timeout: 10s)...")
    try:
        vehicle.wait_heartbeat(timeout=10)
        print(f"  Heartbeat received from system {vehicle.target_system}, component {vehicle.target_component}")
        return vehicle
    except Exception as e:
        print(f"  Connection failed: {str(e)}")
        print("Ensure your vehicle is sending MAVLink messages to 14550 port")
        print("Example QGroundControl setup: Settings > General > MAVLink > Listening Port = 14550")
        sys.exit(1)

def format_param_id(param_id):
```

```
    """格式化16字节参数ID（符合MAVLink协议要求）"""
    if len(param_id) > 16:
        raise ValueError("param_id cannot exceed 16 characters")

    param_id_bytes = bytearray(16)
    for i in range(min(len(param_id), 16)):
        param_id_bytes[i] = ord(param_id[i])

    # 长度<16时添加NULL终止符
    if len(param_id) < 16:
        param_id_bytes[len(param_id)] = 0

    return param_id_bytes

def send_param_set(vehicle, param_id, param_value, param_type):
    """发送参数设置请求"""
    vehicle.mav.param_set_send(
        vehicle.target_system,
        vehicle.target_component,
        format_param_id(param_id),
        param_value,
        param_type
    )
    print(f"  Sent PARAM_SET for '{param_id}' = {param_value} (type: {param_type})")

def send_param_request_read(vehicle, param_id=None, param_index=-1):
    """发送参数读取请求"""
    param_id_bytes = format_param_id(param_id) if param_id else bytearray(16)

    vehicle.mav.param_request_read_send(
        vehicle.target_system,
        vehicle.target_component,
        param_id_bytes,
        param_index
    )

    target = f"index {param_index}" if param_index >= 0 else f"ID '{param_id}'"
    print(f"  Sent PARAM_REQUEST_READ for {target}")

def send_param_request_list(vehicle):
    """发送请求所有参数列表"""
```

```
    vehicle.mav.param_request_list_send(
        vehicle.target_system,
        vehicle.target_component
    )
    print(f"  Sent PARAM_REQUEST_LIST to system {vehicle.target_system}, component {vehicle.target_component}")

def receive_param_value(vehicle, timeout=5.0):
    """接收参数值响应"""
    print(f"  Waiting for PARAM_VALUE ({timeout}s timeout)...")
    msg = vehicle.recv_match(type='PARAM_VALUE', blocking=True, timeout=timeout)

    if not msg:
        print("  Timeout: No PARAM_VALUE received")
        return None

    # 解析参数ID
    param_id = msg.param_id

    # 解析参数值
    param_value = msg.param_value

    # 参数类型映射
    type_names = {
        0: "INT8", 1: "UINT8", 2: "INT16", 3: "UINT16", 4: "INT32",
        5: "UINT32", 6: "INT64", 7: "UINT64", 8: "FLOAT", 9: "DOUBLE"
    }
    param_type = type_names.get(msg.param_type, f"UNKNOWN({msg.param_type})")

    print("\n  Received PARAM_VALUE:")
    print(f"    ID: {param_id}")
    print(f"    Value: {param_value}")
    print(f"    Type: {param_type}")
    print(f"    Index/Total: {msg.param_index}/{msg.param_count}")

    return {
        'id': param_id,
        'value': param_value,
        'type': param_type,
        'index': msg.param_index,
        'count': msg.param_count
    }
```

```
def receive_all_param_values(vehicle, timeout=10.0):
    """接收所有参数值，直到收到 param_count 条消息后退出循环"""
    received_params = []
    expected_count = None
    print(f"  Waiting for all PARAM_VALUE messages...")

    while True:
        msg = vehicle.recv_match(type='PARAM_VALUE', blocking=True,
timeout=timeout)

        if not msg:
            print(f"  Timeout: Received {len(received_params)} para
meters (expected {expected_count if expected_count else 'unknow
n'})")
            break

        param_id = msg.param_id
        param_value = msg.param_value

        # 参数类型映射
        type_names = {
            0: "INT8", 1: "UINT8", 2: "INT16", 3: "UINT16", 4: "INT
32",
            5: "UINT32", 6: "INT64", 7: "UINT64", 8: "FLOAT", 9: "D
OUBLE"
        }
        param_type = type_names.get(msg.param_type, f"UNKNOWN({msg.
param_type})")

        # 首次收到消息时记录总参数数量
        if expected_count is None:
            expected_count = msg.param_count
            print(f"  Total parameters to receive: {expected_coun
t}")

        print(f"  [{msg.param_index}/{expected_count}] {param_id} =
{param_value} ({param_type})")

        received_params.append({
            'id': param_id,
            'value': param_value,
            'type': param_type,
            'index': msg.param_index,
            'count': msg.param_count
```

```
        })

        # 当已接收数量达到预期总数时退出循环
        if len(received_params) >= expected_count:
            print(f"\n  Successfully received all {expected_count}
parameters")
            break

    return received_params

def main():
    print("  Starting MAVLink PARAM_SET Tool (UDP Listening Mode)")
    print("------------------------------------------------------")
    print("This script listens for MAVLink on ALL interfaces:1455
0")
    print("Ensure your vehicle is configured to send to this port")
    print("------------------------------------------------------
\n")

    vehicle = connect_to_vehicle()

    # 操作选择
    print("\nAvailable operations:")
    print("1. Set parameter value")
    print("2. Read parameter value (by ID or index)")
    print("3. Request all parameters (PARAM_REQUEST_LIST)")
    choice = input("\nSelect operation [1-3]: ").strip()

    if choice == '1':
        param_id = input("Enter parameter ID to set: ").strip()
        param_value = float(input("Enter new value: ").strip())

        print("\nParameter types:")
        print("  8 = FLOAT (most common)")
        print("  9 = DOUBLE")
        print("  4 = INT32")
        print("  1 = UINT8")
        param_type = int(input("Enter type number [8]: ").strip() o
r 8)

        send_param_set(vehicle, param_id, param_value, param_type)
        receive_param_value(vehicle)

    elif choice == '2':
        # mode = input("Read by [id/index]: ").lower()
```

```
        mode = 'id'
        if mode == 'id':
            param_id = input("Enter parameter ID: ").strip()
            send_param_request_read(vehicle, param_id=param_id)
            receive_param_value(vehicle)
        elif mode == 'index':
            param_index = int(input("Enter parameter index: "))
            send_param_request_read(vehicle, param_index=param_inde
x)
            receive_param_value(vehicle)
        else:
            print("Invalid mode. Use 'id' or 'index'")

    elif choice == '3':
        print("\n  Requesting all parameters...")
        send_param_request_list(vehicle)
        all_params = receive_all_param_values(vehicle)
        print(f"\n  Total parameters received: {len(all_params)}")

    else:
        print("Invalid choice. Select 1, 2, or 3.")


if __name__ == "__main__":
    main()
```

示例使用说明

获取

运行后，成功建立连接

```
🚀 Starting MAVLink PARAM_SET Tool (UDP Listening Mode)
--------------------------------------------------
This script listens for MAVLink on ALL interfaces:14550
Ensure your vehicle is configured to send to this port
--------------------------------------------------

Listening for MAVLink on: udpin:0.0.0.0:14550
Waiting for HEARTBEAT (timeout: 10s)...
✅ Heartbeat received from system 3, component 0

Available operations:
1. Set parameter value
2. Read parameter value (by ID or index)
3. Request all parameters (PARAM_REQUEST_LIST)
```

输入“2”选择读取参数

```
Select operation [1-3]: 2
```

输入对应参数“ENH_TRANS_STATE”

```
Enter parameter ID: ENH_TRANS_STATE
```

成功返回该参数对应的数值

```
📥 Received PARAM_VALUE:
  🆔 ID: ENH_TRANS_STATE
  🔢 Value: 0.0
```

设置

运行后，成功建立连接

```
🚀 Starting MAVLink PARAM_SET Tool (UDP Listening Mode)
--------------------------------------------------
This script listens for MAVLink on ALL interfaces:14550
Ensure your vehicle is configured to send to this port
--------------------------------------------------

Listening for MAVLink on: udpin:0.0.0.0:14550
Waiting for HEARTBEAT (timeout: 10s)...
✅ Heartbeat received from system 3, component 0

Available operations:
1. Set parameter value
2. Read parameter value (by ID or index)
3. Request all parameters (PARAM_REQUEST_LIST)
```

输入“1”选择设置参数

```
Available operations:
1. Set parameter value
2. Read parameter value (by ID or index)
3. Request all parameters (PARAM_REQUEST_LIST)

Select operation [1-3]: 1
```

输入对应参数“ENH_TRANS_STATE”

```
Enter parameter ID to set: ENH_TRANS_STATE
```

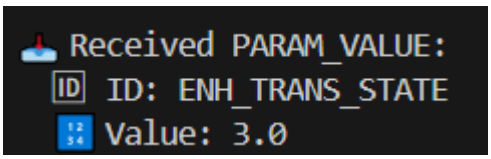
输入要设置的数值

```
Enter new value: 3
```

选择参数类型“1”

```
Parameter types:
  8 = FLOAT (most common)
  9 = DOUBLE
  4 = INT32
  1 = UINT8
Enter type number [8]: 1
```

成功返回设置值



结果（使用QGC查看）

云台功能

控制云台的俯仰和偏航角

MAV_CMD_DO_MOUNT_CONTROL (205)

Number	Field Name	Type	Units	Values	Description
param1	Pitch	float	° (角度)		俯仰角
param2	Roll	float	° (角度)		横滚角
param3	Yaw	float	° (角度)		偏航角
param7	MAV_MOUNT_MODE				云台模式

MAV_MOUNT_MODE 为2时启动由MAVLink发送的 P/R/Y 角度控制。
当param1、param2、param3均为0时复位
返回消息：COMMAND_ACK
返回值 command 205 执行命令id
返回值 result 0或4 0：执行成功，4：执行失败

示例

```
import time
from pymavlink import mavutil
from datetime import datetime

# 获取当前时间（毫秒级）
def get_current_millis():
    return int(time.time() * 1000)

# 连接无人机（UDP端口14550）
master = mavutil.mavlink_connection('udpin:0.0.0.0:14550')
print("等待无人机心跳...")
master.wait_heartbeat()
print("无人机已连接，开始发送云台控制命令")

# 云台控制参数设置
# 参数说明：
#   param1: 横滚角度 (0.0~1.0, 0.5=中立)
#   param2: 俯仰角度 (0.0~1.0, 0.5=中立)
```

```
#   param3: 偏航角度 (0.0~1.0, 0.5=中立)
#   param4: 模式 (0=无模式, 1=锁定, 2=跟踪)
#   param5: 方向 (0=无, 1=上, 2=下, 3=左, 4=右)
#   param6: 未使用
roll = 0   # 云台横滚中立位置
pitch = 20  # 云台俯仰中立位置
yaw = 0    # 云台偏航中立位置
mode = 0     # 无模式（直接控制角度）
direction = 0  # 无方向

# 发送云台控制命令
send_time = get_current_millis()
print(f"[{send_time}] 发送云台控制命令: roll={roll}, pitch={pitch}, ya
w={yaw}, mode={mode}")

master.mav.command_long_send(
    master.target_system,
    master.target_component,  # 云台通常使用组件ID 2 (MAV_COMPONENT_MO
UNT)
    mavutil.mavlink.MAV_CMD_DO_MOUNT_CONTROL,
    0,
    pitch,       # param1
    roll,      # param2
    yaw,         # param3
    0,       # param4
    0, # param5
    0,         # param6 (未使用)
    2
)

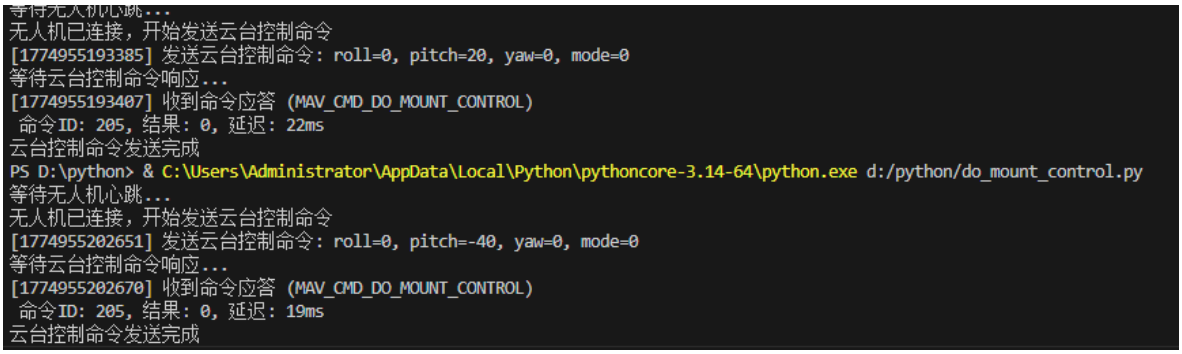
# 等待 COMMAND_ACK 响应（超时5秒）
print("等待云台控制命令响应...")
ack = master.recv_match(
    type='COMMAND_ACK',
    blocking=True,
    timeout=5.0
)

if ack:
    ack_time = get_current_millis()
    delay = ack_time - send_time
    print(f"[{ack_time}] 收到命令应答 (MAV_CMD_DO_MOUNT_CONTROL)")
    print(f" 命令ID: {ack.command}, 结果: {ack.result}, 延迟: {delay}
ms")
else:
```

```
    print(f"[{get_current_millis()}] 超时：未收到云台控制命令应答")

print("云台控制命令发送完成")
```

执行结果




DO NOT

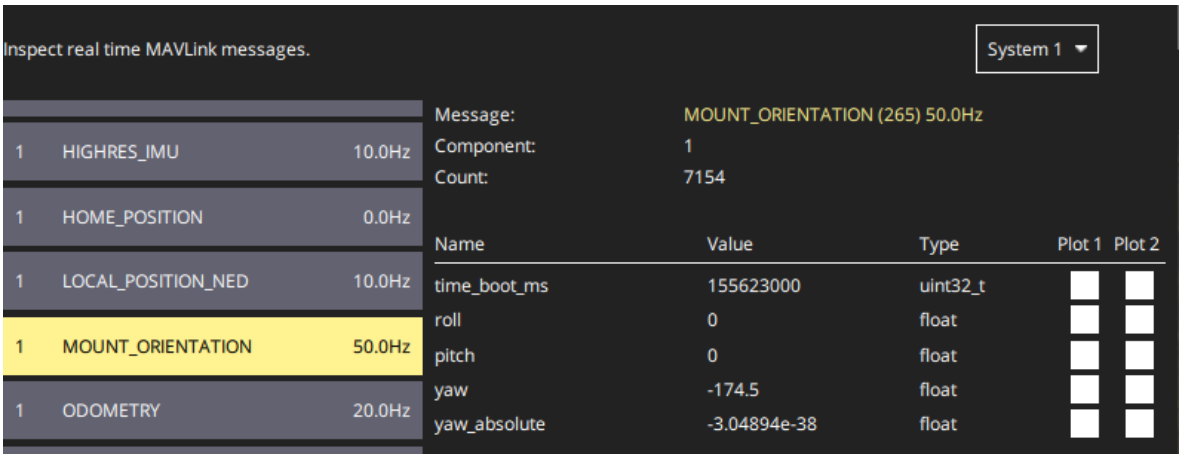




mavlink id 265 MOUNT_ORIENTATION

返回字段名称	类型	单位	描述
time_boot_ms	uint32_t	ms	时间戳
roll	float	deg	翻滚角
pitch	float	deg	俯仰角
yaw	float	deg	偏航角

结果（使用QGC查看）



启用或禁用云台俯仰限位扩展

GIMBAL_EXT_ENAB

mavlink消息：

接收PARAM_REQUEST_READ
发送PARAM_VALUE

参数：

类型	枚举值	说明
bool	0	禁用 俯仰限位扩展 功能
bool	1	启用 俯仰限位扩展 功能

示例

```
import time
import sys
```

```
from pymavlink import mavutil

def connect_to_vehicle():
    """建立 UDP 监听连接（默认监听所有接口的 14550 端口）"""
    connection_string = "udpin:0.0.0.0:14550"
    print(f"Listening for MAVLink on: {connection_string}")
    vehicle = mavutil.mavlink_connection(connection_string)
    print("Waiting for HEARTBEAT (timeout: 10s)...")
    try:
        vehicle.wait_heartbeat(timeout=10)
        print(f"  Heartbeat received from system {vehicle.target_sy
stem}, component {vehicle.target_component}")
        return vehicle
    except Exception as e:
        print(f"  Connection failed: {str(e)}")
        print("Ensure your vehicle is sending MAVLink messages to 1
4550 port")
        print("Example QGroundControl setup: Settings > General > M
AVLink > Listening Port = 14550")
        sys.exit(1)

def format_param_id(param_id):
    """格式化16字节参数ID（符合MAVLink协议要求）"""
    if len(param_id) > 16:
        raise ValueError("param_id cannot exceed 16 characters")

    param_id_bytes = bytearray(16)
    for i in range(min(len(param_id), 16)):
        param_id_bytes[i] = ord(param_id[i])

    # 长度<16时添加NULL终止符
    if len(param_id) < 16:
        param_id_bytes[len(param_id)] = 0

    return param_id_bytes

def send_param_set(vehicle, param_id, param_value, param_type):
    """发送参数设置请求"""
    vehicle.mav.param_set_send(
        vehicle.target_system,
        vehicle.target_component,
        format_param_id(param_id),
        param_value,
        param_type
    )
```

```
    print(f"  Sent PARAM_SET for '{param_id}' = {param_value} (type: {param_type})")

def send_param_request_read(vehicle, param_id=None, param_index=-1):
    """发送参数读取请求"""
    param_id_bytes = format_param_id(param_id) if param_id else bytearray(16)

    vehicle.mav.param_request_read_send(
        vehicle.target_system,
        vehicle.target_component,
        param_id_bytes,
        param_index
    )

    target = f"index {param_index}" if param_index >= 0 else f"ID '{param_id}'"
    print(f"  Sent PARAM_REQUEST_READ for {target}")

def send_param_request_list(vehicle):
    """发送请求所有参数列表"""
    vehicle.mav.param_request_list_send(
        vehicle.target_system,
        vehicle.target_component
    )
    print(f"  Sent PARAM_REQUEST_LIST to system {vehicle.target_system}, component {vehicle.target_component}")

def receive_param_value(vehicle, timeout=5.0):
    """接收参数值响应"""
    print(f"  Waiting for PARAM_VALUE ({timeout}s timeout)...")
    msg = vehicle.recv_match(type='PARAM_VALUE', blocking=True, timeout=timeout)

    if not msg:
        print("  Timeout: No PARAM_VALUE received")
        return None

    # 解析参数ID
    param_id = msg.param_id

    # 解析参数值
    param_value = msg.param_value
```

```
    # 参数类型映射
    type_names = {
        0: "INT8", 1: "UINT8", 2: "INT16", 3: "UINT16", 4: "INT32",
        5: "UINT32", 6: "INT64", 7: "UINT64", 8: "FLOAT", 9: "DOUBL
E"
    }
    param_type = type_names.get(msg.param_type, f"UNKNOWN({msg.para
m_type})")

    print("\n  Received PARAM_VALUE:")
    print(f"    ID: {param_id}")
    print(f"    Value: {param_value}")
    print(f"    Type: {param_type}")
    print(f"    Index/Total: {msg.param_index}/{msg.param_count}")

    return {
        'id': param_id,
        'value': param_value,
        'type': param_type,
        'index': msg.param_index,
        'count': msg.param_count
    }

def receive_all_param_values(vehicle, timeout=10.0):
    """接收所有参数值，直到收到 param_count 条消息后退出循环"""
    received_params = []
    expected_count = None
    print(f"  Waiting for all PARAM_VALUE messages...")

    while True:
        msg = vehicle.recv_match(type='PARAM_VALUE', blocking=True,
timeout=timeout)

        if not msg:
            print(f"  Timeout: Received {len(received_params)} para
meters (expected {expected_count if expected_count else 'unknow
n'})")
            break

        param_id = msg.param_id
        param_value = msg.param_value

        # 参数类型映射
        type_names = {
            0: "INT8", 1: "UINT8", 2: "INT16", 3: "UINT16", 4: "INT
```

```
32",
            5: "UINT32", 6: "INT64", 7: "UINT64", 8: "FLOAT", 9: "D
OUBLE"
        }
        param_type = type_names.get(msg.param_type, f"UNKNOWN({msg.
param_type})")

        # 首次收到消息时记录总参数数量
        if expected_count is None:
            expected_count = msg.param_count
            print(f"  Total parameters to receive: {expected_coun
t}")

        print(f"  [{msg.param_index}/{expected_count}] {param_id} =
{param_value} ({param_type})")

        received_params.append({
            'id': param_id,
            'value': param_value,
            'type': param_type,
            'index': msg.param_index,
            'count': msg.param_count
        })

        # 当已接收数量达到预期总数时退出循环
        if len(received_params) >= expected_count:
            print(f"\n  Successfully received all {expected_count}
parameters")
            break

    return received_params

def main():
    print("  Starting MAVLink PARAM_SET Tool (UDP Listening Mode)")
    print("-----------------------------------------------------")
    print("This script listens for MAVLink on ALL interfaces:1455
0")
    print("Ensure your vehicle is configured to send to this port")
    print("-----------------------------------------------------
\n")

    vehicle = connect_to_vehicle()

    # 操作选择
    print("\nAvailable operations:")
```

```
    print("1. Set parameter value")
    print("2. Read parameter value (by ID or index)")
    print("3. Request all parameters (PARAM_REQUEST_LIST)")
    choice = input("\nSelect operation [1-3]: ").strip()

    if choice == '1':
        param_id = input("Enter parameter ID to set: ").strip()
        param_value = float(input("Enter new value: ").strip())

        print("\nParameter types:")
        print("  8 = FLOAT (most common)")
        print("  9 = DOUBLE")
        print("  4 = INT32")
        print("  1 = UINT8")
        param_type = int(input("Enter type number [8]: ").strip() or 8)

        send_param_set(vehicle, param_id, param_value, param_type)
        receive_param_value(vehicle)

    elif choice == '2':
        # mode = input("Read by [id/index]: ").lower()
        mode = 'id'
        if mode == 'id':
            param_id = input("Enter parameter ID: ").strip()
            send_param_request_read(vehicle, param_id=param_id)
            receive_param_value(vehicle)
        elif mode == 'index':
            param_index = int(input("Enter parameter index: "))
            send_param_request_read(vehicle, param_index=param_index)
            receive_param_value(vehicle)
        else:
            print("Invalid mode. Use 'id' or 'index'")

    elif choice == '3':
        print("\n  Requesting all parameters...")
        send_param_request_list(vehicle)
        all_params = receive_all_param_values(vehicle)
        print(f"\n  Total parameters received: {len(all_params)}")

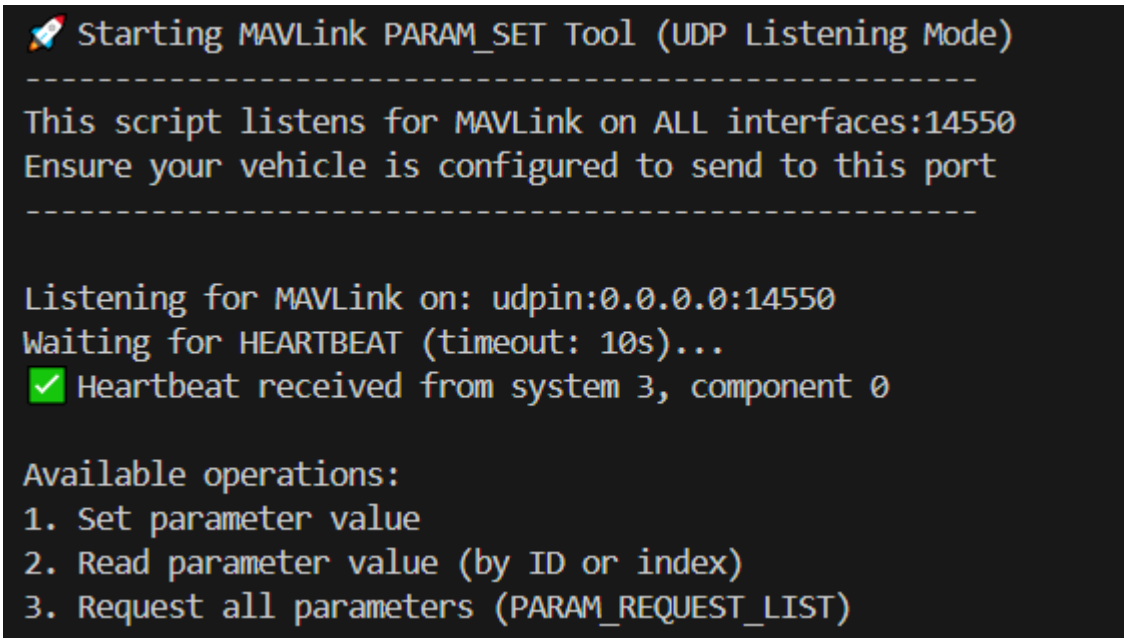
    else:
        print("Invalid choice. Select 1, 2, or 3.")
```

```
if __name__ == "__main__":
    main()
```

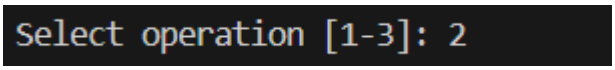
示例使用说明

获取

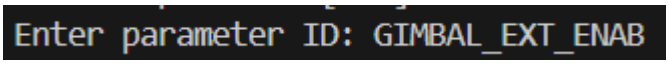
运行后，成功建立连接



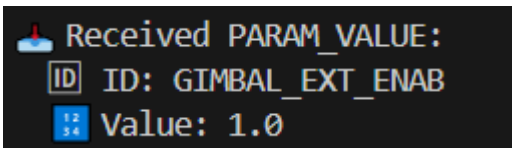
输入“2”选择读取参数



输入对应参数“GIMBAL_EXT_ENAB”



成功返回该参数对应的数值



设置

运行后，成功建立连接

```
🚀 Starting MAVLink PARAM_SET Tool (UDP Listening Mode)
------------------------------------------------------
This script listens for MAVLink on ALL interfaces:14550
Ensure your vehicle is configured to send to this port
------------------------------------------------------

Listening for MAVLink on: udpin:0.0.0.0:14550
Waiting for HEARTBEAT (timeout: 10s)...
✅ Heartbeat received from system 3, component 0

Available operations:
1. Set parameter value
2. Read parameter value (by ID or index)
3. Request all parameters (PARAM_REQUEST_LIST)
```

输入“1”选择设置参数

```
Select operation [1-3]: 1
```

输入对应参数“GIMBAL_EXT_ENAB”

```
Enter parameter ID to set: GIMBAL_EXT_ENAB
```

输入要设置的数值

```
Enter new value: 0
```

选择参数类型“1”

```
Parameter types:
  9 = DOUBLE
  4 = INT32
  1 = UINT8
Enter type number [8]: 1
```

成功返回设置值

```
📥 Received PARAM_VALUE:
   🆔 ID: GIMBAL_EXT_ENAB
   🔢 Value: 0.0
```

云台控制器的最大速度百分比

GIMBAL_MAXSPD_P
GIMBAL_MAXSPD_R
GIMBAL_MAXSPD_Y

mavlink消息：

接收PARAM_REQUEST_READ
发送PARAM_VALUE

参数：

参数名	类型	说明	取值范围
maxSpeedPercentage	uint8_t	最大速度百分比	1 ~ 100

示例

```
import time
import sys
from pymavlink import mavutil

def connect_to_vehicle():
    """建立 UDP 监听连接（默认监听所有接口的 14550 端口）"""
    connection_string = "udpin:0.0.0.0:14550"
    print(f"Listening for MAVLink on: {connection_string}")
    vehicle = mavutil.mavlink_connection(connection_string)
    print("Waiting for HEARTBEAT (timeout: 10s)...")
    try:
        vehicle.wait_heartbeat(timeout=10)
        print(f"  Heartbeat received from system {vehicle.target_sy
stem}, component {vehicle.target_component}")
        return vehicle
    except Exception as e:
        print(f"  Connection failed: {str(e)}")
        print("Ensure your vehicle is sending MAVLink messages to 1
4550 port")
        print("Example QGroundControl setup: Settings > General > M
AVLink > Listening Port = 14550")
        sys.exit(1)

def format_param_id(param_id):
    """格式化16字节参数ID（符合MAVLink协议要求）"""
    if len(param_id) > 16:
        raise ValueError("param_id cannot exceed 16 characters")
```

```
    param_id_bytes = bytearray(16)
    for i in range(min(len(param_id), 16)):
        param_id_bytes[i] = ord(param_id[i])

    # 长度<16时添加NULL终止符
    if len(param_id) < 16:
        param_id_bytes[len(param_id)] = 0

    return param_id_bytes

def send_param_set(vehicle, param_id, param_value, param_type):
    """发送参数设置请求"""
    vehicle.mav.param_set_send(
        vehicle.target_system,
        vehicle.target_component,
        format_param_id(param_id),
        param_value,
        param_type
    )
    print(f"  Sent PARAM_SET for '{param_id}' = {param_value} (type: {param_type})")

def send_param_request_read(vehicle, param_id=None, param_index=-1):
    """发送参数读取请求"""
    param_id_bytes = format_param_id(param_id) if param_id else bytearray(16)

    vehicle.mav.param_request_read_send(
        vehicle.target_system,
        vehicle.target_component,
        param_id_bytes,
        param_index
    )

    target = f"index {param_index}" if param_index >= 0 else f"ID '{param_id}'"
    print(f"  Sent PARAM_REQUEST_READ for {target}")

def send_param_request_list(vehicle):
    """发送请求所有参数列表"""
    vehicle.mav.param_request_list_send(
        vehicle.target_system,
        vehicle.target_component
    )
```

```
    print(f"  Sent PARAM_REQUEST_LIST to system {vehicle.target_system}, component {vehicle.target_component}")

def receive_param_value(vehicle, timeout=5.0):
    """接收参数值响应"""
    print(f"  Waiting for PARAM_VALUE ({timeout}s timeout)...")
    msg = vehicle.recv_match(type='PARAM_VALUE', blocking=True, timeout=timeout)

    if not msg:
        print("  Timeout: No PARAM_VALUE received")
        return None

    # 解析参数ID
    param_id = msg.param_id

    # 解析参数值
    param_value = msg.param_value

    # 参数类型映射
    type_names = {
        0: "INT8", 1: "UINT8", 2: "INT16", 3: "UINT16", 4: "INT32",
        5: "UINT32", 6: "INT64", 7: "UINT64", 8: "FLOAT", 9: "DOUBLE"
    }
    param_type = type_names.get(msg.param_type, f"UNKNOWN({msg.param_type})")

    print("\n  Received PARAM_VALUE:")
    print(f"    ID: {param_id}")
    print(f"    Value: {param_value}")
    print(f"    Type: {param_type}")
    print(f"    Index/Total: {msg.param_index}/{msg.param_count}")

    return {
        'id': param_id,
        'value': param_value,
        'type': param_type,
        'index': msg.param_index,
        'count': msg.param_count
    }

def receive_all_param_values(vehicle, timeout=10.0):
    """接收所有参数值，直到收到 param_count 条消息后退出循环"""
    received_params = []
```

```
    expected_count = None
    print(f"  Waiting for all PARAM_VALUE messages...")

    while True:
        msg = vehicle.recv_match(type='PARAM_VALUE', blocking=True,
timeout=timeout)

        if not msg:
            print(f"  Timeout: Received {len(received_params)} para
meters (expected {expected_count if expected_count else 'unknow
n'})")
            break

        param_id = msg.param_id
        param_value = msg.param_value

        # 参数类型映射
        type_names = {
            0: "INT8", 1: "UINT8", 2: "INT16", 3: "UINT16", 4: "INT
32",
            5: "UINT32", 6: "INT64", 7: "UINT64", 8: "FLOAT", 9: "D
OUBLE"
        }
        param_type = type_names.get(msg.param_type, f"UNKNOWN({msg.
param_type})")

        # 首次收到消息时记录总参数数量
        if expected_count is None:
            expected_count = msg.param_count
            print(f"  Total parameters to receive: {expected_coun
t}")

        print(f"  [{msg.param_index}/{expected_count}] {param_id} =
{param_value} ({param_type})")

        received_params.append({
            'id': param_id,
            'value': param_value,
            'type': param_type,
            'index': msg.param_index,
            'count': msg.param_count
        })

        # 当已接收数量达到预期总数时退出循环
        if len(received_params) >= expected_count:
```

```
            print(f"\n  Successfully received all {expected_count}
parameters")
            break

    return received_params

def main():
    print("  Starting MAVLink PARAM_SET Tool (UDP Listening Mode)")
    print("-----------------------------------------------------")
    print("This script listens for MAVLink on ALL interfaces:1455
0")
    print("Ensure your vehicle is configured to send to this port")
    print("-----------------------------------------------------
\n")

    vehicle = connect_to_vehicle()

    # 操作选择
    print("\nAvailable operations:")
    print("1. Set parameter value")
    print("2. Read parameter value (by ID or index)")
    print("3. Request all parameters (PARAM_REQUEST_LIST)")
    choice = input("\nSelect operation [1-3]: ").strip()

    if choice == '1':
        param_id = input("Enter parameter ID to set: ").strip()
        param_value = float(input("Enter new value: ").strip())

        print("\nParameter types:")
        print("  8 = FLOAT (most common)")
        print("  9 = DOUBLE")
        print("  4 = INT32")
        print("  1 = UINT8")
        param_type = int(input("Enter type number [8]: ").strip() o
r 8)

        send_param_set(vehicle, param_id, param_value, param_type)
        receive_param_value(vehicle)

    elif choice == '2':
        # mode = input("Read by [id/index]: ").lower()
        mode = 'id'
        if mode == 'id':
            param_id = input("Enter parameter ID: ").strip()
            send_param_request_read(vehicle, param_id=param_id)
```

```
            receive_param_value(vehicle)
        elif mode == 'index':
            param_index = int(input("Enter parameter index: "))
            send_param_request_read(vehicle, param_index=param_inde
x)
            receive_param_value(vehicle)
        else:
            print("Invalid mode. Use 'id' or 'index'")

    elif choice == '3':
        print("\n  Requesting all parameters...")
        send_param_request_list(vehicle)
        all_params = receive_all_param_values(vehicle)
        print(f"\n  Total parameters received: {len(all_params)}")

    else:
        print("Invalid choice. Select 1, 2, or 3.")


if __name__ == "__main__":
    main()
```

示例使用说明

获取

运行后，成功建立连接

```
🚀 Starting MAVLink PARAM_SET Tool (UDP Listening Mode)
------------------------------------------------------
This script listens for MAVLink on ALL interfaces:14550
Ensure your vehicle is configured to send to this port
------------------------------------------------------

Listening for MAVLink on: udpin:0.0.0.0:14550
Waiting for HEARTBEAT (timeout: 10s)...
✅ Heartbeat received from system 3, component 0

Available operations:
1. Set parameter value
2. Read parameter value (by ID or index)
3. Request all parameters (PARAM_REQUEST_LIST)
```

输入“2”选择读取参数

```
Select operation [1-3]: 2
```

输入对应参数“GIMBAL_MAXSPD_P”/“GIMBAL_MAXSPD_R”/“GIMBAL_MAXSPD_Y”

```
Enter parameter ID: GIMBAL_MAXSPD_P
```

成功返回该参数对应的数值

```
📩 Received PARAM_VALUE:
   🆔 ID: GIMBAL_MAXSPD_P
   🔢 Value: 50.0
```

设置

运行后，成功建立连接

```
🚀 Starting MAVLink PARAM_SET Tool (UDP Listening Mode)
------------------------------------------------------
This script listens for MAVLink on ALL interfaces:14550
Ensure your vehicle is configured to send to this port
------------------------------------------------------

Listening for MAVLink on: udpin:0.0.0.0:14550
Waiting for HEARTBEAT (timeout: 10s)...
✅ Heartbeat received from system 3, component 0

Available operations:
1. Set parameter value
2. Read parameter value (by ID or index)
3. Request all parameters (PARAM_REQUEST_LIST)
```

输入“1”选择设置参数

```
Select operation [1-3]: 1
```

输入对应参数“GIMBAL_MAXSPD_P”/“GIMBAL_MAXSPD_R”/“GIMBAL_MAXSPD_Y”

```
Enter parameter ID to set: GIMBAL_MAXSPD_P
```

输入要设置的数值

```
Enter new value: 45
```

选择参数类型“1”

```
Parameter types:
  9 = DOUBLE
  4 = INT32
  1 = UINT8
Enter type number [8]: 1
```

成功返回设置值

```
Received PARAM_VALUE:
  ID: GIMBAL_MAXSPD_P
  Value: 45.0
```

设置云台控制器的平滑因子

GIMBAL_SMOOTH_P
GIMBAL_SMOOTH_R
GIMBAL_SMOOTH_Y

mavlink消息：

接收PARAM_REQUEST_READ
发送PARAM_VALUE

参数：

参数名	类型	说明	取值范围
maxSpeedPercentage	uint8_t	最大速度百分比	1 ~ 100

平滑因子（Smooth Factor）。数值越大，云台的加速度越小，启动和停止动作会更加平缓。
取值范围：
0 ~ 30

示例

```
import time
import sys
from pymavlink import mavutil

def connect_to_vehicle():
    """建立 UDP 监听连接（默认监听所有接口的 14550 端口）"""
    connection_string = "udpin:0.0.0.0:14550"
    print(f"Listening for MAVLink on: {connection_string}")
```

```
    vehicle = mavutil.mavlink_connection(connection_string)
    print("Waiting for HEARTBEAT (timeout: 10s)...")
    try:
        vehicle.wait_heartbeat(timeout=10)
        print(f"  Heartbeat received from system {vehicle.target_system}, component {vehicle.target_component}")
        return vehicle
    except Exception as e:
        print(f"  Connection failed: {str(e)}")
        print("Ensure your vehicle is sending MAVLink messages to 14550 port")
        print("Example QGroundControl setup: Settings > General > MAVLink > Listening Port = 14550")
        sys.exit(1)

def format_param_id(param_id):
    """格式化16字节参数ID（符合MAVLink协议要求）"""
    if len(param_id) > 16:
        raise ValueError("param_id cannot exceed 16 characters")

    param_id_bytes = bytearray(16)
    for i in range(min(len(param_id), 16)):
        param_id_bytes[i] = ord(param_id[i])

    # 长度<16时添加NULL终止符
    if len(param_id) < 16:
        param_id_bytes[len(param_id)] = 0

    return param_id_bytes

def send_param_set(vehicle, param_id, param_value, param_type):
    """发送参数设置请求"""
    vehicle.mav.param_set_send(
        vehicle.target_system,
        vehicle.target_component,
        format_param_id(param_id),
        param_value,
        param_type
    )
    print(f"  Sent PARAM_SET for '{param_id}' = {param_value} (type: {param_type})")

def send_param_request_read(vehicle, param_id=None, param_index=-1):
    """发送参数读取请求"""
```

```
    param_id_bytes = format_param_id(param_id) if param_id else byt
earray(16)

    vehicle.mav.param_request_read_send(
        vehicle.target_system,
        vehicle.target_component,
        param_id_bytes,
        param_index
    )

    target = f"index {param_index}" if param_index >= 0 else f"ID
'{param_id}'"
    print(f"  Sent PARAM_REQUEST_READ for {target}")

def send_param_request_list(vehicle):
    """发送请求所有参数列表"""
    vehicle.mav.param_request_list_send(
        vehicle.target_system,
        vehicle.target_component
    )
    print(f"  Sent PARAM_REQUEST_LIST to system {vehicle.target_sys
tem}, component {vehicle.target_component}")

def receive_param_value(vehicle, timeout=5.0):
    """接收参数值响应"""
    print(f"  Waiting for PARAM_VALUE ({timeout}s timeout)...")
    msg = vehicle.recv_match(type='PARAM_VALUE', blocking=True, tim
eout=timeout)

    if not msg:
        print("  Timeout: No PARAM_VALUE received")
        return None

    # 解析参数ID
    param_id = msg.param_id

    # 解析参数值
    param_value = msg.param_value

    # 参数类型映射
    type_names = {
        0: "INT8", 1: "UINT8", 2: "INT16", 3: "UINT16", 4: "INT32",
        5: "UINT32", 6: "INT64", 7: "UINT64", 8: "FLOAT", 9: "DOUBL
E"
    }
```

```
    param_type = type_names.get(msg.param_type, f"UNKNOWN({msg.para
m_type})")

    print("\n  Received PARAM_VALUE:")
    print(f"    ID: {param_id}")
    print(f"    Value: {param_value}")
    print(f"    Type: {param_type}")
    print(f"    Index/Total: {msg.param_index}/{msg.param_count}")

    return {
        'id': param_id,
        'value': param_value,
        'type': param_type,
        'index': msg.param_index,
        'count': msg.param_count
    }

def receive_all_param_values(vehicle, timeout=10.0):
    """接收所有参数值，直到收到 param_count 条消息后退出循环"""
    received_params = []
    expected_count = None
    print(f"  Waiting for all PARAM_VALUE messages...")

    while True:
        msg = vehicle.recv_match(type='PARAM_VALUE', blocking=True,
timeout=timeout)

        if not msg:
            print(f"  Timeout: Received {len(received_params)} para
meters (expected {expected_count if expected_count else 'unknow
n'})")
            break

        param_id = msg.param_id
        param_value = msg.param_value

        # 参数类型映射
        type_names = {
            0: "INT8", 1: "UINT8", 2: "INT16", 3: "UINT16", 4: "INT
32",
            5: "UINT32", 6: "INT64", 7: "UINT64", 8: "FLOAT", 9: "D
OUBLE"
        }
        param_type = type_names.get(msg.param_type, f"UNKNOWN({msg.
param_type})")
```

```
            # 首次收到消息时记录总参数数量
            if expected_count is None:
                expected_count = msg.param_count
                print(f"  Total parameters to receive: {expected_coun
t}")

            print(f"  [{msg.param_index}/{expected_count}] {param_id} =
{param_value} ({param_type})")

            received_params.append({
                'id': param_id,
                'value': param_value,
                'type': param_type,
                'index': msg.param_index,
                'count': msg.param_count
            })

            # 当已接收数量达到预期总数时退出循环
            if len(received_params) >= expected_count:
                print(f"\n  Successfully received all {expected_count}
parameters")
                break

    return received_params

def main():
    print("  Starting MAVLink PARAM_SET Tool (UDP Listening Mode)")
    print("------------------------------------------------------------")
    print("This script listens for MAVLink on ALL interfaces:1455
0")
    print("Ensure your vehicle is configured to send to this port")
    print("------------------------------------------------------------
\n")

    vehicle = connect_to_vehicle()

    # 操作选择
    print("\nAvailable operations:")
    print("1. Set parameter value")
    print("2. Read parameter value (by ID or index)")
    print("3. Request all parameters (PARAM_REQUEST_LIST)")
    choice = input("\nSelect operation [1-3]: ").strip()

    if choice == '1':
```

```
        param_id = input("Enter parameter ID to set: ").strip()
        param_value = float(input("Enter new value: ").strip())

        print("\nParameter types:")
        print("  8 = FLOAT (most common)")
        print("  9 = DOUBLE")
        print("  4 = INT32")
        print("  1 = UINT8")
        param_type = int(input("Enter type number [8]: ").strip() or 8)

        send_param_set(vehicle, param_id, param_value, param_type)
        receive_param_value(vehicle)

    elif choice == '2':
        # mode = input("Read by [id/index]: ").lower()
        mode = 'id'
        if mode == 'id':
            param_id = input("Enter parameter ID: ").strip()
            send_param_request_read(vehicle, param_id=param_id)
            receive_param_value(vehicle)
        elif mode == 'index':
            param_index = int(input("Enter parameter index: "))
            send_param_request_read(vehicle, param_index=param_index)
            receive_param_value(vehicle)
        else:
            print("Invalid mode. Use 'id' or 'index'")

    elif choice == '3':
        print("\n  Requesting all parameters...")
        send_param_request_list(vehicle)
        all_params = receive_all_param_values(vehicle)
        print(f"\n  Total parameters received: {len(all_params)}")

    else:
        print("Invalid choice. Select 1, 2, or 3.")


if __name__ == "__main__":
    main()
```

示例使用说明

获取

运行后，成功建立连接

```
🚀 Starting MAVLink PARAM_SET Tool (UDP Listening Mode)
--------------------------------------------------
This script listens for MAVLink on ALL interfaces:14550
Ensure your vehicle is configured to send to this port
--------------------------------------------------

Listening for MAVLink on: udpin:0.0.0.0:14550
Waiting for HEARTBEAT (timeout: 10s)...
✅ Heartbeat received from system 3, component 0

Available operations:
1. Set parameter value
2. Read parameter value (by ID or index)
3. Request all parameters (PARAM_REQUEST_LIST)
```

输入“2”选择读取参数

```
Select operation [1-3]: 2
```

输入对应参数“GIMBAL_SMOOTH_P”/“GIMBAL_SMOOTH_R”/“GIMBAL_SMOOTH_Y”

```
Enter parameter ID: GIMBAL_SMOOTH_P
```

成功返回该参数对应的数值

```
📥 Received PARAM_VALUE:
  🆔 ID: GIMBAL_SMOOTH_P
  🔢 Value: 1.0
```

设置

运行后，成功建立连接

```
🚀 Starting MAVLink PARAM_SET Tool (UDP Listening Mode)
--------------------------------------------------
This script listens for MAVLink on ALL interfaces:14550
Ensure your vehicle is configured to send to this port
--------------------------------------------------

Listening for MAVLink on: udpin:0.0.0.0:14550
Waiting for HEARTBEAT (timeout: 10s)...
✅ Heartbeat received from system 3, component 0

Available operations:
1. Set parameter value
2. Read parameter value (by ID or index)
3. Request all parameters (PARAM_REQUEST_LIST)
```

输入“1”选择设置参数

```
Select operation [1-3]: 1
```

输入对应参数“GIMBAL_SMOOTH_P”/“GIMBAL_SMOOTH_R”/“GIMBAL_SMOOTH_Y”

```
Enter parameter ID to set: GIMBAL_SMOOTH_P
```

输入要设置的数值

```
Enter new value: 2
```

选择参数类型“1”

```
Parameter types:
  9 = DOUBLE
  4 = INT32
  1 = UINT8
Enter type number [8]: 1
```

成功返回设置值

```
📥 Received PARAM_VALUE:
  🆔 ID: GIMBAL_SMOOTH_P
  🔢 Value: 2.0
```

恢复云台的出厂设置

发送

COMMAND_LONG

示例

```
import time
from pymavlink import mavutil
from enum import IntEnum


# MAV_CMD 执行结果映射
MAV_RESULT_NAMES = {
    mavutil.mavlink.MAV_RESULT_ACCEPTED: "ACCEPTED",
    mavutil.mavlink.MAV_RESULT_TEMPORARILY_REJECTED: "TEMPORARILY_R
EJECTED",
    mavutil.mavlink.MAV_RESULT_DENIED: "DENIED",
    mavutil.mavlink.MAV_RESULT_UNSUPPORTED: "UNSUPPORTED",
    mavutil.mavlink.MAV_RESULT_FAILED: "FAILED",
    mavutil.mavlink.MAV_RESULT_IN_PROGRESS: "IN_PROGRESS",
}


def recv_command_ack(timeout=5):
    """接收并显示 command_ack 消息

    Args:
        timeout: 超时时间（秒）

    Returns:
        ack 消息对象，None 表示超时
    """
    print(f"等待 command_ack 响应 (超时 {timeout}s)...")

    ack = master.recv_match(type='COMMAND_ACK', blocking=True, time
out=timeout)

    if ack is None:
        print("错误: 命令响应超时")
        return None

    # 获取命令名称
    cmd_name = getattr(ack, 'command_name', None)
    if not cmd_name:
        # 尝试通过枚举查找命令名称
```

```
        try:
            cmd_name = mavutil.mavlink.enums.get('MAV_CMD', {}).get
(ack.command, {}).name
        except:
            cmd_name = str(ack.command)

    # 获取结果名称
    result_name = MAV_RESULT_NAMES.get(ack.result, f"UNKNOWN({ack.r
esult})")

    print("=" * 50)
    print(f"命令: {cmd_name} ({ack.command})")
    print(f"结果: {result_name}")
    print(f"进度: {ack.progress * 100:.0f}%" if ack.progress > 0 els
e "")
    # print(f"参数2: {ack.result_param2}")
    print(f"系统ID: {ack.target_system}")
    print(f"组件ID: {ack.target_component}")
    print("=" * 50)

    return ack


class MavDjiCmd(IntEnum):
    MAV_CMD_NAV_LAND = 21
    MAV_DJI_CMD_TRIGGER_FFC=40  # M4TD不支持
    MAV_DJI_CMD_START_RECORD_POINT_CLOUD=41 # M4TD不支持
    MAV_DJI_CMD_STOP_RECORD_POINT_CLOUD=42  # M4TD不支持
    MAV_DJI_CMD_RESTORE_FACTORY_SETTINGS=43 # 正常调用
    MAV_DJI_CMD_APPLY_HIGH_POWER_SYNC=44    # 正常调用
    MAV_DJI_CMD_APPLY_HIGH_POWER_SYNCV2=45  # 正常调用
    MAV_DJI_CMD_START_CONFIRM_LANDING=46    # 不等待确认
    MAV_DJI_CMD_START_FORCE_LANDING=47      # 正常
    MAV_DJI_CMD_EXECUTE_EMERGENCY_BRAKE_ACTION=48   # 正常
    MAV_DJI_CMD_CANCEL_EMERGENCY_BRAKE_ACTION=49    # 正常
    MAV_DJI_CMD_START_SLOW_ROTATE_MOTOR=50  # 正常调用
    MAV_DJI_CMD_STOP_SLOW_ROTATE_MOTOR=51   # 正常调用
    MAV_DJI_CMD_ARREST_FLYING=52            # 正常
    MAV_DJI_CMD_CANCEL_ARREST_FLYING=53     # 正常
    MAV_DJI_CMD_OUTPUT_HIGH_POWER=54        # 正常调用
    TUNNEL_PROTOCOL_ID_INJECT_HMS_ERROR=32770

timestamp = int(time.time() * 1000)
millis = int(time.time() * 1000)
```

```
master = mavutil.mavlink_connection('udpin:0.0.0.0:14550')
print("waiting for heartbeat")
master.wait_heartbeat()
print("heartbeat received")
boot_time = time.time()

def send_command(command, param1=0, param2=0, param3=0, param4=0, p
aram5=0, param6=0, param7=0, wait_ack=True, timeout=5):
    """发送 command_long 并可选接收 command_ack

    Args:
        command: 命令ID (如 MavDjiCmd.TRIGGER_FFC)
        param1~param7: 命令参数
        wait_ack: 是否等待并显示 ACK
        timeout: ACK 超时时间(秒)

    Returns:
        ack 消息对象,None 表示超时或未等待
    """
    print(f"发送命令: {command} ({int(command)})")
    master.mav.command_long_send(
        master.target_system,
        master.target_component,
        int(command),
        0,  # confirmation
        param1, param2, param3, param4, param5, param6, param7
    )

    if wait_ack:
        return recv_command_ack(timeout=timeout)
    return None


def ready():
    send_command(MavDjiCmd.MAV_DJI_CMD_RESTORE_FACTORY_SETTINGS)

ready()
```

示例使用说明

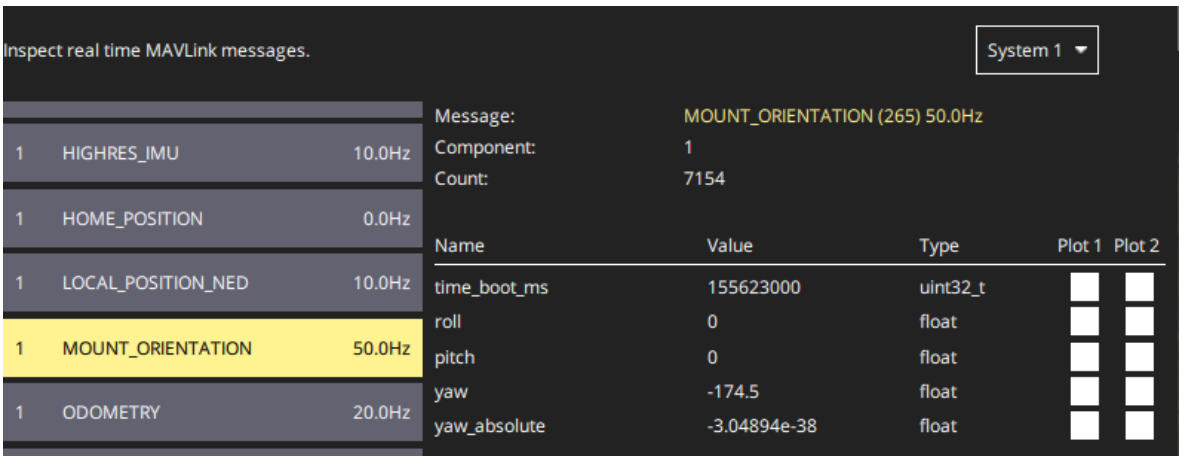
运行后，成功建立连接并发送指令

```
waiting for heartbeat
heartbeat received
发送命令: 43 (43)
等待 command_ack 响应 (超时 5s)...
================================================
命令: 43 (43)
结果: ACCEPTED
```

mavlink id 265 MOUNT_ORIENTATION

返回字段名称	类型	单位	描
time_boot_ms	uint32_t	ms	时
roll	float	deg	翻
pitch	float	deg	俯
yaw	float	deg	偏

结果（使用QGC查看）



GPS定位

mavlink id 24 GPS_RAW_INT

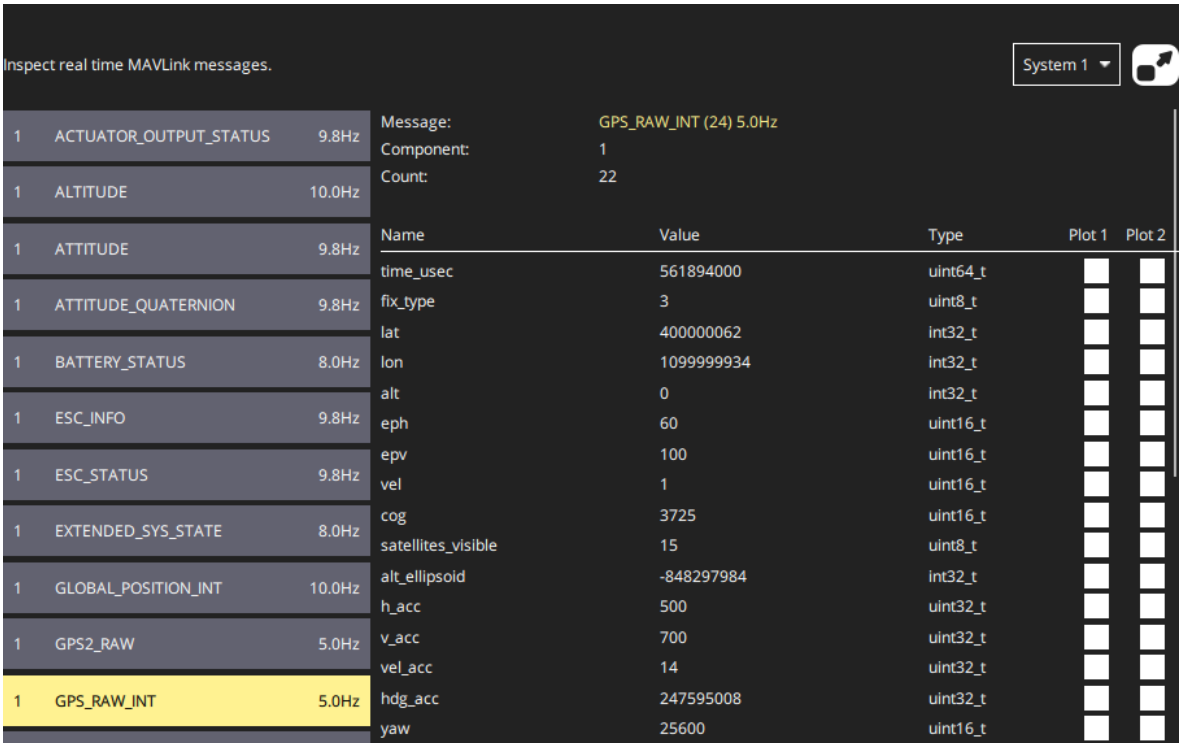
返回字段名称	类型	单位	描
time_usec	uint64_t	us	时
lat	int32_t	degE7	经
lon	int32_t	degE7	纬
alt	int32_t	mm	海
fix_type	uint8_t	GPS_FIX_TYPE	GI
eph	uint16_t	1E2	GI 水 (未
epv	uint16_t	1E2	GI 垂 (未
vel	uint16_t	cm/s	GI UI
cog	uint16_t		运 *1 35 设
h_acc	uint32_t	mm	位
v_acc	uint32_t	mm	高
vel_acc	uint32_t	mm/s	速
satellites_visible	uint16_t		可

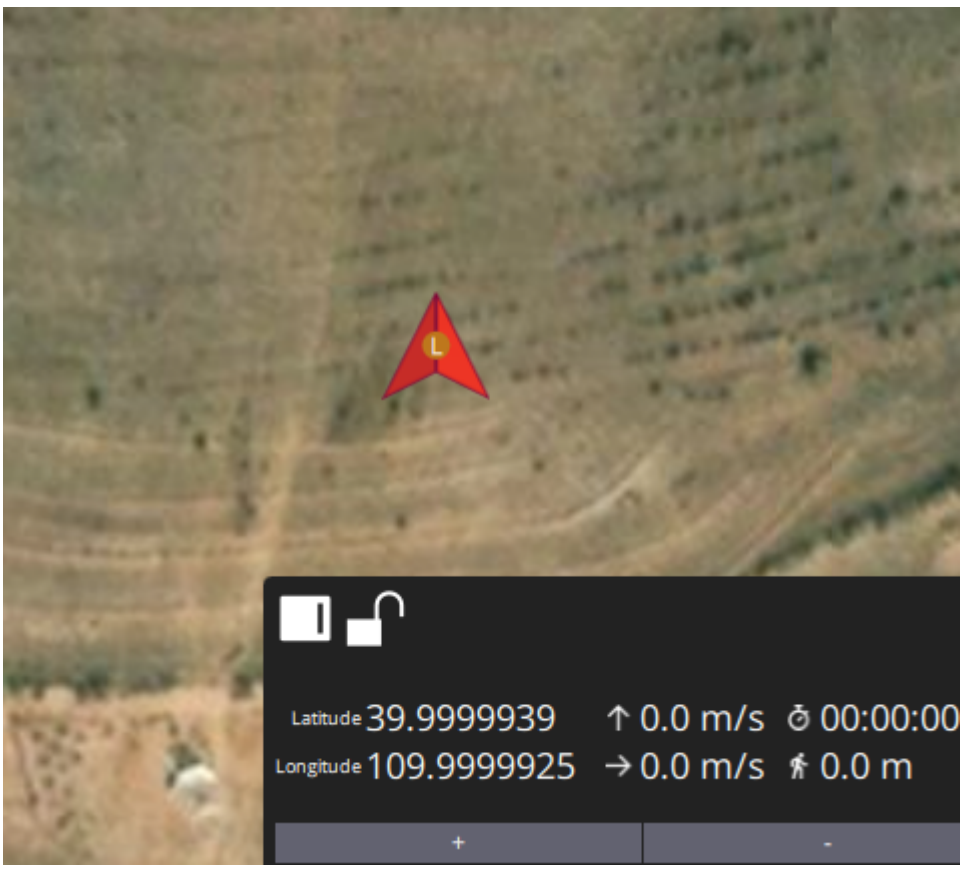
GPS_FIX_TYPE

GPS 定位类型

Value	Field Name	Description
0	GPS_FIX_TYPE_NO_GPS	没有连接 GP
1	GPS_FIX_TYPE_NO_FIX	没有位置信息
2	GPS_FIX_TYPE_2D_FIX	二维位置
3	GPS_FIX_TYPE_3D_FIX	三维位置
4	GPS_FIX_TYPE_DGPS	DGPS/SBAS
5	GPS_FIX_TYPE_RTK_FLOAT	RTK 浮点，
6	GPS_FIX_TYPE_RTK_FIXED	RTK 定位，
7	GPS_FIX_TYPE_STATIC	静态定位，这
8	GPS_FIX_TYPE_PPP	PPP，三维位

结果（使用QGC查看）


Inspect real time MAVLink messages.
System 1
1 ACTUATOR_OUTPUT_STATUS 9.8Hz
1 ALTITUDE 10.0Hz
1 ATTITUDE 9.8Hz
1 ATTITUDE_QUATERNION 9.8Hz
1 BATTERY_STATUS 8.0Hz
1 ESC_INFO 9.8Hz
1 ESC_STATUS 9.8Hz
1 EXTENDED_SYS_STATE 8.0Hz
1 GLOBAL_POSITION_INT 10.0Hz
1 GPS2_RAW 5.0Hz
1 GPS_RAW_INT 5.0Hz
Message: GPS_RAW_INT (24) 5.0Hz
Component: 1
Count: 22
Name Value Type Plot 1 Plot 2
time_usec 561894000 uint64_t
fix_type 3 uint8_t
lat 400000062 int32_t
lon 1099999934 int32_t
alt 0 int32_t
eph 60 uint16_t
epv 100 uint16_t
vel 1 uint16_t
cog 3725 uint16_t
satellites_visible 15 uint8_t
alt_ellipsoid -848297984 int32_t
h_acc 500 uint32_t
v_acc 700 uint32_t
vel_acc 14 uint32_t
hdg_acc 247595008 uint32_t
yaw 25600 uint16_t


Latitude 39.9999939
Longitude 109.9999925
0.0 m/s
0.0 m/s
00:00:00
0.0 m
+
-

回放下载

接口总览

功能	mavlink消息
分块下载选定的相机媒体文件列表	TUNNEL_PROTOCOL_ID_REQUEST
通过文件索引和文件类型下载选定的相机媒体文件	TUNNEL_PROTOCOL_ID_REQUEST
通过文件索引删除选定的相机媒体文件	TUNNEL_PROTOCOL_ID_REUQEST

示例

```
#!/usr/bin/env python3
"""
通过 MAVLink TUNNEL 消息获取相机媒体文件列表（分块下载）

协议说明:
  1. 发送 REQUEST_DOWNLOAD_FILE_LIST (32784) → 请求文件列表
     负载: uint16 sliceStartIndex + uint16 countPerSlice (4 字节, 小端序)
  2. 接收 DOWNLOAD_FILE_LIST (32785) → 文件列表中每个文件以分块方式返回
     分块协议: id(1) + chunk_seq(1) + data(126)
     相同 id 的块属于同一个文件, 按 chunk_seq 拼接后即为 T_DjiCameraManagerFileListInfo
  3. 发送 REQUEST_DOWNLOAD_FILE_TO_LOCAL (32786) → 请求下载文件到设备本地
     负载: uint32 fileIndex + uint8 fileType (5 字节, 小端序)
     注意: 文件内容存储在设备本地磁盘, 不通过 TUNNEL 回传

T_DjiCameraManagerFileListInfo 内存布局 (64-bit, 小端序, 无 pack):
  偏移      大小    字段
  0         256     fileName[256]
  256       4       fileSize (uint32)
  260       4       fileIndex (uint32)
  264       1       createTime.second
  265       1       createTime.minute
```

```
  266       1       createTime.hour
  267       1       createTime.day
  268       1       createTime.month
  269       1       (padding)
  270       2       createTime.year (uint16)
  272       4       type (E_DjiCameraMediaFileType, uint32)
  276       16      attributeData
  292       1       subFileListTotalNum
  293       3       (padding)
  296       8       subFileListInfo (指针, 不可用)
  总计: 304 字节

Usage:
    python tunnel_file_download.py [--list] [--download INDEX] [--f
ile-type TYPE]
"""

import argparse
import struct
import time
import os
from collections import defaultdict
from pymavlink import mavutil

# ================================================================
============
# MAVLink 组件 ID
# ================================================================
============
MAV_COMP_ID_DEFAULT = 12  # DJI 指定组件 ID

# ================================================================
============
# TUNNEL 协议 ID (与 mavlink_tool.h 中 TUNNEL_PROTOCOL_ID 枚举一致)
# ================================================================
============
TUNNEL_PROTOCOL_ID_REQUEST_DOWNLOAD_FILE_LIST    = 32784  # 请求文件列
表
TUNNEL_PROTOCOL_ID_DOWNLOAD_FILE_LIST            = 32785  # 文件列表响
应 (分块)
TUNNEL_PROTOCOL_ID_REQUEST_DOWNLOAD_FILE_TO_LOCAL = 32786  # 请求下
载文件到本地

# ================================================================
============
```

```
# 分片参数（与 E_DjiCameraManagerFileListCountPerSlice 枚举一致）
# ================================================================
===========
FILE_LIST_COUNT_60_PER_SLICE  = 60
FILE_LIST_COUNT_120_PER_SLICE = 120
FILE_LIST_COUNT_ALL_PER_SLICE = 0xFFFF

# ================================================================
===========
# T_DjiCameraManagerFileListInfo 结构体常量
# ================================================================
===========
DJI_FILE_NAME_SIZE_MAX = 256
# 结构体在 64-bit ARM/Linux 上的 sizeof（通过内存布局计算）
FILE_LIST_INFO_SIZE = 304  # bytes

# ================================================================
===========
# 媒体文件类型（E_DjiCameraMediaFileType）
# ================================================================
===========
MEDIA_FILE_TYPE_NAMES = {
    0:  "JPEG",
    1:  "DNG",
    2:  "MOV",
    3:  "MP4",
    4:  "PANORAMA_JPEG",
    5:  "TIFF_SEQ",
    6:  "MP4_IR",
    7:  "JPEG_IR",
    8:  "M4A",        # M4D 可见光 JPEG
    9:  "IR",         # M4D 红外
    10: "WIDE",       # M4D 广角
    14: "PHOTO_DNG",
    15: "PHOTO_TIF",
    16: "PHOTO_TIF_SEQ",
    17: "PHOTO_SSD_RAW",
    18: "PHOTO_SSD_RAW_MULTI",
    19: "VIDEO_ORG",
    20: "PHOTO_ORG",
    21: "AUDIO_WAV",
    22: "VIDEO_DNG",
    23: "VIDEO_SSD_RAW",
    24: "PHOTO_TIFF_8K",
    25: "PHOTO_MP4",
```

```
    26: "PHOTO_JPEG_8K",
    27: "PHOTO_DNG_8K",
    32: "ORG_FILE",  # >= 32 表示原始文件
}


def parse_file_list_info(data: bytes) -> dict:
    """
    从原始字节解析 T_DjiCameraManagerFileListInfo 结构体

    Args:
        data: 文件列表信息的原始字节 (至少 304 字节)

    Returns:
        dict: 解析后的文件信息
    """
    if len(data) < FILE_LIST_INFO_SIZE:
        raise ValueError(f"数据不足: 需要 {FILE_LIST_INFO_SIZE} 字节,
实际 {len(data)} 字节")

    offset = 0
    # fileName: 256 bytes, 以 null 结尾
    raw_name = data[offset:offset + DJI_FILE_NAME_SIZE_MAX]
    offset += DJI_FILE_NAME_SIZE_MAX
    null_pos = raw_name.find(b'\x00')
    file_name = raw_name[:null_pos].decode('utf-8', errors='replac
e') if null_pos >= 0 else raw_name.decode('utf-8', errors='replac
e')

    # fileSize: uint32_t LE
    file_size = struct.unpack_from('<I', data, offset)[0]
    offset += 4

    # fileIndex: uint32_t LE
    file_index = struct.unpack_from('<I', data, offset)[0]
    offset += 4

    # createTime:
    #   second(1) + minute(1) + hour(1) + day(1) + month(1) + pad
(1) + year(2) = 8 bytes
    second, minute, hour, day, month = struct.unpack_from('<BBBBB',
data, offset)
    offset += 5
    # padding 1 byte (for uint16 alignment)
    offset += 1
```

```
    year = struct.unpack_from('<H', data, offset)[0]
    offset += 2

    # type: uint32_t LE (E_DjiCameraMediaFileType)
    file_type = struct.unpack_from('<I', data, offset)[0]
    offset += 4

    # attributeData: 16 bytes (union, raw)
    attribute_data = data[offset:offset + 16]
    offset += 16

    # subFileListTotalNum: uint8_t
    sub_file_count = data[offset]
    offset += 1

    # padding 3 bytes + subFileListInfo pointer 8 bytes (跳过)
    # offset += 3 + 8

    type_name = MEDIA_FILE_TYPE_NAMES.get(file_type, f"UNKNOWN({file_type})")

    return {
        'file_name': file_name,
        'file_size': file_size,
        'file_index': file_index,
        'create_time': f"{year:04d}-{month:02d}-{day:02d} {hour:02d}:{minute:02d}:{second:02d}",
        'year': year, 'month': month, 'day': day, 'hour': hour, 'minute': minute, 'second': second,
        'file_type': file_type,
        'type_name': type_name,
        'sub_file_count': sub_file_count,
        'raw_bytes': data[:FILE_LIST_INFO_SIZE],
    }


def send_file_list_request(mavlink_conn, slice_start_index=0, count_per_slice=FILE_LIST_COUNT_ALL_PER_SLICE):
    """
    发送请求文件列表命令

    Args:
        mavlink_conn: MAVLink 连接对象
        slice_start_index: 分片起始索引 (0 表示从头开始)
        count_per_slice: 每片文件数量 (60/120/0xFFFF)
```

```
    """
    # 构建负载: uint16 sliceStartIndex + uint16 countPerSlice (小端序)
    payload = struct.pack('<HH', slice_start_index, count_per_slice)

    print(f"\n[REQUEST] 请求文件列表: sliceStartIndex={slice_start_index}, countPerSlice={count_per_slice}")

    send_tunnel_message(mavlink_conn, payload, TUNNEL_PROTOCOL_ID_REQUEST_DOWNLOAD_FILE_LIST)


def send_download_request(mavlink_conn, file_index, file_type=32):
    """
    发送下载文件到本地命令

    注意: 此命令仅触发设备端的本地下载, 文件内容不会通过 TUNNEL 回传。

    Args:
        mavlink_conn: MAVLink 连接对象
        file_index: 文件索引
        file_type: 文件类型 (>=32 表示原始文件, <32 表示子文件)
    """
    # 构建负载: uint32 fileIndex + uint8 fileType (小端序)
    payload = struct.pack('<IB', file_index, file_type)
    # 确保是 5 字节
    payload = payload.ljust(5, b'\x00')[:5]

    print(f"\n[REQUEST] 请求下载文件: fileIndex={file_index}, fileType={file_type}")

    send_tunnel_message(mavlink_conn, payload, TUNNEL_PROTOCOL_ID_REQUEST_DOWNLOAD_FILE_TO_LOCAL)


def send_tunnel_message(mavlink_conn, payload: bytes, message_type: int):
    """
    通过 MAVLink TUNNEL 消息发送数据

    Args:
        mavlink_conn: MAVLink 连接对象
        payload: 要发送的数据负载
        message_type: TUNNEL payload_type
```

```
    """
    if len(payload) > 128:
        raise ValueError(f"Payload 过大: {len(payload)} 字节 (最大 128)")

    # 填充到 128 字节
    payload_bytes = payload.ljust(128, b'\x00')

    mavlink_conn.mav.tunnel_send(
        target_system=0,
        target_component=MAV_COMP_ID_DEFAULT,
        payload_length=len(payload),
        payload_type=message_type,
        payload=payload_bytes
    )

    hex_str = ' '.join(f'{b:02X}' for b in payload[:len(payload)])
    print(f"  [TUNNEL SEND] type={message_type}, len={len(payload)}, data=[{hex_str}]")


def receive_file_list(mavlink_conn, timeout: float = 60.0):
    """
    接收并解析文件列表

    文件列表中的每个文件通过分块协议发送:
      - 每个块包含: id(1) + chunk_seq(1) + data(126)
      - 相同 id 的块属于同一个文件
      - 按 chunk_seq 拼接后得到 T_DjiCameraManagerFileListInfo 的原始字节

    Args:
        mavlink_conn: MAVLink 连接对象
        timeout: 接收超时时间(秒)

    Returns:
        list: 解析后的文件信息列表
    """
    print("\n" + "=" * 70)
    print("等待接收文件列表...")
    print("=" * 70)

    # 按消息 id 收集分块
    # { msg_id: { chunk_seq: bytes } }
    chunk_buffers = defaultdict(dict)
```

```
    start_time = time.time()
    received_count = 0
    file_count_estimate = None

    try:
        while True:
            elapsed = time.time() - start_time
            if elapsed >= timeout:
                print(f"\n接收超时 ({timeout}秒)")
                break

            msg = mavlink_conn.recv_match(type='TUNNEL', blocking=T
rue, timeout=1.0)

            if msg is None:
                continue

            payload_type = msg.payload_type

            # 只处理文件列表响应 (32785)
            if payload_type != TUNNEL_PROTOCOL_ID_DOWNLOAD_FILE_LIS
T:
                # 打印其他 TUNNEL 消息以帮助调试
                if payload_type != 0:
                    payload_bytes = bytes(msg.payload)
                    payload_len = msg.payload_length
                    actual_data = payload_bytes[:payload_len]
                    hex_str = ' '.join(f'{b:02X}' for b in actual_d
ata)
                    ascii_str = ''.join(chr(b) if 32 <= b < 127 els
e '.' for b in actual_data)
                    print(f"  [TUNNEL] 收到其他类型消息: type={payload
_type}, len={payload_len}")
                    print(f"    Hex: [{hex_str}]")
                    print(f"    ASCII: {ascii_str}")
                continue

            received_count += 1

            # 解析分块数据
            payload_bytes = bytes(msg.payload)
            payload_len = msg.payload_length

            if payload_len < 2:
```

```
                print(f"  [WARN] payload 太短: {payload_len} 字节, 跳过")
                continue

            # 分块头: id (1 byte) + chunk_seq (1 byte)
            chunk_id = payload_bytes[0]
            chunk_seq = payload_bytes[1]
            # 分块数据: 剩余的 payload
            chunk_data_size = min(payload_len - 2, 126)
            chunk_data = payload_bytes[2:2 + chunk_data_size]

            chunk_buffers[chunk_id][chunk_seq] = chunk_data

            # 计算总块数估算 (每个文件 304 字节 / 126 = 3 块)
            print(f"  [CHUNK] #{received_count}: id={chunk_id}, seq={chunk_seq}, "
                  f"data_len={len(chunk_data)}")

    except KeyboardInterrupt:
        print("\n\n用户中断接收")
    except Exception as e:
        print(f"\n接收异常: {e}")

    print(f"\n共接收 {received_count} 个分块, {len(chunk_buffers)} 个消息组")

    # ============================================================
    # 重组并解析每个文件的完整信息
    # ============================================================
    file_list = []

    # 按 id 排序以保证文件顺序 (id 是递增的)
    for chunk_id in sorted(chunk_buffers.keys()):
        chunks = chunk_buffers[chunk_id]

        # 按 chunk_seq 排序并拼接
        sorted_seqs = sorted(chunks.keys())
        full_data = b''
        for seq in sorted_seqs:
            full_data += chunks[seq]

        print(f"\n  [ASSEMBLE] id={chunk_id}: {len(sorted_seqs)} 块 → {len(full_data)} 字节")
```

```
        if len(full_data) < FILE_LIST_INFO_SIZE:
            print(f"  [WARN] id={chunk_id}: 数据不足 ({len(full_dat
a)} < {FILE_LIST_INFO_SIZE}), 跳过")
            continue

        try:
            info = parse_file_list_info(full_data[:FILE_LIST_INFO_S
IZE])
            file_list.append(info)
        except Exception as e:
            print(f"  [ERROR] id={chunk_id}: 解析失败: {e}")

    return file_list


def format_file_size(size_bytes: int) -> str:
    """格式化文件大小"""
    if size_bytes >= 1024 * 1024 * 1024:
        return f"{size_bytes / (1024 ** 3):.2f} GB"
    elif size_bytes >= 1024 * 1024:
        return f"{size_bytes / (1024 ** 2):.2f} MB"
    elif size_bytes >= 1024:
        return f"{size_bytes / 1024:.2f} KB"
    else:
        return f"{size_bytes} B"


def print_file_list(file_list: list):
    """打印文件列表"""
    if not file_list:
        print("\n[结果] 未发现任何文件 (SD 卡可能为空)")
        return

    print("\n" + "=" * 90)
    print(f"相机媒体文件列表 (共 {len(file_list)} 个文件)")
    print("=" * 90)
    print(f"{'#':>4}  {'索引':>7}  {'文件名':<30} {'大小':>10}  {'时
间':<20}  {'类型':>10}  {'子文件':>6}")
    print("-" * 90)

    for i, f in enumerate(file_list):
        size_str = format_file_size(f['file_size'])
        # 截断过长的文件名
        name = f['file_name'][:28] + '..' if len(f['file_name']) >
```

```
30 else f['file_name']
        print(f"{i:>4}  {f['file_index']:>7}  {name:<30} {size_str:
>10}  {f['create_time']:<20}  "
              f"{f['type_name']:>10}  {f['sub_file_count']:>6}")

    print("-" * 90)
    total_size = sum(f['file_size'] for f in file_list)
    print(f"总计: {len(file_list)} 个文件, {format_file_size(total_si
ze)}")


def interactive_mode(mavlink_conn, file_list: list):
    """
    交互模式: 允许用户选择下载文件

    Args:
        mavlink_conn: MAVLink 连接
        file_list: 文件列表
    """
    if not file_list:
        print("没有可下载的文件")
        return

    while True:
        print("\n" + "-" * 50)
        print("操作选项:")
        print("  d <编号>     - 下载指定文件 (例如: d 0)")
        print("  d <编号> <类型> - 指定文件类型下载 (>=32 原始文件)")
        print("  l            - 重新列出文件")
        print("  q            - 退出")
        print("-" * 50)

        try:
            cmd = input("> ").strip().lower()
        except (EOFError, KeyboardInterrupt):
            print("\n退出")
            break

        if cmd == 'q':
            break
        elif cmd == 'l':
            print_file_list(file_list)
            continue
        elif cmd.startswith('d'):
            parts = cmd.split()
```

```
            if len(parts) < 2:
                print("用法: d <编号> [类型]")
                continue

            try:
                idx = int(parts[1])
            except ValueError:
                print(f"无效的编号: {parts[1]}")
                continue

            if idx < 0 or idx >= len(file_list):
                print(f"编号超出范围 (0-{len(file_list) - 1})")
                continue

            file_type = 32  # 默认下载原始文件
            if len(parts) >= 3:
                try:
                    file_type = int(parts[2])
                except ValueError:
                    print(f"无效的类型: {parts[2]}")
                    continue

            f = file_list[idx]
            print(f"\n即将下载: [{idx}] {f['file_name']} (index={f
['file_index']}, type={file_type})")
            confirm = input("确认下载? (y/N): ").strip().lower()
            if confirm == 'y':
                send_download_request(mavlink_conn, f['file_inde
x'], file_type)
                print("下载命令已发送 (文件将存储在设备本地磁盘)")
                # 短暂等待接收确认/状态消息
                print("等待设备响应...")
                start = time.time()
                while time.time() - start < 5:
                    msg = mavlink_conn.recv_match(type='TUNNEL', bl
ocking=True, timeout=0.5)
                    if msg and msg.payload_type != 0:
                        payload = bytes(msg.payload)
                        text = payload[:msg.payload_length].decode
('utf-8', errors='replace').strip('\x00')
                        if text:
                            print(f"  [响应] type={msg.payload_typ
e}: {text}")
        else:
            print(f"未知命令: {cmd}")
```

```
def main():
    parser = argparse.ArgumentParser(
        description='MAVLink TUNNEL 相机媒体文件列表获取工具',
        formatter_class=argparse.RawDescriptionHelpFormatter,
        epilog="""
示例:
  %(prog)s                              # 列出所有文件并进入交互模式
  %(prog)s --list                       # 仅列出文件
  %(prog)s --list --slice 0 --count 60   # 每片 60 个文件
  %(prog)s --download 5 --file-type 32   # 下载第 5 个文件的原始文件
  %(prog)s --download 3 --file-type 0    # 下载第 3 个文件的 JPEG
  %(prog)s --connect udpin:0.0.0.0:14550  # 指定连接地址
        """
    )
    parser.add_argument('--connect', '-c', default='udpin:0.0.0.0:14550',
                        help='MAVLink 连接字符串 (默认: udpin:0.0.0.0:14550)')
    parser.add_argument('--list', '-l', action='store_true',
                        help='列出相机媒体文件列表')
    parser.add_argument('--download', '-d', type=int, default=None,
                        help='下载指定编号的文件 (与 --list 配合使用, 或单独使用时直接按 fileIndex 下载)')
    parser.add_argument('--file-type', '-t', type=int, default=32,
                        help='文件类型 (>=32 原始文件, <32 子文件, 默认: 32)')
    parser.add_argument('--slice', type=int, default=0,
                        help='文件列表分片起始索引 (默认: 0)')
    parser.add_argument('--count', type=int, default=FILE_LIST_COUNT_ALL_PER_SLICE,
                        help=f'每片文件数量 (60/120/{FILE_LIST_COUNT_ALL_PER_SLICE}=全部, 默认: {FILE_LIST_COUNT_ALL_PER_SLICE})')
    parser.add_argument('--timeout', type=float, default=15.0,
                        help='接收文件列表的超时时间(秒) (默认: 15)')
    parser.add_argument('--no-interactive', '-n', action='store_true',
                        help='禁用交互模式 (仅输出文件列表)')
    args = parser.parse_args()

    print("=" * 70)
    print("MAVLink TUNNEL 相机媒体文件列表获取工具")
    print("=" * 70)
```

```
    # ============================================================
============
    # 建立 MAVLink 连接
    # ============================================================
============
    print(f"\n连接: {args.connect}")
    try:
        mavlink_conn = mavutil.mavlink_connection(args.connect)
    except Exception as e:
        print(f"连接失败: {e}")
        return 1

    # 等待心跳包
    print("等待心跳包...")
    heartbeat = mavlink_conn.wait_heartbeat(timeout=10)
    if heartbeat:
        print(f"已连接! sysid={heartbeat.get_srcSystem()}, compid={h
eartbeat.get_srcComponent()}")
    else:
        print("警告: 未收到心跳包, 继续尝试...")

    file_list = None

    try:
        # ============================================================
============
        # 模式 1: 仅下载文件 (直接使用 fileIndex)
        # ============================================================
============
        if args.download is not None and not args.list:
            send_download_request(mavlink_conn, args.download, arg
s.file_type)
            print("下载命令已发送 (文件将存储在设备本地磁盘)")
            print("等待设备响应...")
            start = time.time()
            while time.time() - start < 5:
                msg = mavlink_conn.recv_match(type='TUNNEL', blocki
ng=True, timeout=0.5)
                if msg and msg.payload_type not in (0, TUNNEL_PROTO
COL_ID_DOWNLOAD_FILE_LIST):
                    payload = bytes(msg.payload)
                    text = payload[:msg.payload_length].decode('utf
-8', errors='replace').strip('\x00')
                    if text:
                        print(f"  [响应] type={msg.payload_type}: {t
```

```
ext}")
            return 0

        # ============================================================
============
        # 模式 2: 列出文件列表
        # ============================================================
============
        send_file_list_request(mavlink_conn, args.slice, args.coun
t)

        # 接收文件列表
        file_list = receive_file_list(mavlink_conn, timeout=args.ti
meout)

        # 打印文件列表
        print_file_list(file_list)

        # ============================================================
============
        # 模式 3: 列出后下载指定文件
        # ============================================================
============
        if args.download is not None and file_list and args.downloa
d < len(file_list):
            f = file_list[args.download]
            print(f"\n下载文件: [{args.download}] {f['file_name']}
(index={f['file_index']}, type={args.file_type})")
            send_download_request(mavlink_conn, f['file_index'], ar
gs.file_type)
            print("下载命令已发送 (文件将存储在设备本地磁盘)")

        # ============================================================
============
        # 交互模式
        # ============================================================
============
        if not args.no_interactive and file_list is not None:
            interactive_mode(mavlink_conn, file_list)

    except KeyboardInterrupt:
        print("\n\n用户中断")
    except Exception as e:
        print(f"\n错误: {e}")
        import traceback
```

```
        traceback.print_exc()
        return 1
    finally:
        mavlink_conn.close()
        print("连接已关闭")

    return 0


if __name__ == "__main__":
    exit(main())
```

电源管理

mavlink id 147 BATTERY_STATUS

电池信息。更新 GCS 并结合飞行控制器的电池状态。

返回字段名称	类型	单位	描述
id	uint8_t		电池编号
battery_function	uint8_t	MAV_BATTERY_FUNCTION	电池所负载的功
temperature	int16_t	℃	当前电池温度。 温度未知。
voltages	uint16_t[10]	mV	电池 1 至 10 的 如果单个电池的 则整体电池电压 其他电压设为 U
current_battery	int16_t	A	当前电量多少安 自动驾驶仪无法
current_consumed	int32_t	mAh	消耗电量。-1： 自动驾驶无法提
battery_remaining	int8_t	%	剩余电池电量百 100]，-1： 自动驾驶仪不会

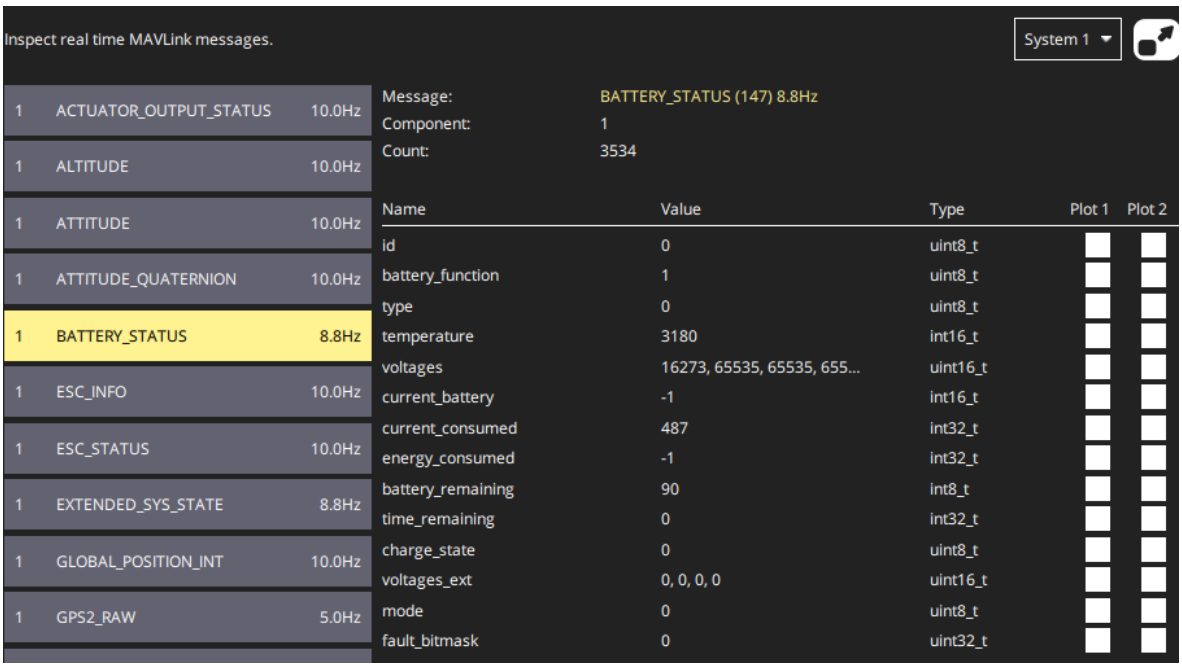
MAV_BATTERY_FUNCTION

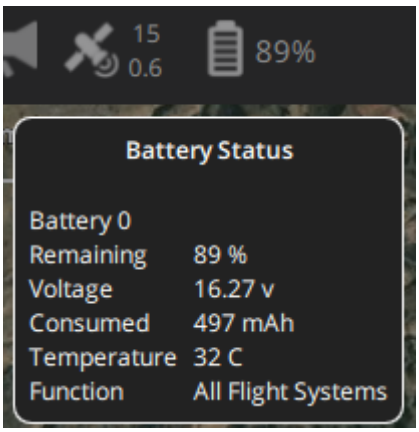
电池所负载的功能模块类型枚举。

Value	Field Name	Descripti
0	MAV_BATTERY_FUNCTION_UNKNOWN	电池负载
1	MAV_BATTERY_FUNCTION_ALL	电池支持

2	MAV_BATTERY_FUNCTION_PROPULSION	推进系统
3	MAV_BATTERY_FUNCTION_AVIONICS	航空电子
4	MAV_BATTERY_FUNCTION_PAYLOAD	载荷设备

结果（使用QGC查看）





向飞机申请高功率供电（同步阻塞方式）

MAV_DJI_CMD_APPLY_HIGH_POWER_SYNC
发送
COMMAND_LONG
接收

类型	枚举值	说明
bool	MAV_RESULT_ACCEPTED	成功
bool	MAV_RESULT_ACCEPTED	失败

示例

```
import time
from pymavlink import mavutil
from enum import IntEnum


# MAV_CMD 执行结果映射
MAV_RESULT_NAMES = {
    mavutil.mavlink.MAV_RESULT_ACCEPTED: "ACCEPTED",
    mavutil.mavlink.MAV_RESULT_TEMPORARILY_REJECTED: "TEMPORARILY_R
EJECTED",
    mavutil.mavlink.MAV_RESULT_DENIED: "DENIED",
    mavutil.mavlink.MAV_RESULT_UNSUPPORTED: "UNSUPPORTED",
    mavutil.mavlink.MAV_RESULT_FAILED: "FAILED",
    mavutil.mavlink.MAV_RESULT_IN_PROGRESS: "IN_PROGRESS",
}


def recv_command_ack(timeout=5):
    """接收并显示 command_ack 消息

    Args:
        timeout: 超时时间（秒）

    Returns:
        ack 消息对象，None 表示超时
    """
    print(f"等待 command_ack 响应 (超时 {timeout}s)...")

    ack = master.recv_match(type='COMMAND_ACK', blocking=True, time
out=timeout)

    if ack is None:
        print("错误: 命令响应超时")
        return None

    # 获取命令名称
```

```
    cmd_name = getattr(ack, 'command_name', None)
    if not cmd_name:
        # 尝试通过枚举查找命令名称
        try:
            cmd_name = mavutil.mavlink.enums.get('MAV_CMD', {}).get
(ack.command, {}).name
        except:
            cmd_name = str(ack.command)

    # 获取结果名称
    result_name = MAV_RESULT_NAMES.get(ack.result, f"UNKNOWN({ack.r
esult})")

    print("=" * 50)
    print(f"命令: {cmd_name} ({ack.command})")
    print(f"结果: {result_name}")
    print(f"进度: {ack.progress * 100:.0f}%" if ack.progress > 0 els
e "")
    # print(f"参数2: {ack.result_param2}")
    print(f"系统ID: {ack.target_system}")
    print(f"组件ID: {ack.target_component}")
    print("=" * 50)

    return ack


class MavDjiCmd(IntEnum):
    MAV_CMD_NAV_LAND = 21
    MAV_DJI_CMD_TRIGGER_FFC=40  # M4TD不支持
    MAV_DJI_CMD_START_RECORD_POINT_CLOUD=41 # M4TD不支持
    MAV_DJI_CMD_STOP_RECORD_POINT_CLOUD=42  # M4TD不支持
    MAV_DJI_CMD_RESTORE_FACTORY_SETTINGS=43 # 正常调用
    MAV_DJI_CMD_APPLY_HIGH_POWER_SYNC=44    # 正常调用
    MAV_DJI_CMD_APPLY_HIGH_POWER_SYNCV2=45  # 正常调用
    MAV_DJI_CMD_START_CONFIRM_LANDING=46    # 不等待确认
    MAV_DJI_CMD_START_FORCE_LANDING=47      # 正常
    MAV_DJI_CMD_EXECUTE_EMERGENCY_BRAKE_ACTION=48   # 正常
    MAV_DJI_CMD_CANCEL_EMERGENCY_BRAKE_ACTION=49    # 正常
    MAV_DJI_CMD_START_SLOW_ROTATE_MOTOR=50  # 正常调用
    MAV_DJI_CMD_STOP_SLOW_ROTATE_MOTOR=51   # 正常调用
    MAV_DJI_CMD_ARREST_FLYING=52            # 正常
    MAV_DJI_CMD_CANCEL_ARREST_FLYING=53     # 正常
    MAV_DJI_CMD_OUTPUT_HIGH_POWER=54        # 正常调用
    TUNNEL_PROTOCOL_ID_INJECT_HMS_ERROR=32770
```

```
timestamp = int(time.time() * 1000)
millis = int(time.time() * 1000)

master = mavutil.mavlink_connection('udpin:0.0.0.0:14550')
print("waiting for heartbeat")
master.wait_heartbeat()
print("heartbeat received")
boot_time = time.time()

def send_command(command, param1=0, param2=0, param3=0, param4=0, p
aram5=0, param6=0, param7=0, wait_ack=True, timeout=5):
    """发送 command_long 并可选接收 command_ack

    Args:
        command: 命令ID (如 MavDjiCmd.TRIGGER_FFC)
        param1~param7: 命令参数
        wait_ack: 是否等待并显示 ACK
        timeout: ACK 超时时间(秒)

    Returns:
        ack 消息对象,None 表示超时或未等待
    """
    print(f"发送命令: {command} ({int(command)})")
    master.mav.command_long_send(
        master.target_system,
        master.target_component,
        int(command),
        0,  # confirmation
        param1, param2, param3, param4, param5, param6, param7
    )

    if wait_ack:
        return recv_command_ack(timeout=timeout)
    return None


def ready():
    send_command(MavDjiCmd.MAV_DJI_CMD_OUTPUT_HIGH_POWER)

ready()
```

自定义HMS功能

向应用程序上报自定义 HMS 错误码

TUNNEL_PROTOCOL_ID_INJECT_HMS_ERROR

mavlink消息：

接收TUNNEL_PROTOCOL_ID_INJECT_HMS_ERROR

参数：

名称	枚举值	描述
DJI_HMS_ERROR_LEVEL_NONE	0	无错误。
DJI_HMS_ERROR_LEVEL_HINT	1	提示级（Hint）
DJI_HMS_ERROR_LEVEL_WARN	2	警告级（Warning）
DJI_HMS_ERROR_LEVEL_CRITICAL	3	严重级（Critical）
DJI_HMS_ERROR_LEVEL_FATAL	4	致命级（Fatal）

示例

```
#!/usr/bin/env python3
"""
通过 MAVLink TUNNEL 消息发送 DJI HMS 结构体数据
HmsErrCode 和 HmsAlarmEnhancedCtrl

Usage:
    python tunnel_hms_send.py
"""

import struct
import time
from pymavlink import mavutil

# MAVLink 组件 ID (DJI 指定)
MAV_COMP_ID_HMS = 12

# HMS 目标组件
```

```
TARGET_COMPONENT = MAV_COMP_ID_HMS

def create_hmserrcode_struct(error_code: int, error_level: int, ena
ble: bool, error_id: int) -> bytes:
    """
    创建 HmsErrCode 结构体 (7字节, packed)

    typedef struct Hms_ErrorCode_Ctrl{
        uint32_t errorCode;       // 4 bytes
        uint32_t errorLevel;       // 4 byte (E_DjiHmsErrorLevel)
        bool enable;              // 1 byte
        uint8_t id;               // 1 byte
    } __attribute__((packed)) HmsErrCode;
    """
    return struct.pack('<IIbB', error_code, error_level, 1 if enabl
e else 0, error_id)


def create_hmsalarminfo_struct(
    alarm_type: int,
    times: int,
    interval: int,
    action: int,
    alarm_id: int
) -> bytes:
    """
    创建 HmsAlarmEnhancedCtrl 结构体 (6字节, packed)

    typedef struct Hms_Alarm_Enhanced_Ctrl{
        uint32_t type;             // 4 byte (E_DjiHmsAlarmEnhanced
Type)
        int8_t times;              // 1 byte
        uint8_t reserved;          // 1 byte
        int16_t interval;          // 2 bytes
        uint32_t action;           // 4 byte (E_DjiHmsAlarmEnhanced
Action)
        uint8_t id;               // 1 byte
    } __attribute__((packed)) HmsAlarmEnhancedCtrl;
    """
    return struct.pack('<IbbhIB', alarm_type, times, 0, interval, a
ction, alarm_id)


def send_tunnel_message(mavlink_conn, payload: bytes, message_type:
int = 0):
```

```
    """
    通过 MAVLink TUNNEL 消息发送数据

    Args:
        mavlink_conn: MAVLink 连接对象
        payload: 要发送的数据负载 (最多 228 字节)
        message_type: 消息类型标识 (可选)
    """
    if len(payload) > 228:
        raise ValueError(f"Payload too large: {len(payload)} bytes
(max 228)")

    # 填充到固定长度 (确保小端序)
    payload_bytes = payload.ljust(228, b'\x00')

    # 发送 TUNNEL 消息
    # MAVLINK_MSG_ID_TUNNEL = 110
    mavlink_conn.mav.tunnel_send(
        target_system=0,            # 广播到所有系统
        target_component=TARGET_COMPONENT,
        payload_length=len(payload),
        payload_type=message_type,
        payload=payload_bytes
    )


def send_hms_error_code(mavlink_conn,message_type: int = 0):
    """
    发送 HmsErrCode 结构体
    示例数据:
        errorCode: 0x1E020001 (设备错误)
        errorLevel: DJI_HMS_ERROR_LEVEL_WARN (2)
        enable: True
        id: 1
    """
    payload = create_hmserrcode_struct(
        error_code=0x1E020001,     # 示例错误码
        error_level=2,              # WARN 级别
        enable=True,
        error_id=1
    )

    # message_type 用于标识 HMS 错误码消息类型
    if message_type==32771:
        payload = create_hmserrcode_struct(
```

```
            error_code=0x1E020001,     # 示例错误码
            error_level=2,              # WARN 级别
            enable=False,
            error_id=1
        )
    send_tunnel_message(mavlink_conn, payload, message_type)
    print(f"[HMS] 发送 HmsErrCode: errorCode=0x1E020001, level=2, en
able=True, id=1")


def send_hms_alarm_enhanced(mavlink_conn):
    """
    发送 HmsAlarmEnhancedCtrl 结构体
    示例数据:
        type: SHAKE_MOTOR (1)
        times: 3
        interval: 500ms
        action: START (1)
        id: 1
    """
    payload = create_hmsalarminfo_struct(
        alarm_type=3,               # SHAKE_MOTOR
        times=3,                    # 振动3次
        interval=500,               # 间隔500ms
        action=1,                    # START
        alarm_id=1
    )

    # message_type 用于标识 HMS 告警控制消息类型
    HMS_ALARM_ENHANCED_TYPE = 32772
    send_tunnel_message(mavlink_conn, payload, HMS_ALARM_ENHANCED_T
YPE)
    print(f"[HMS] 发送 HmsAlarmEnhancedCtrl: type=1, times=3, interv
al=500, action=1, id=1")


def main():
    print("=" * 60)
    print("MAVLink HMS TUNNEL 消息发送工具")
    print("=" * 60)

    # 创建 MAVLink UDP 连接 (监听 14550 端口)
    # udpin:0.0.0.0:14550 表示监听所有网卡地址的 14550 端口
    connection_string = "udpin:0.0.0.0:14550"
    print(f"\n正在连接到: {connection_string}")
```

```
    try:
        mavlink_conn = mavutil.mavlink_connection(connection_strin
g)

        # 等待心跳包以确认连接
        print("等待心跳包...")
        heartbeat = mavlink_conn.wait_heartbeat(timeout=10)
        if heartbeat:
            print(f"已连接! ")
        else:
            print("警告: 未收到心跳包，继续发送...")

        print("\n" + "-" * 60)
        print("开始发送 HMS 消息...")
        print("-" * 60)

        # 发送 HmsErrCode 结构体
        send_hms_error_code(mavlink_conn,32770)
        time.sleep(3)  # 间隔1秒

        # 发送 HmsAlarmEnhancedCtrl 结构体
        send_hms_alarm_enhanced(mavlink_conn)

        print("\n" + "-" * 60)
        print("消息发送完成!")
        print("-" * 60)

        # 保持连接一段时间以便接收响应
        print("\n等待响应 (5秒)...")
        time.sleep(5)

        mavlink_conn.close()
        print("连接已关闭")

    except Exception as e:
        print(f"错误: {e}")
        raise


if __name__ == "__main__":
    main()
```

示例使用说明

运行后，成功建立连接并发送指令

```
正在连接到: udpin:0.0.0.0:14550
等待心跳包...
已连接!

------------------------------------------------------------
开始发送 HMS 消息...
------------------------------------------------------------
[HMS] 发送 HmsErrCode: errorCode=0x1E020001, level=2, enable=True, id=1
[HMS] 发送 HmsAlarmEnhancedCtrl: type=1, times=3, interval=500, action=1, id=1

------------------------------------------------------------
消息发送完成!
------------------------------------------------------------
```

清除应用程序上显示的自定义 HMS 错误码

TUNNEL_PROTOCOL_ID_ELIMINATE_HMS_ERROR

设置

mavlink消息：

接收TUNNEL_PROTOCOL_ID_ELIMINATE_HMS_ERROR

参数：

名称	枚举值	描述
DJI_HMS_ERROR_LEVEL_NONE	0	无错误。
DJI_HMS_ERROR_LEVEL_HINT	1	提示级（Hint）
DJI_HMS_ERROR_LEVEL_WARN	2	警告级（Warning）
DJI_HMS_ERROR_LEVEL_CRITICAL	3	严重级（Critical）
DJI_HMS_ERROR_LEVEL_FATAL	4	致命级（Fatal）

示例

```
#!/usr/bin/env python3
"""
通过 MAVLink TUNNEL 消息发送 DJI HMS 结构体数据
HmsErrCode 和 HmsAlarmEnhancedCtrl
```

```
Usage:
    python tunnel_hms_send.py
"""

import struct
import time
from pymavlink import mavutil

# MAVLink 组件 ID (DJI 指定)
MAV_COMP_ID_HMS = 12

# HMS 目标组件
TARGET_COMPONENT = MAV_COMP_ID_HMS

def create_hmserrcode_struct(error_code: int, error_level: int, ena
ble: bool, error_id: int) -> bytes:
    """
    创建 HmsErrCode 结构体 (7字节, packed)

    typedef struct Hms_ErrorCode_Ctrl{
        uint32_t errorCode;        // 4 bytes
        uint32_t errorLevel;        // 4 byte (E_DjiHmsErrorLevel)
        bool enable;               // 1 byte
        uint8_t id;                // 1 byte
    } __attribute__((packed)) HmsErrCode;
    """
    return struct.pack('<IIbB', error_code, error_level, 1 if enabl
e else 0, error_id)


def create_hmsalarminfo_struct(
    alarm_type: int,
    times: int,
    interval: int,
    action: int,
    alarm_id: int
) -> bytes:
    """
    创建 HmsAlarmEnhancedCtrl 结构体 (6字节, packed)

    typedef struct Hms_Alarm_Enhanced_Ctrl{
        uint32_t type;             // 4 byte (E_DjiHmsAlarmEnhanced
Type)
        int8_t times;              // 1 byte
        uint8_t reserved;          // 1 byte
```

```
        int16_t interval;           // 2 bytes
        uint32_t action;            // 4 byte (E_DjiHmsAlarmEnhanced
Action)
        uint8_t id;                // 1 byte
    } __attribute__((packed)) HmsAlarmEnhancedCtrl;
    """
    return struct.pack('<IbbhIB', alarm_type, times, 0, interval, a
ction, alarm_id)


def send_tunnel_message(mavlink_conn, payload: bytes, message_type:
int = 0):
    """
    通过 MAVLink TUNNEL 消息发送数据

    Args:
        mavlink_conn: MAVLink 连接对象
        payload: 要发送的数据负载 (最多 228 字节)
        message_type: 消息类型标识 (可选)
    """
    if len(payload) > 228:
        raise ValueError(f"Payload too large: {len(payload)} bytes
(max 228)")

    # 填充到固定长度 (确保小端序)
    payload_bytes = payload.ljust(228, b'\x00')

    # 发送 TUNNEL 消息
    # MAVLINK_MSG_ID_TUNNEL = 110
    mavlink_conn.mav.tunnel_send(
        target_system=0,            # 广播到所有系统
        target_component=TARGET_COMPONENT,
        payload_length=len(payload),
        payload_type=message_type,
        payload=payload_bytes
    )


def send_hms_error_code(mavlink_conn,message_type: int = 0):
    """
    发送 HmsErrCode 结构体
    示例数据:
        errorCode: 0x1E020001 (设备错误)
        errorLevel: DJI_HMS_ERROR_LEVEL_WARN (2)
        enable: True
```

```
        id: 1
    """
    payload = create_hmserrcode_struct(
        error_code=0x1E020001,     # 示例错误码
        error_level=2,              # WARN 级别
        enable=True,
        error_id=1
    )

    # message_type 用于标识 HMS 错误码消息类型
    if message_type==32771:
        payload = create_hmserrcode_struct(
            error_code=0x1E020001,     # 示例错误码
            error_level=2,              # WARN 级别
            enable=False,
            error_id=1
        )
    send_tunnel_message(mavlink_conn, payload, message_type)
    print(f"[HMS] 发送 HmsErrCode: errorCode=0x1E020001, level=2, en
able=True, id=1")


def send_hms_alarm_enhanced(mavlink_conn):
    """
    发送 HmsAlarmEnhancedCtrl 结构体
    示例数据:
        type: SHAKE_MOTOR (1)
        times: 3
        interval: 500ms
        action: START (1)
        id: 1
    """
    payload = create_hmsalarminfo_struct(
        alarm_type=3,               # SHAKE_MOTOR
        times=3,                    # 振动3次
        interval=500,               # 间隔500ms
        action=1,                    # START
        alarm_id=1
    )

    # message_type 用于标识 HMS 告警控制消息类型
    HMS_ALARM_ENHANCED_TYPE = 32772
    send_tunnel_message(mavlink_conn, payload, HMS_ALARM_ENHANCED_T
YPE)
    print(f"[HMS] 发送 HmsAlarmEnhancedCtrl: type=1, times=3, interv
```

```
al=500, action=1, id=1")


def main():
    print("=" * 60)
    print("MAVLink HMS TUNNEL 消息发送工具")
    print("=" * 60)

    # 创建 MAVLink UDP 连接 (监听 14550 端口)
    # udpin:0.0.0.0:14550 表示监听所有网卡地址的 14550 端口
    connection_string = "udpin:0.0.0.0:14550"
    print(f"\n正在连接到: {connection_string}")

    try:
        mavlink_conn = mavutil.mavlink_connection(connection_strin
g)

        # 等待心跳包以确认连接
        print("等待心跳包...")
        heartbeat = mavlink_conn.wait_heartbeat(timeout=10)
        if heartbeat:
            print(f"已连接! ")
        else:
            print("警告: 未收到心跳包，继续发送...")

        print("\n" + "-" * 60)
        print("开始发送 HMS 消息...")
        print("-" * 60)

        # 发送 HmsErrCode 结构体
        send_hms_error_code(mavlink_conn,32771)
        time.sleep(3)  # 间隔1秒

        # 发送 HmsAlarmEnhancedCtrl 结构体
        send_hms_alarm_enhanced(mavlink_conn)

        print("\n" + "-" * 60)
        print("消息发送完成!")
        print("-" * 60)

        # 保持连接一段时间以便接收响应
        print("\n等待响应 (5秒)...")
        time.sleep(5)

        mavlink_conn.close()
```

```
        print("连接已关闭")

    except Exception as e:
        print(f"错误: {e}")
        raise


if __name__ == "__main__":
    main()
```

示例使用说明

运行后，成功建立连接并发送指令

```
正在连接到: udpin:0.0.0.0:14550
等待心跳包...
已连接!

------------------------------------------------------------
开始发送 HMS 消息...
------------------------------------------------------------
[HMS] 发送 HmsErrCode: errorCode=0x1E020001, level=2, enable=True, id=1
[HMS] 发送 HmsAlarmEnhancedCtrl: type=1, times=3, interval=500, action=1, id=1

------------------------------------------------------------
消息发送完成!
```

清除应用程序上显示的自定义 HMS 错误码

TUNNEL_PROTOCOL_ID_ELIMINATE_HMS_ERROR

设置

mavlink消息：

接收TUNNEL_PROTOCOL_ID_ELIMINATE_HMS_ERROR

参数：

名称	枚举值	描述
DJI_HMS_ERROR_LEVEL_NONE	0	无错误。
DJI_HMS_ERROR_LEVEL_HINT	1	提示级（Hint）
DJI_HMS_ERROR_LEVEL_WARN	2	警告级（Warning）
DJI_HMS_ERROR_LEVEL_CRITICAL	3	严重级（Critical）

名称	枚举值	描述
DJI_HMS_ERROR_LEVEL_FATAL	4	致命级（Fatal）

示例

```
#!/usr/bin/env python3
"""
通过 MAVLink TUNNEL 消息发送 DJI HMS 结构体数据
HmsErrCode 和 HmsAlarmEnhancedCtrl

Usage:
    python tunnel_hms_send.py
"""

import struct
import time
from pymavlink import mavutil

# MAVLink 组件 ID (DJI 指定)
MAV_COMP_ID_HMS = 12

# HMS 目标组件
TARGET_COMPONENT = MAV_COMP_ID_HMS

def create_hmserrcode_struct(error_code: int, error_level: int, ena
ble: bool, error_id: int) -> bytes:
    """
    创建 HmsErrCode 结构体 (7字节, packed)

    typedef struct Hms_ErrorCode_Ctrl{
        uint32_t errorCode;       // 4 bytes
        uint32_t errorLevel;       // 4 byte (E_DjiHmsErrorLevel)
        bool enable;              // 1 byte
        uint8_t id;               // 1 byte
    } __attribute__((packed)) HmsErrCode;
    """
    return struct.pack('<IIbB', error_code, error_level, 1 if enabl
e else 0, error_id)


def create_hmsalarminfo_struct(
    alarm_type: int,
```

```
    times: int,
    interval: int,
    action: int,
    alarm_id: int
) -> bytes:
    """
    创建 HmsAlarmEnhancedCtrl 结构体 (6字节, packed)

    typedef struct Hms_Alarm_Enhanced_Ctrl{
        uint32_t type;              // 4 byte (E_DjiHmsAlarmEnhanced
Type)
        int8_t times;               // 1 byte
        uint8_t reserved;           // 1 byte
        int16_t interval;           // 2 bytes
        uint32_t action;            // 4 byte (E_DjiHmsAlarmEnhanced
Action)
        uint8_t id;                // 1 byte
    } __attribute__((packed)) HmsAlarmEnhancedCtrl;
    """
    return struct.pack('<IbbhIB', alarm_type, times, 0, interval, a
ction, alarm_id)


def send_tunnel_message(mavlink_conn, payload: bytes, message_type:
int = 0):
    """
    通过 MAVLink TUNNEL 消息发送数据

    Args:
        mavlink_conn: MAVLink 连接对象
        payload: 要发送的数据负载 (最多 228 字节)
        message_type: 消息类型标识 (可选)
    """
    if len(payload) > 228:
        raise ValueError(f"Payload too large: {len(payload)} bytes
(max 228)")

    # 填充到固定长度 (确保小端序)
    payload_bytes = payload.ljust(228, b'\x00')

    # 发送 TUNNEL 消息
    # MAVLINK_MSG_ID_TUNNEL = 110
    mavlink_conn.mav.tunnel_send(
        target_system=0,            # 广播到所有系统
        target_component=TARGET_COMPONENT,
```

```
        payload_length=len(payload),
        payload_type=message_type,
        payload=payload_bytes
    )


def send_hms_error_code(mavlink_conn,message_type: int = 0):
    """
    发送 HmsErrCode 结构体
    示例数据:
        errorCode: 0x1E020001 (设备错误)
        errorLevel: DJI_HMS_ERROR_LEVEL_WARN (2)
        enable: True
        id: 1
    """
    payload = create_hmserrcode_struct(
        error_code=0x1E020001,     # 示例错误码
        error_level=2,              # WARN 级别
        enable=True,
        error_id=1
    )

    # message_type 用于标识 HMS 错误码消息类型
    if message_type==32771:
        payload = create_hmserrcode_struct(
            error_code=0x1E020001,     # 示例错误码
            error_level=2,              # WARN 级别
            enable=False,
            error_id=1
        )
    send_tunnel_message(mavlink_conn, payload, message_type)
    print(f"[HMS] 发送 HmsErrCode: errorCode=0x1E020001, level=2, en
able=True, id=1")


def send_hms_alarm_enhanced(mavlink_conn):
    """
    发送 HmsAlarmEnhancedCtrl 结构体
    示例数据:
        type: SHAKE_MOTOR (1)
        times: 3
        interval: 500ms
        action: START (1)
        id: 1
    """
```

```
    payload = create_hmsalarminfo_struct(
        alarm_type=3,               # SHAKE_MOTOR
        times=3,                    # 振动3次
        interval=500,               # 间隔500ms
        action=1,                    # START
        alarm_id=1
    )

    # message_type 用于标识 HMS 告警控制消息类型
    HMS_ALARM_ENHANCED_TYPE = 32772
    send_tunnel_message(mavlink_conn, payload, HMS_ALARM_ENHANCED_T
YPE)
    print(f"[HMS] 发送 HmsAlarmEnhancedCtrl: type=1, times=3, interv
al=500, action=1, id=1")


def main():
    print("=" * 60)
    print("MAVLink HMS TUNNEL 消息发送工具")
    print("=" * 60)

    # 创建 MAVLink UDP 连接 (监听 14550 端口)
    # udpin:0.0.0.0:14550 表示监听所有网卡地址的 14550 端口
    connection_string = "udpin:0.0.0.0:14550"
    print(f"\n正在连接到: {connection_string}")

    try:
        mavlink_conn = mavutil.mavlink_connection(connection_strin
g)

        # 等待心跳包以确认连接
        print("等待心跳包...")
        heartbeat = mavlink_conn.wait_heartbeat(timeout=10)
        if heartbeat:
            print(f"已连接! ")
        else:
            print("警告: 未收到心跳包，继续发送...")

        print("\n" + "-" * 60)
        print("开始发送 HMS 消息...")
        print("-" * 60)

        # 发送 HmsErrCode 结构体
        send_hms_error_code(mavlink_conn,32772)
        time.sleep(3)  # 间隔1秒
```

```
        # send_hms_error_code(mavlink_conn,32771)
        # time.sleep(3)  # 间隔1秒

        # 发送 HmsAlarmEnhancedCtrl 结构体
        send_hms_alarm_enhanced(mavlink_conn)

        print("\n" + "-" * 60)
        print("消息发送完成!")
        print("-" * 60)

        # 保持连接一段时间以便接收响应
        print("\n等待响应 (5秒)...")
        time.sleep(5)

        mavlink_conn.close()
        print("连接已关闭")

    except Exception as e:
        print(f"错误: {e}")
        raise


if __name__ == "__main__":
    main()
```

示例使用说明

运行后，成功建立连接并发送指令

```
正在连接到: udpin:0.0.0.0:14550
等待心跳包...
已连接!

------------------------------------------------------------
开始发送 HMS 消息...
------------------------------------------------------------
[HMS] 发送 HmsErrCode: errorCode=0x1E020001, level=2, enable=True, id=1
[HMS] 发送 HmsAlarmEnhancedCtrl: type=1, times=3, interval=500, action=1, id=1

------------------------------------------------------------
消息发送完成!
```

精准定位

精准定位

功能说明

RTK Precise Position 功能用于将 DJI PSDK 订阅到的 RTK 定位、速度、航向和解状态转换为 MAVLink 标准定位消息输出。

本功能只负责“PSDK RTK 数据 -> MAVLink 定位消息”的转换，不负责启动网络 RTK 服务。网络 RTK 服务的账号、NTRIP 配置和启停请参考 `network_rtk.md` 。

相关文件

文件	说明
`src/dji_function/dji_sub_callback.cpp`	RTK 订阅、坐标转换、MAVLink 定位消息发送
`src/dji_mavlink/mavlink_tool.h`	MAVLink 工具函数、坐标/角度转换函数

PSDK 订阅 Topic

PSDK Topic	频率	用途
`DJI_FC_SUBSCRIPTION_TOPIC_RTK_POSITION`	5 Hz	RTK 经纬度和高程
`DJI_FC_SUBSCRIPTION_TOPIC_RTK_VELOCITY`	5 Hz	RTK 速度
`DJI_FC_SUBSCRIPTION_TOPIC_RTK_YAW`	5 Hz	RTK 双天线航向
`DJI_FC_SUBSCRIPTION_TOPIC_RTK_POSITION_INFO`	5 Hz	RTK 定位解状态
`DJI_FC_SUBSCRIPTION_TOPIC_RTK_YAW_INFO`	5 Hz	RTK 航向解状态

PSDK Topic	频率	用途
`DJI_FC_SUBSCRIPTION_TOPIC_RTK_CONNECT_STATUS`	5 Hz	RTK 连接状态

输出 MAVLink 消息

MAVLink 消息	说明
`GPS_RAW_INT`	主 GPS/RTK 定位输出
`GPS2_RAW`	第二路 GPS/RTK 定位输出，内容与 RTK 主定位一致
`GPS_INPUT`	给飞控/上位机使用的 GPS 输入格式
`GLOBAL_POSITION_INT`	全局位置、相对高度、速度和航向
`GPS_RTK`	主 RTK 状态
`GPS2_RTK`	第二路 RTK 状态

坐标系和单位

坐标系转换必须保持以下规则，避免 MAVLink 侧坐标混乱。

数据	DJI/PSDK 来源	MAVLink 输出	转换说明
纬度	`RTK_POSITION.latitude`，单位 degree	`lat`，单位 degE7	`degree * 1e7`
经度	`RTK_POSITION.longitude`，单位 degree	`lon`，单位 degE7	`degree * 1e7`
高程	`RTK_POSITION.hfsl`，单位 m	`alt`，单位 mm	`m * 1000`
北向速度	`RTK_VELOCITY.x`	`vn` / `vx`	保持 North 方向
东向速度	`RTK_VELOCITY.y`	`ve` / `vy`	保持 East 方向

数据	DJI/PSDK 来源	MAVLink 输出	转换说明
垂向速度	`RTK_VELOCITY.z`	`vd` / `vz`	MAVLink NED 使用 Down，输出时取负
RTK 航向	`RTK_YAW`	`yaw` / `hdg`，单位 cdeg	DJI RTK yaw 为 ANT1->ANT2，输出前减 90 deg

航向转换

DJI RTK 双天线 yaw 与 MAVLink 车辆航向存在 90 度偏移，代码中统一使用：

```
mavlink_yaw_deg = dji_rtk_yaw_deg - 90
mavlink_yaw_cdeg = normalize_0_360(mavlink_yaw_deg) * 100
```

当结果为 0 deg 时，`GPS_RAW_INT.yaw` 按 MAVLink 约定输出 `36000`；当航向不可用时输出 `UINT16_MAX`。

定位解状态映射

DJI RTK 解状态	MAVLink `fix_type`
窄巷固定解、宽巷固定解、L1 整周模糊度固定解	`GPS_FIX_TYPE_RTK_FIXED`
浮点解、L1 浮点解、无电离层组合浮点解	`GPS_FIX_TYPE_RTK_FLOAT`
单点解、伪距差分解、SBAS 解	`GPS_FIX_TYPE_3D_FIX`
其他/未知	`GPS_FIX_TYPE_NO_FIX`

数据输出策略

RTK 定位有效时，系统优先输出 RTK 定位数据，并临时抑制普通 GPS 位置回调中的低精度 `GPS_RAW_INT` / `GLOBAL_POSITION_INT` 输出。

RTK 定位无效或超过 1 秒未更新时，普通 GPS 输出恢复，避免上位机长时间没有定位消息。

使用流程

1. 启动程序后，订阅模块自动订阅 RTK 相关 Topic。
2. 上位机监听标准 MAVLink 定位消息。
3. 当 RTK 进入 Float 或 Fixed 解时，`GPS_RAW_INT.fix_type` 变为 `GPS_FIX_TYPE_RTK_FLOAT` 或 `GPS_FIX_TYPE_RTK_FIXED`。
4. 如需网络 RTK 数据源，先通过 `network_rtk.md` 中的 Network RTK 功能配置并启动 NTRIP 服务。

Python 例程

以下例程使用 `pymavlink` 监听 RTK 精准定位相关 MAVLink 消息，并打印经纬度、高度、速度、航向和 RTK 解状态。

```
import time
from pymavlink import mavutil


def get_current_millis():
    return int(time.time() * 1000)


def fix_type_name(fix_type):
    names = {
        0: "NO_GPS",
        1: "NO_FIX",
        2: "2D_FIX",
        3: "3D_FIX",
        4: "DGPS",
        5: "RTK_FLOAT",
        6: "RTK_FIXED",
    }
    return names.get(fix_type, f"UNKNOWN({fix_type})")


def yaw_cdeg_to_deg(yaw):
    if yaw == 65535:
        return None
    if yaw == 36000:
        return 0.0
    return yaw / 100.0


# 连接无人机（UDP 端口 14550）
master = mavutil.mavlink_connection("udpin:0.0.0.0:14550")
print("等待无人机心跳...")
```

```
master.wait_heartbeat()
print(f"无人机已连接: system={master.target_system}, component={maste
r.target_component}")
print("开始监听 RTK 精准定位消息")

message_types = [
    "GPS_RAW_INT",
    "GPS2_RAW",
    "GPS_INPUT",
    "GLOBAL_POSITION_INT",
    "GPS_RTK",
    "GPS2_RTK",
]

while True:
    msg = master.recv_match(type=message_types, blocking=True, time
out=5.0)
    now_ms = get_current_millis()

    if msg is None:
        print(f"[{now_ms}] 超时：未收到 RTK/MAVLink 定位消息")
        continue

    msg_type = msg.get_type()

    if msg_type in ("GPS_RAW_INT", "GPS2_RAW"):
        lat_deg = msg.lat / 1e7
        lon_deg = msg.lon / 1e7
        alt_m = msg.alt / 1000.0
        yaw_deg = yaw_cdeg_to_deg(getattr(msg, "yaw", 65535))
        print(
            f"[{now_ms}] {msg_type}: "
            f"fix={fix_type_name(msg.fix_type)}, "
            f"lat={lat_deg:.7f}, lon={lon_deg:.7f}, alt={alt_m:.3f}
m, "
            f"vel={msg.vel / 100.0:.2f}m/s, cog={msg.cog / 100.0:.2
f}deg, "
            f"yaw={yaw_deg}"
        )

    elif msg_type == "GPS_INPUT":
        print(
            f"[{now_ms}] GPS_INPUT: "
            f"fix={fix_type_name(msg.fix_type)}, "
            f"lat={msg.lat / 1e7:.7f}, lon={msg.lon / 1e7:.7f}, alt
```

```
={msg.alt:.3f}m, "
            f"vn={msg.vn:.2f}, ve={msg.ve:.2f}, vd={msg.vd:.2f}"
        )

    elif msg_type == "GLOBAL_POSITION_INT":
        print(
            f"[{now_ms}] GLOBAL_POSITION_INT: "
            f"lat={msg.lat / 1e7:.7f}, lon={msg.lon / 1e7:.7f}, "
            f"alt={msg.alt / 1000.0:.3f}m, relative_alt={msg.relati
ve_alt / 1000.0:.3f}m, "
            f"vx={msg.vx / 100.0:.2f}, vy={msg.vy / 100.0:.2f}, vz=
{msg.vz / 100.0:.2f}, "
            f"hdg={msg.hdg / 100.0:.2f}deg"
        )

    elif msg_type in ("GPS_RTK", "GPS2_RTK"):
        print(
            f"[{now_ms}] {msg_type}: "
            f"health={msg.rtk_health}, rate={msg.rtk_rate}Hz, nsats
={msg.nsats}, "
            f"receiver_id={msg.rtk_receiver_id}"
        )
```

注意事项

- MAVLink `GLOBAL_POSITION_INT` 的速度字段为 NED 坐标系，垂向速度必须使用 Down 为正。
- RTK 服务状态等于 `DJI_TRANSFER` 只代表 RTCM/网络数据传输建立，不等于 RTK 已经收敛。是否可用必须看 RTK 定位解状态。
- `GPS_RTK` / `GPS2_RTK` 中没有 DJI 可直接提供的基线详细值时，对应字段使用未知值或 0 填充。
- 如果地面站只看 `GPS_RAW_INT`，需要确认其支持 `GPS_FIX_TYPE_RTK_FLOAT` 和 `GPS_FIX_TYPE_RTK_FIXED`。

E-port（高级功能）

相机管理

相机曝光模式

EXPOSURE_MODE

设置

mavlink消息：

接收PARAM_SET
发送PARAM_VALUE

获取

mavlink消息

接收PARAM_REQUEST_READ
发送PARAM_VALUE

参数：E_DjiCameraManagerExposureMode

枚举值	值	说明
DJI_CAMERA_MANAGER_EXPOSURE_MODE_PROGRAM_AUTO	1	程序（P Auto
DJI_CAMERA_MANAGER_EXPOSURE_MODE_SHUTTER_PRIORITY	2	快门（S Pri S/T
DJI_CAMERA_MANAGER_EXPOSURE_MODE_APERTURE_PRIORITY	3	光圈（A Pri A/A
DJI_CAMERA_MANAGER_EXPOSURE_MODE_EXPOSURE_MANUAL	4	手动（M M档
DJI_CAMERA_MANAGER_EXPOSURE_MODE_EXPOSURE_UNKNOWN	0xFF	未知

示例

```
import time
import sys
from pymavlink import mavutil

def connect_to_vehicle():
    """建立 UDP 监听连接（默认监听所有接口的 14550 端口）"""
    connection_string = "udpin:0.0.0.0:14550"
    print(f"Listening for MAVLink on: {connection_string}")
    vehicle = mavutil.mavlink_connection(connection_string)
    print("Waiting for HEARTBEAT (timeout: 10s)...")
    try:
        vehicle.wait_heartbeat(timeout=10)
        print(f"  Heartbeat received from system {vehicle.target_sy
stem}, component {vehicle.target_component}")
        return vehicle
    except Exception as e:
        print(f"  Connection failed: {str(e)}")
        print("Ensure your vehicle is sending MAVLink messages to 1
4550 port")
        print("Example QGroundControl setup: Settings > General > M
AVLink > Listening Port = 14550")
        sys.exit(1)

def format_param_id(param_id):
    """格式化16字节参数ID（符合MAVLink协议要求）"""
    if len(param_id) > 16:
        raise ValueError("param_id cannot exceed 16 characters")

    param_id_bytes = bytearray(16)
    for i in range(min(len(param_id), 16)):
        param_id_bytes[i] = ord(param_id[i])

    # 长度<16时添加NULL终止符
    if len(param_id) < 16:
        param_id_bytes[len(param_id)] = 0

    return param_id_bytes

def send_param_set(vehicle, param_id, param_value, param_type):
    """发送参数设置请求"""
    vehicle.mav.param_set_send(
        vehicle.target_system,
        vehicle.target_component,
```

```
        format_param_id(param_id),
        param_value,
        param_type
    )
    print(f"  Sent PARAM_SET for '{param_id}' = {param_value} (typ
e: {param_type})")

def send_param_request_read(vehicle, param_id=None, param_index=-
1):
    """发送参数读取请求"""
    param_id_bytes = format_param_id(param_id) if param_id else byt
earray(16)

    vehicle.mav.param_request_read_send(
        vehicle.target_system,
        vehicle.target_component,
        param_id_bytes,
        param_index
    )

    target = f"index {param_index}" if param_index >= 0 else f"ID
'{param_id}'"
    print(f"  Sent PARAM_REQUEST_READ for {target}")

def send_param_request_list(vehicle):
    """发送请求所有参数列表"""
    vehicle.mav.param_request_list_send(
        vehicle.target_system,
        vehicle.target_component
    )
    print(f"  Sent PARAM_REQUEST_LIST to system {vehicle.target_sys
tem}, component {vehicle.target_component}")

def receive_param_value(vehicle, timeout=5.0):
    """接收参数值响应"""
    print(f"  Waiting for PARAM_VALUE ({timeout}s timeout)...")
    msg = vehicle.recv_match(type='PARAM_VALUE', blocking=True, tim
eout=timeout)

    if not msg:
        print("  Timeout: No PARAM_VALUE received")
        return None

    # 解析参数ID
    param_id = msg.param_id
```

```
    # 解析参数值
    param_value = msg.param_value

    # 参数类型映射
    type_names = {
        0: "INT8", 1: "UINT8", 2: "INT16", 3: "UINT16", 4: "INT32",
        5: "UINT32", 6: "INT64", 7: "UINT64", 8: "FLOAT", 9: "DOUBL
E"
    }
    param_type = type_names.get(msg.param_type, f"UNKNOWN({msg.para
m_type})")

    print("\n  Received PARAM_VALUE:")
    print(f"    ID: {param_id}")
    print(f"    Value: {param_value}")
    print(f"    Type: {param_type}")
    print(f"    Index/Total: {msg.param_index}/{msg.param_count}")

    return {
        'id': param_id,
        'value': param_value,
        'type': param_type,
        'index': msg.param_index,
        'count': msg.param_count
    }

def receive_all_param_values(vehicle, timeout=10.0):
    """接收所有参数值，直到收到 param_count 条消息后退出循环"""
    received_params = []
    expected_count = None
    print(f"  Waiting for all PARAM_VALUE messages...")

    while True:
        msg = vehicle.recv_match(type='PARAM_VALUE', blocking=True,
timeout=timeout)

        if not msg:
            print(f"  Timeout: Received {len(received_params)} para
meters (expected {expected_count if expected_count else 'unknow
n'})")
            break

        param_id = msg.param_id
        param_value = msg.param_value
```

```
        # 参数类型映射
        type_names = {
            0: "INT8", 1: "UINT8", 2: "INT16", 3: "UINT16", 4: "INT
32",
            5: "UINT32", 6: "INT64", 7: "UINT64", 8: "FLOAT", 9: "D
OUBLE"
        }
        param_type = type_names.get(msg.param_type, f"UNKNOWN({msg.
param_type})")

        # 首次收到消息时记录总参数数量
        if expected_count is None:
            expected_count = msg.param_count
            print(f"  Total parameters to receive: {expected_coun
t}")

        print(f"  [{msg.param_index}/{expected_count}] {param_id} =
{param_value} ({param_type})")

        received_params.append({
            'id': param_id,
            'value': param_value,
            'type': param_type,
            'index': msg.param_index,
            'count': msg.param_count
        })

        # 当已接收数量达到预期总数时退出循环
        if len(received_params) >= expected_count:
            print(f"\n  Successfully received all {expected_count}
parameters")
            break

    return received_params

def main():
    print("  Starting MAVLink PARAM_SET Tool (UDP Listening Mode)")
    print("-----------------------------------------------------")
    print("This script listens for MAVLink on ALL interfaces:1455
0")
    print("Ensure your vehicle is configured to send to this port")
    print("-----------------------------------------------------
\n")
```

```
    vehicle = connect_to_vehicle()

    # 操作选择
    print("\nAvailable operations:")
    print("1. Set parameter value")
    print("2. Read parameter value (by ID or index)")
    print("3. Request all parameters (PARAM_REQUEST_LIST)")
    choice = input("\nSelect operation [1-3]: ").strip()

    if choice == '1':
        param_id = input("Enter parameter ID to set: ").strip()
        param_value = float(input("Enter new value: ").strip())

        print("\nParameter types:")
        print("  8 = FLOAT (most common)")
        print("  9 = DOUBLE")
        print("  4 = INT32")
        print("  1 = UINT8")
        param_type = int(input("Enter type number [8]: ").strip() or 8)

        send_param_set(vehicle, param_id, param_value, param_type)
        receive_param_value(vehicle)

    elif choice == '2':
        # mode = input("Read by [id/index]: ").lower()
        mode = 'id'
        if mode == 'id':
            param_id = input("Enter parameter ID: ").strip()
            send_param_request_read(vehicle, param_id=param_id)
            receive_param_value(vehicle)
        elif mode == 'index':
            param_index = int(input("Enter parameter index: "))
            send_param_request_read(vehicle, param_index=param_index)
            receive_param_value(vehicle)
        else:
            print("Invalid mode. Use 'id' or 'index'")

    elif choice == '3':
        print("\n  Requesting all parameters...")
        send_param_request_list(vehicle)
        all_params = receive_all_param_values(vehicle)
        print(f"\n  Total parameters received: {len(all_params)}")
```

```
    else:
        print("Invalid choice. Select 1, 2, or 3.")


if __name__ == "__main__":
    main()
```

示例使用说明

获取

运行后，成功建立连接

```
🚀 Starting MAVLink PARAM_SET Tool (UDP Listening Mode)
------------------------------------------------------
This script listens for MAVLink on ALL interfaces:14550
Ensure your vehicle is configured to send to this port
------------------------------------------------------

Listening for MAVLink on: udpin:0.0.0.0:14550
Waiting for HEARTBEAT (timeout: 10s)...
✅ Heartbeat received from system 3, component 0

Available operations:
1. Set parameter value
2. Read parameter value (by ID or index)
3. Request all parameters (PARAM_REQUEST_LIST)
```

输入“2”选择读取参数

```
Select operation [1-3]: 2
```

输入对应参数“EXPOSURE_MODE”

```
Enter parameter ID: EXPOSURE_MODE
📤 Sent PARAM_REQUEST_READ for ID 'EXPOSURE_MODE'
```

成功返回该参数对应的数值

```
📥 Received PARAM_VALUE:
   🆔 ID: EXPOSURE_MODE
   🔢 Value: 1.0
```

设置

运行后，成功建立连接

```
🚀 Starting MAVLink PARAM_SET Tool (UDP Listening Mode)
------------------------------------------------------
This script listens for MAVLink on ALL interfaces:14550
Ensure your vehicle is configured to send to this port
------------------------------------------------------

Listening for MAVLink on: udpin:0.0.0.0:14550
Waiting for HEARTBEAT (timeout: 10s)...
✅ Heartbeat received from system 3, component 0

Available operations:
1. Set parameter value
2. Read parameter value (by ID or index)
3. Request all parameters (PARAM_REQUEST_LIST)
```

输入“1”选择设置参数

```
Select operation [1-3]: 1
```

输入对应参数“EXPOSURE_MODE”

```
Enter parameter ID to set: EXPOSURE_MODE
```

输入要设置的数值

```
Enter new value: 2
```

选择参数类型“1”

```
Parameter types:
  9 = DOUBLE
  4 = INT32
  1 = UINT8
Enter type number [8]: 1
```

成功返回设置值

```
📤 Sent PARAM_SET for 'EXPOSURE_MODE' = 2.0 (type: 1)
⏳ Waiting for PARAM_VALUE (5.0s timeout)...

📥 Received PARAM_VALUE:
  🆔 ID: EXPOSURE_MODE
  🔢 Value: 2.0
```

相机感光度

ISO

设置

mavlink消息：

接收PARAM_SET
发送PARAM_VALUE

获取

mavlink消息

接收PARAM_REQUEST_READ
发送PARAM_VALUE

参数：E_DjiCameraManagerISO

枚举值	值	说明
DJI_CAMERA_MANAGER_ISO_AUTO	0x00	自动 ISO
DJI_CAMERA_MANAGER_ISO_100	0x03	ISO 100
DJI_CAMERA_MANAGER_ISO_200	0x04	ISO 200
DJI_CAMERA_MANAGER_ISO_400	0x05	ISO 400
DJI_CAMERA_MANAGER_ISO_800	0x06	ISO 800
DJI_CAMERA_MANAGER_ISO_1600	0x07	ISO 1600
DJI_CAMERA_MANAGER_ISO_3200	0x08	ISO 3200
DJI_CAMERA_MANAGER_ISO_6400	0x09	ISO 6400
DJI_CAMERA_MANAGER_ISO_12800	0x0A	ISO 12800
DJI_CAMERA_MANAGER_ISO_25600	0x0B	ISO 25600
DJI_CAMERA_MANAGER_ISO_FIXED	0xFF	固定 ISO（由相机固件决定）

注意：该功能仅在相机曝光模式设置为“手动（Exposure_Manual）时可用。

示例

```
import time
import sys
from pymavlink import mavutil

def connect_to_vehicle():
    """建立 UDP 监听连接（默认监听所有接口的 14550 端口）"""
    connection_string = "udpin:0.0.0.0:14550"
    print(f"Listening for MAVLink on: {connection_string}")
    vehicle = mavutil.mavlink_connection(connection_string)
    print("Waiting for HEARTBEAT (timeout: 10s)...")
    try:
        vehicle.wait_heartbeat(timeout=10)
        print(f"  Heartbeat received from system {vehicle.target_system}, component {vehicle.target_component}")
        return vehicle
    except Exception as e:
        print(f"  Connection failed: {str(e)}")
        print("Ensure your vehicle is sending MAVLink messages to 14550 port")
        print("Example QGroundControl setup: Settings > General > MAVLink > Listening Port = 14550")
        sys.exit(1)

def format_param_id(param_id):
    """格式化16字节参数ID（符合MAVLink协议要求）"""
    if len(param_id) > 16:
        raise ValueError("param_id cannot exceed 16 characters")

    param_id_bytes = bytearray(16)
    for i in range(min(len(param_id), 16)):
        param_id_bytes[i] = ord(param_id[i])

    # 长度<16时添加NULL终止符
    if len(param_id) < 16:
        param_id_bytes[len(param_id)] = 0

    return param_id_bytes

def send_param_set(vehicle, param_id, param_value, param_type):
    """发送参数设置请求"""
    vehicle.mav.param_set_send(
        vehicle.target_system,
        vehicle.target_component,
        format_param_id(param_id),
        param_value,
```

```
        param_type
    )
    print(f"  Sent PARAM_SET for '{param_id}' = {param_value} (type: {param_type})")

def send_param_request_read(vehicle, param_id=None, param_index=-1):
    """发送参数读取请求"""
    param_id_bytes = format_param_id(param_id) if param_id else bytearray(16)

    vehicle.mav.param_request_read_send(
        vehicle.target_system,
        vehicle.target_component,
        param_id_bytes,
        param_index
    )

    target = f"index {param_index}" if param_index >= 0 else f"ID '{param_id}'"
    print(f"  Sent PARAM_REQUEST_READ for {target}")

def send_param_request_list(vehicle):
    """发送请求所有参数列表"""
    vehicle.mav.param_request_list_send(
        vehicle.target_system,
        vehicle.target_component
    )
    print(f"  Sent PARAM_REQUEST_LIST to system {vehicle.target_system}, component {vehicle.target_component}")

def receive_param_value(vehicle, timeout=5.0):
    """接收参数值响应"""
    print(f"  Waiting for PARAM_VALUE ({timeout}s timeout)...")
    msg = vehicle.recv_match(type='PARAM_VALUE', blocking=True, timeout=timeout)

    if not msg:
        print("  Timeout: No PARAM_VALUE received")
        return None

    # 解析参数ID
    param_id = msg.param_id

    # 解析参数值
```

```
    param_value = msg.param_value

    # 参数类型映射
    type_names = {
        0: "INT8", 1: "UINT8", 2: "INT16", 3: "UINT16", 4: "INT32",
        5: "UINT32", 6: "INT64", 7: "UINT64", 8: "FLOAT", 9: "DOUBL
E"
    }
    param_type = type_names.get(msg.param_type, f"UNKNOWN({msg.para
m_type})")

    print("\n  Received PARAM_VALUE:")
    print(f"    ID: {param_id}")
    print(f"    Value: {param_value}")
    print(f"    Type: {param_type}")
    print(f"    Index/Total: {msg.param_index}/{msg.param_count}")

    return {
        'id': param_id,
        'value': param_value,
        'type': param_type,
        'index': msg.param_index,
        'count': msg.param_count
    }

def receive_all_param_values(vehicle, timeout=10.0):
    """接收所有参数值，直到收到 param_count 条消息后退出循环"""
    received_params = []
    expected_count = None
    print(f"  Waiting for all PARAM_VALUE messages...")

    while True:
        msg = vehicle.recv_match(type='PARAM_VALUE', blocking=True,
timeout=timeout)

        if not msg:
            print(f"  Timeout: Received {len(received_params)} para
meters (expected {expected_count if expected_count else 'unknow
n'})")
            break

        param_id = msg.param_id
        param_value = msg.param_value

        # 参数类型映射
```

```
        type_names = {
            0: "INT8", 1: "UINT8", 2: "INT16", 3: "UINT16", 4: "INT
32",
            5: "UINT32", 6: "INT64", 7: "UINT64", 8: "FLOAT", 9: "D
OUBLE"
        }
        param_type = type_names.get(msg.param_type, f"UNKNOWN({msg.
param_type})")

        # 首次收到消息时记录总参数数量
        if expected_count is None:
            expected_count = msg.param_count
            print(f"  Total parameters to receive: {expected_coun
t}")

        print(f"  [{msg.param_index}/{expected_count}] {param_id} =
{param_value} ({param_type})")

        received_params.append({
            'id': param_id,
            'value': param_value,
            'type': param_type,
            'index': msg.param_index,
            'count': msg.param_count
        })

        # 当已接收数量达到预期总数时退出循环
        if len(received_params) >= expected_count:
            print(f"\n  Successfully received all {expected_count}
parameters")
            break

    return received_params

def main():
    print("  Starting MAVLink PARAM_SET Tool (UDP Listening Mode)")
    print("------------------------------------------------------")
    print("This script listens for MAVLink on ALL interfaces:1455
0")
    print("Ensure your vehicle is configured to send to this port")
    print("------------------------------------------------------
\n")

    vehicle = connect_to_vehicle()
```

```
    # 操作选择
    print("\nAvailable operations:")
    print("1. Set parameter value")
    print("2. Read parameter value (by ID or index)")
    print("3. Request all parameters (PARAM_REQUEST_LIST)")
    choice = input("\nSelect operation [1-3]: ").strip()

    if choice == '1':
        param_id = input("Enter parameter ID to set: ").strip()
        param_value = float(input("Enter new value: ").strip())

        print("\nParameter types:")
        print("  8 = FLOAT (most common)")
        print("  9 = DOUBLE")
        print("  4 = INT32")
        print("  1 = UINT8")
        param_type = int(input("Enter type number [8]: ").strip() or 8)

        send_param_set(vehicle, param_id, param_value, param_type)
        receive_param_value(vehicle)

    elif choice == '2':
        # mode = input("Read by [id/index]: ").lower()
        mode = 'id'
        if mode == 'id':
            param_id = input("Enter parameter ID: ").strip()
            send_param_request_read(vehicle, param_id=param_id)
            receive_param_value(vehicle)
        elif mode == 'index':
            param_index = int(input("Enter parameter index: "))
            send_param_request_read(vehicle, param_index=param_index)
            receive_param_value(vehicle)
        else:
            print("Invalid mode. Use 'id' or 'index'")

    elif choice == '3':
        print("\n  Requesting all parameters...")
        send_param_request_list(vehicle)
        all_params = receive_all_param_values(vehicle)
        print(f"\n  Total parameters received: {len(all_params)}")

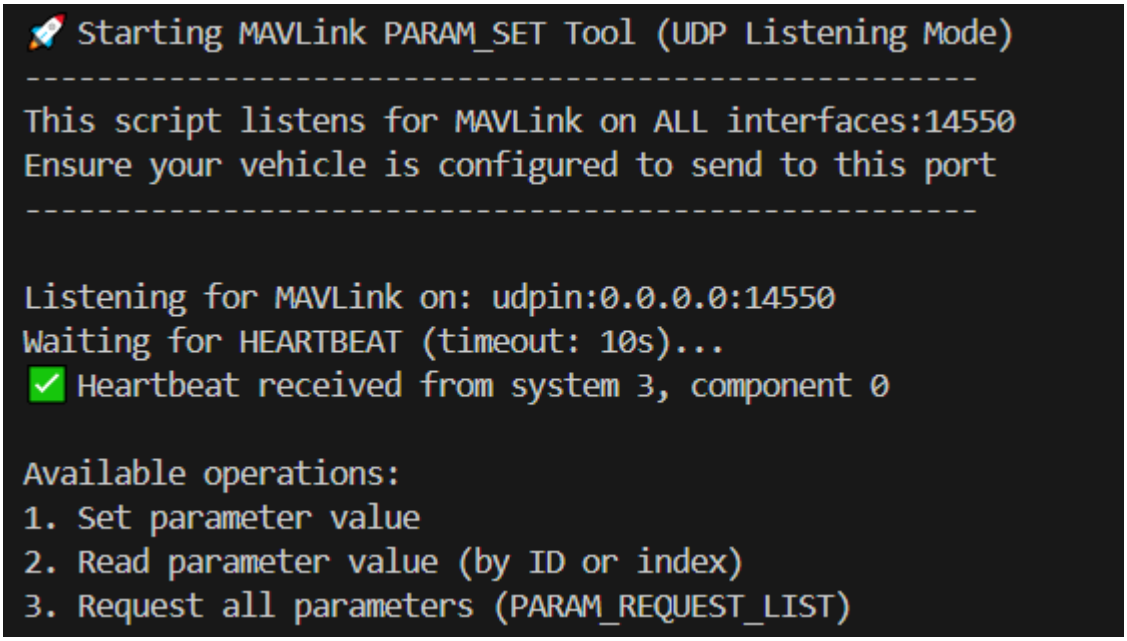
    else:
        print("Invalid choice. Select 1, 2, or 3.")
```

```
if __name__ == "__main__":
    main()
```

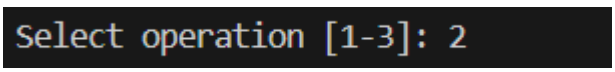
示例使用说明

获取

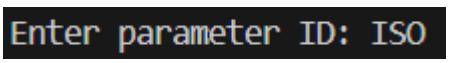
运行后，成功建立连接



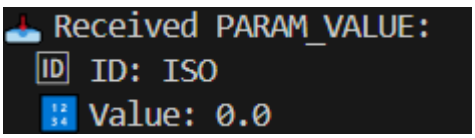
输入“2”选择读取参数



输入对应参数“ISO”



成功返回该参数对应的数值



设置

运行后，成功建立连接

```
🚀 Starting MAVLink PARAM_SET Tool (UDP Listening Mode)
--------------------------------------------------
This script listens for MAVLink on ALL interfaces:14550
Ensure your vehicle is configured to send to this port
--------------------------------------------------

Listening for MAVLink on: udpin:0.0.0.0:14550
Waiting for HEARTBEAT (timeout: 10s)...
✅ Heartbeat received from system 3, component 0

Available operations:
1. Set parameter value
2. Read parameter value (by ID or index)
3. Request all parameters (PARAM_REQUEST_LIST)
```

输入“1”选择设置参数

```
Select operation [1-3]: 1
```

输入对应参数“ISO”

```
Enter parameter ID to set: ISO
```

输入要设置的数值

```
Enter new value: 1
```

选择参数类型“1”

```
Enter new value: 1
```

成功返回设置值

```
📥 Received PARAM_VALUE:
  🆔 ID: ISO
  🔢 Value: 1.0
```

快门速度

SHUTTER_SPEED

设置

mavlink消息：

接收PARAM_SET
发送PARAM_VALUE

获取

mavlink消息

接收PARAM_REQUEST_READ
发送PARAM_VALUE

参数：

名称	枚举值	说明
DJI_CAMERA_MANAGER_SHUTTER_SPEED_1_8000	0	1/8000 s
DJI_CAMERA_MANAGER_SHUTTER_SPEED_1_6400	1	1/6400 s
DJI_CAMERA_MANAGER_SHUTTER_SPEED_1_6000	2	1/6000 s
DJI_CAMERA_MANAGER_SHUTTER_SPEED_1_5000	3	1/5000 s
DJI_CAMERA_MANAGER_SHUTTER_SPEED_1_4000	4	1/4000 s
DJI_CAMERA_MANAGER_SHUTTER_SPEED_1_3200	5	1/3200 s
DJI_CAMERA_MANAGER_SHUTTER_SPEED_1_3000	6	1/3000 s
DJI_CAMERA_MANAGER_SHUTTER_SPEED_1_2500	7	1/2500 s
DJI_CAMERA_MANAGER_SHUTTER_SPEED_1_2000	8	1/2000 s
DJI_CAMERA_MANAGER_SHUTTER_SPEED_1_1600	9	1/1600 s
DJI_CAMERA_MANAGER_SHUTTER_SPEED_1_1500	10	1/1500 s
DJI_CAMERA_MANAGER_SHUTTER_SPEED_1_1250	11	1/1250 s
DJI_CAMERA_MANAGER_SHUTTER_SPEED_1_1000	12	1/1000 s
DJI_CAMERA_MANAGER_SHUTTER_SPEED_1_800	13	1/800 s
DJI_CAMERA_MANAGER_SHUTTER_SPEED_1_725	14	1/725 s
DJI_CAMERA_MANAGER_SHUTTER_SPEED_1_640	15	1/640 s
DJI_CAMERA_MANAGER_SHUTTER_SPEED_1_500	16	1/500 s

名称	枚举值	说明
DJI_CAMERA_MANAGER_SHUTTER_SPEED_1_400	17	1/400 s
DJI_CAMERA_MANAGER_SHUTTER_SPEED_1_350	18	1/350 s
DJI_CAMERA_MANAGER_SHUTTER_SPEED_1_320	19	1/320 s
DJI_CAMERA_MANAGER_SHUTTER_SPEED_1_250	20	1/250 s
DJI_CAMERA_MANAGER_SHUTTER_SPEED_1_240	21	1/240 s
DJI_CAMERA_MANAGER_SHUTTER_SPEED_1_200	22	1/200 s
DJI_CAMERA_MANAGER_SHUTTER_SPEED_1_180	23	1/180 s
DJI_CAMERA_MANAGER_SHUTTER_SPEED_1_160	24	1/160 s
DJI_CAMERA_MANAGER_SHUTTER_SPEED_1_125	25	1/125 s
DJI_CAMERA_MANAGER_SHUTTER_SPEED_1_120	26	1/120 s
DJI_CAMERA_MANAGER_SHUTTER_SPEED_1_100	27	1/100 s
DJI_CAMERA_MANAGER_SHUTTER_SPEED_1_90	28	1/90 s
DJI_CAMERA_MANAGER_SHUTTER_SPEED_1_80	29	1/80 s
DJI_CAMERA_MANAGER_SHUTTER_SPEED_1_60	30	1/60 s
DJI_CAMERA_MANAGER_SHUTTER_SPEED_1_50	31	1/50 s
DJI_CAMERA_MANAGER_SHUTTER_SPEED_1_40	32	1/40 s
DJI_CAMERA_MANAGER_SHUTTER_SPEED_1_30	33	1/30 s
DJI_CAMERA_MANAGER_SHUTTER_SPEED_1_25	34	1/25 s
DJI_CAMERA_MANAGER_SHUTTER_SPEED_1_20	35	1/20 s
DJI_CAMERA_MANAGER_SHUTTER_SPEED_1_15	36	1/15 s
DJI_CAMERA_MANAGER_SHUTTER_SPEED_1_12DOT5	37	1/12.5 s
DJI_CAMERA_MANAGER_SHUTTER_SPEED_1_10	38	1/10 s
DJI_CAMERA_MANAGER_SHUTTER_SPEED_1_8	39	1/8 s

名称	枚举值	说明
DJI_CAMERA_MANAGER_SHUTTER_SPEED_1_6DOT25	40	1/6.25 s
DJI_CAMERA_MANAGER_SHUTTER_SPEED_1_5	41	1/5 s
DJI_CAMERA_MANAGER_SHUTTER_SPEED_1_4	42	1/4 s
DJI_CAMERA_MANAGER_SHUTTER_SPEED_1_3	43	1/3 s
DJI_CAMERA_MANAGER_SHUTTER_SPEED_1_2DOT5	44	1/2.5 s
DJI_CAMERA_MANAGER_SHUTTER_SPEED_1_2	45	1/2 s
DJI_CAMERA_MANAGER_SHUTTER_SPEED_1_1DOT67	46	1/1.67 s
DJI_CAMERA_MANAGER_SHUTTER_SPEED_1_1DOT25	47	1/1.25 s
DJI_CAMERA_MANAGER_SHUTTER_SPEED_1	48	1.0 s
DJI_CAMERA_MANAGER_SHUTTER_SPEED_1DOT3	49	1.3 s
DJI_CAMERA_MANAGER_SHUTTER_SPEED_1DOT6	50	1.6 s
DJI_CAMERA_MANAGER_SHUTTER_SPEED_2	51	2.0 s
DJI_CAMERA_MANAGER_SHUTTER_SPEED_2DOT5	52	2.5 s
DJI_CAMERA_MANAGER_SHUTTER_SPEED_3	53	3.0 s
DJI_CAMERA_MANAGER_SHUTTER_SPEED_3DOT2	54	3.2 s
DJI_CAMERA_MANAGER_SHUTTER_SPEED_4	55	4.0 s
DJI_CAMERA_MANAGER_SHUTTER_SPEED_5	56	5.0 s
DJI_CAMERA_MANAGER_SHUTTER_SPEED_6	57	6.0 s
DJI_CAMERA_MANAGER_SHUTTER_SPEED_7	58	7.0 s
DJI_CAMERA_MANAGER_SHUTTER_SPEED_8	59	8.0 s
DJI_CAMERA_MANAGER_SHUTTER_SPEED_9	60	9.0 s
DJI_CAMERA_MANAGER_SHUTTER_SPEED_10	61	10.0 s
DJI_CAMERA_MANAGER_SHUTTER_SPEED_13	62	13.0 s

名称	枚举值	说明
DJI_CAMERA_MANAGER_SHUTTER_SPEED_15	63	15.0 s
DJI_CAMERA_MANAGER_SHUTTER_SPEED_20	64	20.0 s
DJI_CAMERA_MANAGER_SHUTTER_SPEED_25	65	25.0 s
DJI_CAMERA_MANAGER_SHUTTER_SPEED_30	66	30.0 s
DJI_CAMERA_MANAGER_SHUTTER_SPEED_UNKNOWN	0xFF	未知快门速度

注意：该功能仅在相机曝光模式设置为“手动（Exposure_Manual）”或“快门优先（SHUTTER_PRIORITY）”时可用。

示例

```
import time
import sys
from pymavlink import mavutil

def connect_to_vehicle():
    """建立 UDP 监听连接（默认监听所有接口的 14550 端口）"""
    connection_string = "udpin:0.0.0.0:14550"
    print(f"Listening for MAVLink on: {connection_string}")
    vehicle = mavutil.mavlink_connection(connection_string)
    print("Waiting for HEARTBEAT (timeout: 10s)...")
    try:
        vehicle.wait_heartbeat(timeout=10)
        print(f"  Heartbeat received from system {vehicle.target_system}, component {vehicle.target_component}")
        return vehicle
    except Exception as e:
        print(f"  Connection failed: {str(e)}")
        print("Ensure your vehicle is sending MAVLink messages to 14550 port")
        print("Example QGroundControl setup: Settings > General > MAVLink > Listening Port = 14550")
        sys.exit(1)

def format_param_id(param_id):
    """格式化16字节参数ID（符合MAVLink协议要求）"""
    if len(param_id) > 16:
        raise ValueError("param_id cannot exceed 16 characters")
```

```
    param_id_bytes = bytearray(16)
    for i in range(min(len(param_id), 16)):
        param_id_bytes[i] = ord(param_id[i])

    # 长度<16时添加NULL终止符
    if len(param_id) < 16:
        param_id_bytes[len(param_id)] = 0

    return param_id_bytes

def send_param_set(vehicle, param_id, param_value, param_type):
    """发送参数设置请求"""
    vehicle.mav.param_set_send(
        vehicle.target_system,
        vehicle.target_component,
        format_param_id(param_id),
        param_value,
        param_type
    )
    print(f"  Sent PARAM_SET for '{param_id}' = {param_value} (typ
e: {param_type})")

def send_param_request_read(vehicle, param_id=None, param_index=-
1):
    """发送参数读取请求"""
    param_id_bytes = format_param_id(param_id) if param_id else byt
earray(16)

    vehicle.mav.param_request_read_send(
        vehicle.target_system,
        vehicle.target_component,
        param_id_bytes,
        param_index
    )

    target = f"index {param_index}" if param_index >= 0 else f"ID
'{param_id}'"
    print(f"  Sent PARAM_REQUEST_READ for {target}")

def send_param_request_list(vehicle):
    """发送请求所有参数列表"""
    vehicle.mav.param_request_list_send(
        vehicle.target_system,
        vehicle.target_component
```

```
    )
    print(f"  Sent PARAM_REQUEST_LIST to system {vehicle.target_system}, component {vehicle.target_component}")

def receive_param_value(vehicle, timeout=5.0):
    """接收参数值响应"""
    print(f"  Waiting for PARAM_VALUE ({timeout}s timeout)...")
    msg = vehicle.recv_match(type='PARAM_VALUE', blocking=True, timeout=timeout)

    if not msg:
        print("  Timeout: No PARAM_VALUE received")
        return None

    # 解析参数ID
    param_id = msg.param_id

    # 解析参数值
    param_value = msg.param_value

    # 参数类型映射
    type_names = {
        0: "INT8", 1: "UINT8", 2: "INT16", 3: "UINT16", 4: "INT32",
        5: "UINT32", 6: "INT64", 7: "UINT64", 8: "FLOAT", 9: "DOUBLE"
    }
    param_type = type_names.get(msg.param_type, f"UNKNOWN({msg.param_type})")

    print("\n  Received PARAM_VALUE:")
    print(f"    ID: {param_id}")
    print(f"    Value: {param_value}")
    print(f"    Type: {param_type}")
    print(f"    Index/Total: {msg.param_index}/{msg.param_count}")

    return {
        'id': param_id,
        'value': param_value,
        'type': param_type,
        'index': msg.param_index,
        'count': msg.param_count
    }

def receive_all_param_values(vehicle, timeout=10.0):
    """接收所有参数值，直到收到 param_count 条消息后退出循环"""
```

```
    received_params = []
    expected_count = None
    print(f"  Waiting for all PARAM_VALUE messages...")

    while True:
        msg = vehicle.recv_match(type='PARAM_VALUE', blocking=True,
timeout=timeout)

        if not msg:
            print(f"  Timeout: Received {len(received_params)} para
meters (expected {expected_count if expected_count else 'unknow
n'})")
            break

        param_id = msg.param_id
        param_value = msg.param_value

        # 参数类型映射
        type_names = {
            0: "INT8", 1: "UINT8", 2: "INT16", 3: "UINT16", 4: "INT
32",
            5: "UINT32", 6: "INT64", 7: "UINT64", 8: "FLOAT", 9: "D
OUBLE"
        }
        param_type = type_names.get(msg.param_type, f"UNKNOWN({msg.
param_type})")

        # 首次收到消息时记录总参数数量
        if expected_count is None:
            expected_count = msg.param_count
            print(f"  Total parameters to receive: {expected_coun
t}")

        print(f"  [{msg.param_index}/{expected_count}] {param_id} =
{param_value} ({param_type})")

        received_params.append({
            'id': param_id,
            'value': param_value,
            'type': param_type,
            'index': msg.param_index,
            'count': msg.param_count
        })

        # 当已接收数量达到预期总数时退出循环
```

```
        if len(received_params) >= expected_count:
            print(f"\n  Successfully received all {expected_count} parameters")
            break

    return received_params

def main():
    print("  Starting MAVLink PARAM_SET Tool (UDP Listening Mode)")
    print("------------------------------------------------------")
    print("This script listens for MAVLink on ALL interfaces:14550")
    print("Ensure your vehicle is configured to send to this port")
    print("------------------------------------------------------\n")

    vehicle = connect_to_vehicle()

    # 操作选择
    print("\nAvailable operations:")
    print("1. Set parameter value")
    print("2. Read parameter value (by ID or index)")
    print("3. Request all parameters (PARAM_REQUEST_LIST)")
    choice = input("\nSelect operation [1-3]: ").strip()

    if choice == '1':
        param_id = input("Enter parameter ID to set: ").strip()
        param_value = float(input("Enter new value: ").strip())

        print("\nParameter types:")
        print("  8 = FLOAT (most common)")
        print("  9 = DOUBLE")
        print("  4 = INT32")
        print("  1 = UINT8")
        param_type = int(input("Enter type number [8]: ").strip() or 8)

        send_param_set(vehicle, param_id, param_value, param_type)
        receive_param_value(vehicle)

    elif choice == '2':
        # mode = input("Read by [id/index]: ").lower()
        mode = 'id'
        if mode == 'id':
            param_id = input("Enter parameter ID: ").strip()
```

```
            send_param_request_read(vehicle, param_id=param_id)
            receive_param_value(vehicle)
        elif mode == 'index':
            param_index = int(input("Enter parameter index: "))
            send_param_request_read(vehicle, param_index=param_inde
x)
            receive_param_value(vehicle)
        else:
            print("Invalid mode. Use 'id' or 'index'")

    elif choice == '3':
        print("\n  Requesting all parameters...")
        send_param_request_list(vehicle)
        all_params = receive_all_param_values(vehicle)
        print(f"\n  Total parameters received: {len(all_params)}")

    else:
        print("Invalid choice. Select 1, 2, or 3.")


if __name__ == "__main__":
    main()
```

示例使用说明

获取

运行后，成功建立连接

```
🚀 Starting MAVLink PARAM_SET Tool (UDP Listening Mode)
------------------------------------------------------
This script listens for MAVLink on ALL interfaces:14550
Ensure your vehicle is configured to send to this port
------------------------------------------------------

Listening for MAVLink on: udpin:0.0.0.0:14550
Waiting for HEARTBEAT (timeout: 10s)...
✅ Heartbeat received from system 3, component 0

Available operations:
1. Set parameter value
2. Read parameter value (by ID or index)
3. Request all parameters (PARAM_REQUEST_LIST)
```

输入“2”选择读取参数

```
Select operation [1-3]: 2
```

输入对应参数“SHUTTER_SPEED”

```
Enter parameter ID: SHUTTER_SPEED
```

成功返回该参数对应的数值

```
📥 Received PARAM_VALUE:
  🆔 ID: SHUTTER_SPEED
  🔢 Value: 10.0
```

设置

运行后，成功建立连接

```
🚀 Starting MAVLink PARAM_SET Tool (UDP Listening Mode)
--------------------------------------------------
This script listens for MAVLink on ALL interfaces:14550
Ensure your vehicle is configured to send to this port
--------------------------------------------------

Listening for MAVLink on: udpin:0.0.0.0:14550
Waiting for HEARTBEAT (timeout: 10s)...
✅ Heartbeat received from system 3, component 0

Available operations:
1. Set parameter value
2. Read parameter value (by ID or index)
3. Request all parameters (PARAM_REQUEST_LIST)
```

输入“1”选择设置参数

```
Select operation [1-3]: 1
```

输入对应参数“SHUTTER_SPEED”

```
Enter parameter ID to set: SHUTTER_SPEED
```

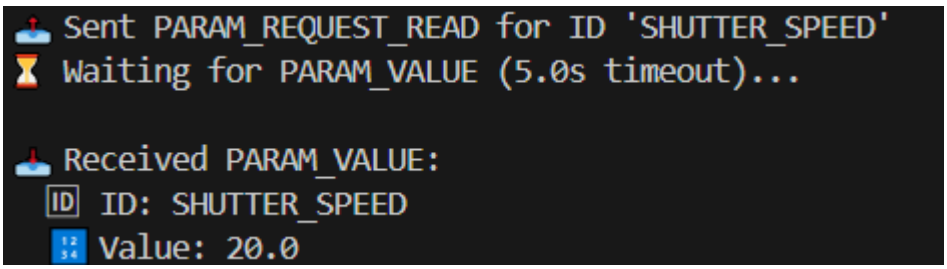
输入要设置的数值

```
Enter new value: 20
```

选择参数类型“1”

```
Enter type number [8]: 1
```

成功返回设置值



曝光补偿值

EXPOSURE_COMP

设置

mavlink消息：

接收PARAM_SET
发送PARAM_VALUE

获取

mavlink消息

接收PARAM_REQUEST_READ
发送PARAM_VALUE

参数：

名称	枚举值	说明
DJI_CAMERA_MANAGER_EXPOSURE_COMPENSATION_N_5_0	1	-5.0 EV 曝
DJI_CAMERA_MANAGER_EXPOSURE_COMPENSATION_N_4_7	2	-4.7 EV 曝
DJI_CAMERA_MANAGER_EXPOSURE_COMPENSATION_N_4_3	3	-4.3 EV 曝
DJI_CAMERA_MANAGER_EXPOSURE_COMPENSATION_N_4_0	4	-4.0 EV 曝
DJI_CAMERA_MANAGER_EXPOSURE_COMPENSATION_N_3_7	5	-3.7 EV 曝
DJI_CAMERA_MANAGER_EXPOSURE_COMPENSATION_N_3_3	6	-3.3 EV 曝
DJI_CAMERA_MANAGER_EXPOSURE_COMPENSATION_N_3_0	7	-3.0 EV 曝
DJI_CAMERA_MANAGER_EXPOSURE_COMPENSATION_N_2_7	8	-2.7 EV 曝

名称	枚举值	说明
DJI_CAMERA_MANAGER_EXPOSURE_COMPENSATION_N_2_3	9	-2.3 EV 曝
DJI_CAMERA_MANAGER_EXPOSURE_COMPENSATION_N_2_0	10	-2.0 EV 曝
DJI_CAMERA_MANAGER_EXPOSURE_COMPENSATION_N_1_7	11	-1.7 EV 曝
DJI_CAMERA_MANAGER_EXPOSURE_COMPENSATION_N_1_3	12	-1.3 EV 曝
DJI_CAMERA_MANAGER_EXPOSURE_COMPENSATION_N_1_0	13	-1.0 EV 曝
DJI_CAMERA_MANAGER_EXPOSURE_COMPENSATION_N_0_7	14	-0.7 EV 曝
DJI_CAMERA_MANAGER_EXPOSURE_COMPENSATION_N_0_3	15	-0.3 EV 曝
DJI_CAMERA_MANAGER_EXPOSURE_COMPENSATION_N_0_0	16	0.0 EV 曝
DJI_CAMERA_MANAGER_EXPOSURE_COMPENSATION_P_0_3	17	+0.3 EV 曝
DJI_CAMERA_MANAGER_EXPOSURE_COMPENSATION_P_0_7	18	+0.7 EV 曝
DJI_CAMERA_MANAGER_EXPOSURE_COMPENSATION_P_1_0	19	+1.0 EV 曝
DJI_CAMERA_MANAGER_EXPOSURE_COMPENSATION_P_1_3	20	+1.3 EV 曝
DJI_CAMERA_MANAGER_EXPOSURE_COMPENSATION_P_1_7	21	+1.7 EV 曝
DJI_CAMERA_MANAGER_EXPOSURE_COMPENSATION_P_2_0	22	+2.0 EV 曝
DJI_CAMERA_MANAGER_EXPOSURE_COMPENSATION_P_2_3	23	+2.3 EV 曝
DJI_CAMERA_MANAGER_EXPOSURE_COMPENSATION_P_2_7	24	+2.7 EV 曝
DJI_CAMERA_MANAGER_EXPOSURE_COMPENSATION_P_3_0	25	+3.0 EV 曝
DJI_CAMERA_MANAGER_EXPOSURE_COMPENSATION_P_3_3	26	+3.3 EV 曝
DJI_CAMERA_MANAGER_EXPOSURE_COMPENSATION_P_3_7	27	+3.7 EV 曝
DJI_CAMERA_MANAGER_EXPOSURE_COMPENSATION_P_4_0	28	+4.0 EV 曝
DJI_CAMERA_MANAGER_EXPOSURE_COMPENSATION_P_4_3	29	+4.3 EV 曝
DJI_CAMERA_MANAGER_EXPOSURE_COMPENSATION_P_4_7	30	+4.7 EV 曝
DJI_CAMERA_MANAGER_EXPOSURE_COMPENSATION_P_5_0	31	+5.0 EV 曝

名称	枚举值	说明
DJI_CAMERA_MANAGER_EXPOSURE_COMPENSATION_FIXED	0xFF	固定曝光（由相机

示例

```
import time
import sys
from pymavlink import mavutil

def connect_to_vehicle():
    """建立 UDP 监听连接（默认监听所有接口的 14550 端口）"""
    connection_string = "udpin:0.0.0.0:14550"
    print(f"Listening for MAVLink on: {connection_string}")
    vehicle = mavutil.mavlink_connection(connection_string)
    print("Waiting for HEARTBEAT (timeout: 10s)...")
    try:
        vehicle.wait_heartbeat(timeout=10)
        print(f"  Heartbeat received from system {vehicle.target_system}, component {vehicle.target_component}")
        return vehicle
    except Exception as e:
        print(f"  Connection failed: {str(e)}")
        print("Ensure your vehicle is sending MAVLink messages to 14550 port")
        print("Example QGroundControl setup: Settings > General > MAVLink > Listening Port = 14550")
        sys.exit(1)

def format_param_id(param_id):
    """格式化16字节参数ID（符合MAVLink协议要求）"""
    if len(param_id) > 16:
        raise ValueError("param_id cannot exceed 16 characters")

    param_id_bytes = bytearray(16)
    for i in range(min(len(param_id), 16)):
        param_id_bytes[i] = ord(param_id[i])

    # 长度<16时添加NULL终止符
    if len(param_id) < 16:
        param_id_bytes[len(param_id)] = 0
```

```
    return param_id_bytes

def send_param_set(vehicle, param_id, param_value, param_type):
    """发送参数设置请求"""
    vehicle.mav.param_set_send(
        vehicle.target_system,
        vehicle.target_component,
        format_param_id(param_id),
        param_value,
        param_type
    )
    print(f"  Sent PARAM_SET for '{param_id}' = {param_value} (type: {param_type})")

def send_param_request_read(vehicle, param_id=None, param_index=-1):
    """发送参数读取请求"""
    param_id_bytes = format_param_id(param_id) if param_id else bytearray(16)

    vehicle.mav.param_request_read_send(
        vehicle.target_system,
        vehicle.target_component,
        param_id_bytes,
        param_index
    )

    target = f"index {param_index}" if param_index >= 0 else f"ID '{param_id}'"
    print(f"  Sent PARAM_REQUEST_READ for {target}")

def send_param_request_list(vehicle):
    """发送请求所有参数列表"""
    vehicle.mav.param_request_list_send(
        vehicle.target_system,
        vehicle.target_component
    )
    print(f"  Sent PARAM_REQUEST_LIST to system {vehicle.target_system}, component {vehicle.target_component}")

def receive_param_value(vehicle, timeout=5.0):
    """接收参数值响应"""
    print(f"  Waiting for PARAM_VALUE ({timeout}s timeout)...")
    msg = vehicle.recv_match(type='PARAM_VALUE', blocking=True, timeout=timeout)
```

```
    if not msg:
        print("  Timeout: No PARAM_VALUE received")
        return None

    # 解析参数ID
    param_id = msg.param_id

    # 解析参数值
    param_value = msg.param_value

    # 参数类型映射
    type_names = {
        0: "INT8", 1: "UINT8", 2: "INT16", 3: "UINT16", 4: "INT32",
        5: "UINT32", 6: "INT64", 7: "UINT64", 8: "FLOAT", 9: "DOUBL
E"
    }
    param_type = type_names.get(msg.param_type, f"UNKNOWN({msg.para
m_type})")

    print("\n  Received PARAM_VALUE:")
    print(f"    ID: {param_id}")
    print(f"    Value: {param_value}")
    print(f"    Type: {param_type}")
    print(f"    Index/Total: {msg.param_index}/{msg.param_count}")

    return {
        'id': param_id,
        'value': param_value,
        'type': param_type,
        'index': msg.param_index,
        'count': msg.param_count
    }

def receive_all_param_values(vehicle, timeout=10.0):
    """接收所有参数值，直到收到 param_count 条消息后退出循环"""
    received_params = []
    expected_count = None
    print(f"  Waiting for all PARAM_VALUE messages...")

    while True:
        msg = vehicle.recv_match(type='PARAM_VALUE', blocking=True,
timeout=timeout)

        if not msg:
```

```
            print(f"  Timeout: Received {len(received_params)} parameters (expected {expected_count if expected_count else 'unknown'})")
            break

        param_id = msg.param_id
        param_value = msg.param_value

        # 参数类型映射
        type_names = {
            0: "INT8", 1: "UINT8", 2: "INT16", 3: "UINT16", 4: "INT32",
            5: "UINT32", 6: "INT64", 7: "UINT64", 8: "FLOAT", 9: "DOUBLE"
        }
        param_type = type_names.get(msg.param_type, f"UNKNOWN({msg.param_type})")

        # 首次收到消息时记录总参数数量
        if expected_count is None:
            expected_count = msg.param_count
            print(f"  Total parameters to receive: {expected_count}")

        print(f"  [{msg.param_index}/{expected_count}] {param_id} = {param_value} ({param_type})")

        received_params.append({
            'id': param_id,
            'value': param_value,
            'type': param_type,
            'index': msg.param_index,
            'count': msg.param_count
        })

        # 当已接收数量达到预期总数时退出循环
        if len(received_params) >= expected_count:
            print(f"\n  Successfully received all {expected_count} parameters")
            break

    return received_params

def main():
    print("  Starting MAVLink PARAM_SET Tool (UDP Listening Mode)")
```

```
    print("-----------------------------------------------------")
    print("This script listens for MAVLink on ALL interfaces:1455
0")
    print("Ensure your vehicle is configured to send to this port")
    print("-----------------------------------------------------
\n")

    vehicle = connect_to_vehicle()

    # 操作选择
    print("\nAvailable operations:")
    print("1. Set parameter value")
    print("2. Read parameter value (by ID or index)")
    print("3. Request all parameters (PARAM_REQUEST_LIST)")
    choice = input("\nSelect operation [1-3]: ").strip()

    if choice == '1':
        param_id = input("Enter parameter ID to set: ").strip()
        param_value = float(input("Enter new value: ").strip())

        print("\nParameter types:")
        print("  8 = FLOAT (most common)")
        print("  9 = DOUBLE")
        print("  4 = INT32")
        print("  1 = UINT8")
        param_type = int(input("Enter type number [8]: ").strip() o
r 8)

        send_param_set(vehicle, param_id, param_value, param_type)
        receive_param_value(vehicle)

    elif choice == '2':
        # mode = input("Read by [id/index]: ").lower()
        mode = 'id'
        if mode == 'id':
            param_id = input("Enter parameter ID: ").strip()
            send_param_request_read(vehicle, param_id=param_id)
            receive_param_value(vehicle)
        elif mode == 'index':
            param_index = int(input("Enter parameter index: "))
            send_param_request_read(vehicle, param_index=param_inde
x)
            receive_param_value(vehicle)
        else:
            print("Invalid mode. Use 'id' or 'index'")
```

```
    elif choice == '3':
        print("\n  Requesting all parameters...")
        send_param_request_list(vehicle)
        all_params = receive_all_param_values(vehicle)
        print(f"\n  Total parameters received: {len(all_params)}")

    else:
        print("Invalid choice. Select 1, 2, or 3.")


if __name__ == "__main__":
    main()
```

示例使用说明

获取

运行后，成功建立连接

```
🚀 Starting MAVLink PARAM_SET Tool (UDP Listening Mode)
------------------------------------------------------
This script listens for MAVLink on ALL interfaces:14550
Ensure your vehicle is configured to send to this port
------------------------------------------------------

Listening for MAVLink on: udpin:0.0.0.0:14550
Waiting for HEARTBEAT (timeout: 10s)...
✅ Heartbeat received from system 3, component 0

Available operations:
1. Set parameter value
2. Read parameter value (by ID or index)
3. Request all parameters (PARAM_REQUEST_LIST)
```

输入“2”选择读取参数

```
Select operation [1-3]: 2
```

输入对应参数“EXPOSURE_COMP”

```
Enter parameter ID: EXPOSURE_COMP
```

成功返回该参数对应的数值

```
📥 Received PARAM_VALUE:
  🆔 ID: EXPOSURE_COMP
  🔢 Value: 16.0
```

设置

运行后，成功建立连接

```
🚀 Starting MAVLink PARAM_SET Tool (UDP Listening Mode)
--------------------------------------------------
This script listens for MAVLink on ALL interfaces:14550
Ensure your vehicle is configured to send to this port
--------------------------------------------------

Listening for MAVLink on: udpin:0.0.0.0:14550
Waiting for HEARTBEAT (timeout: 10s)...
✅ Heartbeat received from system 3, component 0

Available operations:
1. Set parameter value
2. Read parameter value (by ID or index)
3. Request all parameters (PARAM_REQUEST_LIST)
```

输入“1”选择设置参数

```
Select operation [1-3]: 1
```

输入对应参数“EXPOSURE_COMP”

```
Enter parameter ID to set: EXPOSURE_COMP
```

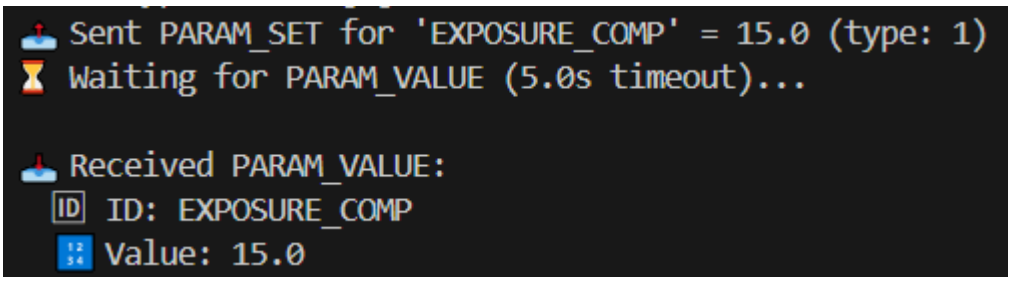
输入要设置的数值

```
Enter new value: 15
```

选择参数类型“1”

```
Enter type number [8]: 1
```

成功返回设置值



曝光锁定是否使能

AE_LOCK

设置

mavlink消息：

接收PARAM_SET
发送PARAM_VALUE

获取

mavlink消息

接收PARAM_REQUEST_READ
发送PARAM_VALUE

参数：

类型	枚举值	说明
bool	0	禁用 曝光锁定
bool	1	启用 曝光锁定

示例

```
import time
import sys
from pymavlink import mavutil

def connect_to_vehicle():
    """建立 UDP 监听连接（默认监听所有接口的 14550 端口）"""
    connection_string = "udpin:0.0.0.0:14550"
    print(f"Listening for MAVLink on: {connection_string}")
```

```
    vehicle = mavutil.mavlink_connection(connection_string)
    print("Waiting for HEARTBEAT (timeout: 10s)...")
    try:
        vehicle.wait_heartbeat(timeout=10)
        print(f"  Heartbeat received from system {vehicle.target_sy
stem}, component {vehicle.target_component}")
        return vehicle
    except Exception as e:
        print(f"  Connection failed: {str(e)}")
        print("Ensure your vehicle is sending MAVLink messages to 1
4550 port")
        print("Example QGroundControl setup: Settings > General > M
AVLink > Listening Port = 14550")
        sys.exit(1)

def format_param_id(param_id):
    """格式化16字节参数ID（符合MAVLink协议要求）"""
    if len(param_id) > 16:
        raise ValueError("param_id cannot exceed 16 characters")

    param_id_bytes = bytearray(16)
    for i in range(min(len(param_id), 16)):
        param_id_bytes[i] = ord(param_id[i])

    # 长度<16时添加NULL终止符
    if len(param_id) < 16:
        param_id_bytes[len(param_id)] = 0

    return param_id_bytes

def send_param_set(vehicle, param_id, param_value, param_type):
    """发送参数设置请求"""
    vehicle.mav.param_set_send(
        vehicle.target_system,
        vehicle.target_component,
        format_param_id(param_id),
        param_value,
        param_type
    )
    print(f"  Sent PARAM_SET for '{param_id}' = {param_value} (typ
e: {param_type})")

def send_param_request_read(vehicle, param_id=None, param_index=-
1):
    """发送参数读取请求"""
```

```
    param_id_bytes = format_param_id(param_id) if param_id else byt
earray(16)

    vehicle.mav.param_request_read_send(
        vehicle.target_system,
        vehicle.target_component,
        param_id_bytes,
        param_index
    )

    target = f"index {param_index}" if param_index >= 0 else f"ID
'{param_id}'"
    print(f"  Sent PARAM_REQUEST_READ for {target}")

def send_param_request_list(vehicle):
    """发送请求所有参数列表"""
    vehicle.mav.param_request_list_send(
        vehicle.target_system,
        vehicle.target_component
    )
    print(f"  Sent PARAM_REQUEST_LIST to system {vehicle.target_sys
tem}, component {vehicle.target_component}")

def receive_param_value(vehicle, timeout=5.0):
    """接收参数值响应"""
    print(f"  Waiting for PARAM_VALUE ({timeout}s timeout)...")
    msg = vehicle.recv_match(type='PARAM_VALUE', blocking=True, tim
eout=timeout)

    if not msg:
        print("  Timeout: No PARAM_VALUE received")
        return None

    # 解析参数ID
    param_id = msg.param_id

    # 解析参数值
    param_value = msg.param_value

    # 参数类型映射
    type_names = {
        0: "INT8", 1: "UINT8", 2: "INT16", 3: "UINT16", 4: "INT32",
        5: "UINT32", 6: "INT64", 7: "UINT64", 8: "FLOAT", 9: "DOUBL
E"
    }
```

```
    param_type = type_names.get(msg.param_type, f"UNKNOWN({msg.para
m_type})")

    print("\n  Received PARAM_VALUE:")
    print(f"    ID: {param_id}")
    print(f"    Value: {param_value}")
    print(f"    Type: {param_type}")
    print(f"    Index/Total: {msg.param_index}/{msg.param_count}")

    return {
        'id': param_id,
        'value': param_value,
        'type': param_type,
        'index': msg.param_index,
        'count': msg.param_count
    }

def receive_all_param_values(vehicle, timeout=10.0):
    """接收所有参数值，直到收到 param_count 条消息后退出循环"""
    received_params = []
    expected_count = None
    print(f"  Waiting for all PARAM_VALUE messages...")

    while True:
        msg = vehicle.recv_match(type='PARAM_VALUE', blocking=True,
timeout=timeout)

        if not msg:
            print(f"  Timeout: Received {len(received_params)} para
meters (expected {expected_count if expected_count else 'unknow
n'})")
            break

        param_id = msg.param_id
        param_value = msg.param_value

        # 参数类型映射
        type_names = {
            0: "INT8", 1: "UINT8", 2: "INT16", 3: "UINT16", 4: "INT
32",
            5: "UINT32", 6: "INT64", 7: "UINT64", 8: "FLOAT", 9: "D
OUBLE"
        }
        param_type = type_names.get(msg.param_type, f"UNKNOWN({msg.
param_type})")
```

```
            # 首次收到消息时记录总参数数量
            if expected_count is None:
                expected_count = msg.param_count
                print(f"  Total parameters to receive: {expected_coun
t}")

            print(f"  [{msg.param_index}/{expected_count}] {param_id} =
{param_value} ({param_type})")

            received_params.append({
                'id': param_id,
                'value': param_value,
                'type': param_type,
                'index': msg.param_index,
                'count': msg.param_count
            })

            # 当已接收数量达到预期总数时退出循环
            if len(received_params) >= expected_count:
                print(f"\n  Successfully received all {expected_count}
parameters")
                break

    return received_params

def main():
    print("  Starting MAVLink PARAM_SET Tool (UDP Listening Mode)")
    print("------------------------------------------------------------")
    print("This script listens for MAVLink on ALL interfaces:1455
0")
    print("Ensure your vehicle is configured to send to this port")
    print("------------------------------------------------------------
\n")

    vehicle = connect_to_vehicle()

    # 操作选择
    print("\nAvailable operations:")
    print("1. Set parameter value")
    print("2. Read parameter value (by ID or index)")
    print("3. Request all parameters (PARAM_REQUEST_LIST)")
    choice = input("\nSelect operation [1-3]: ").strip()

    if choice == '1':
```

```
        param_id = input("Enter parameter ID to set: ").strip()
        param_value = float(input("Enter new value: ").strip())

        print("\nParameter types:")
        print("  8 = FLOAT (most common)")
        print("  9 = DOUBLE")
        print("  4 = INT32")
        print("  1 = UINT8")
        param_type = int(input("Enter type number [8]: ").strip() or 8)

        send_param_set(vehicle, param_id, param_value, param_type)
        receive_param_value(vehicle)

    elif choice == '2':
        # mode = input("Read by [id/index]: ").lower()
        mode = 'id'
        if mode == 'id':
            param_id = input("Enter parameter ID: ").strip()
            send_param_request_read(vehicle, param_id=param_id)
            receive_param_value(vehicle)
        elif mode == 'index':
            param_index = int(input("Enter parameter index: "))
            send_param_request_read(vehicle, param_index=param_index)
            receive_param_value(vehicle)
        else:
            print("Invalid mode. Use 'id' or 'index'")

    elif choice == '3':
        print("\n  Requesting all parameters...")
        send_param_request_list(vehicle)
        all_params = receive_all_param_values(vehicle)
        print(f"\n  Total parameters received: {len(all_params)}")

    else:
        print("Invalid choice. Select 1, 2, or 3.")


if __name__ == "__main__":
    main()
```

示例使用说明

获取

运行后，成功建立连接

```
🚀 Starting MAVLink PARAM_SET Tool (UDP Listening Mode)
--------------------------------------------------
This script listens for MAVLink on ALL interfaces:14550
Ensure your vehicle is configured to send to this port
--------------------------------------------------

Listening for MAVLink on: udpin:0.0.0.0:14550
Waiting for HEARTBEAT (timeout: 10s)...
✅ Heartbeat received from system 3, component 0

Available operations:
1. Set parameter value
2. Read parameter value (by ID or index)
3. Request all parameters (PARAM_REQUEST_LIST)
```

输入“2”选择读取参数

```
Select operation [1-3]: 2
```

输入对应参数“AE_LOCK”

```
Enter parameter ID: AE_LOCK
```

成功返回该参数对应的数值

```
📥 Received PARAM_VALUE:
  🆔 ID: AE_LOCK
  🔢 Value: 0.0
```

设置

运行后，成功建立连接

```
🚀 Starting MAVLink PARAM_SET Tool (UDP Listening Mode)
--------------------------------------------------
This script listens for MAVLink on ALL interfaces:14550
Ensure your vehicle is configured to send to this port
--------------------------------------------------

Listening for MAVLink on: udpin:0.0.0.0:14550
Waiting for HEARTBEAT (timeout: 10s)...
✅ Heartbeat received from system 3, component 0

Available operations:
1. Set parameter value
2. Read parameter value (by ID or index)
3. Request all parameters (PARAM_REQUEST_LIST)
```

输入“1”选择设置参数

```
1. Set parameter value
2. Read parameter value (by ID or index)
3. Request all parameters (PARAM_REQUEST_LIST)

Select operation [1-3]: 1
```

输入对应参数“AE_LOCK”

```
Enter parameter ID to set: AE_LOCK
```

输入要设置的参数

```
Enter new value: 1
```

选择参数类型“1”

```
  8 = FLOAT (most common)
  9 = DOUBLE
  4 = INT32
  1 = UINT8
Enter type number [8]: 1
```

成功返回设置值

```
📥 Received PARAM_VALUE:
  🆔 ID: AE_LOCK
  🔢 Value: 1.0
```

开始录像

MAV_CMD_VIDEO_START_CAPTURE 2500
返回command = MAV_CMD_REQUEST_MESSAGE
返回值 param1 2500 执行命令id
返回值 param2 0或1 是否执行成功
返回值 param3 失败代码
0 无异常
1 无法初始化相机控制
2 设置相机为录像模式
3 获取相机版本失败
4 相机录像执行失败

示例

```
import time
from pymavlink import mavutil

timestamp = int(time.time() * 1000)
millis = int(time.time() * 1000)

master = mavutil.mavlink_connection('udpin:0.0.0.0:14550')
master.wait_heartbeat()
boot_time = time.time()

def videoStart():
    master.mav.command_long_send(
        master.target_system,
        master.target_component,
        mavutil.mavlink.MAV_CMD_VIDEO_START_CAPTURE,
        0,
        0, 0, 0, 0, 0, 0, 0
    )

videoStart()
```

执行示例

```
PS D:\python> & C:\Users\Administrator\AppData\Local\Python\pythoncore-3.14-64\python.exe d:/python/tmp.py
PS D:\python>
```

结果



具体录像行为相关信息需用遥控器查看

MAV_CMD_VIDEO_STOP_CAPTURE 2501

返回command = MAV_CMD_REQUEST_MESSAGE
返回值 param1 2501 执行命令id
返回值 param2 0或1 是否执行成功
返回值 param3 失败代码
0 无异常
1 无法初始化相机控制
2 设置相机为录像模式
3 获取相机版本失败
4 相机录像执行失败

示例

```
import time
from pymavlink import mavutil

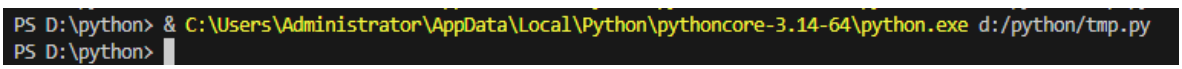
timestamp = int(time.time() * 1000)
millis = int(time.time() * 1000)

master = mavutil.mavlink_connection('udpin:0.0.0.0:14550')
master.wait_heartbeat()
boot_time = time.time()

def videoStop():
    master.mav.command_long_send(
        master.target_system,
        master.target_component,
        mavutil.mavlink.MAV_CMD_VIDEO_STOP_CAPTURE,
        0,
        0, 0, 0, 0, 0, 0, 0
    )

videoStop()
```

执行示例



结果

```
发送 CAMERA_CAPTURE_STATUS: video_status=0, recording_time_ms=0
video stop..
```

具体录像行为相关信息需用遥控器查看

获取相机流媒体数据源范围

STRM_SRC_RNG

参数：

字段	类型	说明
size	uint8_t	有效列表长度（最大16）
value	union	枚举取值列表（根据具体类型选择其中一个数组）
minValue	uint32_t	数值范围最小值
maxValue	uint32_t	数值范围最大值

value（联合体说明）

该字段为联合体，同一时间仅有一个数组有效：

字段	类型	说明
size	uint8_t	有效列表长度（最大16）
photoStorageFormat	E_DjiCameraManagerPhotoStorageFormat[16]	照片存储格式枚
videoStorageFormat	E_DjiCameraManagerVideoStorageFormat[16]	视频存储格式枚
photoRatioFormat	E_DjiCameraManagerPhotoRatio[16]	照片比例枚举列
streamSource	E_DjiCameraManagerStreamSource[16]	视频流源枚举列
streamStorage	E_DjiCameraManagerStreamStorage[16]	视频存储介质枚
nightSceneMode	E_DjiCameraManagerNightSceneMode[16]	夜景模式枚举列
meteringMode	E_DjiCameraManagerMeteringMode[16]	测光模式枚举列

字段	类型	说明
minValue	uint32_t	数值范围最小值（适用于数值型
maxValue	uint32_t	数值范围最大值（适用于数值型

示例

```
import time
import sys
import struct
import ctypes
from pymavlink import mavutil

def connect_to_vehicle():
    """建立 UDP 监听连接（默认监听所有接口的 14550 端口）"""
    connection_string = "udpin:0.0.0.0:14550"
    print(f"Listening for MAVLink on: {connection_string}")
    vehicle = mavutil.mavlink_connection(connection_string)
    print("Waiting for HEARTBEAT (timeout: 10s)...")
    try:
        vehicle.wait_heartbeat(timeout=10)
        print(f"  Heartbeat received from system {vehicle.target_system}, component {vehicle.target_component}")
        return vehicle
    except Exception as e:
        print(f"  Connection failed: {str(e)}")
        print("Ensure your vehicle is sending MAVLink messages to 14550 port")
        print("Example QGroundControl setup: Settings > General > MAVLink > Listening Port = 14550")
        sys.exit(1)

def format_param_id(param_id):
    """格式化16字节参数ID（符合MAVLink协议要求）"""
    if len(param_id) > 16:
        raise ValueError("param_id cannot exceed 16 characters")

    param_id_bytes = bytearray(16)
    for i in range(min(len(param_id), 16)):
        param_id_bytes[i] = ord(param_id[i])
```

```
    # 长度<16时添加NULL终止符
    if len(param_id) < 16:
        param_id_bytes[len(param_id)] = 0

    return param_id_bytes

# ============ 结构体打包函数 ============

def pack_uint8_xy(x, y):
    """打包 {uint8_t x; uint8_t y} 结构体 (2字节)"""
    return struct.pack('<BB', x, y)

def pack_float32_xy(x, y):
    """打包 {float32 x; float32 y} 结构体 (8字节)"""
    return struct.pack('<ff', x, y)

def pack_float32_lxlyrxry(LX, LY, RX, RY):
    """打包 {float32 LX; float32 LY; float32 RX; float32 RY;} 结构体
(16字节)"""
    return struct.pack('<ffff', LX, LY, RX, RY)

def pack_ering_mode(size, eringMode, minValue, maxValue):
    """打包 {uint8_t size; uint8_t eringMode[16]; uint32_t minValue;
uint32_t maxValue;} 结构体 (25字节)

    Args:
        size: uint8_t, eringMode数组的有效长度
        eringMode: list of int (0-255), 16个uint8_t值
        minValue: uint32_t, 最小值
        maxValue: uint32_t, 最大值

    Returns:
        bytes: 打包后的二进制数据
    """
    # 确保eringMode是16字节
    eringMode_bytes = bytearray(16)
    for i in range(min(len(eringMode), 16)):
        eringMode_bytes[i] = eringMode[i] & 0xFF

    return struct.pack('<B16sII', size, eringMode_bytes, minValue,
maxValue)

def format_param_value(param_value):
    """格式化128字节参数值（接受二进制输入）"""
```

```
    if isinstance(param_value, str):
        # 如果是十六进制字符串（如 "48656c6c6f"），先转换为字节
        if param_value.startswith('0x') or all(c in '0123456789abcd
efABCDEF' for c in param_value.replace(' ', '')):
            hex_str = param_value.replace('0x', '').replace(' ',
'')
            if len(hex_str) % 2 == 0:
                param_value = bytes.fromhex(hex_str)
            else:
                raise ValueError("Invalid hex string: must have eve
n number of characters")
        else:
            # 普通字符串转字节
            param_value = param_value.encode('utf-8')

    if len(param_value) > 128:
        raise ValueError("param_value cannot exceed 128 bytes")

    param_value_bytes = bytearray(128)
    param_value_bytes[:len(param_value)] = param_value

    return param_value_bytes

def hex_dump(data, bytes_per_line=16):
    """格式化二进制数据为 hex + ASCII 展示（类 hexdump 风格）

    Returns:
        str: 格式化后的多行字符串
    """
    lines = []
    for i in range(0, len(data), bytes_per_line):
        chunk = data[i:i + bytes_per_line]
        hex_part = ' '.join(f'{b:02X}' for b in chunk)
        ascii_part = ''.join(chr(b) if 32 <= b < 127 else '.' for b
in chunk)
        lines.append(f"  {i:04X}  {hex_part:<{bytes_per_line*3}}
{ascii_part}")
    return '\n'.join(lines)

def send_param_ext_request_read(vehicle, param_id=None, param_index
=-1):
    """发送参数读取请求"""
    param_id_bytes = format_param_id(param_id) if param_id else byt
earray(16)
```

```
    vehicle.mav.param_ext_request_read_send(
        vehicle.target_system,
        vehicle.target_component,
        param_id_bytes,
        param_index
    )

    target = f"index {param_index}" if param_index >= 0 else f"ID
'{param_id}'"
    print(f"  Sent PARAM_EXT_REQUEST_READ for {target}")

def send_param_ext_request_list(vehicle):
    """发送请求所有参数列表"""
    vehicle.mav.param_ext_request_list_send(
        vehicle.target_system,
        vehicle.target_component
    )
    print(f"  Sent PARAM_EXT_REQUEST_LIST to system {vehicle.target
_system}, component {vehicle.target_component}")

def send_param_ext_set(vehicle, param_id, param_value, param_type):
    """发送参数设置请求"""
    vehicle.mav.param_ext_set_send(
        vehicle.target_system,
        vehicle.target_component,
        format_param_id(param_id),
        format_param_value(param_value),
        param_type
    )
    print(f"  Sent PARAM_EXT_SET for '{param_id}' = '{param_value}'
(type: {param_type})")

def receive_param_ext_value(vehicle, timeout=5.0):
    """接收参数值响应"""
    print(f"  Waiting for PARAM_EXT_VALUE ({timeout}s timeout)...")
    msg = vehicle.recv_match(type='PARAM_EXT_VALUE', blocking=True,
timeout=timeout)

    if not msg:
        print("  Timeout: No PARAM_EXT_VALUE received")
        return None

    # 解析参数ID
    param_id = msg.param_id
```

```
    # 解析参数值（直接获取128字节原始数据）
    param_value_raw = msg.param_value
    # byte_pointer = (ctypes.c_ubyte * 128).from_buffer_copy(msg.pa
ram_value)
    if isinstance(param_value_raw, (bytes, bytearray)):
        param_value_bytes = bytes(param_value_raw)
    # param_value_bytes = bytes(byte_pointer)
    elif isinstance(param_value_raw, str):
        # 替换无法编码的字符（如 \ufffd）为 0x00，还原为128字节
        param_value_bytes = param_value_raw.encode('latin1', errors
='replace')
    else:
        param_value_bytes = bytes(param_value_raw)

    # 参数类型映射
    type_names = {
        0: "INT8", 1: "UINT8", 2: "INT16", 3: "UINT16", 4: "INT32",
        5: "UINT32", 6: "INT64", 7: "UINT64", 8: "FLOAT", 9: "FLOA
T",
        10: "CHAR", 11: "CHAR", 12: "UINT8_ENUM", 13: "STRING"
    }
    param_type = type_names.get(msg.param_type, f"UNKNOWN({msg.para
m_type})")

    print("\n  Received PARAM_EXT_VALUE:")
    print(f"    ID: {param_id}")
    print(msg.get_msgbuf().hex())
    print(f"    Type: {param_type}  |  Data: {len(param_value_byte
s)} bytes")
    print(f"    Index/Total: {msg.param_index}/{msg.param_count}")
    print(hex_dump(param_value_bytes))

    return {
        'id': param_id,
        'value': param_value_bytes,
        'type': param_type,
        'index': msg.param_index,
        'count': msg.param_count
    }


def receive_all_param_ext_values(vehicle, timeout=10.0):
    """接收所有参数值，直到收到 param_count 条消息后退出循环"""
    received_params = []
    expected_count = None
```

```
    print(f"  Waiting for all PARAM_EXT_VALUE messages...")

    while True:
        msg = vehicle.recv_match(type='PARAM_EXT_VALUE', blocking=True, timeout=timeout)

        if not msg:
            print(f"  Timeout: Received {len(received_params)} parameters (expected {expected_count if expected_count else 'unknown'})")
            break

        param_id = msg.param_id
        param_value_raw = msg.param_value

        # 直接获取128字节原始数据
        if isinstance(param_value_raw, (bytes, bytearray)):
            param_value_bytes = bytes(param_value_raw)
        elif isinstance(param_value_raw, str):
            # 替换无法编码的字符（如 \ufffd）为 0x00
            param_value_bytes = param_value_raw.encode('latin1', errors='replace')
        else:
            param_value_bytes = bytes(param_value_raw)

        # 参数类型映射
        type_names = {
            0: "INT8", 1: "UINT8", 2: "INT16", 3: "UINT16", 4: "INT32",
            5: "UINT32", 6: "INT64", 7: "UINT64", 8: "FLOAT", 9: "DOUBLE",
            10: "CHAR", 11: "BOOL", 12: "UINT8_ENUM", 13: "STRING"
        }
        param_type = type_names.get(msg.param_type, f"UNKNOWN({msg.param_type})")

        # 首次收到消息时记录总参数数量
        if expected_count is None:
            expected_count = msg.param_count
            print(f"  Total parameters to receive: {expected_count}")

        print(f"  [{msg.param_index}/{expected_count}] {param_id} [{param_type}][{len(param_value_bytes)}B]")
        print(hex_dump(param_value_bytes))
```

```
            received_params.append({
                'id': param_id,
                'value': param_value_bytes,
                'type': param_type,
                'index': msg.param_index,
                'count': msg.param_count
            })

            # 当已接收数量达到预期总数时退出循环
            if len(received_params) >= expected_count:
                print(f"\n  Successfully received all {expected_count} parameters")
                break

    return received_params

def receive_param_ext_ack(vehicle, timeout=5.0):
    """接收参数设置确认"""
    print(f"  Waiting for PARAM_EXT_ACK ({timeout}s timeout)...")
    msg = vehicle.recv_match(type='PARAM_EXT_ACK', blocking=True, timeout=timeout)

    if not msg:
        print("  Timeout: No PARAM_EXT_ACK received")
        return None

    param_id = msg.param_id
    param_value = msg.param_value

    # 结果代码映射
    result_codes = {
        0: "  PARAM_ACK_ACCEPTED",
        1: "⚠ PARAM_ACK_VALUE_UNSUPPORTED",
        2: "  PARAM_ACK_FAILED",
        3: "  PARAM_ACK_IN_PROGRESS"
    }
    result = result_codes.get(msg.param_result, f"  UNKNOWN({msg.param_result})")

    print("\n  Received PARAM_EXT_ACK:")
    print(f"    ID: {param_id}")
    print(f"    Value: {param_value}")
    print(f"    Result: {result}")
```

```
    return {
        'id': param_id,
        'value': param_value,
        'result': result,
        'result_code': msg.param_result
    }

def main():
    print("  Starting MAVLink PARAM_EXT Tool (UDP Listening Mode)")
    print("------------------------------------------------------")
    print("This script listens for MAVLink on ALL interfaces:1455
0")
    print("Ensure your vehicle is configured to send to this port")
    print("------------------------------------------------------
\n")

    vehicle = connect_to_vehicle()

    # 操作选择
    print("\nAvailable operations:")
    print("1. Read parameter value (by ID or index)")
    print("2. Set parameter value (raw/hex/string)")
    print("3. Request all parameters (PARAM_EXT_REQUEST_LIST)")
    print("4. Set struct: uint8_xy {x, y}")
    print("5. Set struct: float32_xy {x, y}")
    print("6. Set struct: float32_lxlyrxry {LX, LY, RX, RY}")
    print("7. Set struct: ering_mode {size, eringMode[16], minValu
e, maxValue}")
    choice = input("\nSelect operation [1-7]: ").strip()

    if choice == '1':
        # mode = input("Read by [id/index]: ").lower()
        mode = 'id'
        if mode == 'id':
            param_id = input("Enter parameter ID: ").strip()
            send_param_ext_request_read(vehicle, param_id=param_id)
            receive_param_ext_value(vehicle)
        elif mode == 'index':
            param_index = int(input("Enter parameter index: "))
            send_param_ext_request_read(vehicle, param_index=param_
index)
            receive_param_ext_value(vehicle)
        else:
            print("Invalid mode. Use 'id' or 'index'")
```

```
    elif choice == '2':
        param_id = input("Enter parameter ID to set: ").strip()

        print("\nEnter parameter value (supports):")
        print("  - Plain string (e.g., hello)")
        print("  - Hex string (e.g., 48656c6c6f or 0x48656c6c6f)")
        print("  - Raw bytes (e.g., b'\\x48\\x65\\x6c\\x6c\\x6f')")
        param_value_input = input("Enter value: ").strip()

        # 解析输入值
        if param_value_input.startswith("b'") or param_value_input.
startswith('b"'):
            # Python字节字面量格式
            param_value = eval(param_value_input)
        elif param_value_input.startswith('0x') or all(c in '012345
6789abcdefABCDEF ' for c in param_value_input):
            # 十六进制格式
            hex_str = param_value_input.replace('0x', '').replace('
', '')
            try:
                param_value = bytes.fromhex(hex_str)
            except ValueError:
                param_value = param_value_input.encode('utf-8')
        else:
            # 默认作为字符串处理
            param_value = param_value_input.encode('utf-8')

        print("\nParameter types:")
        print("  8 = FLOAT (most common)")
        print("  13 = STRING")
        print("  10 = CHAR (raw bytes)")
        print("  Other types require specific binary formats")
        param_type = int(input("Enter type number [8]: ") or 8)

        send_param_ext_set(vehicle, param_id, param_value, param_ty
pe)
        receive_param_ext_ack(vehicle)

    elif choice == '4':
        """设置 uint8_xy 结构体 {uint8_t x; uint8_t y}"""
        param_id = input("Enter parameter ID: ").strip()
        x = int(input("Enter x (0-255): "))
        y = int(input("Enter y (0-255): "))
        param_value = pack_uint8_xy(x, y)
        print(f"  Packed: {param_value.hex()} ({len(param_value)} b
```

```
ytes)")
        send_param_ext_set(vehicle, param_id, param_value, 10)  # C
HAR type
        receive_param_ext_ack(vehicle)

    elif choice == '5':
        """设置 float32_xy 结构体 {float32 x; float32 y}"""
        param_id = input("Enter parameter ID: ").strip()
        x = float(input("Enter x: "))
        y = float(input("Enter y: "))
        param_value = pack_float32_xy(x, y)
        print(f"  Packed: {param_value.hex()} ({len(param_value)} b
ytes)")
        send_param_ext_set(vehicle, param_id, param_value, 10)  # C
HAR type
        receive_param_ext_ack(vehicle)

    elif choice == '6':
        """设置 float32_lxlyrxry 结构体 {float32 LX; float32 LY; floa
t32 RX; float32 RY;}"""
        param_id = input("Enter parameter ID: ").strip()
        LX = float(input("Enter LX: "))
        LY = float(input("Enter LY: "))
        RX = float(input("Enter RX: "))
        RY = float(input("Enter RY: "))
        param_value = pack_float32_lxlyrxry(LX, LY, RX, RY)
        print(f"  Packed: {param_value.hex()} ({len(param_value)} b
ytes)")
        send_param_ext_set(vehicle, param_id, param_value, 10)  # C
HAR type
        receive_param_ext_ack(vehicle)

    elif choice == '7':
        """设置 ering_mode 结构体 {uint8_t size; uint8_t eringMode[1
6]; uint32_t minValue; uint32_t maxValue;}"""
        param_id = input("Enter parameter ID: ").strip()
        size = int(input("Enter size (0-16): "))
        eringMode = []
        print("Enter 16 eringMode values (0-255), one per line:")
        for i in range(16):
            val = int(input(f"  eringMode[{i}]: "))
            eringMode.append(val)
        minValue = int(input("Enter minValue: "))
        maxValue = int(input("Enter maxValue: "))
        param_value = pack_ering_mode(size, eringMode, minValue, ma
```

```
xValue)
        print(f"  Packed: {param_value.hex()} ({len(param_value)} bytes)")
        send_param_ext_set(vehicle, param_id, param_value, 10)  # CHAR type
        receive_param_ext_ack(vehicle)

    elif choice == '3':
        print("\n  Requesting all parameters...")
        send_param_ext_request_list(vehicle)
        all_params = receive_all_param_ext_values(vehicle)
        print(f"\n  Total parameters received: {len(all_params)}")

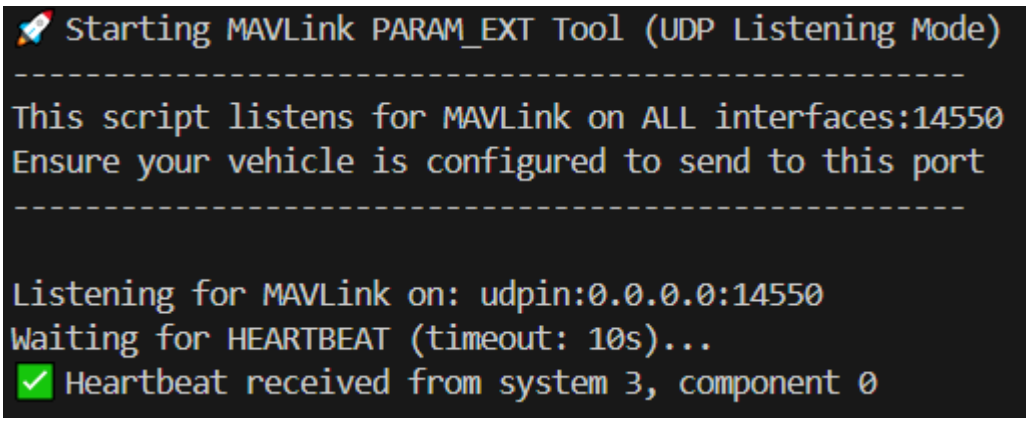
    else:
        print("Invalid choice. Select 1, 2, or 3.")


if __name__ == "__main__":
    main()
```

示例使用说明

获取

运行后，成功建立连接



输入“1”选择读取参数

```
Available operations:
1. Read parameter value (by ID or index)
2. Set parameter value (raw/hex/string)
3. Request all parameters (PARAM_EXT_REQUEST_LIST)
4. Set struct: uint8_xy {x, y}
5. Set struct: float32_xy {x, y}
6. Set struct: float32_lxlyrxry {LX, LY, RX, RY}
7. Set struct: ering_mode {size, eringMode[16], minValue, maxValue}

Select operation [1-7]: 1
```

输入对应参数“STRM_SRC_RNG”

```
Enter parameter ID: STRM_SRC_RNG
```

成功返回该参数对应的数值

```
Received PARAM_EXT_VALUE:
  ID: STRM_SRC_RNG
fd9500000103014201001200090053545 24d5f5352435f524e4700000000047afd7100000000100000002000000030000008 0ffffff0000000000000007800000005000000 6821ddb2aaaa0000800e004cffff0000600b004cffff0000507afd71ffff0000f0e19bb2000000000000
```

字段	类型	说明
videoStorageFormat	E_DjiCameraManagerVideoStorageFormat[16]	视频存储格式枚
photoRatioFormat	E_DjiCameraManagerPhotoRatio[16]	照片比例枚举列
streamSource	E_DjiCameraManagerStreamSource[16]	视频流源枚举列
streamStorage	E_DjiCameraManagerStreamStorage[16]	视频存储介质枚
nightSceneMode	E_DjiCameraManagerNightSceneMode[16]	夜景模式枚举列
meteringMode	E_DjiCameraManagerMeteringMode[16]	测光模式枚举列

示例

```
import time
import sys
import struct
import ctypes
from pymavlink import mavutil

def connect_to_vehicle():
    """建立 UDP 监听连接（默认监听所有接口的 14550 端口）"""
    connection_string = "udpin:0.0.0.0:14550"
    print(f"Listening for MAVLink on: {connection_string}")
    vehicle = mavutil.mavlink_connection(connection_string)
    print("Waiting for HEARTBEAT (timeout: 10s)...")
    try:
        vehicle.wait_heartbeat(timeout=10)
        print(f"  Heartbeat received from system {vehicle.target_sy
stem}, component {vehicle.target_component}")
        return vehicle
    except Exception as e:
        print(f"  Connection failed: {str(e)}")
        print("Ensure your vehicle is sending MAVLink messages to 1
4550 port")
        print("Example QGroundControl setup: Settings > General > M
AVLink > Listening Port = 14550")
        sys.exit(1)

def format_param_id(param_id):
    """格式化16字节参数ID（符合MAVLink协议要求）"""
    if len(param_id) > 16:
```

```
        raise ValueError("param_id cannot exceed 16 characters")

    param_id_bytes = bytearray(16)
    for i in range(min(len(param_id), 16)):
        param_id_bytes[i] = ord(param_id[i])

    # 长度<16时添加NULL终止符
    if len(param_id) < 16:
        param_id_bytes[len(param_id)] = 0

    return param_id_bytes

# ============ 结构体打包函数 ============

def pack_uint8_xy(x, y):
    """打包 {uint8_t x; uint8_t y} 结构体 (2字节)"""
    return struct.pack('<BB', x, y)

def pack_float32_xy(x, y):
    """打包 {float32 x; float32 y} 结构体 (8字节)"""
    return struct.pack('<ff', x, y)

def pack_float32_lxlyrxry(LX, LY, RX, RY):
    """打包 {float32 LX; float32 LY; float32 RX; float32 RY;} 结构体
(16字节)"""
    return struct.pack('<ffff', LX, LY, RX, RY)

def pack_ering_mode(size, eringMode, minValue, maxValue):
    """打包 {uint8_t size; uint8_t eringMode[16]; uint32_t minValue;
uint32_t maxValue;} 结构体 (25字节)

    Args:
        size: uint8_t, eringMode数组的有效长度
        eringMode: list of int (0-255), 16个uint8_t值
        minValue: uint32_t, 最小值
        maxValue: uint32_t, 最大值

    Returns:
        bytes: 打包后的二进制数据
    """
    # 确保eringMode是16字节
    eringMode_bytes = bytearray(16)
    for i in range(min(len(eringMode), 16)):
        eringMode_bytes[i] = eringMode[i] & 0xFF
```

```
    return struct.pack('<B16sII', size, eringMode_bytes, minValue,
maxValue)

def format_param_value(param_value):
    """格式化128字节参数值（接受二进制输入）"""
    if isinstance(param_value, str):
        # 如果是十六进制字符串（如 "48656c6c6f"），先转换为字节
        if param_value.startswith('0x') or all(c in '0123456789abcd
efABCDEF' for c in param_value.replace(' ', '')):
            hex_str = param_value.replace('0x', '').replace(' ',
'')
            if len(hex_str) % 2 == 0:
                param_value = bytes.fromhex(hex_str)
            else:
                raise ValueError("Invalid hex string: must have eve
n number of characters")
        else:
            # 普通字符串转字节
            param_value = param_value.encode('utf-8')

    if len(param_value) > 128:
        raise ValueError("param_value cannot exceed 128 bytes")

    param_value_bytes = bytearray(128)
    param_value_bytes[:len(param_value)] = param_value

    return param_value_bytes

def hex_dump(data, bytes_per_line=16):
    """格式化二进制数据为 hex + ASCII 展示（类 hexdump 风格）

    Returns:
        str: 格式化后的多行字符串
    """
    lines = []
    for i in range(0, len(data), bytes_per_line):
        chunk = data[i:i + bytes_per_line]
        hex_part = ' '.join(f'{b:02X}' for b in chunk)
        ascii_part = ''.join(chr(b) if 32 <= b < 127 else '.' for b
in chunk)
        lines.append(f"  {i:04X}  {hex_part:<{bytes_per_line*3}}
{ascii_part}")
    return '\n'.join(lines)

def send_param_ext_request_read(vehicle, param_id=None, param_index
```

```
=-1):
    """发送参数读取请求"""
    param_id_bytes = format_param_id(param_id) if param_id else byt
earray(16)

    vehicle.mav.param_ext_request_read_send(
        vehicle.target_system,
        vehicle.target_component,
        param_id_bytes,
        param_index
    )

    target = f"index {param_index}" if param_index >= 0 else f"ID
'{param_id}'"
    print(f"  Sent PARAM_EXT_REQUEST_READ for {target}")

def send_param_ext_request_list(vehicle):
    """发送请求所有参数列表"""
    vehicle.mav.param_ext_request_list_send(
        vehicle.target_system,
        vehicle.target_component
    )
    print(f"  Sent PARAM_EXT_REQUEST_LIST to system {vehicle.target
_system}, component {vehicle.target_component}")

def send_param_ext_set(vehicle, param_id, param_value, param_type):
    """发送参数设置请求"""
    vehicle.mav.param_ext_set_send(
        vehicle.target_system,
        vehicle.target_component,
        format_param_id(param_id),
        format_param_value(param_value),
        param_type
    )
    print(f"  Sent PARAM_EXT_SET for '{param_id}' = '{param_value}'
(type: {param_type})")

def receive_param_ext_value(vehicle, timeout=5.0):
    """接收参数值响应"""
    print(f"  Waiting for PARAM_EXT_VALUE ({timeout}s timeout)...")
    msg = vehicle.recv_match(type='PARAM_EXT_VALUE', blocking=True,
timeout=timeout)

    if not msg:
        print("  Timeout: No PARAM_EXT_VALUE received")
```

```
            return None

        # 解析参数ID
        param_id = msg.param_id

        # 解析参数值（直接获取128字节原始数据）
        param_value_raw = msg.param_value
        # byte_pointer = (ctypes.c_ubyte * 128).from_buffer_copy(msg.pa
ram_value)
        if isinstance(param_value_raw, (bytes, bytearray)):
            param_value_bytes = bytes(param_value_raw)
        # param_value_bytes = bytes(byte_pointer)
        elif isinstance(param_value_raw, str):
            # 替换无法编码的字符（如 \ufffd）为 0x00，还原为128字节
            param_value_bytes = param_value_raw.encode('latin1', errors
='replace')
        else:
            param_value_bytes = bytes(param_value_raw)

        # 参数类型映射
        type_names = {
            0: "INT8", 1: "UINT8", 2: "INT16", 3: "UINT16", 4: "INT32",
            5: "UINT32", 6: "INT64", 7: "UINT64", 8: "FLOAT", 9: "FLOA
T",
            10: "CHAR", 11: "CHAR", 12: "UINT8_ENUM", 13: "STRING"
        }
        param_type = type_names.get(msg.param_type, f"UNKNOWN({msg.para
m_type})")

        print("\n  Received PARAM_EXT_VALUE:")
        print(f"    ID: {param_id}")
        print(f"    Type: {param_type}  |  Data: {len(param_value_byte
s)} bytes")
        print(f"    Index/Total: {msg.param_index}/{msg.param_count}")
        print(hex_dump(param_value_bytes))

        return {
            'id': param_id,
            'value': param_value_bytes,
            'type': param_type,
            'index': msg.param_index,
            'count': msg.param_count
        }
```

```
def receive_all_param_ext_values(vehicle, timeout=10.0):
    """接收所有参数值，直到收到 param_count 条消息后退出循环"""
    received_params = []
    expected_count = None
    print(f"  Waiting for all PARAM_EXT_VALUE messages...")

    while True:
        msg = vehicle.recv_match(type='PARAM_EXT_VALUE', blocking=True, timeout=timeout)

        if not msg:
            print(f"  Timeout: Received {len(received_params)} parameters (expected {expected_count if expected_count else 'unknown'})")
            break

        param_id = msg.param_id
        param_value_raw = msg.param_value

        # 直接获取128字节原始数据
        if isinstance(param_value_raw, (bytes, bytearray)):
            param_value_bytes = bytes(param_value_raw)
        elif isinstance(param_value_raw, str):
            # 替换无法编码的字符（如 \ufffd）为 0x00
            param_value_bytes = param_value_raw.encode('latin1', errors='replace')
        else:
            param_value_bytes = bytes(param_value_raw)

        # 参数类型映射
        type_names = {
            0: "INT8", 1: "UINT8", 2: "INT16", 3: "UINT16", 4: "INT32",
            5: "UINT32", 6: "INT64", 7: "UINT64", 8: "FLOAT", 9: "DOUBLE",
            10: "CHAR", 11: "BOOL", 12: "UINT8_ENUM", 13: "STRING"
        }
        param_type = type_names.get(msg.param_type, f"UNKNOWN({msg.param_type})")

        # 首次收到消息时记录总参数数量
        if expected_count is None:
            expected_count = msg.param_count
            print(f"  Total parameters to receive: {expected_count}")
```

```
            print(f"  [{msg.param_index}/{expected_count}] {param_id}
[{param_type}][{len(param_value_bytes)}B]")
            print(hex_dump(param_value_bytes))

            received_params.append({
                'id': param_id,
                'value': param_value_bytes,
                'type': param_type,
                'index': msg.param_index,
                'count': msg.param_count
            })

            # 当已接收数量达到预期总数时退出循环
            if len(received_params) >= expected_count:
                print(f"\n  Successfully received all {expected_count}
parameters")
                break

    return received_params

def receive_param_ext_ack(vehicle, timeout=5.0):
    """接收参数设置确认"""
    print(f"  Waiting for PARAM_EXT_ACK ({timeout}s timeout)...")
    msg = vehicle.recv_match(type='PARAM_EXT_ACK', blocking=True, t
imeout=timeout)

    if not msg:
        print("  Timeout: No PARAM_EXT_ACK received")
        return None

    param_id = msg.param_id
    param_value = msg.param_value

    # 结果代码映射
    result_codes = {
        0: "  PARAM_ACK_ACCEPTED",
        1: "⚠ PARAM_ACK_VALUE_UNSUPPORTED",
        2: "  PARAM_ACK_FAILED",
        3: "  PARAM_ACK_IN_PROGRESS"
    }
    result = result_codes.get(msg.param_result, f"  UNKNOWN({msg.pa
ram_result})")

    print("\n  Received PARAM_EXT_ACK:")
```

```
    print(f"    ID: {param_id}")
    print(f"    Value: {param_value}")
    print(f"    Result: {result}")

    return {
        'id': param_id,
        'value': param_value,
        'result': result,
        'result_code': msg.param_result
    }

def main():
    print("  Starting MAVLink PARAM_EXT Tool (UDP Listening Mode)")
    print("------------------------------------------------------")
    print("This script listens for MAVLink on ALL interfaces:1455
0")
    print("Ensure your vehicle is configured to send to this port")
    print("------------------------------------------------------
\n")

    vehicle = connect_to_vehicle()

    # 操作选择
    print("\nAvailable operations:")
    print("1. Read parameter value (by ID or index)")
    print("2. Set parameter value (raw/hex/string)")
    print("3. Request all parameters (PARAM_EXT_REQUEST_LIST)")
    print("4. Set struct: uint8_xy {x, y}")
    print("5. Set struct: float32_xy {x, y}")
    print("6. Set struct: float32_lxlyrxry {LX, LY, RX, RY}")
    print("7. Set struct: ering_mode {size, eringMode[16], minValu
e, maxValue}")
    choice = input("\nSelect operation [1-7]: ").strip()

    if choice == '1':
        # mode = input("Read by [id/index]: ").lower()
        mode = 'id'
        if mode == 'id':
            param_id = input("Enter parameter ID: ").strip()
            send_param_ext_request_read(vehicle, param_id=param_id)
            receive_param_ext_value(vehicle)
        elif mode == 'index':
            param_index = int(input("Enter parameter index: "))
            send_param_ext_request_read(vehicle, param_index=param_
index)
```

```
            receive_param_ext_value(vehicle)
        else:
            print("Invalid mode. Use 'id' or 'index'")

    elif choice == '2':
        param_id = input("Enter parameter ID to set: ").strip()

        print("\nEnter parameter value (supports):")
        print("  - Plain string (e.g., hello)")
        print("  - Hex string (e.g., 48656c6c6f or 0x48656c6c6f)")
        print("  - Raw bytes (e.g., b'\\x48\\x65\\x6c\\x6c\\x6f')")
        param_value_input = input("Enter value: ").strip()

        # 解析输入值
        if param_value_input.startswith("b'") or param_value_input.
startswith('b"'):
            # Python字节字面量格式
            param_value = eval(param_value_input)
        elif param_value_input.startswith('0x') or all(c in '012345
6789abcdefABCDEF ' for c in param_value_input):
            # 十六进制格式
            hex_str = param_value_input.replace('0x', '').replace('
', '')
            try:
                param_value = bytes.fromhex(hex_str)
            except ValueError:
                param_value = param_value_input.encode('utf-8')
        else:
            # 默认作为字符串处理
            param_value = param_value_input.encode('utf-8')

        print("\nParameter types:")
        print("  8 = FLOAT (most common)")
        print("  13 = STRING")
        print("  10 = CHAR (raw bytes)")
        print("  Other types require specific binary formats")
        param_type = int(input("Enter type number [8]: ") or 8)

        send_param_ext_set(vehicle, param_id, param_value, param_ty
pe)
        receive_param_ext_ack(vehicle)

    elif choice == '4':
        """设置 uint8_xy 结构体 {uint8_t x; uint8_t y}"""
        param_id = input("Enter parameter ID: ").strip()
```

```
        x = int(input("Enter x (0-255): "))
        y = int(input("Enter y (0-255): "))
        param_value = pack_uint8_xy(x, y)
        print(f"  Packed: {param_value.hex()} ({len(param_value)} bytes)")
        send_param_ext_set(vehicle, param_id, param_value, 10)  # CHAR type
        receive_param_ext_ack(vehicle)

    elif choice == '5':
        """设置 float32_xy 结构体 {float32 x; float32 y}"""
        param_id = input("Enter parameter ID: ").strip()
        x = float(input("Enter x: "))
        y = float(input("Enter y: "))
        param_value = pack_float32_xy(x, y)
        print(f"  Packed: {param_value.hex()} ({len(param_value)} bytes)")
        send_param_ext_set(vehicle, param_id, param_value, 10)  # CHAR type
        receive_param_ext_ack(vehicle)

    elif choice == '6':
        """设置 float32_lxlyrxry 结构体 {float32 LX; float32 LY; float32 RX; float32 RY;}"""
        param_id = input("Enter parameter ID: ").strip()
        LX = float(input("Enter LX: "))
        LY = float(input("Enter LY: "))
        RX = float(input("Enter RX: "))
        RY = float(input("Enter RY: "))
        param_value = pack_float32_lxlyrxry(LX, LY, RX, RY)
        print(f"  Packed: {param_value.hex()} ({len(param_value)} bytes)")
        send_param_ext_set(vehicle, param_id, param_value, 10)  # CHAR type
        receive_param_ext_ack(vehicle)

    elif choice == '7':
        """设置 ering_mode 结构体 {uint8_t size; uint8_t eringMode[16]; uint32_t minValue; uint32_t maxValue;}"""
        param_id = input("Enter parameter ID: ").strip()
        size = int(input("Enter size (0-16): "))
        eringMode = []
        print("Enter 16 eringMode values (0-255), one per line:")
        for i in range(16):
            val = int(input(f"  eringMode[{i}]: "))
```

```
            eringMode.append(val)
        minValue = int(input("Enter minValue: "))
        maxValue = int(input("Enter maxValue: "))
        param_value = pack_ering_mode(size, eringMode, minValue, ma
xValue)
        print(f"  Packed: {param_value.hex()} ({len(param_value)} b
ytes)")
        send_param_ext_set(vehicle, param_id, param_value, 10)  # C
HAR type
        receive_param_ext_ack(vehicle)

    elif choice == '3':
        print("\n  Requesting all parameters...")
        send_param_ext_request_list(vehicle)
        all_params = receive_all_param_ext_values(vehicle)
        print(f"\n  Total parameters received: {len(all_params)}")

    else:
        print("Invalid choice. Select 1, 2, or 3.")


if __name__ == "__main__":
    main()
```

运行后，成功建立连接

```
🚀 Starting MAVLink PARAM_EXT Tool (UDP Listening Mode)
------------------------------------------------------
This script listens for MAVLink on ALL interfaces:14550
Ensure your vehicle is configured to send to this port
------------------------------------------------------

Listening for MAVLink on: udpin:0.0.0.0:14550
Waiting for HEARTBEAT (timeout: 10s)...
✅ Heartbeat received from system 3, component 0
```

输入“1”选择设置参数

```
Available operations:
1. Read parameter value (by ID or index)
2. Set parameter value (raw/hex/string)
3. Request all parameters (PARAM_EXT_REQUEST_LIST)
4. Set struct: uint8_xy {x, y}
5. Set struct: float32_xy {x, y}
6. Set struct: float32_lxlyrxry {LX, LY, RX, RY}
7. Set struct: ering_mode {size, eringMode[16], minValue, maxValue}

Select operation [1-7]: 1
```

输入对应参数“STRM_SRC_RNG”

```
Enter parameter ID: STRM_SRC_RNG
```

成功返回完整消息

```
📩 Received PARAM_EXT_VALUE:
  🆔 ID: STRM_SRC_RNG
fd950000000301420100010000005354524d5f5352435f524e470000000004fad671000000000100000002000000030000080ffffff0000000
0000000007800000005000000068214babaaaa0000f00c0040ffff0000800e0040ffff000050fad671ffff0000f0e109ab0000000000000000000
0000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000b251c
```

相机流媒体数据源

STREAM_SOURCE

设置

mavlink消息：

接收PARAM_SET
发送PARAM_VALUE

mavlink消息：

接收PARAM_SET
发送PARAM_VALUE

获取

mavlink消息

接收PARAM_REQUEST_READ
发送PARAM_VALUE

参数：

名称	枚举值	说明
DJI_CAMERA_MANAGER_SOURCE_DEFAULT_CAM	0x0	默认相机源
DJI_CAMERA_MANAGER_SOURCE_WIDE_CAM	0x1	广角相机
DJI_CAMERA_MANAGER_SOURCE_ZOOM_CAM	0x2	变焦相机
DJI_CAMERA_MANAGER_SOURCE_IR_CAM	0x3	红外相机
DJI_CAMERA_MANAGER_SOURCE_VISIBLE_CAM	0x7	可见光相机

示例

```
import time
import sys
from pymavlink import mavutil

def connect_to_vehicle():
    """建立 UDP 监听连接（默认监听所有接口的 14550 端口）"""
    connection_string = "udpin:0.0.0.0:14550"
    print(f"Listening for MAVLink on: {connection_string}")
    vehicle = mavutil.mavlink_connection(connection_string)
    print("Waiting for HEARTBEAT (timeout: 10s)...")
    try:
        vehicle.wait_heartbeat(timeout=10)
        print(f"  Heartbeat received from system {vehicle.target_system}, component {vehicle.target_component}")
        return vehicle
    except Exception as e:
        print(f"  Connection failed: {str(e)}")
        print("Ensure your vehicle is sending MAVLink messages to 14550 port")
        print("Example QGroundControl setup: Settings > General > MAVLink > Listening Port = 14550")
        sys.exit(1)

def format_param_id(param_id):
    """格式化16字节参数ID（符合MAVLink协议要求）"""
    if len(param_id) > 16:
        raise ValueError("param_id cannot exceed 16 characters")

    param_id_bytes = bytearray(16)
    for i in range(min(len(param_id), 16)):
        param_id_bytes[i] = ord(param_id[i])
```

```
    # 长度<16时添加NULL终止符
    if len(param_id) < 16:
        param_id_bytes[len(param_id)] = 0

    return param_id_bytes

def send_param_set(vehicle, param_id, param_value, param_type):
    """发送参数设置请求"""
    vehicle.mav.param_set_send(
        vehicle.target_system,
        vehicle.target_component,
        format_param_id(param_id),
        param_value,
        param_type
    )
    print(f"  Sent PARAM_SET for '{param_id}' = {param_value} (type: {param_type})")

def send_param_request_read(vehicle, param_id=None, param_index=-1):
    """发送参数读取请求"""
    param_id_bytes = format_param_id(param_id) if param_id else bytearray(16)

    vehicle.mav.param_request_read_send(
        vehicle.target_system,
        vehicle.target_component,
        param_id_bytes,
        param_index
    )

    target = f"index {param_index}" if param_index >= 0 else f"ID '{param_id}'"
    print(f"  Sent PARAM_REQUEST_READ for {target}")

def send_param_request_list(vehicle):
    """发送请求所有参数列表"""
    vehicle.mav.param_request_list_send(
        vehicle.target_system,
        vehicle.target_component
    )
    print(f"  Sent PARAM_REQUEST_LIST to system {vehicle.target_system}, component {vehicle.target_component}")
```

```
def receive_param_value(vehicle, timeout=5.0):
    """接收参数值响应"""
    print(f"  Waiting for PARAM_VALUE ({timeout}s timeout)...")
    msg = vehicle.recv_match(type='PARAM_VALUE', blocking=True, tim
eout=timeout)

    if not msg:
        print("  Timeout: No PARAM_VALUE received")
        return None

    # 解析参数ID
    param_id = msg.param_id

    # 解析参数值
    param_value = msg.param_value

    # 参数类型映射
    type_names = {
        0: "INT8", 1: "UINT8", 2: "INT16", 3: "UINT16", 4: "INT32",
        5: "UINT32", 6: "INT64", 7: "UINT64", 8: "FLOAT", 9: "DOUBL
E"
    }
    param_type = type_names.get(msg.param_type, f"UNKNOWN({msg.para
m_type})")

    print("\n  Received PARAM_VALUE:")
    print(f"    ID: {param_id}")
    print(f"    Value: {param_value}")
    print(f"    Type: {param_type}")
    print(f"    Index/Total: {msg.param_index}/{msg.param_count}")

    return {
        'id': param_id,
        'value': param_value,
        'type': param_type,
        'index': msg.param_index,
        'count': msg.param_count
    }

def receive_all_param_values(vehicle, timeout=10.0):
    """接收所有参数值，直到收到 param_count 条消息后退出循环"""
    received_params = []
    expected_count = None
    print(f"  Waiting for all PARAM_VALUE messages...")
```

```
    while True:
        msg = vehicle.recv_match(type='PARAM_VALUE', blocking=True, timeout=timeout)

        if not msg:
            print(f"  Timeout: Received {len(received_params)} parameters (expected {expected_count if expected_count else 'unknown'})")
            break

        param_id = msg.param_id
        param_value = msg.param_value

        # 参数类型映射
        type_names = {
            0: "INT8", 1: "UINT8", 2: "INT16", 3: "UINT16", 4: "INT32",
            5: "UINT32", 6: "INT64", 7: "UINT64", 8: "FLOAT", 9: "DOUBLE"
        }
        param_type = type_names.get(msg.param_type, f"UNKNOWN({msg.param_type})")

        # 首次收到消息时记录总参数数量
        if expected_count is None:
            expected_count = msg.param_count
            print(f"  Total parameters to receive: {expected_count}")

        print(f"  [{msg.param_index}/{expected_count}] {param_id} = {param_value} ({param_type})")

        received_params.append({
            'id': param_id,
            'value': param_value,
            'type': param_type,
            'index': msg.param_index,
            'count': msg.param_count
        })

        # 当已接收数量达到预期总数时退出循环
        if len(received_params) >= expected_count:
            print(f"\n  Successfully received all {expected_count} parameters")
            break
```

```
    return received_params

def main():
    print("  Starting MAVLink PARAM_SET Tool (UDP Listening Mode)")
    print("-------------------------------------------------------")
    print("This script listens for MAVLink on ALL interfaces:1455
0")
    print("Ensure your vehicle is configured to send to this port")
    print("-------------------------------------------------------
\n")

    vehicle = connect_to_vehicle()

    # 操作选择
    print("\nAvailable operations:")
    print("1. Set parameter value")
    print("2. Read parameter value (by ID or index)")
    print("3. Request all parameters (PARAM_REQUEST_LIST)")
    choice = input("\nSelect operation [1-3]: ").strip()

    if choice == '1':
        param_id = input("Enter parameter ID to set: ").strip()
        param_value = float(input("Enter new value: ").strip())

        print("\nParameter types:")
        print("  8 = FLOAT (most common)")
        print("  9 = DOUBLE")
        print("  4 = INT32")
        print("  1 = UINT8")
        param_type = int(input("Enter type number [8]: ").strip() o
r 8)

        send_param_set(vehicle, param_id, param_value, param_type)
        receive_param_value(vehicle)

    elif choice == '2':
        # mode = input("Read by [id/index]: ").lower()
        mode = 'id'
        if mode == 'id':
            param_id = input("Enter parameter ID: ").strip()
            send_param_request_read(vehicle, param_id=param_id)
            receive_param_value(vehicle)
        elif mode == 'index':
            param_index = int(input("Enter parameter index: "))
```

```
            send_param_request_read(vehicle, param_index=param_inde
x)
            receive_param_value(vehicle)
        else:
            print("Invalid mode. Use 'id' or 'index'")

    elif choice == '3':
        print("\n  Requesting all parameters...")
        send_param_request_list(vehicle)
        all_params = receive_all_param_values(vehicle)
        print(f"\n  Total parameters received: {len(all_params)}")

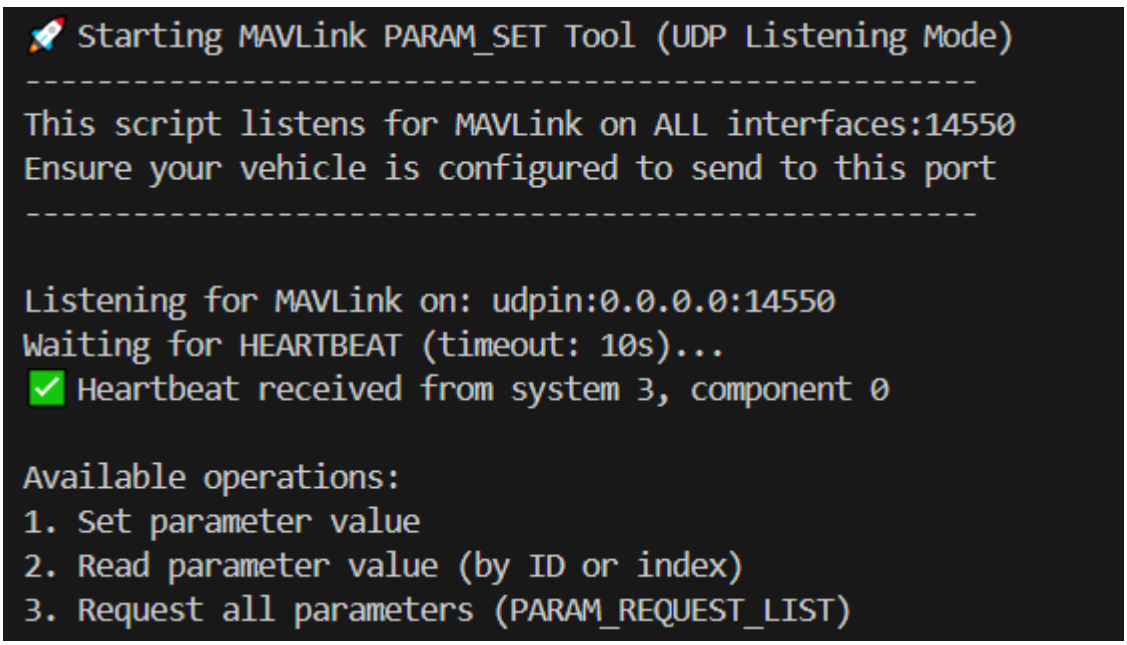
    else:
        print("Invalid choice. Select 1, 2, or 3.")


if __name__ == "__main__":
    main()
```

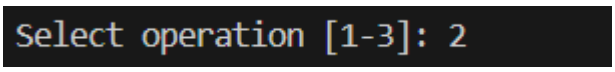
示例使用说明

获取

运行后，成功建立连接



输入“2”选择读取参数



输入对应参数“STREAM_SOURCE”

```
Enter parameter ID: STREAM_SOURCE
```

成功返回该参数对应的数值

```
📩 Received PARAM_VALUE:
  🆔 ID: STREAM_SOURCE
  🔢 Value: 1.0
```

设置

运行后，成功建立连接

```
🚀 Starting MAVLink PARAM_SET Tool (UDP Listening Mode)
------------------------------------------------------
This script listens for MAVLink on ALL interfaces:14550
Ensure your vehicle is configured to send to this port
------------------------------------------------------

Listening for MAVLink on: udpin:0.0.0.0:14550
Waiting for HEARTBEAT (timeout: 10s)...
✅ Heartbeat received from system 3, component 0

Available operations:
1. Set parameter value
2. Read parameter value (by ID or index)
3. Request all parameters (PARAM_REQUEST_LIST)
```

输入“1”选择设置参数

```
Select operation [1-3]: 1
```

输入对应参数“STREAM_SOURCE”

```
Enter parameter ID to set: STREAM_SOURCE
```

输入要设置的数值

```
Enter new value: 2
```

选择参数类型“1”

```
  8 = FLOAT (most common)
  9 = DOUBLE
  4 = INT32
  1 = UINT8
Enter type number [8]: 1
```

成功返回设置值

```
📥 Received PARAM_VALUE:
  🆔 ID: STREAM_SOURCE
  🔢 Value: 2.0
```

获取照片存储格式范围

PHT_STORE_FMTRNG

mavlink消息：

接收PARAM_EXT_REQUEST_READ
发送PARAM_EXT_VALUE

参数：

字段	类型	说明
size	uint8_t	有效列表长度（最大16）
minValue	uint32_t	数值范围最小值
maxValue	uint32_t	数值范围最大值

union（同一时刻仅生效一个字段）

字段	类型	说明
photoStorageFormat	E_DjiCameraManagerPhotoStorageFormat[16]	照片存储格式枚
videoStorageFormat	E_DjiCameraManagerVideoStorageFormat[16]	视频存储格式枚
photoRatioFormat	E_DjiCameraManagerPhotoRatio[16]	照片比例枚举列
streamSource	E_DjiCameraManagerStreamSource[16]	视频流源枚举列
streamStorage	E_DjiCameraManagerStreamStorage[16]	视频存储介质枚

字段	类型	说明
nightSceneMode	E_DjiCameraManagerNightSceneMode[16]	夜景模式枚举列
meteringMode	E_DjiCameraManagerMeteringMode[16]	测光模式枚举列

示例

```
import time
import sys
import struct
import ctypes
from pymavlink import mavutil

def connect_to_vehicle():
    """建立 UDP 监听连接（默认监听所有接口的 14550 端口）"""
    connection_string = "udpin:0.0.0.0:14550"
    print(f"Listening for MAVLink on: {connection_string}")
    vehicle = mavutil.mavlink_connection(connection_string)
    print("Waiting for HEARTBEAT (timeout: 10s)...")
    try:
        vehicle.wait_heartbeat(timeout=10)
        print(f"  Heartbeat received from system {vehicle.target_sy
stem}, component {vehicle.target_component}")
        return vehicle
    except Exception as e:
        print(f"  Connection failed: {str(e)}")
        print("Ensure your vehicle is sending MAVLink messages to 1
4550 port")
        print("Example QGroundControl setup: Settings > General > M
AVLink > Listening Port = 14550")
        sys.exit(1)

def format_param_id(param_id):
    """格式化16字节参数ID（符合MAVLink协议要求）"""
    if len(param_id) > 16:
        raise ValueError("param_id cannot exceed 16 characters")

    param_id_bytes = bytearray(16)
    for i in range(min(len(param_id), 16)):
        param_id_bytes[i] = ord(param_id[i])

    # 长度<16时添加NULL终止符
    if len(param_id) < 16:
```

```
        param_id_bytes[len(param_id)] = 0

    return param_id_bytes

# ============ 结构体打包函数 ============

def pack_uint8_xy(x, y):
    """打包 {uint8_t x; uint8_t y} 结构体 (2字节)"""
    return struct.pack('<BB', x, y)

def pack_float32_xy(x, y):
    """打包 {float32 x; float32 y} 结构体 (8字节)"""
    return struct.pack('<ff', x, y)

def pack_float32_lxlyrxry(LX, LY, RX, RY):
    """打包 {float32 LX; float32 LY; float32 RX; float32 RY;} 结构体
(16字节)"""
    return struct.pack('<ffff', LX, LY, RX, RY)

def pack_ering_mode(size, eringMode, minValue, maxValue):
    """打包 {uint8_t size; uint8_t eringMode[16]; uint32_t minValue;
uint32_t maxValue;} 结构体 (25字节)

    Args:
        size: uint8_t, eringMode数组的有效长度
        eringMode: list of int (0-255), 16个uint8_t值
        minValue: uint32_t, 最小值
        maxValue: uint32_t, 最大值

    Returns:
        bytes: 打包后的二进制数据
    """
    # 确保eringMode是16字节
    eringMode_bytes = bytearray(16)
    for i in range(min(len(eringMode), 16)):
        eringMode_bytes[i] = eringMode[i] & 0xFF

    return struct.pack('<B16sII', size, eringMode_bytes, minValue,
maxValue)

def format_param_value(param_value):
    """格式化128字节参数值（接受二进制输入）"""
    if isinstance(param_value, str):
        # 如果是十六进制字符串（如 "48656c6c6f"），先转换为字节
        if param_value.startswith('0x') or all(c in '0123456789abcd
```

```
efABCDEF' for c in param_value.replace(' ', '')):
            hex_str = param_value.replace('0x', '').replace(' ',
'')
            if len(hex_str) % 2 == 0:
                param_value = bytes.fromhex(hex_str)
            else:
                raise ValueError("Invalid hex string: must have eve
n number of characters")
        else:
            # 普通字符串转字节
            param_value = param_value.encode('utf-8')

    if len(param_value) > 128:
        raise ValueError("param_value cannot exceed 128 bytes")

    param_value_bytes = bytearray(128)
    param_value_bytes[:len(param_value)] = param_value

    return param_value_bytes

def hex_dump(data, bytes_per_line=16):
    """格式化二进制数据为 hex + ASCII 展示（类 hexdump 风格）

    Returns:
        str: 格式化后的多行字符串
    """
    lines = []
    for i in range(0, len(data), bytes_per_line):
        chunk = data[i:i + bytes_per_line]
        hex_part = ' '.join(f'{b:02X}' for b in chunk)
        ascii_part = ''.join(chr(b) if 32 <= b < 127 else '.' for b
in chunk)
        lines.append(f"  {i:04X}  {hex_part:<{bytes_per_line*3}}
{ascii_part}")
    return '\n'.join(lines)

def send_param_ext_request_read(vehicle, param_id=None, param_index
=-1):
    """发送参数读取请求"""
    param_id_bytes = format_param_id(param_id) if param_id else byt
earray(16)

    vehicle.mav.param_ext_request_read_send(
        vehicle.target_system,
        vehicle.target_component,
```

```
        param_id_bytes,
        param_index
    )

    target = f"index {param_index}" if param_index >= 0 else f"ID
'{param_id}'"
    print(f"  Sent PARAM_EXT_REQUEST_READ for {target}")

def send_param_ext_request_list(vehicle):
    """发送请求所有参数列表"""
    vehicle.mav.param_ext_request_list_send(
        vehicle.target_system,
        vehicle.target_component
    )
    print(f"  Sent PARAM_EXT_REQUEST_LIST to system {vehicle.target
_system}, component {vehicle.target_component}")

def send_param_ext_set(vehicle, param_id, param_value, param_type):
    """发送参数设置请求"""
    vehicle.mav.param_ext_set_send(
        vehicle.target_system,
        vehicle.target_component,
        format_param_id(param_id),
        format_param_value(param_value),
        param_type
    )
    print(f"  Sent PARAM_EXT_SET for '{param_id}' = '{param_value}'
(type: {param_type})")

def receive_param_ext_value(vehicle, timeout=5.0):
    """接收参数值响应"""
    print(f"  Waiting for PARAM_EXT_VALUE ({timeout}s timeout)...")
    msg = vehicle.recv_match(type='PARAM_EXT_VALUE', blocking=True,
timeout=timeout)

    if not msg:
        print("  Timeout: No PARAM_EXT_VALUE received")
        return None

    # 解析参数ID
    param_id = msg.param_id

    # 解析参数值（直接获取128字节原始数据）
    param_value_raw = msg.param_value
    # byte_pointer = (ctypes.c_ubyte * 128).from_buffer_copy(msg.pa
```

```
ram_value)
    if isinstance(param_value_raw, (bytes, bytearray)):
        param_value_bytes = bytes(param_value_raw)
    # param_value_bytes = bytes(byte_pointer)
    elif isinstance(param_value_raw, str):
        # 替换无法编码的字符（如 \ufffd）为 0x00，还原为128字节
        param_value_bytes = param_value_raw.encode('latin1', errors
='replace')
    else:
        param_value_bytes = bytes(param_value_raw)

    # 参数类型映射
    type_names = {
        0: "INT8", 1: "UINT8", 2: "INT16", 3: "UINT16", 4: "INT32",
        5: "UINT32", 6: "INT64", 7: "UINT64", 8: "FLOAT", 9: "FLOA
T",
        10: "CHAR", 11: "CHAR", 12: "UINT8_ENUM", 13: "STRING"
    }
    param_type = type_names.get(msg.param_type, f"UNKNOWN({msg.para
m_type})")

    print("\n  Received PARAM_EXT_VALUE:")
    print(f"    ID: {param_id}")
    print(f"    Type: {param_type}  |  Data: {len(param_value_byte
s)} bytes")
    print(f"    Index/Total: {msg.param_index}/{msg.param_count}")
    print(hex_dump(param_value_bytes))

    return {
        'id': param_id,
        'value': param_value_bytes,
        'type': param_type,
        'index': msg.param_index,
        'count': msg.param_count
    }


def receive_all_param_ext_values(vehicle, timeout=10.0):
    """接收所有参数值，直到收到 param_count 条消息后退出循环"""
    received_params = []
    expected_count = None
    print(f"  Waiting for all PARAM_EXT_VALUE messages...")

    while True:
        msg = vehicle.recv_match(type='PARAM_EXT_VALUE', blocking=T
```

```
rue, timeout=timeout)

        if not msg:
            print(f"  Timeout: Received {len(received_params)} para
meters (expected {expected_count if expected_count else 'unknow
n'})")
            break

        param_id = msg.param_id
        param_value_raw = msg.param_value

        # 直接获取128字节原始数据
        if isinstance(param_value_raw, (bytes, bytearray)):
            param_value_bytes = bytes(param_value_raw)
        elif isinstance(param_value_raw, str):
            # 替换无法编码的字符（如 \ufffd）为 0x00
            param_value_bytes = param_value_raw.encode('latin1', er
rors='replace')
        else:
            param_value_bytes = bytes(param_value_raw)

        # 参数类型映射
        type_names = {
            0: "INT8", 1: "UINT8", 2: "INT16", 3: "UINT16", 4: "INT
32",
            5: "UINT32", 6: "INT64", 7: "UINT64", 8: "FLOAT", 9: "D
OUBLE",
            10: "CHAR", 11: "BOOL", 12: "UINT8_ENUM", 13: "STRING"
        }
        param_type = type_names.get(msg.param_type, f"UNKNOWN({msg.
param_type})")

        # 首次收到消息时记录总参数数量
        if expected_count is None:
            expected_count = msg.param_count
            print(f"  Total parameters to receive: {expected_coun
t}")

        print(f"  [{msg.param_index}/{expected_count}] {param_id}
[{param_type}][{len(param_value_bytes)}B]")
        print(hex_dump(param_value_bytes))

        received_params.append({
            'id': param_id,
            'value': param_value_bytes,
```

```
            'type': param_type,
            'index': msg.param_index,
            'count': msg.param_count
        })

        # 当已接收数量达到预期总数时退出循环
        if len(received_params) >= expected_count:
            print(f"\n  Successfully received all {expected_count} parameters")
            break

    return received_params

def receive_param_ext_ack(vehicle, timeout=5.0):
    """接收参数设置确认"""
    print(f"  Waiting for PARAM_EXT_ACK ({timeout}s timeout)...")
    msg = vehicle.recv_match(type='PARAM_EXT_ACK', blocking=True, timeout=timeout)

    if not msg:
        print("  Timeout: No PARAM_EXT_ACK received")
        return None

    param_id = msg.param_id
    param_value = msg.param_value

    # 结果代码映射
    result_codes = {
        0: "  PARAM_ACK_ACCEPTED",
        1: "⚠ PARAM_ACK_VALUE_UNSUPPORTED",
        2: "  PARAM_ACK_FAILED",
        3: "  PARAM_ACK_IN_PROGRESS"
    }
    result = result_codes.get(msg.param_result, f"  UNKNOWN({msg.param_result})")

    print("\n  Received PARAM_EXT_ACK:")
    print(f"    ID: {param_id}")
    print(f"    Value: {param_value}")
    print(f"    Result: {result}")

    return {
        'id': param_id,
        'value': param_value,
        'result': result,
```

```
        'result_code': msg.param_result
    }

def main():
    print("  Starting MAVLink PARAM_EXT Tool (UDP Listening Mode)")
    print("------------------------------------------------------")
    print("This script listens for MAVLink on ALL interfaces:1455
0")
    print("Ensure your vehicle is configured to send to this port")
    print("------------------------------------------------------
\n")

    vehicle = connect_to_vehicle()

    # 操作选择
    print("\nAvailable operations:")
    print("1. Read parameter value (by ID or index)")
    print("2. Set parameter value (raw/hex/string)")
    print("3. Request all parameters (PARAM_EXT_REQUEST_LIST)")
    print("4. Set struct: uint8_xy {x, y}")
    print("5. Set struct: float32_xy {x, y}")
    print("6. Set struct: float32_lxlyrxry {LX, LY, RX, RY}")
    print("7. Set struct: ering_mode {size, eringMode[16], minValu
e, maxValue}")
    choice = input("\nSelect operation [1-7]: ").strip()

    if choice == '1':
        # mode = input("Read by [id/index]: ").lower()
        mode = 'id'
        if mode == 'id':
            param_id = input("Enter parameter ID: ").strip()
            send_param_ext_request_read(vehicle, param_id=param_id)
            receive_param_ext_value(vehicle)
        elif mode == 'index':
            param_index = int(input("Enter parameter index: "))
            send_param_ext_request_read(vehicle, param_index=param_
index)
            receive_param_ext_value(vehicle)
        else:
            print("Invalid mode. Use 'id' or 'index'")

    elif choice == '2':
        param_id = input("Enter parameter ID to set: ").strip()

        print("\nEnter parameter value (supports):")
```

```
        print("  - Plain string (e.g., hello)")
        print("  - Hex string (e.g., 48656c6c6f or 0x48656c6c6f)")
        print("  - Raw bytes (e.g., b'\\x48\\x65\\x6c\\x6c\\x6f')")
        param_value_input = input("Enter value: ").strip()

        # 解析输入值
        if param_value_input.startswith("b'") or param_value_input.
startswith('b"'):
            # Python字节字面量格式
            param_value = eval(param_value_input)
        elif param_value_input.startswith('0x') or all(c in '012345
6789abcdefABCDEF ' for c in param_value_input):
            # 十六进制格式
            hex_str = param_value_input.replace('0x', '').replace('
', '')
            try:
                param_value = bytes.fromhex(hex_str)
            except ValueError:
                param_value = param_value_input.encode('utf-8')
        else:
            # 默认作为字符串处理
            param_value = param_value_input.encode('utf-8')

        print("\nParameter types:")
        print("  8 = FLOAT (most common)")
        print("  13 = STRING")
        print("  10 = CHAR (raw bytes)")
        print("  Other types require specific binary formats")
        param_type = int(input("Enter type number [8]: ") or 8)

        send_param_ext_set(vehicle, param_id, param_value, param_ty
pe)
        receive_param_ext_ack(vehicle)

    elif choice == '4':
        """设置 uint8_xy 结构体 {uint8_t x; uint8_t y}"""
        param_id = input("Enter parameter ID: ").strip()
        x = int(input("Enter x (0-255): "))
        y = int(input("Enter y (0-255): "))
        param_value = pack_uint8_xy(x, y)
        print(f"  Packed: {param_value.hex()} ({len(param_value)} b
ytes)")
        send_param_ext_set(vehicle, param_id, param_value, 10)  # C
HAR type
        receive_param_ext_ack(vehicle)
```

```
    elif choice == '5':
        """设置 float32_xy 结构体 {float32 x; float32 y}"""
        param_id = input("Enter parameter ID: ").strip()
        x = float(input("Enter x: "))
        y = float(input("Enter y: "))
        param_value = pack_float32_xy(x, y)
        print(f"  Packed: {param_value.hex()} ({len(param_value)} bytes)")
        send_param_ext_set(vehicle, param_id, param_value, 10)  # CHAR type
        receive_param_ext_ack(vehicle)

    elif choice == '6':
        """设置 float32_lxlyrxry 结构体 {float32 LX; float32 LY; float32 RX; float32 RY;}"""
        param_id = input("Enter parameter ID: ").strip()
        LX = float(input("Enter LX: "))
        LY = float(input("Enter LY: "))
        RX = float(input("Enter RX: "))
        RY = float(input("Enter RY: "))
        param_value = pack_float32_lxlyrxry(LX, LY, RX, RY)
        print(f"  Packed: {param_value.hex()} ({len(param_value)} bytes)")
        send_param_ext_set(vehicle, param_id, param_value, 10)  # CHAR type
        receive_param_ext_ack(vehicle)

    elif choice == '7':
        """设置 ering_mode 结构体 {uint8_t size; uint8_t eringMode[16]; uint32_t minValue; uint32_t maxValue;}"""
        param_id = input("Enter parameter ID: ").strip()
        size = int(input("Enter size (0-16): "))
        eringMode = []
        print("Enter 16 eringMode values (0-255), one per line:")
        for i in range(16):
            val = int(input(f"  eringMode[{i}]: "))
            eringMode.append(val)
        minValue = int(input("Enter minValue: "))
        maxValue = int(input("Enter maxValue: "))
        param_value = pack_ering_mode(size, eringMode, minValue, maxValue)
        print(f"  Packed: {param_value.hex()} ({len(param_value)} bytes)")
        send_param_ext_set(vehicle, param_id, param_value, 10)  # C
```

```
HAR type
            receive_param_ext_ack(vehicle)

        elif choice == '3':
            print("\n  Requesting all parameters...")
            send_param_ext_request_list(vehicle)
            all_params = receive_all_param_ext_values(vehicle)
            print(f"\n  Total parameters received: {len(all_params)}")

        else:
            print("Invalid choice. Select 1, 2, or 3.")


if __name__ == "__main__":
    main()
```

示例使用说明

运行后，成功建立连接

```
🚀 Starting MAVLink PARAM_EXT Tool (UDP Listening Mode)
------------------------------------------------------
This script listens for MAVLink on ALL interfaces:14550
Ensure your vehicle is configured to send to this port
------------------------------------------------------

Listening for MAVLink on: udpin:0.0.0.0:14550
Waiting for HEARTBEAT (timeout: 10s)...
✅ Heartbeat received from system 3, component 0
```

输入“1”选择读取参数

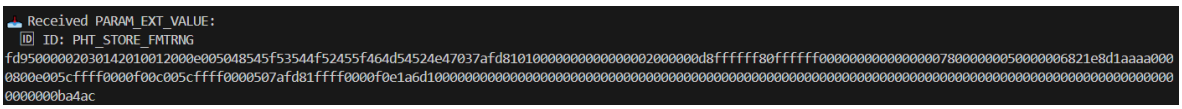
```
Available operations:
1. Read parameter value (by ID or index)
2. Set parameter value (raw/hex/string)
3. Request all parameters (PARAM_EXT_REQUEST_LIST)
4. Set struct: uint8_xy {x, y}
5. Set struct: float32_xy {x, y}
6. Set struct: float32_lxlyrxry {LX, LY, RX, RY}
7. Set struct: ering_mode {size, eringMode[16], minValue, maxValue}

Select operation [1-7]: 1
```

输入对应参数“PHT_STORE_FMTRNG”

```
Enter parameter ID: PHT_STORE_FMTRNG
```

成功返回完整消息


Received PARAM_EXT_VALUE:
ID: PHT_STORE_FMTRNG
fd950000020301420100120000e005048545f53544f52455f464d54524e47037afd8101000000000000000020000000d8fffff80ffffff000000000000000078000000050000006821e8d1aaaa0000800e005cffff0000f00c005cffff0000507afd81ffff0000f0e1a6d1000000000000000

```
def connect_to_vehicle():
    """建立 UDP 监听连接（默认监听所有接口的 14550 端口）"""
    connection_string = "udpin:0.0.0.0:14550"
    print(f"Listening for MAVLink on: {connection_string}")
    vehicle = mavutil.mavlink_connection(connection_string)
    print("Waiting for HEARTBEAT (timeout: 10s)...")
    try:
        vehicle.wait_heartbeat(timeout=10)
        print(f"  Heartbeat received from system {vehicle.target_system}, component {vehicle.target_component}")
        return vehicle
    except Exception as e:
        print(f"  Connection failed: {str(e)}")
        print("Ensure your vehicle is sending MAVLink messages to 14550 port")
        print("Example QGroundControl setup: Settings > General > MAVLink > Listening Port = 14550")
        sys.exit(1)

def format_param_id(param_id):
    """格式化16字节参数ID（符合MAVLink协议要求）"""
    if len(param_id) > 16:
        raise ValueError("param_id cannot exceed 16 characters")

    param_id_bytes = bytearray(16)
    for i in range(min(len(param_id), 16)):
        param_id_bytes[i] = ord(param_id[i])

    # 长度<16时添加NULL终止符
    if len(param_id) < 16:
        param_id_bytes[len(param_id)] = 0

    return param_id_bytes

def send_param_set(vehicle, param_id, param_value, param_type):
    """发送参数设置请求"""
    vehicle.mav.param_set_send(
        vehicle.target_system,
        vehicle.target_component,
        format_param_id(param_id),
        param_value,
        param_type
    )
    print(f"  Sent PARAM_SET for '{param_id}' = {param_value} (type: {param_type})")
```

```
def send_param_request_read(vehicle, param_id=None, param_index=-
1):
    """发送参数读取请求"""
    param_id_bytes = format_param_id(param_id) if param_id else byt
earray(16)

    vehicle.mav.param_request_read_send(
        vehicle.target_system,
        vehicle.target_component,
        param_id_bytes,
        param_index
    )

    target = f"index {param_index}" if param_index >= 0 else f"ID
'{param_id}'"
    print(f"  Sent PARAM_REQUEST_READ for {target}")

def send_param_request_list(vehicle):
    """发送请求所有参数列表"""
    vehicle.mav.param_request_list_send(
        vehicle.target_system,
        vehicle.target_component
    )
    print(f"  Sent PARAM_REQUEST_LIST to system {vehicle.target_sys
tem}, component {vehicle.target_component}")

def receive_param_value(vehicle, timeout=5.0):
    """接收参数值响应"""
    print(f"  Waiting for PARAM_VALUE ({timeout}s timeout)...")
    msg = vehicle.recv_match(type='PARAM_VALUE', blocking=True, tim
eout=timeout)

    if not msg:
        print("  Timeout: No PARAM_VALUE received")
        return None

    # 解析参数ID
    param_id = msg.param_id

    # 解析参数值
    param_value = msg.param_value

    # 参数类型映射
    type_names = {
```

```
        0: "INT8", 1: "UINT8", 2: "INT16", 3: "UINT16", 4: "INT32",
        5: "UINT32", 6: "INT64", 7: "UINT64", 8: "FLOAT", 9: "DOUBL
E"
    }
    param_type = type_names.get(msg.param_type, f"UNKNOWN({msg.para
m_type})")

    print("\n  Received PARAM_VALUE:")
    print(f"    ID: {param_id}")
    print(f"    Value: {param_value}")
    print(f"    Type: {param_type}")
    print(f"    Index/Total: {msg.param_index}/{msg.param_count}")

    return {
        'id': param_id,
        'value': param_value,
        'type': param_type,
        'index': msg.param_index,
        'count': msg.param_count
    }

def receive_all_param_values(vehicle, timeout=10.0):
    """接收所有参数值，直到收到 param_count 条消息后退出循环"""
    received_params = []
    expected_count = None
    print(f"  Waiting for all PARAM_VALUE messages...")

    while True:
        msg = vehicle.recv_match(type='PARAM_VALUE', blocking=True,
timeout=timeout)

        if not msg:
            print(f"  Timeout: Received {len(received_params)} para
meters (expected {expected_count if expected_count else 'unknow
n'})")
            break

        param_id = msg.param_id
        param_value = msg.param_value

        # 参数类型映射
        type_names = {
            0: "INT8", 1: "UINT8", 2: "INT16", 3: "UINT16", 4: "INT
32",
            5: "UINT32", 6: "INT64", 7: "UINT64", 8: "FLOAT", 9: "D
```

```
OUBLE"
        }
        param_type = type_names.get(msg.param_type, f"UNKNOWN({msg.
param_type})")

        # 首次收到消息时记录总参数数量
        if expected_count is None:
            expected_count = msg.param_count
            print(f"  Total parameters to receive: {expected_coun
t}")

        print(f"  [{msg.param_index}/{expected_count}] {param_id} =
{param_value} ({param_type})")

        received_params.append({
            'id': param_id,
            'value': param_value,
            'type': param_type,
            'index': msg.param_index,
            'count': msg.param_count
        })

        # 当已接收数量达到预期总数时退出循环
        if len(received_params) >= expected_count:
            print(f"\n  Successfully received all {expected_count}
parameters")
            break

    return received_params

def main():
    print("  Starting MAVLink PARAM_SET Tool (UDP Listening Mode)")
    print("-----------------------------------------------------")
    print("This script listens for MAVLink on ALL interfaces:1455
0")
    print("Ensure your vehicle is configured to send to this port")
    print("-----------------------------------------------------
\n")

    vehicle = connect_to_vehicle()

    # 操作选择
    print("\nAvailable operations:")
    print("1. Set parameter value")
    print("2. Read parameter value (by ID or index)")
```

```
    print("3. Request all parameters (PARAM_REQUEST_LIST)")
    choice = input("\nSelect operation [1-3]: ").strip()

    if choice == '1':
        param_id = input("Enter parameter ID to set: ").strip()
        param_value = float(input("Enter new value: ").strip())

        print("\nParameter types:")
        print("  8 = FLOAT (most common)")
        print("  9 = DOUBLE")
        print("  4 = INT32")
        print("  1 = UINT8")
        param_type = int(input("Enter type number [8]: ").strip() or 8)

        send_param_set(vehicle, param_id, param_value, param_type)
        receive_param_value(vehicle)

    elif choice == '2':
        # mode = input("Read by [id/index]: ").lower()
        mode = 'id'
        if mode == 'id':
            param_id = input("Enter parameter ID: ").strip()
            send_param_request_read(vehicle, param_id=param_id)
            receive_param_value(vehicle)
        elif mode == 'index':
            param_index = int(input("Enter parameter index: "))
            send_param_request_read(vehicle, param_index=param_index)
            receive_param_value(vehicle)
        else:
            print("Invalid mode. Use 'id' or 'index'")

    elif choice == '3':
        print("\n  Requesting all parameters...")
        send_param_request_list(vehicle)
        all_params = receive_all_param_values(vehicle)
        print(f"\n  Total parameters received: {len(all_params)}")

    else:
        print("Invalid choice. Select 1, 2, or 3.")


if __name__ == "__main__":
    main()
```

示例使用说明

获取

运行后，成功建立连接

```
🚀 Starting MAVLink PARAM_SET Tool (UDP Listening Mode)
------------------------------------------------------
This script listens for MAVLink on ALL interfaces:14550
Ensure your vehicle is configured to send to this port
------------------------------------------------------

Listening for MAVLink on: udpin:0.0.0.0:14550
Waiting for HEARTBEAT (timeout: 10s)...
✅ Heartbeat received from system 3, component 0

Available operations:
1. Set parameter value
2. Read parameter value (by ID or index)
3. Request all parameters (PARAM_REQUEST_LIST)
```

输入“2”选择读取参数

```
Select operation [1-3]: 2
```

输入对应参数“PHOTO_FORMAT”

```
Enter parameter ID: PHOTO_FORMAT
```

成功返回该参数对应的数值

```
📥 Received PARAM_VALUE:
  🆔 ID: PHOTO_FORMAT
  🔢 Value: 1.0
```

设置

运行后，成功建立连接

```
🚀 Starting MAVLink PARAM_SET Tool (UDP Listening Mode)
--------------------------------------------------
This script listens for MAVLink on ALL interfaces:14550
Ensure your vehicle is configured to send to this port
--------------------------------------------------

Listening for MAVLink on: udpin:0.0.0.0:14550
Waiting for HEARTBEAT (timeout: 10s)...
✅ Heartbeat received from system 3, component 0

Available operations:
1. Set parameter value
2. Read parameter value (by ID or index)
3. Request all parameters (PARAM_REQUEST_LIST)
```

输入“1”选择设置参数

```
Available operations:
1. Set parameter value
2. Read parameter value (by ID or index)
3. Request all parameters (PARAM_REQUEST_LIST)

Select operation [1-3]: 1
```

输入对应参数“PHOTO_FORMAT”

```
Enter parameter ID to set: PHOTO_FORMAT
```

选择参数类型“1”

```
Parameter types:
  8 = FLOAT (most common)
  9 = DOUBLE
  4 = INT32
  1 = UINT8
Enter type number [8]: 1
```

发送成功返回设置值

```
📩 Received PARAM_VALUE:
   🆔 ID: PHOTO_FORMAT
   🔢 Value: 2.0
```

获取视频存储格式范围

VID_FMT_RNG

mavlink消息：

接收PARAM_EXT_REQUEST_READ
发送PARAM_EXT_VALUE

参数：

字段	类型	说明
size	uint8_t	有效列表长度（最大16）
minValue	uint32_t	数值范围最小值
maxValue	uint32_t	数值范围最大值

union（同一时刻仅生效一个字段）

字段	类型	说明
photoStorageFormat	E_DjiCameraManagerPhotoStorageFormat[16]	照片存储格式枚
videoStorageFormat	E_DjiCameraManagerVideoStorageFormat[16]	视频存储格式枚
photoRatioFormat	E_DjiCameraManagerPhotoRatio[16]	照片比例枚举列
streamSource	E_DjiCameraManagerStreamSource[16]	视频流源枚举列
streamStorage	E_DjiCameraManagerStreamStorage[16]	视频存储介质枚
nightSceneMode	E_DjiCameraManagerNightSceneMode[16]	夜景模式枚举列
meteringMode	E_DjiCameraManagerMeteringMode[16]	测光模式枚举列

示例

```
import time
import sys
import struct
import ctypes
from pymavlink import mavutil
```

```
def connect_to_vehicle():
    """建立 UDP 监听连接（默认监听所有接口的 14550 端口）"""
    connection_string = "udpin:0.0.0.0:14550"
    print(f"Listening for MAVLink on: {connection_string}")
    vehicle = mavutil.mavlink_connection(connection_string)
    print("Waiting for HEARTBEAT (timeout: 10s)...")
    try:
        vehicle.wait_heartbeat(timeout=10)
        print(f"  Heartbeat received from system {vehicle.target_sy
stem}, component {vehicle.target_component}")
        return vehicle
    except Exception as e:
        print(f"  Connection failed: {str(e)}")
        print("Ensure your vehicle is sending MAVLink messages to 1
4550 port")
        print("Example QGroundControl setup: Settings > General > M
AVLink > Listening Port = 14550")
        sys.exit(1)

def format_param_id(param_id):
    """格式化16字节参数ID（符合MAVLink协议要求）"""
    if len(param_id) > 16:
        raise ValueError("param_id cannot exceed 16 characters")

    param_id_bytes = bytearray(16)
    for i in range(min(len(param_id), 16)):
        param_id_bytes[i] = ord(param_id[i])

    # 长度<16时添加NULL终止符
    if len(param_id) < 16:
        param_id_bytes[len(param_id)] = 0

    return param_id_bytes

# ============ 结构体打包函数 ============

def pack_uint8_xy(x, y):
    """打包 {uint8_t x; uint8_t y} 结构体 (2字节)"""
    return struct.pack('<BB', x, y)

def pack_float32_xy(x, y):
    """打包 {float32 x; float32 y} 结构体 (8字节)"""
    return struct.pack('<ff', x, y)
```

```
def pack_float32_lxlyrxry(LX, LY, RX, RY):
    """打包 {float32 LX; float32 LY; float32 RX; float32 RY;} 结构体 (16字节)"""
    return struct.pack('<ffff', LX, LY, RX, RY)

def pack_ering_mode(size, eringMode, minValue, maxValue):
    """打包 {uint8_t size; uint8_t eringMode[16]; uint32_t minValue; uint32_t maxValue;} 结构体 (25字节)

    Args:
        size: uint8_t, eringMode数组的有效长度
        eringMode: list of int (0-255), 16个uint8_t值
        minValue: uint32_t, 最小值
        maxValue: uint32_t, 最大值

    Returns:
        bytes: 打包后的二进制数据
    """
    # 确保eringMode是16字节
    eringMode_bytes = bytearray(16)
    for i in range(min(len(eringMode), 16)):
        eringMode_bytes[i] = eringMode[i] & 0xFF

    return struct.pack('<B16sII', size, eringMode_bytes, minValue, maxValue)

def format_param_value(param_value):
    """格式化128字节参数值（接受二进制输入）"""
    if isinstance(param_value, str):
        # 如果是十六进制字符串（如 "48656c6c6f"），先转换为字节
        if param_value.startswith('0x') or all(c in '0123456789abcdefABCDEF' for c in param_value.replace(' ', '')):
            hex_str = param_value.replace('0x', '').replace(' ', '')
            if len(hex_str) % 2 == 0:
                param_value = bytes.fromhex(hex_str)
            else:
                raise ValueError("Invalid hex string: must have even number of characters")
        else:
            # 普通字符串转字节
            param_value = param_value.encode('utf-8')

    if len(param_value) > 128:
        raise ValueError("param_value cannot exceed 128 bytes")
```

```
    param_value_bytes = bytearray(128)
    param_value_bytes[:len(param_value)] = param_value

    return param_value_bytes

def hex_dump(data, bytes_per_line=16):
    """格式化二进制数据为 hex + ASCII 展示（类 hexdump 风格）

    Returns:
        str: 格式化后的多行字符串
    """
    lines = []
    for i in range(0, len(data), bytes_per_line):
        chunk = data[i:i + bytes_per_line]
        hex_part = ' '.join(f'{b:02X}' for b in chunk)
        ascii_part = ''.join(chr(b) if 32 <= b < 127 else '.' for b
in chunk)
        lines.append(f"  {i:04X}  {hex_part:<{bytes_per_line*3}}
{ascii_part}")
    return '\n'.join(lines)

def send_param_ext_request_read(vehicle, param_id=None, param_index
=-1):
    """发送参数读取请求"""
    param_id_bytes = format_param_id(param_id) if param_id else byt
earray(16)

    vehicle.mav.param_ext_request_read_send(
        vehicle.target_system,
        vehicle.target_component,
        param_id_bytes,
        param_index
    )

    target = f"index {param_index}" if param_index >= 0 else f"ID
'{param_id}'"
    print(f"  Sent PARAM_EXT_REQUEST_READ for {target}")

def send_param_ext_request_list(vehicle):
    """发送请求所有参数列表"""
    vehicle.mav.param_ext_request_list_send(
        vehicle.target_system,
        vehicle.target_component
    )
```

```
    print(f"  Sent PARAM_EXT_REQUEST_LIST to system {vehicle.target
_system}, component {vehicle.target_component}")

def send_param_ext_set(vehicle, param_id, param_value, param_type):
    """发送参数设置请求"""
    vehicle.mav.param_ext_set_send(
        vehicle.target_system,
        vehicle.target_component,
        format_param_id(param_id),
        format_param_value(param_value),
        param_type
    )
    print(f"  Sent PARAM_EXT_SET for '{param_id}' = '{param_value}'
(type: {param_type})")

def receive_param_ext_value(vehicle, timeout=5.0):
    """接收参数值响应"""
    print(f"  Waiting for PARAM_EXT_VALUE ({timeout}s timeout)...")
    msg = vehicle.recv_match(type='PARAM_EXT_VALUE', blocking=True,
timeout=timeout)

    if not msg:
        print("  Timeout: No PARAM_EXT_VALUE received")
        return None

    # 解析参数ID
    param_id = msg.param_id

    # 解析参数值（直接获取128字节原始数据）
    param_value_raw = msg.param_value
    # byte_pointer = (ctypes.c_ubyte * 128).from_buffer_copy(msg.pa
ram_value)
    if isinstance(param_value_raw, (bytes, bytearray)):
        param_value_bytes = bytes(param_value_raw)
    # param_value_bytes = bytes(byte_pointer)
    elif isinstance(param_value_raw, str):
        # 替换无法编码的字符（如 \ufffd）为 0x00，还原为128字节
        param_value_bytes = param_value_raw.encode('latin1', errors
='replace')
    else:
        param_value_bytes = bytes(param_value_raw)

    # 参数类型映射
    type_names = {
        0: "INT8", 1: "UINT8", 2: "INT16", 3: "UINT16", 4: "INT32",
```

```
        5: "UINT32", 6: "INT64", 7: "UINT64", 8: "FLOAT", 9: "FLOAT",
        10: "CHAR", 11: "CHAR", 12: "UINT8_ENUM", 13: "STRING"
    }
    param_type = type_names.get(msg.param_type, f"UNKNOWN({msg.param_type})")

    print("\n  Received PARAM_EXT_VALUE:")
    print(f"    ID: {param_id}")
    print(f"    Type: {param_type}  |  Data: {len(param_value_bytes)} bytes")
    print(f"    Index/Total: {msg.param_index}/{msg.param_count}")
    print(hex_dump(param_value_bytes))

    return {
        'id': param_id,
        'value': param_value_bytes,
        'type': param_type,
        'index': msg.param_index,
        'count': msg.param_count
    }


def receive_all_param_ext_values(vehicle, timeout=10.0):
    """接收所有参数值，直到收到 param_count 条消息后退出循环"""
    received_params = []
    expected_count = None
    print(f"  Waiting for all PARAM_EXT_VALUE messages...")

    while True:
        msg = vehicle.recv_match(type='PARAM_EXT_VALUE', blocking=True, timeout=timeout)

        if not msg:
            print(f"  Timeout: Received {len(received_params)} parameters (expected {expected_count if expected_count else 'unknown'})")
            break

        param_id = msg.param_id
        param_value_raw = msg.param_value

        # 直接获取128字节原始数据
        if isinstance(param_value_raw, (bytes, bytearray)):
            param_value_bytes = bytes(param_value_raw)
```

```
        elif isinstance(param_value_raw, str):
            # 替换无法编码的字符（如 \ufffd）为 0x00
            param_value_bytes = param_value_raw.encode('latin1', er
rors='replace')
        else:
            param_value_bytes = bytes(param_value_raw)

        # 参数类型映射
        type_names = {
            0: "INT8", 1: "UINT8", 2: "INT16", 3: "UINT16", 4: "INT
32",
            5: "UINT32", 6: "INT64", 7: "UINT64", 8: "FLOAT", 9: "D
OUBLE",
            10: "CHAR", 11: "BOOL", 12: "UINT8_ENUM", 13: "STRING"
        }
        param_type = type_names.get(msg.param_type, f"UNKNOWN({msg.
param_type})")

        # 首次收到消息时记录总参数数量
        if expected_count is None:
            expected_count = msg.param_count
            print(f"  Total parameters to receive: {expected_coun
t}")

        print(f"  [{msg.param_index}/{expected_count}] {param_id}
[{param_type}][{len(param_value_bytes)}B]")
        print(hex_dump(param_value_bytes))

        received_params.append({
            'id': param_id,
            'value': param_value_bytes,
            'type': param_type,
            'index': msg.param_index,
            'count': msg.param_count
        })

        # 当已接收数量达到预期总数时退出循环
        if len(received_params) >= expected_count:
            print(f"\n  Successfully received all {expected_count}
parameters")
            break

    return received_params

def receive_param_ext_ack(vehicle, timeout=5.0):
```

```
    """接收参数设置确认"""
    print(f"  Waiting for PARAM_EXT_ACK ({timeout}s timeout)...")
    msg = vehicle.recv_match(type='PARAM_EXT_ACK', blocking=True, t
imeout=timeout)

    if not msg:
        print("  Timeout: No PARAM_EXT_ACK received")
        return None

    param_id = msg.param_id
    param_value = msg.param_value

    # 结果代码映射
    result_codes = {
        0: "  PARAM_ACK_ACCEPTED",
        1: "⚠ PARAM_ACK_VALUE_UNSUPPORTED",
        2: "  PARAM_ACK_FAILED",
        3: "  PARAM_ACK_IN_PROGRESS"
    }
    result = result_codes.get(msg.param_result, f"  UNKNOWN({msg.pa
ram_result})")

    print("\n  Received PARAM_EXT_ACK:")
    print(f"    ID: {param_id}")
    print(f"    Value: {param_value}")
    print(f"    Result: {result}")

    return {
        'id': param_id,
        'value': param_value,
        'result': result,
        'result_code': msg.param_result
    }

def main():
    print("  Starting MAVLink PARAM_EXT Tool (UDP Listening Mode)")
    print("-------------------------------------------------------")
    print("This script listens for MAVLink on ALL interfaces:1455
0")
    print("Ensure your vehicle is configured to send to this port")
    print("-------------------------------------------------------
\n")

    vehicle = connect_to_vehicle()
```

```
    # 操作选择
    print("\nAvailable operations:")
    print("1. Read parameter value (by ID or index)")
    print("2. Set parameter value (raw/hex/string)")
    print("3. Request all parameters (PARAM_EXT_REQUEST_LIST)")
    print("4. Set struct: uint8_xy {x, y}")
    print("5. Set struct: float32_xy {x, y}")
    print("6. Set struct: float32_lxlyrxry {LX, LY, RX, RY}")
    print("7. Set struct: ering_mode {size, eringMode[16], minValu
e, maxValue}")
    choice = input("\nSelect operation [1-7]: ").strip()

    if choice == '1':
        # mode = input("Read by [id/index]: ").lower()
        mode = 'id'
        if mode == 'id':
            param_id = input("Enter parameter ID: ").strip()
            send_param_ext_request_read(vehicle, param_id=param_id)
            receive_param_ext_value(vehicle)
        elif mode == 'index':
            param_index = int(input("Enter parameter index: "))
            send_param_ext_request_read(vehicle, param_index=param_
index)
            receive_param_ext_value(vehicle)
        else:
            print("Invalid mode. Use 'id' or 'index'")

    elif choice == '2':
        param_id = input("Enter parameter ID to set: ").strip()

        print("\nEnter parameter value (supports):")
        print("  - Plain string (e.g., hello)")
        print("  - Hex string (e.g., 48656c6c6f or 0x48656c6c6f)")
        print("  - Raw bytes (e.g., b'\\x48\\x65\\x6c\\x6c\\x6f')")
        param_value_input = input("Enter value: ").strip()

        # 解析输入值
        if param_value_input.startswith("b'") or param_value_input.
startswith('b"'):
            # Python字节字面量格式
            param_value = eval(param_value_input)
        elif param_value_input.startswith('0x') or all(c in '012345
6789abcdefABCDEF ' for c in param_value_input):
            # 十六进制格式
            hex_str = param_value_input.replace('0x', '').replace('
```

```
', '')
            try:
                param_value = bytes.fromhex(hex_str)
            except ValueError:
                param_value = param_value_input.encode('utf-8')
        else:
            # 默认作为字符串处理
            param_value = param_value_input.encode('utf-8')

        print("\nParameter types:")
        print("  8 = FLOAT (most common)")
        print("  13 = STRING")
        print("  10 = CHAR (raw bytes)")
        print("  Other types require specific binary formats")
        param_type = int(input("Enter type number [8]: ") or 8)

        send_param_ext_set(vehicle, param_id, param_value, param_ty
pe)
        receive_param_ext_ack(vehicle)

    elif choice == '4':
        """设置 uint8_xy 结构体 {uint8_t x; uint8_t y}"""
        param_id = input("Enter parameter ID: ").strip()
        x = int(input("Enter x (0-255): "))
        y = int(input("Enter y (0-255): "))
        param_value = pack_uint8_xy(x, y)
        print(f"  Packed: {param_value.hex()} ({len(param_value)} b
ytes)")
        send_param_ext_set(vehicle, param_id, param_value, 10)  # C
HAR type
        receive_param_ext_ack(vehicle)

    elif choice == '5':
        """设置 float32_xy 结构体 {float32 x; float32 y}"""
        param_id = input("Enter parameter ID: ").strip()
        x = float(input("Enter x: "))
        y = float(input("Enter y: "))
        param_value = pack_float32_xy(x, y)
        print(f"  Packed: {param_value.hex()} ({len(param_value)} b
ytes)")
        send_param_ext_set(vehicle, param_id, param_value, 10)  # C
HAR type
        receive_param_ext_ack(vehicle)

    elif choice == '6':
```

```
        """设置 float32_lxlyrxry 结构体 {float32 LX; float32 LY; floa
t32 RX; float32 RY;}"""
        param_id = input("Enter parameter ID: ").strip()
        LX = float(input("Enter LX: "))
        LY = float(input("Enter LY: "))
        RX = float(input("Enter RX: "))
        RY = float(input("Enter RY: "))
        param_value = pack_float32_lxlyrxry(LX, LY, RX, RY)
        print(f"  Packed: {param_value.hex()} ({len(param_value)} b
ytes)")
        send_param_ext_set(vehicle, param_id, param_value, 10)  # C
HAR type
        receive_param_ext_ack(vehicle)

    elif choice == '7':
        """设置 ering_mode 结构体 {uint8_t size; uint8_t eringMode[1
6]; uint32_t minValue; uint32_t maxValue;}"""
        param_id = input("Enter parameter ID: ").strip()
        size = int(input("Enter size (0-16): "))
        eringMode = []
        print("Enter 16 eringMode values (0-255), one per line:")
        for i in range(16):
            val = int(input(f"  eringMode[{i}]: "))
            eringMode.append(val)
        minValue = int(input("Enter minValue: "))
        maxValue = int(input("Enter maxValue: "))
        param_value = pack_ering_mode(size, eringMode, minValue, ma
xValue)
        print(f"  Packed: {param_value.hex()} ({len(param_value)} b
ytes)")
        send_param_ext_set(vehicle, param_id, param_value, 10)  # C
HAR type
        receive_param_ext_ack(vehicle)

    elif choice == '3':
        print("\n  Requesting all parameters...")
        send_param_ext_request_list(vehicle)
        all_params = receive_all_param_ext_values(vehicle)
        print(f"\n  Total parameters received: {len(all_params)}")

    else:
        print("Invalid choice. Select 1, 2, or 3.")
```

```
if __name__ == "__main__":
    main()
```

示例使用说明

运行后，成功建立连接

```
🚀 Starting MAVLink PARAM_EXT Tool (UDP Listening Mode)
------------------------------------------------------
This script listens for MAVLink on ALL interfaces:14550
Ensure your vehicle is configured to send to this port
------------------------------------------------------

Listening for MAVLink on: udpin:0.0.0.0:14550
Waiting for HEARTBEAT (timeout: 10s)...
✅ Heartbeat received from system 3, component 0
```

输入“1”选择设置参数

```
Available operations:
1. Read parameter value (by ID or index)
2. Set parameter value (raw/hex/string)
3. Request all parameters (PARAM_EXT_REQUEST_LIST)
4. Set struct: uint8_xy {x, y}
5. Set struct: float32_xy {x, y}
6. Set struct: float32_lxlyrxry {LX, LY, RX, RY}
7. Set struct: ering_mode {size, eringMode[16], minValue, maxValue}

Select operation [1-7]: 1
```

输入对应参数“VID_FMT_RNG”

```
Enter parameter ID: VID_FMT_RNG
```

成功返回完整消息

```
📥 Received PARAM_EXT_VALUE:
  🆔 ID: VID_FMT_RNG
fd950000030301420100120000a005649445f464d545f524e470000000000017afd8101000000407afd81ffff0000d8ffffff80ffffff0000000000000000780000000500000006821e8d1aaaa0000f00c005cffff0000600b005cffff0000507afd81ffff0000f0e1a6d1000000000000000
```

接收PARAM_SET
发送PARAM_VALUE

获取

mavlink消息

接收PARAM_REQUEST_READ
发送PARAM_VALUE

参数：

名称	枚举值	说明
DJI_CAMERA_MANAGER_VIDEO_STORAGE_FORMAT_MOV	0	MOV视频格式
DJI_CAMERA_MANAGER_VIDEO_STORAGE_FORMAT_MP4	1	MP4视频格式

示例

```
import time
import sys
from pymavlink import mavutil

def connect_to_vehicle():
    """建立 UDP 监听连接（默认监听所有接口的 14550 端口）"""
    connection_string = "udpin:0.0.0.0:14550"
    print(f"Listening for MAVLink on: {connection_string}")
    vehicle = mavutil.mavlink_connection(connection_string)
    print("Waiting for HEARTBEAT (timeout: 10s)...")
    try:
        vehicle.wait_heartbeat(timeout=10)
        print(f"  Heartbeat received from system {vehicle.target_system}, component {vehicle.target_component}")
        return vehicle
    except Exception as e:
        print(f"  Connection failed: {str(e)}")
        print("Ensure your vehicle is sending MAVLink messages to 14550 port")
        print("Example QGroundControl setup: Settings > General > MAVLink > Listening Port = 14550")
        sys.exit(1)

def format_param_id(param_id):
```

```
    """格式化16字节参数ID（符合MAVLink协议要求）"""
    if len(param_id) > 16:
        raise ValueError("param_id cannot exceed 16 characters")

    param_id_bytes = bytearray(16)
    for i in range(min(len(param_id), 16)):
        param_id_bytes[i] = ord(param_id[i])

    # 长度<16时添加NULL终止符
    if len(param_id) < 16:
        param_id_bytes[len(param_id)] = 0

    return param_id_bytes

def send_param_set(vehicle, param_id, param_value, param_type):
    """发送参数设置请求"""
    vehicle.mav.param_set_send(
        vehicle.target_system,
        vehicle.target_component,
        format_param_id(param_id),
        param_value,
        param_type
    )
    print(f"  Sent PARAM_SET for '{param_id}' = {param_value} (type: {param_type})")

def send_param_request_read(vehicle, param_id=None, param_index=-1):
    """发送参数读取请求"""
    param_id_bytes = format_param_id(param_id) if param_id else bytearray(16)

    vehicle.mav.param_request_read_send(
        vehicle.target_system,
        vehicle.target_component,
        param_id_bytes,
        param_index
    )

    target = f"index {param_index}" if param_index >= 0 else f"ID '{param_id}'"
    print(f"  Sent PARAM_REQUEST_READ for {target}")

def send_param_request_list(vehicle):
    """发送请求所有参数列表"""
```

```
    vehicle.mav.param_request_list_send(
        vehicle.target_system,
        vehicle.target_component
    )
    print(f"  Sent PARAM_REQUEST_LIST to system {vehicle.target_system}, component {vehicle.target_component}")

def receive_param_value(vehicle, timeout=5.0):
    """接收参数值响应"""
    print(f"  Waiting for PARAM_VALUE ({timeout}s timeout)...")
    msg = vehicle.recv_match(type='PARAM_VALUE', blocking=True, timeout=timeout)

    if not msg:
        print("  Timeout: No PARAM_VALUE received")
        return None

    # 解析参数ID
    param_id = msg.param_id

    # 解析参数值
    param_value = msg.param_value

    # 参数类型映射
    type_names = {
        0: "INT8", 1: "UINT8", 2: "INT16", 3: "UINT16", 4: "INT32",
        5: "UINT32", 6: "INT64", 7: "UINT64", 8: "FLOAT", 9: "DOUBLE"
    }
    param_type = type_names.get(msg.param_type, f"UNKNOWN({msg.param_type})")

    print("\n  Received PARAM_VALUE:")
    print(f"    ID: {param_id}")
    print(f"    Value: {param_value}")
    print(f"    Type: {param_type}")
    print(f"    Index/Total: {msg.param_index}/{msg.param_count}")

    return {
        'id': param_id,
        'value': param_value,
        'type': param_type,
        'index': msg.param_index,
        'count': msg.param_count
    }
```

```
def receive_all_param_values(vehicle, timeout=10.0):
    """接收所有参数值，直到收到 param_count 条消息后退出循环"""
    received_params = []
    expected_count = None
    print(f"  Waiting for all PARAM_VALUE messages...")

    while True:
        msg = vehicle.recv_match(type='PARAM_VALUE', blocking=True,
timeout=timeout)

        if not msg:
            print(f"  Timeout: Received {len(received_params)} para
meters (expected {expected_count if expected_count else 'unknow
n'})")
            break

        param_id = msg.param_id
        param_value = msg.param_value

        # 参数类型映射
        type_names = {
            0: "INT8", 1: "UINT8", 2: "INT16", 3: "UINT16", 4: "INT
32",
            5: "UINT32", 6: "INT64", 7: "UINT64", 8: "FLOAT", 9: "D
OUBLE"
        }
        param_type = type_names.get(msg.param_type, f"UNKNOWN({msg.
param_type})")

        # 首次收到消息时记录总参数数量
        if expected_count is None:
            expected_count = msg.param_count
            print(f"  Total parameters to receive: {expected_coun
t}")

        print(f"  [{msg.param_index}/{expected_count}] {param_id} =
{param_value} ({param_type})")

        received_params.append({
            'id': param_id,
            'value': param_value,
            'type': param_type,
            'index': msg.param_index,
            'count': msg.param_count
```

```
        })

        # 当已接收数量达到预期总数时退出循环
        if len(received_params) >= expected_count:
            print(f"\n  Successfully received all {expected_count}
parameters")
            break

    return received_params

def main():
    print("  Starting MAVLink PARAM_SET Tool (UDP Listening Mode)")
    print("------------------------------------------------------")
    print("This script listens for MAVLink on ALL interfaces:1455
0")
    print("Ensure your vehicle is configured to send to this port")
    print("------------------------------------------------------
\n")

    vehicle = connect_to_vehicle()

    # 操作选择
    print("\nAvailable operations:")
    print("1. Set parameter value")
    print("2. Read parameter value (by ID or index)")
    print("3. Request all parameters (PARAM_REQUEST_LIST)")
    choice = input("\nSelect operation [1-3]: ").strip()

    if choice == '1':
        param_id = input("Enter parameter ID to set: ").strip()
        param_value = float(input("Enter new value: ").strip())

        print("\nParameter types:")
        print("  8 = FLOAT (most common)")
        print("  9 = DOUBLE")
        print("  4 = INT32")
        print("  1 = UINT8")
        param_type = int(input("Enter type number [8]: ").strip() o
r 8)

        send_param_set(vehicle, param_id, param_value, param_type)
        receive_param_value(vehicle)

    elif choice == '2':
        # mode = input("Read by [id/index]: ").lower()
```

```
        mode = 'id'
        if mode == 'id':
            param_id = input("Enter parameter ID: ").strip()
            send_param_request_read(vehicle, param_id=param_id)
            receive_param_value(vehicle)
        elif mode == 'index':
            param_index = int(input("Enter parameter index: "))
            send_param_request_read(vehicle, param_index=param_inde
x)
            receive_param_value(vehicle)
        else:
            print("Invalid mode. Use 'id' or 'index'")

    elif choice == '3':
        print("\n  Requesting all parameters...")
        send_param_request_list(vehicle)
        all_params = receive_all_param_values(vehicle)
        print(f"\n  Total parameters received: {len(all_params)}")

    else:
        print("Invalid choice. Select 1, 2, or 3.")


if __name__ == "__main__":
    main()
```

示例使用说明

获取

运行后，成功建立连接

```
🚀 Starting MAVLink PARAM_SET Tool (UDP Listening Mode)
--------------------------------------------------
This script listens for MAVLink on ALL interfaces:14550
Ensure your vehicle is configured to send to this port
--------------------------------------------------

Listening for MAVLink on: udpin:0.0.0.0:14550
Waiting for HEARTBEAT (timeout: 10s)...
✅ Heartbeat received from system 3, component 0

Available operations:
1. Set parameter value
2. Read parameter value (by ID or index)
3. Request all parameters (PARAM_REQUEST_LIST)
```

输入“2”选择读取参数

```
Select operation [1-3]: 2
```

输入对应参数“VID_STORE_FMT”

```
Enter parameter ID: VID_STORE_FMT
```

成功返回该参数对应的数值

```
📥 Received PARAM_VALUE:
  🆔 ID: VID_STORE_FMT
  🔢 Value: 1.0
```

设置

运行后，成功建立连接

```
🚀 Starting MAVLink PARAM_SET Tool (UDP Listening Mode)
--------------------------------------------------
This script listens for MAVLink on ALL interfaces:14550
Ensure your vehicle is configured to send to this port
--------------------------------------------------

Listening for MAVLink on: udpin:0.0.0.0:14550
Waiting for HEARTBEAT (timeout: 10s)...
✅ Heartbeat received from system 3, component 0

Available operations:
1. Set parameter value
2. Read parameter value (by ID or index)
3. Request all parameters (PARAM_REQUEST_LIST)
```

输入“1”选择设置参数

```
Enter parameter ID to set: VID_STORE_FMT
```

输入对应参数“VID_STORE_FMT”

```
Enter parameter ID to set: VID_STORE_FMT
```

输入要设置的数值

```
Enter new value: 1
```

选择参数类型“1”

```
Parameter types:
  8 = FLOAT (most common)
  9 = DOUBLE
  4 = INT32
  1 = UINT8
Enter type number [8]: 1
```

发送成功返回设置值

```
📥 Received PARAM_VALUE:
  🆔 ID: VID_STORE_FMT
  🔢 Value: 1.0
```

获取照片比例范围

PHT_RATIO_RNG

mavlink消息：

接收PARAM_EXT_REQUEST_READ
发送PARAM_EXT_VALUE

参数：

字段	类型	说明
size	uint8_t	有效列表长度（最大16）
minValue	uint32_t	数值范围最小值
maxValue	uint32_t	数值范围最大值

union（同一时刻仅生效一个字段）

字段	类型	说明
photoStorageFormat	E_DjiCameraManagerPhotoStorageFormat[16]	照片存储格式枚
videoStorageFormat	E_DjiCameraManagerVideoStorageFormat[16]	视频存储格式枚
photoRatioFormat	E_DjiCameraManagerPhotoRatio[16]	照片比例枚举列
streamSource	E_DjiCameraManagerStreamSource[16]	视频流源枚举列
streamStorage	E_DjiCameraManagerStreamStorage[16]	视频存储介质枚
nightSceneMode	E_DjiCameraManagerNightSceneMode[16]	夜景模式枚举列
meteringMode	E_DjiCameraManagerMeteringMode[16]	测光模式枚举列

示例

```
import time
import sys
import struct
import ctypes
from pymavlink import mavutil

def connect_to_vehicle():
    """建立 UDP 监听连接（默认监听所有接口的 14550 端口）"""
```

```
    connection_string = "udpin:0.0.0.0:14550"
    print(f"Listening for MAVLink on: {connection_string}")
    vehicle = mavutil.mavlink_connection(connection_string)
    print("Waiting for HEARTBEAT (timeout: 10s)...")
    try:
        vehicle.wait_heartbeat(timeout=10)
        print(f"  Heartbeat received from system {vehicle.target_sy
stem}, component {vehicle.target_component}")
        return vehicle
    except Exception as e:
        print(f"  Connection failed: {str(e)}")
        print("Ensure your vehicle is sending MAVLink messages to 1
4550 port")
        print("Example QGroundControl setup: Settings > General > M
AVLink > Listening Port = 14550")
        sys.exit(1)

def format_param_id(param_id):
    """格式化16字节参数ID（符合MAVLink协议要求）"""
    if len(param_id) > 16:
        raise ValueError("param_id cannot exceed 16 characters")

    param_id_bytes = bytearray(16)
    for i in range(min(len(param_id), 16)):
        param_id_bytes[i] = ord(param_id[i])

    # 长度<16时添加NULL终止符
    if len(param_id) < 16:
        param_id_bytes[len(param_id)] = 0

    return param_id_bytes

# ============ 结构体打包函数 ============

def pack_uint8_xy(x, y):
    """打包 {uint8_t x; uint8_t y} 结构体 (2字节)"""
    return struct.pack('<BB', x, y)

def pack_float32_xy(x, y):
    """打包 {float32 x; float32 y} 结构体 (8字节)"""
    return struct.pack('<ff', x, y)

def pack_float32_lxlyrxry(LX, LY, RX, RY):
    """打包 {float32 LX; float32 LY; float32 RX; float32 RY;} 结构体
(16字节)"""
```

```
    return struct.pack('<ffff', LX, LY, RX, RY)

def pack_ering_mode(size, eringMode, minValue, maxValue):
    """打包 {uint8_t size; uint8_t eringMode[16]; uint32_t minValue; uint32_t maxValue;} 结构体 (25字节)

    Args:
        size: uint8_t, eringMode数组的有效长度
        eringMode: list of int (0-255), 16个uint8_t值
        minValue: uint32_t, 最小值
        maxValue: uint32_t, 最大值

    Returns:
        bytes: 打包后的二进制数据
    """
    # 确保eringMode是16字节
    eringMode_bytes = bytearray(16)
    for i in range(min(len(eringMode), 16)):
        eringMode_bytes[i] = eringMode[i] & 0xFF

    return struct.pack('<B16sII', size, eringMode_bytes, minValue, maxValue)

def format_param_value(param_value):
    """格式化128字节参数值（接受二进制输入）"""
    if isinstance(param_value, str):
        # 如果是十六进制字符串（如 "48656c6c6f"），先转换为字节
        if param_value.startswith('0x') or all(c in '0123456789abcdefABCDEF' for c in param_value.replace(' ', '')):
            hex_str = param_value.replace('0x', '').replace(' ', '')
            if len(hex_str) % 2 == 0:
                param_value = bytes.fromhex(hex_str)
            else:
                raise ValueError("Invalid hex string: must have even number of characters")
        else:
            # 普通字符串转字节
            param_value = param_value.encode('utf-8')

    if len(param_value) > 128:
        raise ValueError("param_value cannot exceed 128 bytes")

    param_value_bytes = bytearray(128)
    param_value_bytes[:len(param_value)] = param_value
```

```
    return param_value_bytes

def hex_dump(data, bytes_per_line=16):
    """格式化二进制数据为 hex + ASCII 展示（类 hexdump 风格）

    Returns:
        str: 格式化后的多行字符串
    """
    lines = []
    for i in range(0, len(data), bytes_per_line):
        chunk = data[i:i + bytes_per_line]
        hex_part = ' '.join(f'{b:02X}' for b in chunk)
        ascii_part = ''.join(chr(b) if 32 <= b < 127 else '.' for b
in chunk)
        lines.append(f"  {i:04X}  {hex_part:<{bytes_per_line*3}}
{ascii_part}")
    return '\n'.join(lines)

def send_param_ext_request_read(vehicle, param_id=None, param_index
=-1):
    """发送参数读取请求"""
    param_id_bytes = format_param_id(param_id) if param_id else byt
earray(16)

    vehicle.mav.param_ext_request_read_send(
        vehicle.target_system,
        vehicle.target_component,
        param_id_bytes,
        param_index
    )

    target = f"index {param_index}" if param_index >= 0 else f"ID
'{param_id}'"
    print(f"  Sent PARAM_EXT_REQUEST_READ for {target}")

def send_param_ext_request_list(vehicle):
    """发送请求所有参数列表"""
    vehicle.mav.param_ext_request_list_send(
        vehicle.target_system,
        vehicle.target_component
    )
    print(f"  Sent PARAM_EXT_REQUEST_LIST to system {vehicle.target
_system}, component {vehicle.target_component}")
```

```
def send_param_ext_set(vehicle, param_id, param_value, param_type):
    """发送参数设置请求"""
    vehicle.mav.param_ext_set_send(
        vehicle.target_system,
        vehicle.target_component,
        format_param_id(param_id),
        format_param_value(param_value),
        param_type
    )
    print(f"  Sent PARAM_EXT_SET for '{param_id}' = '{param_value}'
(type: {param_type})")

def receive_param_ext_value(vehicle, timeout=5.0):
    """接收参数值响应"""
    print(f"  Waiting for PARAM_EXT_VALUE ({timeout}s timeout)...")
    msg = vehicle.recv_match(type='PARAM_EXT_VALUE', blocking=True,
timeout=timeout)

    if not msg:
        print("  Timeout: No PARAM_EXT_VALUE received")
        return None

    # 解析参数ID
    param_id = msg.param_id

    # 解析参数值（直接获取128字节原始数据）
    param_value_raw = msg.param_value
    # byte_pointer = (ctypes.c_ubyte * 128).from_buffer_copy(msg.pa
ram_value)
    if isinstance(param_value_raw, (bytes, bytearray)):
        param_value_bytes = bytes(param_value_raw)
    # param_value_bytes = bytes(byte_pointer)
    elif isinstance(param_value_raw, str):
        # 替换无法编码的字符（如 \ufffd）为 0x00，还原为128字节
        param_value_bytes = param_value_raw.encode('latin1', errors
='replace')
    else:
        param_value_bytes = bytes(param_value_raw)

    # 参数类型映射
    type_names = {
        0: "INT8", 1: "UINT8", 2: "INT16", 3: "UINT16", 4: "INT32",
        5: "UINT32", 6: "INT64", 7: "UINT64", 8: "FLOAT", 9: "FLOA
T",
        10: "CHAR", 11: "CHAR", 12: "UINT8_ENUM", 13: "STRING"
```

```
    }
    param_type = type_names.get(msg.param_type, f"UNKNOWN({msg.para
m_type})")

    print("\n  Received PARAM_EXT_VALUE:")
    print(f"    ID: {param_id}")
    print(f"    Type: {param_type}  |  Data: {len(param_value_byte
s)} bytes")
    print(f"    Index/Total: {msg.param_index}/{msg.param_count}")
    print(hex_dump(param_value_bytes))

    return {
        'id': param_id,
        'value': param_value_bytes,
        'type': param_type,
        'index': msg.param_index,
        'count': msg.param_count
    }


def receive_all_param_ext_values(vehicle, timeout=10.0):
    """接收所有参数值，直到收到 param_count 条消息后退出循环"""
    received_params = []
    expected_count = None
    print(f"  Waiting for all PARAM_EXT_VALUE messages...")

    while True:
        msg = vehicle.recv_match(type='PARAM_EXT_VALUE', blocking=T
rue, timeout=timeout)

        if not msg:
            print(f"  Timeout: Received {len(received_params)} para
meters (expected {expected_count if expected_count else 'unknow
n'})")
            break

        param_id = msg.param_id
        param_value_raw = msg.param_value

        # 直接获取128字节原始数据
        if isinstance(param_value_raw, (bytes, bytearray)):
            param_value_bytes = bytes(param_value_raw)
        elif isinstance(param_value_raw, str):
            # 替换无法编码的字符（如 \ufffd）为 0x00
            param_value_bytes = param_value_raw.encode('latin1', er
```

```
rors='replace')
        else:
            param_value_bytes = bytes(param_value_raw)

        # 参数类型映射
        type_names = {
            0: "INT8", 1: "UINT8", 2: "INT16", 3: "UINT16", 4: "INT
32",
            5: "UINT32", 6: "INT64", 7: "UINT64", 8: "FLOAT", 9: "D
OUBLE",
            10: "CHAR", 11: "BOOL", 12: "UINT8_ENUM", 13: "STRING"
        }
        param_type = type_names.get(msg.param_type, f"UNKNOWN({msg.
param_type})")

        # 首次收到消息时记录总参数数量
        if expected_count is None:
            expected_count = msg.param_count
            print(f"  Total parameters to receive: {expected_coun
t}")

        print(f"  [{msg.param_index}/{expected_count}] {param_id}
[{param_type}][{len(param_value_bytes)}B]")
        print(hex_dump(param_value_bytes))

        received_params.append({
            'id': param_id,
            'value': param_value_bytes,
            'type': param_type,
            'index': msg.param_index,
            'count': msg.param_count
        })

        # 当已接收数量达到预期总数时退出循环
        if len(received_params) >= expected_count:
            print(f"\n  Successfully received all {expected_count}
parameters")
            break

    return received_params

def receive_param_ext_ack(vehicle, timeout=5.0):
    """接收参数设置确认"""
    print(f"  Waiting for PARAM_EXT_ACK ({timeout}s timeout)...")
    msg = vehicle.recv_match(type='PARAM_EXT_ACK', blocking=True, t
```

```
imeout=timeout)

    if not msg:
        print("  Timeout: No PARAM_EXT_ACK received")
        return None

    param_id = msg.param_id
    param_value = msg.param_value

    # 结果代码映射
    result_codes = {
        0: "  PARAM_ACK_ACCEPTED",
        1: "⚠ PARAM_ACK_VALUE_UNSUPPORTED",
        2: "  PARAM_ACK_FAILED",
        3: "  PARAM_ACK_IN_PROGRESS"
    }
    result = result_codes.get(msg.param_result, f"  UNKNOWN({msg.pa
ram_result})")

    print("\n  Received PARAM_EXT_ACK:")
    print(f"    ID: {param_id}")
    print(f"    Value: {param_value}")
    print(f"    Result: {result}")

    return {
        'id': param_id,
        'value': param_value,
        'result': result,
        'result_code': msg.param_result
    }

def main():
    print("  Starting MAVLink PARAM_EXT Tool (UDP Listening Mode)")
    print("-----------------------------------------------------")
    print("This script listens for MAVLink on ALL interfaces:1455
0")
    print("Ensure your vehicle is configured to send to this port")
    print("-----------------------------------------------------
\n")

    vehicle = connect_to_vehicle()

    # 操作选择
    print("\nAvailable operations:")
    print("1. Read parameter value (by ID or index)")
```

```
    print("2. Set parameter value (raw/hex/string)")
    print("3. Request all parameters (PARAM_EXT_REQUEST_LIST)")
    print("4. Set struct: uint8_xy {x, y}")
    print("5. Set struct: float32_xy {x, y}")
    print("6. Set struct: float32_lxlyrxry {LX, LY, RX, RY}")
    print("7. Set struct: ering_mode {size, eringMode[16], minValu
e, maxValue}")
    choice = input("\nSelect operation [1-7]: ").strip()

    if choice == '1':
        # mode = input("Read by [id/index]: ").lower()
        mode = 'id'
        if mode == 'id':
            param_id = input("Enter parameter ID: ").strip()
            send_param_ext_request_read(vehicle, param_id=param_id)
            receive_param_ext_value(vehicle)
        elif mode == 'index':
            param_index = int(input("Enter parameter index: "))
            send_param_ext_request_read(vehicle, param_index=param_
index)
            receive_param_ext_value(vehicle)
        else:
            print("Invalid mode. Use 'id' or 'index'")

    elif choice == '2':
        param_id = input("Enter parameter ID to set: ").strip()

        print("\nEnter parameter value (supports):")
        print("  - Plain string (e.g., hello)")
        print("  - Hex string (e.g., 48656c6c6f or 0x48656c6c6f)")
        print("  - Raw bytes (e.g., b'\\x48\\x65\\x6c\\x6c\\x6f')")
        param_value_input = input("Enter value: ").strip()

        # 解析输入值
        if param_value_input.startswith("b'") or param_value_input.
startswith('b"'):
            # Python字节字面量格式
            param_value = eval(param_value_input)
        elif param_value_input.startswith('0x') or all(c in '012345
6789abcdefABCDEF ' for c in param_value_input):
            # 十六进制格式
            hex_str = param_value_input.replace('0x', '').replace('
', '')
            try:
                param_value = bytes.fromhex(hex_str)
```

```
            except ValueError:
                param_value = param_value_input.encode('utf-8')
        else:
            # 默认作为字符串处理
            param_value = param_value_input.encode('utf-8')

        print("\nParameter types:")
        print("  8 = FLOAT (most common)")
        print("  13 = STRING")
        print("  10 = CHAR (raw bytes)")
        print("  Other types require specific binary formats")
        param_type = int(input("Enter type number [8]: ") or 8)

        send_param_ext_set(vehicle, param_id, param_value, param_ty
pe)
        receive_param_ext_ack(vehicle)

    elif choice == '4':
        """设置 uint8_xy 结构体 {uint8_t x; uint8_t y}"""
        param_id = input("Enter parameter ID: ").strip()
        x = int(input("Enter x (0-255): "))
        y = int(input("Enter y (0-255): "))
        param_value = pack_uint8_xy(x, y)
        print(f"  Packed: {param_value.hex()} ({len(param_value)} b
ytes)")
        send_param_ext_set(vehicle, param_id, param_value, 10)  # C
HAR type
        receive_param_ext_ack(vehicle)

    elif choice == '5':
        """设置 float32_xy 结构体 {float32 x; float32 y}"""
        param_id = input("Enter parameter ID: ").strip()
        x = float(input("Enter x: "))
        y = float(input("Enter y: "))
        param_value = pack_float32_xy(x, y)
        print(f"  Packed: {param_value.hex()} ({len(param_value)} b
ytes)")
        send_param_ext_set(vehicle, param_id, param_value, 10)  # C
HAR type
        receive_param_ext_ack(vehicle)

    elif choice == '6':
        """设置 float32_lxlyrxry 结构体 {float32 LX; float32 LY; floa
t32 RX; float32 RY;}"""
        param_id = input("Enter parameter ID: ").strip()
```

```
        LX = float(input("Enter LX: "))
        LY = float(input("Enter LY: "))
        RX = float(input("Enter RX: "))
        RY = float(input("Enter RY: "))
        param_value = pack_float32_lxlyrxry(LX, LY, RX, RY)
        print(f"  Packed: {param_value.hex()} ({len(param_value)} bytes)")
        send_param_ext_set(vehicle, param_id, param_value, 10)  # CHAR type
        receive_param_ext_ack(vehicle)

    elif choice == '7':
        """设置 ering_mode 结构体 {uint8_t size; uint8_t eringMode[16]; uint32_t minValue; uint32_t maxValue;}"""
        param_id = input("Enter parameter ID: ").strip()
        size = int(input("Enter size (0-16): "))
        eringMode = []
        print("Enter 16 eringMode values (0-255), one per line:")
        for i in range(16):
            val = int(input(f"  eringMode[{i}]: "))
            eringMode.append(val)
        minValue = int(input("Enter minValue: "))
        maxValue = int(input("Enter maxValue: "))
        param_value = pack_ering_mode(size, eringMode, minValue, maxValue)
        print(f"  Packed: {param_value.hex()} ({len(param_value)} bytes)")
        send_param_ext_set(vehicle, param_id, param_value, 10)  # CHAR type
        receive_param_ext_ack(vehicle)

    elif choice == '3':
        print("\n  Requesting all parameters...")
        send_param_ext_request_list(vehicle)
        all_params = receive_all_param_ext_values(vehicle)
        print(f"\n  Total parameters received: {len(all_params)}")

    else:
        print("Invalid choice. Select 1, 2, or 3.")


if __name__ == "__main__":
    main()
```

示例使用说明

运行后，成功建立连接

```
🚀 Starting MAVLink PARAM_EXT Tool (UDP Listening Mode)
------------------------------------------------------------
This script listens for MAVLink on ALL interfaces:14550
Ensure your vehicle is configured to send to this port
------------------------------------------------------------

Listening for MAVLink on: udpin:0.0.0.0:14550
Waiting for HEARTBEAT (timeout: 10s)...
✅ Heartbeat received from system 3, component 0
```

输入“1”选择设置参数

```
Available operations:
1. Read parameter value (by ID or index)
2. Set parameter value (raw/hex/string)
3. Request all parameters (PARAM_EXT_REQUEST_LIST)
4. Set struct: uint8_xy {x, y}
5. Set struct: float32_xy {x, y}
6. Set struct: float32_lxlyrxry {LX, LY, RX, RY}
7. Set struct: ering_mode {size, eringMode[16], minValue, maxValue}

Select operation [1-7]: 1
```

输入对应参数“PHT_RATIO_RNG”

```
Enter parameter ID: PHT_RATIO_RNG
```

成功返回完整消息

```
📥 Received PARAM_EXT_VALUE:
  🆔 ID: PHT_RATIO_RNG
fd950000000301420100120000b005048545f524154494f5f524e47000000019ad28000000000409ad280ffff0000d8ffffff80ffffff0000000000000000780000000500000006821decdaaaa0000f00c004cffff0000800e004cffff0000509ad280ffff0000f0e19ccd0000000000000000
```

接收PARAM_REQUEST_READ
发送PARAM_VALUE

参数：

名称	枚举值	说明
DJI_CAMERA_MANAGER_PHOTO_RATIO_4X3	0	4:3 照片比例
DJI_CAMERA_MANAGER_PHOTO_RATIO_16X9	1	16:9 照片比例
DJI_CAMERA_MANAGER_PHOTO_RATIO_3X2	2	3:2 照片比例
DJI_CAMERA_MANAGER_PHOTO_RATIO_1X1	3	1:1 正方形比例
DJI_CAMERA_MANAGER_PHOTO_RATIO_18X3	4	18:3 超宽比例
DJI_CAMERA_MANAGER_PHOTO_RATIO_5X4	5	5:4 照片比例

示例

```
import time
import sys
from pymavlink import mavutil

def connect_to_vehicle():
    """建立 UDP 监听连接（默认监听所有接口的 14550 端口）"""
    connection_string = "udpin:0.0.0.0:14550"
    print(f"Listening for MAVLink on: {connection_string}")
    vehicle = mavutil.mavlink_connection(connection_string)
    print("Waiting for HEARTBEAT (timeout: 10s)...")
    try:
        vehicle.wait_heartbeat(timeout=10)
        print(f"  Heartbeat received from system {vehicle.target_sy
stem}, component {vehicle.target_component}")
        return vehicle
    except Exception as e:
        print(f"  Connection failed: {str(e)}")
        print("Ensure your vehicle is sending MAVLink messages to 1
4550 port")
        print("Example QGroundControl setup: Settings > General > M
AVLink > Listening Port = 14550")
        sys.exit(1)

def format_param_id(param_id):
```

```
    """格式化16字节参数ID（符合MAVLink协议要求）"""
    if len(param_id) > 16:
        raise ValueError("param_id cannot exceed 16 characters")

    param_id_bytes = bytearray(16)
    for i in range(min(len(param_id), 16)):
        param_id_bytes[i] = ord(param_id[i])

    # 长度<16时添加NULL终止符
    if len(param_id) < 16:
        param_id_bytes[len(param_id)] = 0

    return param_id_bytes

def send_param_set(vehicle, param_id, param_value, param_type):
    """发送参数设置请求"""
    vehicle.mav.param_set_send(
        vehicle.target_system,
        vehicle.target_component,
        format_param_id(param_id),
        param_value,
        param_type
    )
    print(f"  Sent PARAM_SET for '{param_id}' = {param_value} (type: {param_type})")

def send_param_request_read(vehicle, param_id=None, param_index=-1):
    """发送参数读取请求"""
    param_id_bytes = format_param_id(param_id) if param_id else bytearray(16)

    vehicle.mav.param_request_read_send(
        vehicle.target_system,
        vehicle.target_component,
        param_id_bytes,
        param_index
    )

    target = f"index {param_index}" if param_index >= 0 else f"ID '{param_id}'"
    print(f"  Sent PARAM_REQUEST_READ for {target}")

def send_param_request_list(vehicle):
    """发送请求所有参数列表"""
```

```
    vehicle.mav.param_request_list_send(
        vehicle.target_system,
        vehicle.target_component
    )
    print(f"  Sent PARAM_REQUEST_LIST to system {vehicle.target_system}, component {vehicle.target_component}")

def receive_param_value(vehicle, timeout=5.0):
    """接收参数值响应"""
    print(f"  Waiting for PARAM_VALUE ({timeout}s timeout)...")
    msg = vehicle.recv_match(type='PARAM_VALUE', blocking=True, timeout=timeout)

    if not msg:
        print("  Timeout: No PARAM_VALUE received")
        return None

    # 解析参数ID
    param_id = msg.param_id

    # 解析参数值
    param_value = msg.param_value

    # 参数类型映射
    type_names = {
        0: "INT8", 1: "UINT8", 2: "INT16", 3: "UINT16", 4: "INT32",
        5: "UINT32", 6: "INT64", 7: "UINT64", 8: "FLOAT", 9: "DOUBLE"
    }
    param_type = type_names.get(msg.param_type, f"UNKNOWN({msg.param_type})")

    print("\n  Received PARAM_VALUE:")
    print(f"    ID: {param_id}")
    print(f"    Value: {param_value}")
    print(f"    Type: {param_type}")
    print(f"    Index/Total: {msg.param_index}/{msg.param_count}")

    return {
        'id': param_id,
        'value': param_value,
        'type': param_type,
        'index': msg.param_index,
        'count': msg.param_count
    }
```

```
def receive_all_param_values(vehicle, timeout=10.0):
    """接收所有参数值，直到收到 param_count 条消息后退出循环"""
    received_params = []
    expected_count = None
    print(f"  Waiting for all PARAM_VALUE messages...")

    while True:
        msg = vehicle.recv_match(type='PARAM_VALUE', blocking=True,
timeout=timeout)

        if not msg:
            print(f"  Timeout: Received {len(received_params)} para
meters (expected {expected_count if expected_count else 'unknow
n'})")
            break

        param_id = msg.param_id
        param_value = msg.param_value

        # 参数类型映射
        type_names = {
            0: "INT8", 1: "UINT8", 2: "INT16", 3: "UINT16", 4: "INT
32",
            5: "UINT32", 6: "INT64", 7: "UINT64", 8: "FLOAT", 9: "D
OUBLE"
        }
        param_type = type_names.get(msg.param_type, f"UNKNOWN({msg.
param_type})")

        # 首次收到消息时记录总参数数量
        if expected_count is None:
            expected_count = msg.param_count
            print(f"  Total parameters to receive: {expected_coun
t}")

        print(f"  [{msg.param_index}/{expected_count}] {param_id} =
{param_value} ({param_type})")

        received_params.append({
            'id': param_id,
            'value': param_value,
            'type': param_type,
            'index': msg.param_index,
            'count': msg.param_count
```

```
        })

        # 当已接收数量达到预期总数时退出循环
        if len(received_params) >= expected_count:
            print(f"\n  Successfully received all {expected_count}
parameters")
            break

    return received_params

def main():
    print("  Starting MAVLink PARAM_SET Tool (UDP Listening Mode)")
    print("------------------------------------------------------")
    print("This script listens for MAVLink on ALL interfaces:1455
0")
    print("Ensure your vehicle is configured to send to this port")
    print("------------------------------------------------------
\n")

    vehicle = connect_to_vehicle()

    # 操作选择
    print("\nAvailable operations:")
    print("1. Set parameter value")
    print("2. Read parameter value (by ID or index)")
    print("3. Request all parameters (PARAM_REQUEST_LIST)")
    choice = input("\nSelect operation [1-3]: ").strip()

    if choice == '1':
        param_id = input("Enter parameter ID to set: ").strip()
        param_value = float(input("Enter new value: ").strip())

        print("\nParameter types:")
        print("  8 = FLOAT (most common)")
        print("  9 = DOUBLE")
        print("  4 = INT32")
        print("  1 = UINT8")
        param_type = int(input("Enter type number [8]: ").strip() o
r 8)

        send_param_set(vehicle, param_id, param_value, param_type)
        receive_param_value(vehicle)

    elif choice == '2':
        # mode = input("Read by [id/index]: ").lower()
```

```
        mode = 'id'
        if mode == 'id':
            param_id = input("Enter parameter ID: ").strip()
            send_param_request_read(vehicle, param_id=param_id)
            receive_param_value(vehicle)
        elif mode == 'index':
            param_index = int(input("Enter parameter index: "))
            send_param_request_read(vehicle, param_index=param_inde
x)
            receive_param_value(vehicle)
        else:
            print("Invalid mode. Use 'id' or 'index'")

    elif choice == '3':
        print("\n  Requesting all parameters...")
        send_param_request_list(vehicle)
        all_params = receive_all_param_values(vehicle)
        print(f"\n  Total parameters received: {len(all_params)}")

    else:
        print("Invalid choice. Select 1, 2, or 3.")


if __name__ == "__main__":
    main()
```

示例使用说明

获取

运行后，成功建立连接

```
🚀 Starting MAVLink PARAM_SET Tool (UDP Listening Mode)
--------------------------------------------------
This script listens for MAVLink on ALL interfaces:14550
Ensure your vehicle is configured to send to this port
--------------------------------------------------

Listening for MAVLink on: udpin:0.0.0.0:14550
Waiting for HEARTBEAT (timeout: 10s)...
✅ Heartbeat received from system 3, component 0

Available operations:
1. Set parameter value
2. Read parameter value (by ID or index)
3. Request all parameters (PARAM_REQUEST_LIST)
```

输入“2”选择读取参数

```
Select operation [1-3]: 2
```

输入对应参数“PHOTO_RATIO”

```
Enter parameter ID: PHOTO_RATIO
```

成功返回该参数对应的数值

```
📥 Received PARAM_VALUE:
  🆔 ID: PHOTO_RATIO
  🔢 Value: 0.0
```

设置

运行后，成功建立连接

```
🚀 Starting MAVLink PARAM_SET Tool (UDP Listening Mode)
------------------------------------------------------
This script listens for MAVLink on ALL interfaces:14550
Ensure your vehicle is configured to send to this port
------------------------------------------------------

Listening for MAVLink on: udpin:0.0.0.0:14550
Waiting for HEARTBEAT (timeout: 10s)...
✅ Heartbeat received from system 3, component 0

Available operations:
1. Set parameter value
2. Read parameter value (by ID or index)
3. Request all parameters (PARAM_REQUEST_LIST)
```

输入“1”选择设置参数

```
Available operations:
1. Set parameter value
2. Read parameter value (by ID or index)
3. Request all parameters (PARAM_REQUEST_LIST)

Select operation [1-3]: 1
```

输入对应参数“PHOTO_RATIO”

```
Enter parameter ID to set: PHOTO_RATIO
```

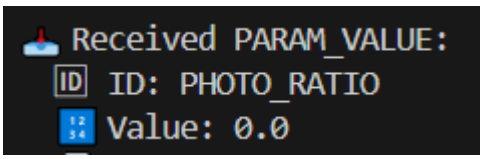
输入要设置的数值

```
Enter new value: 0
```

选择参数类型“1”

```
Parameter types:
  8 = FLOAT (most common)
  9 = DOUBLE
  4 = INT32
  1 = UINT8
Enter type number [8]: 1
```

成功返回设置值



获取夜景模式范围

NGT_SCN_MD_RNG

mavlink消息：

接收PARAM_EXT_REQUEST_READ
发送PARAM_EXT_VALUE

参数：

字段	类型	说明
size	uint8_t	有效列表长度（最大16）
minValue	uint32_t	数值范围最小值
maxValue	uint32_t	数值范围最大值

union（同一时刻仅生效一个字段）

字段	类型	说明
photoStorageFormat	E_DjiCameraManagerPhotoStorageFormat[16]	照片存储格式枚
videoStorageFormat	E_DjiCameraManagerVideoStorageFormat[16]	视频存储格式枚
photoRatioFormat	E_DjiCameraManagerPhotoRatio[16]	照片比例枚举列
streamSource	E_DjiCameraManagerStreamSource[16]	视频流源枚举列
streamStorage	E_DjiCameraManagerStreamStorage[16]	视频存储介质枚
nightSceneMode	E_DjiCameraManagerNightSceneMode[16]	夜景模式枚举列
meteringMode	E_DjiCameraManagerMeteringMode[16]	测光模式枚举列

示例

```
import time
import sys
import struct
import ctypes
from pymavlink import mavutil

def connect_to_vehicle():
    """建立 UDP 监听连接（默认监听所有接口的 14550 端口）"""
    connection_string = "udpin:0.0.0.0:14550"
    print(f"Listening for MAVLink on: {connection_string}")
    vehicle = mavutil.mavlink_connection(connection_string)
    print("Waiting for HEARTBEAT (timeout: 10s)...")
    try:
        vehicle.wait_heartbeat(timeout=10)
        print(f"  Heartbeat received from system {vehicle.target_system}, component {vehicle.target_component}")
        return vehicle
    except Exception as e:
        print(f"  Connection failed: {str(e)}")
        print("Ensure your vehicle is sending MAVLink messages to 14550 port")
        print("Example QGroundControl setup: Settings > General > MAVLink > Listening Port = 14550")
        sys.exit(1)

def format_param_id(param_id):
    """格式化16字节参数ID（符合MAVLink协议要求）"""
    if len(param_id) > 16:
        raise ValueError("param_id cannot exceed 16 characters")

    param_id_bytes = bytearray(16)
    for i in range(min(len(param_id), 16)):
        param_id_bytes[i] = ord(param_id[i])

    # 长度<16时添加NULL终止符
    if len(param_id) < 16:
        param_id_bytes[len(param_id)] = 0

    return param_id_bytes

# ============ 结构体打包函数 ============

def pack_uint8_xy(x, y):
    """打包 {uint8_t x; uint8_t y} 结构体 (2字节)"""
    return struct.pack('<BB', x, y)
```

```
def pack_float32_xy(x, y):
    """打包 {float32 x; float32 y} 结构体 (8字节)"""
    return struct.pack('<ff', x, y)

def pack_float32_lxlyrxry(LX, LY, RX, RY):
    """打包 {float32 LX; float32 LY; float32 RX; float32 RY;} 结构体 (16字节)"""
    return struct.pack('<ffff', LX, LY, RX, RY)

def pack_ering_mode(size, eringMode, minValue, maxValue):
    """打包 {uint8_t size; uint8_t eringMode[16]; uint32_t minValue; uint32_t maxValue;} 结构体 (25字节)

    Args:
        size: uint8_t, eringMode数组的有效长度
        eringMode: list of int (0-255), 16个uint8_t值
        minValue: uint32_t, 最小值
        maxValue: uint32_t, 最大值

    Returns:
        bytes: 打包后的二进制数据
    """
    # 确保eringMode是16字节
    eringMode_bytes = bytearray(16)
    for i in range(min(len(eringMode), 16)):
        eringMode_bytes[i] = eringMode[i] & 0xFF

    return struct.pack('<B16sII', size, eringMode_bytes, minValue, maxValue)

def format_param_value(param_value):
    """格式化128字节参数值（接受二进制输入）"""
    if isinstance(param_value, str):
        # 如果是十六进制字符串（如 "48656c6c6f"），先转换为字节
        if param_value.startswith('0x') or all(c in '0123456789abcdefABCDEF' for c in param_value.replace(' ', '')):
            hex_str = param_value.replace('0x', '').replace(' ', '')
            if len(hex_str) % 2 == 0:
                param_value = bytes.fromhex(hex_str)
            else:
                raise ValueError("Invalid hex string: must have even number of characters")
        else:
```

```
            # 普通字符串转字节
            param_value = param_value.encode('utf-8')

    if len(param_value) > 128:
        raise ValueError("param_value cannot exceed 128 bytes")

    param_value_bytes = bytearray(128)
    param_value_bytes[:len(param_value)] = param_value

    return param_value_bytes

def hex_dump(data, bytes_per_line=16):
    """格式化二进制数据为 hex + ASCII 展示（类 hexdump 风格）

    Returns:
        str: 格式化后的多行字符串
    """
    lines = []
    for i in range(0, len(data), bytes_per_line):
        chunk = data[i:i + bytes_per_line]
        hex_part = ' '.join(f'{b:02X}' for b in chunk)
        ascii_part = ''.join(chr(b) if 32 <= b < 127 else '.' for b
in chunk)
        lines.append(f"  {i:04X}  {hex_part:<{bytes_per_line*3}}
{ascii_part}")
    return '\n'.join(lines)

def send_param_ext_request_read(vehicle, param_id=None, param_index
=-1):
    """发送参数读取请求"""
    param_id_bytes = format_param_id(param_id) if param_id else byt
earray(16)

    vehicle.mav.param_ext_request_read_send(
        vehicle.target_system,
        vehicle.target_component,
        param_id_bytes,
        param_index
    )

    target = f"index {param_index}" if param_index >= 0 else f"ID
'{param_id}'"
    print(f"  Sent PARAM_EXT_REQUEST_READ for {target}")

def send_param_ext_request_list(vehicle):
```

```
    """发送请求所有参数列表"""
    vehicle.mav.param_ext_request_list_send(
        vehicle.target_system,
        vehicle.target_component
    )
    print(f"  Sent PARAM_EXT_REQUEST_LIST to system {vehicle.target
_system}, component {vehicle.target_component}")

def send_param_ext_set(vehicle, param_id, param_value, param_type):
    """发送参数设置请求"""
    vehicle.mav.param_ext_set_send(
        vehicle.target_system,
        vehicle.target_component,
        format_param_id(param_id),
        format_param_value(param_value),
        param_type
    )
    print(f"  Sent PARAM_EXT_SET for '{param_id}' = '{param_value}'
(type: {param_type})")

def receive_param_ext_value(vehicle, timeout=5.0):
    """接收参数值响应"""
    print(f"  Waiting for PARAM_EXT_VALUE ({timeout}s timeout)...")
    msg = vehicle.recv_match(type='PARAM_EXT_VALUE', blocking=True,
timeout=timeout)

    if not msg:
        print("  Timeout: No PARAM_EXT_VALUE received")
        return None

    # 解析参数ID
    param_id = msg.param_id

    # 解析参数值（直接获取128字节原始数据）
    param_value_raw = msg.param_value
    # byte_pointer = (ctypes.c_ubyte * 128).from_buffer_copy(msg.pa
ram_value)
    if isinstance(param_value_raw, (bytes, bytearray)):
        param_value_bytes = bytes(param_value_raw)
    # param_value_bytes = bytes(byte_pointer)
    elif isinstance(param_value_raw, str):
        # 替换无法编码的字符（如 \ufffd）为 0x00，还原为128字节
        param_value_bytes = param_value_raw.encode('latin1', errors
='replace')
    else:
```

```
        param_value_bytes = bytes(param_value_raw)

    # 参数类型映射
    type_names = {
        0: "INT8", 1: "UINT8", 2: "INT16", 3: "UINT16", 4: "INT32",
        5: "UINT32", 6: "INT64", 7: "UINT64", 8: "FLOAT", 9: "FLOA
T",
        10: "CHAR", 11: "CHAR", 12: "UINT8_ENUM", 13: "STRING"
    }
    param_type = type_names.get(msg.param_type, f"UNKNOWN({msg.para
m_type})")

    print("\n  Received PARAM_EXT_VALUE:")
    print(f"    ID: {param_id}")
    print(f"    Type: {param_type}  |  Data: {len(param_value_byte
s)} bytes")
    print(f"    Index/Total: {msg.param_index}/{msg.param_count}")
    print(hex_dump(param_value_bytes))

    return {
        'id': param_id,
        'value': param_value_bytes,
        'type': param_type,
        'index': msg.param_index,
        'count': msg.param_count
    }


def receive_all_param_ext_values(vehicle, timeout=10.0):
    """接收所有参数值，直到收到 param_count 条消息后退出循环"""
    received_params = []
    expected_count = None
    print(f"  Waiting for all PARAM_EXT_VALUE messages...")

    while True:
        msg = vehicle.recv_match(type='PARAM_EXT_VALUE', blocking=T
rue, timeout=timeout)

        if not msg:
            print(f"  Timeout: Received {len(received_params)} para
meters (expected {expected_count if expected_count else 'unknow
n'})")
            break

        param_id = msg.param_id
```

```
        param_value_raw = msg.param_value

        # 直接获取128字节原始数据
        if isinstance(param_value_raw, (bytes, bytearray)):
            param_value_bytes = bytes(param_value_raw)
        elif isinstance(param_value_raw, str):
            # 替换无法编码的字符（如 \ufffd）为 0x00
            param_value_bytes = param_value_raw.encode('latin1', er
rors='replace')
        else:
            param_value_bytes = bytes(param_value_raw)

        # 参数类型映射
        type_names = {
            0: "INT8", 1: "UINT8", 2: "INT16", 3: "UINT16", 4: "INT
32",
            5: "UINT32", 6: "INT64", 7: "UINT64", 8: "FLOAT", 9: "D
OUBLE",
            10: "CHAR", 11: "BOOL", 12: "UINT8_ENUM", 13: "STRING"
        }
        param_type = type_names.get(msg.param_type, f"UNKNOWN({msg.
param_type})")

        # 首次收到消息时记录总参数数量
        if expected_count is None:
            expected_count = msg.param_count
            print(f"  Total parameters to receive: {expected_coun
t}")

        print(f"  [{msg.param_index}/{expected_count}] {param_id}
[{param_type}][{len(param_value_bytes)}B]")
        print(hex_dump(param_value_bytes))

        received_params.append({
            'id': param_id,
            'value': param_value_bytes,
            'type': param_type,
            'index': msg.param_index,
            'count': msg.param_count
        })

        # 当已接收数量达到预期总数时退出循环
        if len(received_params) >= expected_count:
            print(f"\n  Successfully received all {expected_count}
parameters")
```

```
            break

    return received_params

def receive_param_ext_ack(vehicle, timeout=5.0):
    """接收参数设置确认"""
    print(f"  Waiting for PARAM_EXT_ACK ({timeout}s timeout)...")
    msg = vehicle.recv_match(type='PARAM_EXT_ACK', blocking=True, t
imeout=timeout)

    if not msg:
        print("  Timeout: No PARAM_EXT_ACK received")
        return None

    param_id = msg.param_id
    param_value = msg.param_value

    # 结果代码映射
    result_codes = {
        0: "  PARAM_ACK_ACCEPTED",
        1: "⚠ PARAM_ACK_VALUE_UNSUPPORTED",
        2: "  PARAM_ACK_FAILED",
        3: "  PARAM_ACK_IN_PROGRESS"
    }
    result = result_codes.get(msg.param_result, f"  UNKNOWN({msg.pa
ram_result})")

    print("\n  Received PARAM_EXT_ACK:")
    print(f"    ID: {param_id}")
    print(f"    Value: {param_value}")
    print(f"    Result: {result}")

    return {
        'id': param_id,
        'value': param_value,
        'result': result,
        'result_code': msg.param_result
    }

def main():
    print("  Starting MAVLink PARAM_EXT Tool (UDP Listening Mode)")
    print("-----------------------------------------------------")
    print("This script listens for MAVLink on ALL interfaces:1455
0")
    print("Ensure your vehicle is configured to send to this port")
```

```
    print("-------------------------------------------------------
\n")

    vehicle = connect_to_vehicle()

    # 操作选择
    print("\nAvailable operations:")
    print("1. Read parameter value (by ID or index)")
    print("2. Set parameter value (raw/hex/string)")
    print("3. Request all parameters (PARAM_EXT_REQUEST_LIST)")
    print("4. Set struct: uint8_xy {x, y}")
    print("5. Set struct: float32_xy {x, y}")
    print("6. Set struct: float32_lxlyrxry {LX, LY, RX, RY}")
    print("7. Set struct: ering_mode {size, eringMode[16], minValu
e, maxValue}")
    choice = input("\nSelect operation [1-7]: ").strip()

    if choice == '1':
        # mode = input("Read by [id/index]: ").lower()
        mode = 'id'
        if mode == 'id':
            param_id = input("Enter parameter ID: ").strip()
            send_param_ext_request_read(vehicle, param_id=param_id)
            receive_param_ext_value(vehicle)
        elif mode == 'index':
            param_index = int(input("Enter parameter index: "))
            send_param_ext_request_read(vehicle, param_index=param_
index)
            receive_param_ext_value(vehicle)
        else:
            print("Invalid mode. Use 'id' or 'index'")

    elif choice == '2':
        param_id = input("Enter parameter ID to set: ").strip()

        print("\nEnter parameter value (supports):")
        print("  - Plain string (e.g., hello)")
        print("  - Hex string (e.g., 48656c6c6f or 0x48656c6c6f)")
        print("  - Raw bytes (e.g., b'\\x48\\x65\\x6c\\x6c\\x6f')")
        param_value_input = input("Enter value: ").strip()

        # 解析输入值
        if param_value_input.startswith("b'") or param_value_input.
startswith('b"'):
            # Python字节字面量格式
```

```
            param_value = eval(param_value_input)
        elif param_value_input.startswith('0x') or all(c in '012345
6789abcdefABCDEF ' for c in param_value_input):
            # 十六进制格式
            hex_str = param_value_input.replace('0x', '').replace('
', '')
            try:
                param_value = bytes.fromhex(hex_str)
            except ValueError:
                param_value = param_value_input.encode('utf-8')
        else:
            # 默认作为字符串处理
            param_value = param_value_input.encode('utf-8')

        print("\nParameter types:")
        print("  8 = FLOAT (most common)")
        print("  13 = STRING")
        print("  10 = CHAR (raw bytes)")
        print("  Other types require specific binary formats")
        param_type = int(input("Enter type number [8]: ") or 8)

        send_param_ext_set(vehicle, param_id, param_value, param_ty
pe)
        receive_param_ext_ack(vehicle)

    elif choice == '4':
        """设置 uint8_xy 结构体 {uint8_t x; uint8_t y}"""
        param_id = input("Enter parameter ID: ").strip()
        x = int(input("Enter x (0-255): "))
        y = int(input("Enter y (0-255): "))
        param_value = pack_uint8_xy(x, y)
        print(f"  Packed: {param_value.hex()} ({len(param_value)} b
ytes)")
        send_param_ext_set(vehicle, param_id, param_value, 10)  # C
HAR type
        receive_param_ext_ack(vehicle)

    elif choice == '5':
        """设置 float32_xy 结构体 {float32 x; float32 y}"""
        param_id = input("Enter parameter ID: ").strip()
        x = float(input("Enter x: "))
        y = float(input("Enter y: "))
        param_value = pack_float32_xy(x, y)
        print(f"  Packed: {param_value.hex()} ({len(param_value)} b
ytes)")
```

```
        send_param_ext_set(vehicle, param_id, param_value, 10)  # C
HAR type
        receive_param_ext_ack(vehicle)

    elif choice == '6':
        """设置 float32_lxlyrxry 结构体 {float32 LX; float32 LY; floa
t32 RX; float32 RY;}"""
        param_id = input("Enter parameter ID: ").strip()
        LX = float(input("Enter LX: "))
        LY = float(input("Enter LY: "))
        RX = float(input("Enter RX: "))
        RY = float(input("Enter RY: "))
        param_value = pack_float32_lxlyrxry(LX, LY, RX, RY)
        print(f"  Packed: {param_value.hex()} ({len(param_value)} b
ytes)")
        send_param_ext_set(vehicle, param_id, param_value, 10)  # C
HAR type
        receive_param_ext_ack(vehicle)

    elif choice == '7':
        """设置 ering_mode 结构体 {uint8_t size; uint8_t eringMode[1
6]; uint32_t minValue; uint32_t maxValue;}"""
        param_id = input("Enter parameter ID: ").strip()
        size = int(input("Enter size (0-16): "))
        eringMode = []
        print("Enter 16 eringMode values (0-255), one per line:")
        for i in range(16):
            val = int(input(f"  eringMode[{i}]: "))
            eringMode.append(val)
        minValue = int(input("Enter minValue: "))
        maxValue = int(input("Enter maxValue: "))
        param_value = pack_ering_mode(size, eringMode, minValue, ma
xValue)
        print(f"  Packed: {param_value.hex()} ({len(param_value)} b
ytes)")
        send_param_ext_set(vehicle, param_id, param_value, 10)  # C
HAR type
        receive_param_ext_ack(vehicle)

    elif choice == '3':
        print("\n  Requesting all parameters...")
        send_param_ext_request_list(vehicle)
        all_params = receive_all_param_ext_values(vehicle)
        print(f"\n  Total parameters received: {len(all_params)}")
```

```
    else:
        print("Invalid choice. Select 1, 2, or 3.")


if __name__ == "__main__":
    main()
```

示例使用说明

运行后，成功建立连接

```
🚀 Starting MAVLink PARAM_EXT Tool (UDP Listening Mode)
------------------------------------------------------
This script listens for MAVLink on ALL interfaces:14550
Ensure your vehicle is configured to send to this port
------------------------------------------------------

Listening for MAVLink on: udpin:0.0.0.0:14550
Waiting for HEARTBEAT (timeout: 10s)...
✅ Heartbeat received from system 3, component 0
```

输入“1”选择设置参数

```
Available operations:
1. Read parameter value (by ID or index)
2. Set parameter value (raw/hex/string)
3. Request all parameters (PARAM_EXT_REQUEST_LIST)
4. Set struct: uint8_xy {x, y}
5. Set struct: float32_xy {x, y}
6. Set struct: float32_lxlyrxry {LX, LY, RX, RY}
7. Set struct: ering_mode {size, eringMode[16], minValue, maxValue}

Select operation [1-7]: 1
```

输入对应参数“NGT_SCN_MD_RNG”

```
Enter parameter ID: NGT_SCN_MD_RNG
```

成功返回完整消息

```
📥 Received PARAM_EXT_VALUE:
  🆔 ID: NGT_SCN_MD_RNG
fd9500000003014201001200 0c004e47545f53434e5f4d445f524e470000037afd4d0000000001000000020000 00d8ffffff80ffffff00000000
0000000078000000050000006821 9ab5aaaa0000f00c0044ffff0000800e0044ffff0000507afd4dffff0000f0e158b5000000000000000000000000
0000000000000000000000000000 0000000000000000000000000000000000000000000000000000000000000000000000000000b06d6
```

夜景模式

NIGHT_SCENE_MODE

mavlink消息：

接收PARAM_SET
发送PARAM_VALUE

获取

mavlink消息

接收PARAM_REQUEST_READ
发送PARAM_VALUE

参数：

名称	枚举值	说明
DJI_CAMERA_MANAGER_NIGHT_SCENE_MODE_DISABLE	0	关闭夜景模式
DJI_CAMERA_MANAGER_NIGHT_SCENE_MODE_ENABLE	1	开启夜景模式
DJI_CAMERA_MANAGER_NIGHT_SCENE_MODE_AUTO	2	自动夜景模式

示例

```
import time
import sys
from pymavlink import mavutil

def connect_to_vehicle():
    """建立 UDP 监听连接（默认监听所有接口的 14550 端口）"""
    connection_string = "udpin:0.0.0.0:14550"
    print(f"Listening for MAVLink on: {connection_string}")
    vehicle = mavutil.mavlink_connection(connection_string)
    print("Waiting for HEARTBEAT (timeout: 10s)...")
    try:
        vehicle.wait_heartbeat(timeout=10)
        print(f"  Heartbeat received from system {vehicle.target_system}, component {vehicle.target_component}")
        return vehicle
    except Exception as e:
        print(f"  Connection failed: {str(e)}")
        print("Ensure your vehicle is sending MAVLink messages to 14550 port")
        print("Example QGroundControl setup: Settings > General > MAVLink > Listening Port = 14550")
```

```
        sys.exit(1)

def format_param_id(param_id):
    """格式化16字节参数ID（符合MAVLink协议要求）"""
    if len(param_id) > 16:
        raise ValueError("param_id cannot exceed 16 characters")

    param_id_bytes = bytearray(16)
    for i in range(min(len(param_id), 16)):
        param_id_bytes[i] = ord(param_id[i])

    # 长度<16时添加NULL终止符
    if len(param_id) < 16:
        param_id_bytes[len(param_id)] = 0

    return param_id_bytes

def send_param_set(vehicle, param_id, param_value, param_type):
    """发送参数设置请求"""
    vehicle.mav.param_set_send(
        vehicle.target_system,
        vehicle.target_component,
        format_param_id(param_id),
        param_value,
        param_type
    )
    print(f"  Sent PARAM_SET for '{param_id}' = {param_value} (type: {param_type})")

def send_param_request_read(vehicle, param_id=None, param_index=-1):
    """发送参数读取请求"""
    param_id_bytes = format_param_id(param_id) if param_id else bytearray(16)

    vehicle.mav.param_request_read_send(
        vehicle.target_system,
        vehicle.target_component,
        param_id_bytes,
        param_index
    )

    target = f"index {param_index}" if param_index >= 0 else f"ID '{param_id}'"
    print(f"  Sent PARAM_REQUEST_READ for {target}")
```

```
def send_param_request_list(vehicle):
    """发送请求所有参数列表"""
    vehicle.mav.param_request_list_send(
        vehicle.target_system,
        vehicle.target_component
    )
    print(f"  Sent PARAM_REQUEST_LIST to system {vehicle.target_sys
tem}, component {vehicle.target_component}")

def receive_param_value(vehicle, timeout=5.0):
    """接收参数值响应"""
    print(f"  Waiting for PARAM_VALUE ({timeout}s timeout)...")
    msg = vehicle.recv_match(type='PARAM_VALUE', blocking=True, tim
eout=timeout)

    if not msg:
        print("  Timeout: No PARAM_VALUE received")
        return None

    # 解析参数ID
    param_id = msg.param_id

    # 解析参数值
    param_value = msg.param_value

    # 参数类型映射
    type_names = {
        0: "INT8", 1: "UINT8", 2: "INT16", 3: "UINT16", 4: "INT32",
        5: "UINT32", 6: "INT64", 7: "UINT64", 8: "FLOAT", 9: "DOUBL
E"
    }
    param_type = type_names.get(msg.param_type, f"UNKNOWN({msg.para
m_type})")

    print("\n  Received PARAM_VALUE:")
    print(f"    ID: {param_id}")
    print(f"    Value: {param_value}")
    print(f"    Type: {param_type}")
    print(f"    Index/Total: {msg.param_index}/{msg.param_count}")

    return {
        'id': param_id,
        'value': param_value,
        'type': param_type,
```

```
        'index': msg.param_index,
        'count': msg.param_count
    }

def receive_all_param_values(vehicle, timeout=10.0):
    """接收所有参数值，直到收到 param_count 条消息后退出循环"""
    received_params = []
    expected_count = None
    print(f"  Waiting for all PARAM_VALUE messages...")

    while True:
        msg = vehicle.recv_match(type='PARAM_VALUE', blocking=True,
timeout=timeout)

        if not msg:
            print(f"  Timeout: Received {len(received_params)} para
meters (expected {expected_count if expected_count else 'unknow
n'})")
            break

        param_id = msg.param_id
        param_value = msg.param_value

        # 参数类型映射
        type_names = {
            0: "INT8", 1: "UINT8", 2: "INT16", 3: "UINT16", 4: "INT
32",
            5: "UINT32", 6: "INT64", 7: "UINT64", 8: "FLOAT", 9: "D
OUBLE"
        }
        param_type = type_names.get(msg.param_type, f"UNKNOWN({msg.
param_type})")

        # 首次收到消息时记录总参数数量
        if expected_count is None:
            expected_count = msg.param_count
            print(f"  Total parameters to receive: {expected_coun
t}")

        print(f"  [{msg.param_index}/{expected_count}] {param_id} =
{param_value} ({param_type})")

        received_params.append({
            'id': param_id,
            'value': param_value,
```

```
            'type': param_type,
            'index': msg.param_index,
            'count': msg.param_count
        })

        # 当已接收数量达到预期总数时退出循环
        if len(received_params) >= expected_count:
            print(f"\n  Successfully received all {expected_count} parameters")
            break

    return received_params

def main():
    print("  Starting MAVLink PARAM_SET Tool (UDP Listening Mode)")
    print("------------------------------------------------------")
    print("This script listens for MAVLink on ALL interfaces:14550")
    print("Ensure your vehicle is configured to send to this port")
    print("------------------------------------------------------\n")

    vehicle = connect_to_vehicle()

    # 操作选择
    print("\nAvailable operations:")
    print("1. Set parameter value")
    print("2. Read parameter value (by ID or index)")
    print("3. Request all parameters (PARAM_REQUEST_LIST)")
    choice = input("\nSelect operation [1-3]: ").strip()

    if choice == '1':
        param_id = input("Enter parameter ID to set: ").strip()
        param_value = float(input("Enter new value: ").strip())

        print("\nParameter types:")
        print("  8 = FLOAT (most common)")
        print("  9 = DOUBLE")
        print("  4 = INT32")
        print("  1 = UINT8")
        param_type = int(input("Enter type number [8]: ").strip() or 8)

        send_param_set(vehicle, param_id, param_value, param_type)
        receive_param_value(vehicle)
```

```
    elif choice == '2':
        # mode = input("Read by [id/index]: ").lower()
        mode = 'id'
        if mode == 'id':
            param_id = input("Enter parameter ID: ").strip()
            send_param_request_read(vehicle, param_id=param_id)
            receive_param_value(vehicle)
        elif mode == 'index':
            param_index = int(input("Enter parameter index: "))
            send_param_request_read(vehicle, param_index=param_inde
x)
            receive_param_value(vehicle)
        else:
            print("Invalid mode. Use 'id' or 'index'")

    elif choice == '3':
        print("\n  Requesting all parameters...")
        send_param_request_list(vehicle)
        all_params = receive_all_param_values(vehicle)
        print(f"\n  Total parameters received: {len(all_params)}")

    else:
        print("Invalid choice. Select 1, 2, or 3.")


if __name__ == "__main__":
    main()
```

示例使用说明

获取

运行后，成功建立连接

```
🚀 Starting MAVLink PARAM_SET Tool (UDP Listening Mode)
--------------------------------------------------
This script listens for MAVLink on ALL interfaces:14550
Ensure your vehicle is configured to send to this port
--------------------------------------------------

Listening for MAVLink on: udpin:0.0.0.0:14550
Waiting for HEARTBEAT (timeout: 10s)...
✅ Heartbeat received from system 3, component 0

Available operations:
1. Set parameter value
2. Read parameter value (by ID or index)
3. Request all parameters (PARAM_REQUEST_LIST)
```

输入“2”选择读取参数

```
Select operation [1-3]: 2
```

输入对应参数“NIGHT_SCENE_MODE”

```
Enter parameter ID: NIGHT_SCENE_MODE
```

成功返回该参数对应的数值

```
📥 Received PARAM_VALUE:
  🆔 ID: NIGHT_SCENE_MODE
  🔢 Value: 0.0
```

设置

运行后，成功建立连接

```
🚀 Starting MAVLink PARAM_SET Tool (UDP Listening Mode)
--------------------------------------------------
This script listens for MAVLink on ALL interfaces:14550
Ensure your vehicle is configured to send to this port
--------------------------------------------------

Listening for MAVLink on: udpin:0.0.0.0:14550
Waiting for HEARTBEAT (timeout: 10s)...
✅ Heartbeat received from system 3, component 0

Available operations:
1. Set parameter value
2. Read parameter value (by ID or index)
3. Request all parameters (PARAM_REQUEST_LIST)
```

输入“1”选择设置参数

```
Available operations:
1. Set parameter value
2. Read parameter value (by ID or index)
3. Request all parameters (PARAM_REQUEST_LIST)

Select operation [1-3]: 1
```

输入对应参数“NIGHT_SCENE_MODE”

```
Enter parameter ID to set: NIGHT_SCENE_MODE
```

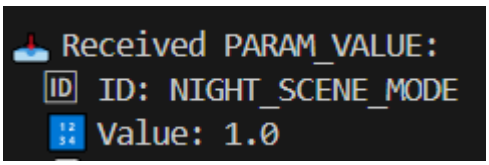
输入要设置的数值

```
Enter new value: 1
```

选择参数类型“1”

```
Parameter types:
  8 = FLOAT (most common)
  9 = DOUBLE
  4 = INT32
  1 = UINT8
Enter type number [8]: 1
```

成功返回设置值



获取拍摄或录像时流媒体数据源的存储范围

STRM_STORE_RNG

mavlink消息：

接收PARAM_EXT_REQUEST_READ
发送PARAM_EXT_VALUE

参数：

字段	类型	说明
size	uint8_t	有效列表长度（最大16）
minValue	uint32_t	数值范围最小值
maxValue	uint32_t	数值范围最大值

union（同一时刻仅生效一个字段）

字段	类型	说明
photoStorageFormat	E_DjiCameraManagerPhotoStorageFormat[16]	照片存储格式枚
videoStorageFormat	E_DjiCameraManagerVideoStorageFormat[16]	视频存储格式枚
photoRatioFormat	E_DjiCameraManagerPhotoRatio[16]	照片比例枚举列
streamSource	E_DjiCameraManagerStreamSource[16]	视频流源枚举列
streamStorage	E_DjiCameraManagerStreamStorage[16]	视频存储介质枚
nightSceneMode	E_DjiCameraManagerNightSceneMode[16]	夜景模式枚举列
meteringMode	E_DjiCameraManagerMeteringMode[16]	测光模式枚举列

示例

```
import time
import sys
import struct
import ctypes
from pymavlink import mavutil

def connect_to_vehicle():
    """建立 UDP 监听连接（默认监听所有接口的 14550 端口）"""
    connection_string = "udpin:0.0.0.0:14550"
    print(f"Listening for MAVLink on: {connection_string}")
    vehicle = mavutil.mavlink_connection(connection_string)
    print("Waiting for HEARTBEAT (timeout: 10s)...")
    try:
        vehicle.wait_heartbeat(timeout=10)
        print(f"  Heartbeat received from system {vehicle.target_system}, component {vehicle.target_component}")
        return vehicle
    except Exception as e:
        print(f"  Connection failed: {str(e)}")
        print("Ensure your vehicle is sending MAVLink messages to 14550 port")
        print("Example QGroundControl setup: Settings > General > MAVLink > Listening Port = 14550")
        sys.exit(1)

def format_param_id(param_id):
    """格式化16字节参数ID（符合MAVLink协议要求）"""
    if len(param_id) > 16:
        raise ValueError("param_id cannot exceed 16 characters")

    param_id_bytes = bytearray(16)
    for i in range(min(len(param_id), 16)):
        param_id_bytes[i] = ord(param_id[i])

    # 长度<16时添加NULL终止符
    if len(param_id) < 16:
        param_id_bytes[len(param_id)] = 0

    return param_id_bytes

# ============ 结构体打包函数 ============

def pack_uint8_xy(x, y):
    """打包 {uint8_t x; uint8_t y} 结构体 (2字节)"""
    return struct.pack('<BB', x, y)
```

```
def pack_float32_xy(x, y):
    """打包 {float32 x; float32 y} 结构体 (8字节)"""
    return struct.pack('<ff', x, y)

def pack_float32_lxlyrxry(LX, LY, RX, RY):
    """打包 {float32 LX; float32 LY; float32 RX; float32 RY;} 结构体 (16字节)"""
    return struct.pack('<ffff', LX, LY, RX, RY)

def pack_ering_mode(size, eringMode, minValue, maxValue):
    """打包 {uint8_t size; uint8_t eringMode[16]; uint32_t minValue; uint32_t maxValue;} 结构体 (25字节)

    Args:
        size: uint8_t, eringMode数组的有效长度
        eringMode: list of int (0-255), 16个uint8_t值
        minValue: uint32_t, 最小值
        maxValue: uint32_t, 最大值

    Returns:
        bytes: 打包后的二进制数据
    """
    # 确保eringMode是16字节
    eringMode_bytes = bytearray(16)
    for i in range(min(len(eringMode), 16)):
        eringMode_bytes[i] = eringMode[i] & 0xFF

    return struct.pack('<B16sII', size, eringMode_bytes, minValue, maxValue)

def format_param_value(param_value):
    """格式化128字节参数值（接受二进制输入）"""
    if isinstance(param_value, str):
        # 如果是十六进制字符串（如 "48656c6c6f"），先转换为字节
        if param_value.startswith('0x') or all(c in '0123456789abcdefABCDEF' for c in param_value.replace(' ', '')):
            hex_str = param_value.replace('0x', '').replace(' ', '')
            if len(hex_str) % 2 == 0:
                param_value = bytes.fromhex(hex_str)
            else:
                raise ValueError("Invalid hex string: must have even number of characters")
        else:
```

```
            # 普通字符串转字节
            param_value = param_value.encode('utf-8')

    if len(param_value) > 128:
        raise ValueError("param_value cannot exceed 128 bytes")

    param_value_bytes = bytearray(128)
    param_value_bytes[:len(param_value)] = param_value

    return param_value_bytes

def hex_dump(data, bytes_per_line=16):
    """格式化二进制数据为 hex + ASCII 展示（类 hexdump 风格）

    Returns:
        str: 格式化后的多行字符串
    """
    lines = []
    for i in range(0, len(data), bytes_per_line):
        chunk = data[i:i + bytes_per_line]
        hex_part = ' '.join(f'{b:02X}' for b in chunk)
        ascii_part = ''.join(chr(b) if 32 <= b < 127 else '.' for b
in chunk)
        lines.append(f"  {i:04X}  {hex_part:<{bytes_per_line*3}}
{ascii_part}")
    return '\n'.join(lines)

def send_param_ext_request_read(vehicle, param_id=None, param_index
=-1):
    """发送参数读取请求"""
    param_id_bytes = format_param_id(param_id) if param_id else byt
earray(16)

    vehicle.mav.param_ext_request_read_send(
        vehicle.target_system,
        vehicle.target_component,
        param_id_bytes,
        param_index
    )

    target = f"index {param_index}" if param_index >= 0 else f"ID
'{param_id}'"
    print(f"  Sent PARAM_EXT_REQUEST_READ for {target}")

def send_param_ext_request_list(vehicle):
```

```
    """发送请求所有参数列表"""
    vehicle.mav.param_ext_request_list_send(
        vehicle.target_system,
        vehicle.target_component
    )
    print(f"  Sent PARAM_EXT_REQUEST_LIST to system {vehicle.target
_system}, component {vehicle.target_component}")

def send_param_ext_set(vehicle, param_id, param_value, param_type):
    """发送参数设置请求"""
    vehicle.mav.param_ext_set_send(
        vehicle.target_system,
        vehicle.target_component,
        format_param_id(param_id),
        format_param_value(param_value),
        param_type
    )
    print(f"  Sent PARAM_EXT_SET for '{param_id}' = '{param_value}'
(type: {param_type})")

def receive_param_ext_value(vehicle, timeout=5.0):
    """接收参数值响应"""
    print(f"  Waiting for PARAM_EXT_VALUE ({timeout}s timeout)...")
    msg = vehicle.recv_match(type='PARAM_EXT_VALUE', blocking=True,
timeout=timeout)

    if not msg:
        print("  Timeout: No PARAM_EXT_VALUE received")
        return None

    # 解析参数ID
    param_id = msg.param_id

    # 解析参数值（直接获取128字节原始数据）
    param_value_raw = msg.param_value
    # byte_pointer = (ctypes.c_ubyte * 128).from_buffer_copy(msg.pa
ram_value)
    if isinstance(param_value_raw, (bytes, bytearray)):
        param_value_bytes = bytes(param_value_raw)
    # param_value_bytes = bytes(byte_pointer)
    elif isinstance(param_value_raw, str):
        # 替换无法编码的字符（如 \ufffd）为 0x00，还原为128字节
        param_value_bytes = param_value_raw.encode('latin1', errors
='replace')
    else:
```

```
        param_value_bytes = bytes(param_value_raw)

    # 参数类型映射
    type_names = {
        0: "INT8", 1: "UINT8", 2: "INT16", 3: "UINT16", 4: "INT32",
        5: "UINT32", 6: "INT64", 7: "UINT64", 8: "FLOAT", 9: "FLOA
T",
        10: "CHAR", 11: "CHAR", 12: "UINT8_ENUM", 13: "STRING"
    }
    param_type = type_names.get(msg.param_type, f"UNKNOWN({msg.para
m_type})")

    print("\n  Received PARAM_EXT_VALUE:")
    print(f"    ID: {param_id}")
    print(f"    Type: {param_type}  |  Data: {len(param_value_byte
s)} bytes")
    print(f"    Index/Total: {msg.param_index}/{msg.param_count}")
    print(hex_dump(param_value_bytes))

    return {
        'id': param_id,
        'value': param_value_bytes,
        'type': param_type,
        'index': msg.param_index,
        'count': msg.param_count
    }


def receive_all_param_ext_values(vehicle, timeout=10.0):
    """接收所有参数值，直到收到 param_count 条消息后退出循环"""
    received_params = []
    expected_count = None
    print(f"  Waiting for all PARAM_EXT_VALUE messages...")

    while True:
        msg = vehicle.recv_match(type='PARAM_EXT_VALUE', blocking=T
rue, timeout=timeout)

        if not msg:
            print(f"  Timeout: Received {len(received_params)} para
meters (expected {expected_count if expected_count else 'unknow
n'})")
            break

        param_id = msg.param_id
```

```
        param_value_raw = msg.param_value

        # 直接获取128字节原始数据
        if isinstance(param_value_raw, (bytes, bytearray)):
            param_value_bytes = bytes(param_value_raw)
        elif isinstance(param_value_raw, str):
            # 替换无法编码的字符（如 \ufffd）为 0x00
            param_value_bytes = param_value_raw.encode('latin1', er
rors='replace')
        else:
            param_value_bytes = bytes(param_value_raw)

        # 参数类型映射
        type_names = {
            0: "INT8", 1: "UINT8", 2: "INT16", 3: "UINT16", 4: "INT
32",
            5: "UINT32", 6: "INT64", 7: "UINT64", 8: "FLOAT", 9: "D
OUBLE",
            10: "CHAR", 11: "BOOL", 12: "UINT8_ENUM", 13: "STRING"
        }
        param_type = type_names.get(msg.param_type, f"UNKNOWN({msg.
param_type})")

        # 首次收到消息时记录总参数数量
        if expected_count is None:
            expected_count = msg.param_count
            print(f"  Total parameters to receive: {expected_coun
t}")

        print(f"  [{msg.param_index}/{expected_count}] {param_id}
[{param_type}][{len(param_value_bytes)}B]")
        print(hex_dump(param_value_bytes))

        received_params.append({
            'id': param_id,
            'value': param_value_bytes,
            'type': param_type,
            'index': msg.param_index,
            'count': msg.param_count
        })

        # 当已接收数量达到预期总数时退出循环
        if len(received_params) >= expected_count:
            print(f"\n  Successfully received all {expected_count}
parameters")
```

```
            break

    return received_params

def receive_param_ext_ack(vehicle, timeout=5.0):
    """接收参数设置确认"""
    print(f"  Waiting for PARAM_EXT_ACK ({timeout}s timeout)...")
    msg = vehicle.recv_match(type='PARAM_EXT_ACK', blocking=True, t
imeout=timeout)

    if not msg:
        print("  Timeout: No PARAM_EXT_ACK received")
        return None

    param_id = msg.param_id
    param_value = msg.param_value

    # 结果代码映射
    result_codes = {
        0: "  PARAM_ACK_ACCEPTED",
        1: "⚠ PARAM_ACK_VALUE_UNSUPPORTED",
        2: "  PARAM_ACK_FAILED",
        3: "  PARAM_ACK_IN_PROGRESS"
    }
    result = result_codes.get(msg.param_result, f"  UNKNOWN({msg.pa
ram_result})")

    print("\n  Received PARAM_EXT_ACK:")
    print(f"    ID: {param_id}")
    print(f"    Value: {param_value}")
    print(f"    Result: {result}")

    return {
        'id': param_id,
        'value': param_value,
        'result': result,
        'result_code': msg.param_result
    }

def main():
    print("  Starting MAVLink PARAM_EXT Tool (UDP Listening Mode)")
    print("-----------------------------------------------------")
    print("This script listens for MAVLink on ALL interfaces:1455
0")
    print("Ensure your vehicle is configured to send to this port")
```

```
    print("-------------------------------------------------------
\n")

    vehicle = connect_to_vehicle()

    # 操作选择
    print("\nAvailable operations:")
    print("1. Read parameter value (by ID or index)")
    print("2. Set parameter value (raw/hex/string)")
    print("3. Request all parameters (PARAM_EXT_REQUEST_LIST)")
    print("4. Set struct: uint8_xy {x, y}")
    print("5. Set struct: float32_xy {x, y}")
    print("6. Set struct: float32_lxlyrxry {LX, LY, RX, RY}")
    print("7. Set struct: ering_mode {size, eringMode[16], minValu
e, maxValue}")
    choice = input("\nSelect operation [1-7]: ").strip()

    if choice == '1':
        # mode = input("Read by [id/index]: ").lower()
        mode = 'id'
        if mode == 'id':
            param_id = input("Enter parameter ID: ").strip()
            send_param_ext_request_read(vehicle, param_id=param_id)
            receive_param_ext_value(vehicle)
        elif mode == 'index':
            param_index = int(input("Enter parameter index: "))
            send_param_ext_request_read(vehicle, param_index=param_
index)
            receive_param_ext_value(vehicle)
        else:
            print("Invalid mode. Use 'id' or 'index'")

    elif choice == '2':
        param_id = input("Enter parameter ID to set: ").strip()

        print("\nEnter parameter value (supports):")
        print("  - Plain string (e.g., hello)")
        print("  - Hex string (e.g., 48656c6c6f or 0x48656c6c6f)")
        print("  - Raw bytes (e.g., b'\\x48\\x65\\x6c\\x6c\\x6f')")
        param_value_input = input("Enter value: ").strip()

        # 解析输入值
        if param_value_input.startswith("b'") or param_value_input.
startswith('b"'):
            # Python字节字面量格式
```

```
            param_value = eval(param_value_input)
        elif param_value_input.startswith('0x') or all(c in '012345
6789abcdefABCDEF ' for c in param_value_input):
            # 十六进制格式
            hex_str = param_value_input.replace('0x', '').replace('
', '')
            try:
                param_value = bytes.fromhex(hex_str)
            except ValueError:
                param_value = param_value_input.encode('utf-8')
        else:
            # 默认作为字符串处理
            param_value = param_value_input.encode('utf-8')

        print("\nParameter types:")
        print("  8 = FLOAT (most common)")
        print("  13 = STRING")
        print("  10 = CHAR (raw bytes)")
        print("  Other types require specific binary formats")
        param_type = int(input("Enter type number [8]: ") or 8)

        send_param_ext_set(vehicle, param_id, param_value, param_ty
pe)
        receive_param_ext_ack(vehicle)

    elif choice == '4':
        """设置 uint8_xy 结构体 {uint8_t x; uint8_t y}"""
        param_id = input("Enter parameter ID: ").strip()
        x = int(input("Enter x (0-255): "))
        y = int(input("Enter y (0-255): "))
        param_value = pack_uint8_xy(x, y)
        print(f"  Packed: {param_value.hex()} ({len(param_value)} b
ytes)")
        send_param_ext_set(vehicle, param_id, param_value, 10)  # C
HAR type
        receive_param_ext_ack(vehicle)

    elif choice == '5':
        """设置 float32_xy 结构体 {float32 x; float32 y}"""
        param_id = input("Enter parameter ID: ").strip()
        x = float(input("Enter x: "))
        y = float(input("Enter y: "))
        param_value = pack_float32_xy(x, y)
        print(f"  Packed: {param_value.hex()} ({len(param_value)} b
ytes)")
```

```
        send_param_ext_set(vehicle, param_id, param_value, 10)  # C
HAR type
        receive_param_ext_ack(vehicle)

    elif choice == '6':
        """设置 float32_lxlyrxry 结构体 {float32 LX; float32 LY; floa
t32 RX; float32 RY;}"""
        param_id = input("Enter parameter ID: ").strip()
        LX = float(input("Enter LX: "))
        LY = float(input("Enter LY: "))
        RX = float(input("Enter RX: "))
        RY = float(input("Enter RY: "))
        param_value = pack_float32_lxlyrxry(LX, LY, RX, RY)
        print(f"  Packed: {param_value.hex()} ({len(param_value)} b
ytes)")
        send_param_ext_set(vehicle, param_id, param_value, 10)  # C
HAR type
        receive_param_ext_ack(vehicle)

    elif choice == '7':
        """设置 ering_mode 结构体 {uint8_t size; uint8_t eringMode[1
6]; uint32_t minValue; uint32_t maxValue;}"""
        param_id = input("Enter parameter ID: ").strip()
        size = int(input("Enter size (0-16): "))
        eringMode = []
        print("Enter 16 eringMode values (0-255), one per line:")
        for i in range(16):
            val = int(input(f"  eringMode[{i}]: "))
            eringMode.append(val)
        minValue = int(input("Enter minValue: "))
        maxValue = int(input("Enter maxValue: "))
        param_value = pack_ering_mode(size, eringMode, minValue, ma
xValue)
        print(f"  Packed: {param_value.hex()} ({len(param_value)} b
ytes)")
        send_param_ext_set(vehicle, param_id, param_value, 10)  # C
HAR type
        receive_param_ext_ack(vehicle)

    elif choice == '3':
        print("\n  Requesting all parameters...")
        send_param_ext_request_list(vehicle)
        all_params = receive_all_param_ext_values(vehicle)
        print(f"\n  Total parameters received: {len(all_params)}")
```

```
    else:
        print("Invalid choice. Select 1, 2, or 3.")


if __name__ == "__main__":
    main()
```

运行后，成功建立连接

```
🚀 Starting MAVLink PARAM_EXT Tool (UDP Listening Mode)
------------------------------------------------------
This script listens for MAVLink on ALL interfaces:14550
Ensure your vehicle is configured to send to this port
------------------------------------------------------

Listening for MAVLink on: udpin:0.0.0.0:14550
Waiting for HEARTBEAT (timeout: 10s)...
✅ Heartbeat received from system 3, component 0
```

输入“1”选择设置参数

```
Available operations:
1. Read parameter value (by ID or index)
2. Set parameter value (raw/hex/string)
3. Request all parameters (PARAM_EXT_REQUEST_LIST)
4. Set struct: uint8_xy {x, y}
5. Set struct: float32_xy {x, y}
6. Set struct: float32_lxlyrxry {LX, LY, RX, RY}
7. Set struct: ering_mode {size, eringMode[16], minValue, maxValue}

Select operation [1-7]: 1
```

输入对应参数“模板”

```
Enter parameter ID: STRM_STORE_RNG
```

成功返回完整消息

```
📥 Received PARAM_EXT_VALUE:
  🆔 ID: STRM_STORE_RNG
fd9500000103014201001200d005354524d5f53544f52455f524e47000003fad671000000000300000007000000d8ffffff80ffffff0000000
0000000078000000050000068214babaaaa0000800e0040ffff0000600b0040ffff000050fad671ffff0000f0e109ab0000000000000000000
0000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000b601e
```

同步分屏缩放功能

SYNC_SPLT_ZM

设置

mavlink消息：

接收PARAM_SET
发送PARAM_VALUE

获取

mavlink消息

接收PARAM_REQUEST_READ
发送PARAM_VALUE

参数：

类型	枚举值	说明
bool	0	禁用同步分屏缩放功能
bool	1	启用同步分屏缩放功能

示例

```
import time
import sys
from pymavlink import mavutil

def connect_to_vehicle():
    """建立 UDP 监听连接（默认监听所有接口的 14550 端口）"""
    connection_string = "udpin:0.0.0.0:14550"
    print(f"Listening for MAVLink on: {connection_string}")
    vehicle = mavutil.mavlink_connection(connection_string)
    print("Waiting for HEARTBEAT (timeout: 10s)...")
    try:
        vehicle.wait_heartbeat(timeout=10)
        print(f"  Heartbeat received from system {vehicle.target_sy
stem}, component {vehicle.target_component}")
        return vehicle
    except Exception as e:
        print(f"  Connection failed: {str(e)}")
        print("Ensure your vehicle is sending MAVLink messages to 1
4550 port")
        print("Example QGroundControl setup: Settings > General > M
AVLink > Listening Port = 14550")
        sys.exit(1)
```

```
def format_param_id(param_id):
    """格式化16字节参数ID（符合MAVLink协议要求）"""
    if len(param_id) > 16:
        raise ValueError("param_id cannot exceed 16 characters")

    param_id_bytes = bytearray(16)
    for i in range(min(len(param_id), 16)):
        param_id_bytes[i] = ord(param_id[i])

    # 长度<16时添加NULL终止符
    if len(param_id) < 16:
        param_id_bytes[len(param_id)] = 0

    return param_id_bytes

def send_param_set(vehicle, param_id, param_value, param_type):
    """发送参数设置请求"""
    vehicle.mav.param_set_send(
        vehicle.target_system,
        vehicle.target_component,
        format_param_id(param_id),
        param_value,
        param_type
    )
    print(f"  Sent PARAM_SET for '{param_id}' = {param_value} (type: {param_type})")

def send_param_request_read(vehicle, param_id=None, param_index=-1):
    """发送参数读取请求"""
    param_id_bytes = format_param_id(param_id) if param_id else bytearray(16)

    vehicle.mav.param_request_read_send(
        vehicle.target_system,
        vehicle.target_component,
        param_id_bytes,
        param_index
    )

    target = f"index {param_index}" if param_index >= 0 else f"ID '{param_id}'"
    print(f"  Sent PARAM_REQUEST_READ for {target}")
```

```
def send_param_request_list(vehicle):
    """发送请求所有参数列表"""
    vehicle.mav.param_request_list_send(
        vehicle.target_system,
        vehicle.target_component
    )
    print(f"  Sent PARAM_REQUEST_LIST to system {vehicle.target_sys
tem}, component {vehicle.target_component}")

def receive_param_value(vehicle, timeout=5.0):
    """接收参数值响应"""
    print(f"  Waiting for PARAM_VALUE ({timeout}s timeout)...")
    msg = vehicle.recv_match(type='PARAM_VALUE', blocking=True, tim
eout=timeout)

    if not msg:
        print("  Timeout: No PARAM_VALUE received")
        return None

    # 解析参数ID
    param_id = msg.param_id

    # 解析参数值
    param_value = msg.param_value

    # 参数类型映射
    type_names = {
        0: "INT8", 1: "UINT8", 2: "INT16", 3: "UINT16", 4: "INT32",
        5: "UINT32", 6: "INT64", 7: "UINT64", 8: "FLOAT", 9: "DOUBL
E"
    }
    param_type = type_names.get(msg.param_type, f"UNKNOWN({msg.para
m_type})")

    print("\n  Received PARAM_VALUE:")
    print(f"    ID: {param_id}")
    print(f"    Value: {param_value}")
    print(f"    Type: {param_type}")
    print(f"    Index/Total: {msg.param_index}/{msg.param_count}")

    return {
        'id': param_id,
        'value': param_value,
        'type': param_type,
        'index': msg.param_index,
```

```
        'count': msg.param_count
    }

def receive_all_param_values(vehicle, timeout=10.0):
    """接收所有参数值，直到收到 param_count 条消息后退出循环"""
    received_params = []
    expected_count = None
    print(f"  Waiting for all PARAM_VALUE messages...")

    while True:
        msg = vehicle.recv_match(type='PARAM_VALUE', blocking=True,
timeout=timeout)

        if not msg:
            print(f"  Timeout: Received {len(received_params)} para
meters (expected {expected_count if expected_count else 'unknow
n'})")
            break

        param_id = msg.param_id
        param_value = msg.param_value

        # 参数类型映射
        type_names = {
            0: "INT8", 1: "UINT8", 2: "INT16", 3: "UINT16", 4: "INT
32",
            5: "UINT32", 6: "INT64", 7: "UINT64", 8: "FLOAT", 9: "D
OUBLE"
        }
        param_type = type_names.get(msg.param_type, f"UNKNOWN({msg.
param_type})")

        # 首次收到消息时记录总参数数量
        if expected_count is None:
            expected_count = msg.param_count
            print(f"  Total parameters to receive: {expected_coun
t}")

        print(f"  [{msg.param_index}/{expected_count}] {param_id} =
{param_value} ({param_type})")

        received_params.append({
            'id': param_id,
            'value': param_value,
            'type': param_type,
```

```
            'index': msg.param_index,
            'count': msg.param_count
        })

        # 当已接收数量达到预期总数时退出循环
        if len(received_params) >= expected_count:
            print(f"\n  Successfully received all {expected_count} parameters")
            break

    return received_params

def main():
    print("  Starting MAVLink PARAM_SET Tool (UDP Listening Mode)")
    print("----------------------------------------------------")
    print("This script listens for MAVLink on ALL interfaces:14550")
    print("Ensure your vehicle is configured to send to this port")
    print("----------------------------------------------------\n")

    vehicle = connect_to_vehicle()

    # 操作选择
    print("\nAvailable operations:")
    print("1. Set parameter value")
    print("2. Read parameter value (by ID or index)")
    print("3. Request all parameters (PARAM_REQUEST_LIST)")
    choice = input("\nSelect operation [1-3]: ").strip()

    if choice == '1':
        param_id = input("Enter parameter ID to set: ").strip()
        param_value = float(input("Enter new value: ").strip())

        print("\nParameter types:")
        print("  8 = FLOAT (most common)")
        print("  9 = DOUBLE")
        print("  4 = INT32")
        print("  1 = UINT8")
        param_type = int(input("Enter type number [8]: ").strip() or 8)

        send_param_set(vehicle, param_id, param_value, param_type)
        receive_param_value(vehicle)
```

```
    elif choice == '2':
        # mode = input("Read by [id/index]: ").lower()
        mode = 'id'
        if mode == 'id':
            param_id = input("Enter parameter ID: ").strip()
            send_param_request_read(vehicle, param_id=param_id)
            receive_param_value(vehicle)
        elif mode == 'index':
            param_index = int(input("Enter parameter index: "))
            send_param_request_read(vehicle, param_index=param_inde
x)
            receive_param_value(vehicle)
        else:
            print("Invalid mode. Use 'id' or 'index'")

    elif choice == '3':
        print("\n  Requesting all parameters...")
        send_param_request_list(vehicle)
        all_params = receive_all_param_values(vehicle)
        print(f"\n  Total parameters received: {len(all_params)}")

    else:
        print("Invalid choice. Select 1, 2, or 3.")


if __name__ == "__main__":
    main()
```

示例使用说明

获取

运行后，成功建立连接

```
🚀 Starting MAVLink PARAM_SET Tool (UDP Listening Mode)
--------------------------------------------------
This script listens for MAVLink on ALL interfaces:14550
Ensure your vehicle is configured to send to this port
--------------------------------------------------

Listening for MAVLink on: udpin:0.0.0.0:14550
Waiting for HEARTBEAT (timeout: 10s)...
✅ Heartbeat received from system 3, component 0

Available operations:
1. Set parameter value
2. Read parameter value (by ID or index)
3. Request all parameters (PARAM_REQUEST_LIST)
```

输入“2”选择读取参数

```
Select operation [1-3]: 2
```

输入对应参数“SYNC_SPLT_ZM”

```
Enter parameter ID: SYNC_SPLT_ZM
```

成功返回该参数对应的数值

```
📥 Received PARAM_VALUE:
  🆔 ID: SYNC_SPLT_ZM
  🔢 Value: 0.0
```

相机测光模式

METER_MODE

设置

mavlink消息：

接收PARAM_SET
发送PARAM_VALUE

获取

mavlink消息

接收PARAM_REQUEST_READ
发送PARAM_VALUE

参数：

枚举值	值	说明
DJI_CAMERA_MANAGER_METERING_MODE_CENTRAL	0	中央重点测光（Center-Weighted Metering）
DJI_CAMERA_MANAGER_METERING_MODE_AVERAGE	1	平均测光（Average Metering）
DJI_CAMERA_MANAGER_METERING_MODE_SPOT	2	点测光（Spot Metering）

示例

```
import time
import sys
from pymavlink import mavutil

def connect_to_vehicle():
    """建立 UDP 监听连接（默认监听所有接口的 14550 端口）"""
    connection_string = "udpin:0.0.0.0:14550"
    print(f"Listening for MAVLink on: {connection_string}")
    vehicle = mavutil.mavlink_connection(connection_string)
    print("Waiting for HEARTBEAT (timeout: 10s)...")
    try:
        vehicle.wait_heartbeat(timeout=10)
        print(f"  Heartbeat received from system {vehicle.target_system}, component {vehicle.target_component}")
        return vehicle
    except Exception as e:
        print(f"  Connection failed: {str(e)}")
        print("Ensure your vehicle is sending MAVLink messages to 14550 port")
        print("Example QGroundControl setup: Settings > General > MAVLink > Listening Port = 14550")
        sys.exit(1)

def format_param_id(param_id):
```

```
    """格式化16字节参数ID（符合MAVLink协议要求）"""
    if len(param_id) > 16:
        raise ValueError("param_id cannot exceed 16 characters")

    param_id_bytes = bytearray(16)
    for i in range(min(len(param_id), 16)):
        param_id_bytes[i] = ord(param_id[i])

    # 长度<16时添加NULL终止符
    if len(param_id) < 16:
        param_id_bytes[len(param_id)] = 0

    return param_id_bytes

def send_param_set(vehicle, param_id, param_value, param_type):
    """发送参数设置请求"""
    vehicle.mav.param_set_send(
        vehicle.target_system,
        vehicle.target_component,
        format_param_id(param_id),
        param_value,
        param_type
    )
    print(f"  Sent PARAM_SET for '{param_id}' = {param_value} (type: {param_type})")

def send_param_request_read(vehicle, param_id=None, param_index=-1):
    """发送参数读取请求"""
    param_id_bytes = format_param_id(param_id) if param_id else bytearray(16)

    vehicle.mav.param_request_read_send(
        vehicle.target_system,
        vehicle.target_component,
        param_id_bytes,
        param_index
    )

    target = f"index {param_index}" if param_index >= 0 else f"ID '{param_id}'"
    print(f"  Sent PARAM_REQUEST_READ for {target}")

def send_param_request_list(vehicle):
    """发送请求所有参数列表"""
```

```
    vehicle.mav.param_request_list_send(
        vehicle.target_system,
        vehicle.target_component
    )
    print(f"  Sent PARAM_REQUEST_LIST to system {vehicle.target_system}, component {vehicle.target_component}")

def receive_param_value(vehicle, timeout=5.0):
    """接收参数值响应"""
    print(f"  Waiting for PARAM_VALUE ({timeout}s timeout)...")
    msg = vehicle.recv_match(type='PARAM_VALUE', blocking=True, timeout=timeout)

    if not msg:
        print("  Timeout: No PARAM_VALUE received")
        return None

    # 解析参数ID
    param_id = msg.param_id

    # 解析参数值
    param_value = msg.param_value

    # 参数类型映射
    type_names = {
        0: "INT8", 1: "UINT8", 2: "INT16", 3: "UINT16", 4: "INT32",
        5: "UINT32", 6: "INT64", 7: "UINT64", 8: "FLOAT", 9: "DOUBLE"
    }
    param_type = type_names.get(msg.param_type, f"UNKNOWN({msg.param_type})")

    print("\n  Received PARAM_VALUE:")
    print(f"    ID: {param_id}")
    print(f"    Value: {param_value}")
    print(f"    Type: {param_type}")
    print(f"    Index/Total: {msg.param_index}/{msg.param_count}")

    return {
        'id': param_id,
        'value': param_value,
        'type': param_type,
        'index': msg.param_index,
        'count': msg.param_count
    }
```

```
def receive_all_param_values(vehicle, timeout=10.0):
    """接收所有参数值，直到收到 param_count 条消息后退出循环"""
    received_params = []
    expected_count = None
    print(f"  Waiting for all PARAM_VALUE messages...")

    while True:
        msg = vehicle.recv_match(type='PARAM_VALUE', blocking=True,
timeout=timeout)

        if not msg:
            print(f"  Timeout: Received {len(received_params)} para
meters (expected {expected_count if expected_count else 'unknow
n'})")
            break

        param_id = msg.param_id
        param_value = msg.param_value

        # 参数类型映射
        type_names = {
            0: "INT8", 1: "UINT8", 2: "INT16", 3: "UINT16", 4: "INT
32",
            5: "UINT32", 6: "INT64", 7: "UINT64", 8: "FLOAT", 9: "D
OUBLE"
        }
        param_type = type_names.get(msg.param_type, f"UNKNOWN({msg.
param_type})")

        # 首次收到消息时记录总参数数量
        if expected_count is None:
            expected_count = msg.param_count
            print(f"  Total parameters to receive: {expected_coun
t}")

        print(f"  [{msg.param_index}/{expected_count}] {param_id} =
{param_value} ({param_type})")

        received_params.append({
            'id': param_id,
            'value': param_value,
            'type': param_type,
            'index': msg.param_index,
            'count': msg.param_count
```

```
        })

        # 当已接收数量达到预期总数时退出循环
        if len(received_params) >= expected_count:
            print(f"\n  Successfully received all {expected_count}
parameters")
            break

    return received_params

def main():
    print("  Starting MAVLink PARAM_SET Tool (UDP Listening Mode)")
    print("------------------------------------------------------")
    print("This script listens for MAVLink on ALL interfaces:1455
0")
    print("Ensure your vehicle is configured to send to this port")
    print("------------------------------------------------------
\n")

    vehicle = connect_to_vehicle()

    # 操作选择
    print("\nAvailable operations:")
    print("1. Set parameter value")
    print("2. Read parameter value (by ID or index)")
    print("3. Request all parameters (PARAM_REQUEST_LIST)")
    choice = input("\nSelect operation [1-3]: ").strip()

    if choice == '1':
        param_id = input("Enter parameter ID to set: ").strip()
        param_value = float(input("Enter new value: ").strip())

        print("\nParameter types:")
        print("  8 = FLOAT (most common)")
        print("  9 = DOUBLE")
        print("  4 = INT32")
        print("  1 = UINT8")
        param_type = int(input("Enter type number [8]: ").strip() o
r 8)

        send_param_set(vehicle, param_id, param_value, param_type)
        receive_param_value(vehicle)

    elif choice == '2':
        # mode = input("Read by [id/index]: ").lower()
```

```
        mode = 'id'
        if mode == 'id':
            param_id = input("Enter parameter ID: ").strip()
            send_param_request_read(vehicle, param_id=param_id)
            receive_param_value(vehicle)
        elif mode == 'index':
            param_index = int(input("Enter parameter index: "))
            send_param_request_read(vehicle, param_index=param_inde
x)
            receive_param_value(vehicle)
        else:
            print("Invalid mode. Use 'id' or 'index'")

    elif choice == '3':
        print("\n  Requesting all parameters...")
        send_param_request_list(vehicle)
        all_params = receive_all_param_values(vehicle)
        print(f"\n  Total parameters received: {len(all_params)}")

    else:
        print("Invalid choice. Select 1, 2, or 3.")


if __name__ == "__main__":
    main()
```

示例使用说明

获取

运行后，成功建立连接

```
🚀 Starting MAVLink PARAM_SET Tool (UDP Listening Mode)
--------------------------------------------------
This script listens for MAVLink on ALL interfaces:14550
Ensure your vehicle is configured to send to this port
--------------------------------------------------

Listening for MAVLink on: udpin:0.0.0.0:14550
Waiting for HEARTBEAT (timeout: 10s)...
✅ Heartbeat received from system 3, component 0

Available operations:
1. Set parameter value
2. Read parameter value (by ID or index)
3. Request all parameters (PARAM_REQUEST_LIST)
```

输入“2”选择读取参数

```
Select operation [1-3]: 2
```

输入对应参数“METER_MODE”

```
Enter parameter ID: METER_MODE
```

成功返回该参数对应的数值

```
📥 Received PARAM_VALUE:
  🆔 ID: METER_MODE
  🔢 Value: 1.0
```

设置

运行后，成功建立连接

```
🚀 Starting MAVLink PARAM_SET Tool (UDP Listening Mode)
------------------------------------------------------
This script listens for MAVLink on ALL interfaces:14550
Ensure your vehicle is configured to send to this port
------------------------------------------------------

Listening for MAVLink on: udpin:0.0.0.0:14550
Waiting for HEARTBEAT (timeout: 10s)...
✅ Heartbeat received from system 3, component 0

Available operations:
1. Set parameter value
2. Read parameter value (by ID or index)
3. Request all parameters (PARAM_REQUEST_LIST)
```

输入“1”选择设置参数

```
Select operation [1-3]: 1
```

输入对应参数“METER_MODE”

```
Enter parameter ID to set: METER_MODE
```

输入要设置的数值

```
Enter new value: 0
```

选择参数类型“1”

```
Parameter types:
  9 = DOUBLE
  4 = INT32
  1 = UINT8
Enter type number [8]: 1
```

成功返回设置值

```
📥 Received PARAM_VALUE:
  🆔 ID: METER_MODE
  🔢 Value: 0.0
```

MAV_CMD_DO_TRIGGER_CONTROL 203

单张拍摄

返回消息：COMMAND_ACK
返回值 command 203 执行命令id
返回值 result 0或4 0：执行成功，4：执行失败

示例

```
import time
from pymavlink import mavutil

timestamp = int(time.time() * 1000)
millis = int(time.time() * 1000)

master = mavutil.mavlink_connection('udpin:0.0.0.0:14550')
master.wait_heartbeat()
boot_time = time.time()

def shoot():
    master.mav.command_long_send(
        master.target_system,
        master.target_component,
        mavutil.mavlink.MAV_CMD_DO_DIGICAM_CONTROL,
        0,
        0, 0, 0, 0, 0, 0, 0
    )

shoot()
```

执行

```
PS D:\python> & C:\Users\Administrator\AppData\Local\Python\pythoncore-3.14-64\python.exe d:/python/tmp.py
PS D:\python>
```

结果

```
isShootPhoto is true
1666.048                user        [Info]      mavlink_connector.cpp:6275 切换拍照模式

1666.575                linker      [Warn]              dji_command.c:1025 <0x66ce>Command async send retry: index = 1, retryTimes = 3, 0x0A06->0x0100(
0x0001) 0x0001
1667.076                linker      [Warn]              dji_command.c:1025 <0x66ce>Command async send retry: index = 1, retryTimes = 2, 0x0A06->0x0100(
0x0001) 0x0001
1667.238                linker      [Warn]              dji_command.c:910  Received invalid ack,<0x66ce> 0x0100(0x0001)->0x0A06(0x00CA) 0x0001
1667.245                linker      [Warn]              dji_command.c:910  Received invalid ack,<0x66ce> 0x0100(0x0001)->0x0A06(0x00CA) 0x0001

SD卡总容量:60097MB  SD卡剩余容量:6485MB
1668.747                user        [Info]      mavlink_connector.cpp:6362 Mounted position 1 camera start to shoot photo
shoot photo..
```

MAV_CMD_VIDEO_STOP_CAPTURE 2501

返回command = MAV_CMD_REQUEST_MESSAGE
返回值 param1 2501 执行命令id

返回值 param2 0或1 是否执行成功
返回值 param3 失败代码
0 无异常
1 无法初始化相机控制
2 设置相机为录像模式
3 获取相机版本失败
4 相机录像执行失败

示例

```
import time
from pymavlink import mavutil

timestamp = int(time.time() * 1000)
millis = int(time.time() * 1000)

master = mavutil.mavlink_connection('udpin:0.0.0.0:14550')
master.wait_heartbeat()
boot_time = time.time()

def videoStop():
    master.mav.command_long_send(
        master.target_system,
        master.target_component,
        mavutil.mavlink.MAV_CMD_VIDEO_STOP_CAPTURE,
        0,
        0, 0, 0, 0, 0, 0, 0
    )

videoStop()
```

执行

```
PS D:\python> & C:\Users\Administrator\AppData\Local\Python\pythoncore-3.14-64\python.exe d:/python/tmp.py
PS D:\python>
```

结果

```
发送 CAMERA_CAPTURE_STATUS: video_status=0, recording_time_ms=0
video stop..
```

具体录像行为相关信息需用遥控器查看

MAV_CMD_VIDEO_STOP_CAPTURE 2501

返回command = MAV_CMD_REQUEST_MESSAGE
返回值 param1 2501 执行命令id

返回值 param2 0或1 是否执行成功
返回值 param3 失败代码
0 无异常
1 无法初始化相机控制
2 设置相机为录像模式
3 获取相机版本失败
4 相机录像执行失败

示例

```
import time
from pymavlink import mavutil

timestamp = int(time.time() * 1000)
millis = int(time.time() * 1000)

master = mavutil.mavlink_connection('udpin:0.0.0.0:14550')
master.wait_heartbeat()
boot_time = time.time()

def videoStop():
    master.mav.command_long_send(
        master.target_system,
        master.target_component,
        mavutil.mavlink.MAV_CMD_VIDEO_STOP_CAPTURE,
        0,
        0, 0, 0, 0, 0, 0, 0
    )

videoStop()
```

执行

```
PS D:\python> & C:\Users\Administrator\AppData\Local\Python\pythoncore-3.14-64\python.exe d:/python/tmp.py
PS D:\python>
```

结果

```
发送 CAMERA_CAPTURE_STATUS: video_status=0, recording_time_ms=0
video stop..
```

具体录像行为相关信息需用遥控器查看

飞行控制

mavlink id 11 SET_MODE

设置系统模式。由于该模式是针对整架飞机的，而非仅针对某个组件，因此不存在目标组件ID。

字段名	类型	单位	描
target_system	uint8_t		要
custom_mode	uint32_t	见px4 custom_mode	要

PX4 custom_mode

飞行模式	值
POSCTL	0x00030000
TAKEOFF	0x02040000
HOLD	0x03040000
MISSION	0x04040000
RTL	0x05040000
LAND	0x06040000

执行任务功能：确保已上传任务内容后，将此消息将custom_mode设为任务模式并发送给无人机。

MAV_CMD_NAV_TAKEOFF 22

返回消息：COMMAND_ACK
返回值 command 22 执行命令id
返回值 result 0或4 0：执行成功，4：执行失败

示例

```
import time
from pymavlink import mavutil

timestamp = int(time.time() * 1000)
millis = int(time.time() * 1000)

master = mavutil.mavlink_connection('udpin:0.0.0.0:14550')
master.wait_heartbeat()
boot_time = time.time()

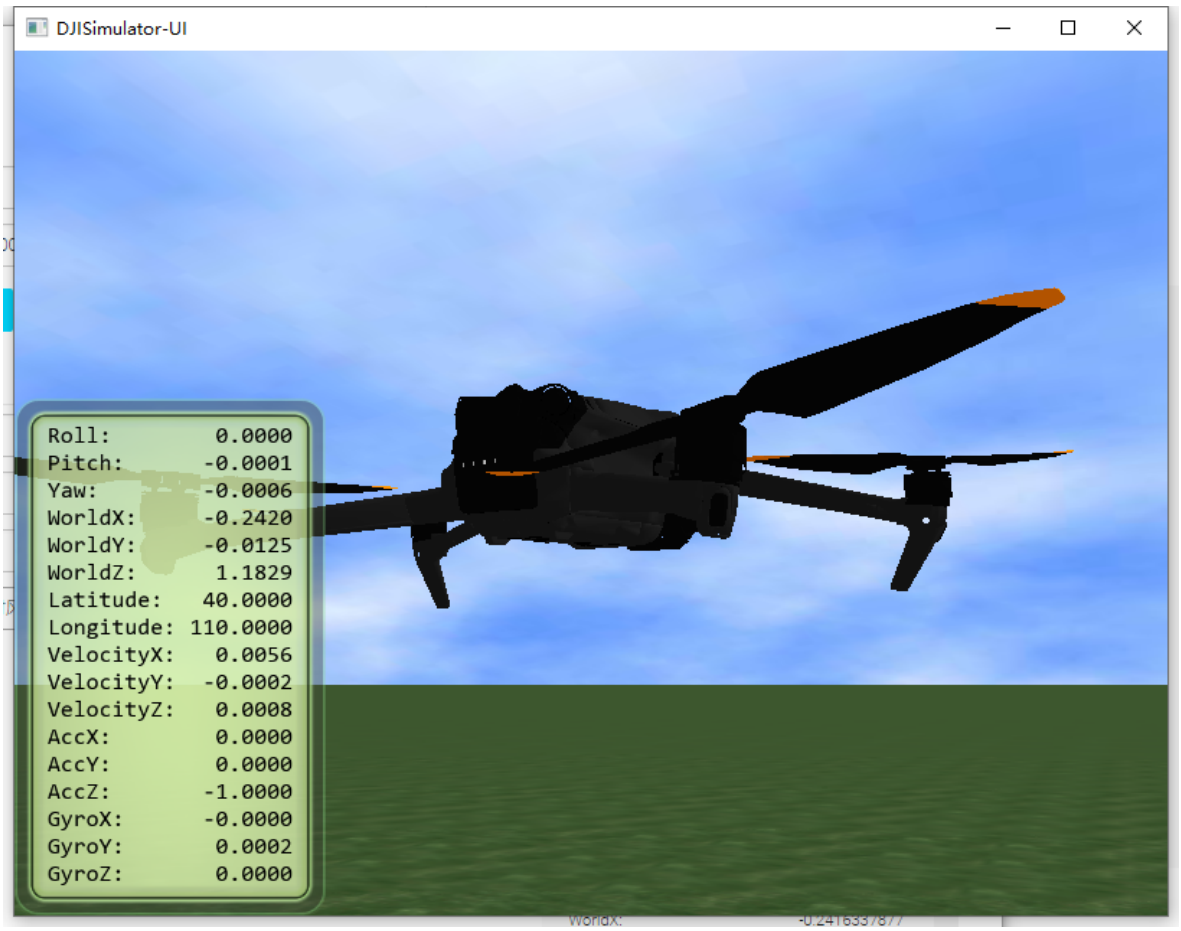
def ready():
    master.mav.command_long_send(
        master.target_system,
        master.target_component,
        mavutil.mavlink.MAV_CMD_NAV_TAKEOFF,
        0,
        0, 0, 0, 0, 0, 0, 0
    )

ready()
```

执行

```
PS D:\python> & C:\Users\Administrator\AppData\Local\Python\pythoncore-3.14-64\python.exe d:/python/tmp.py
PS D:\python>
```

结果

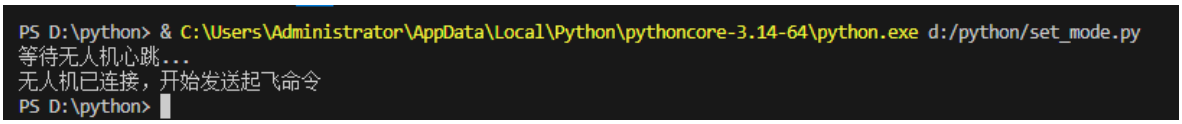


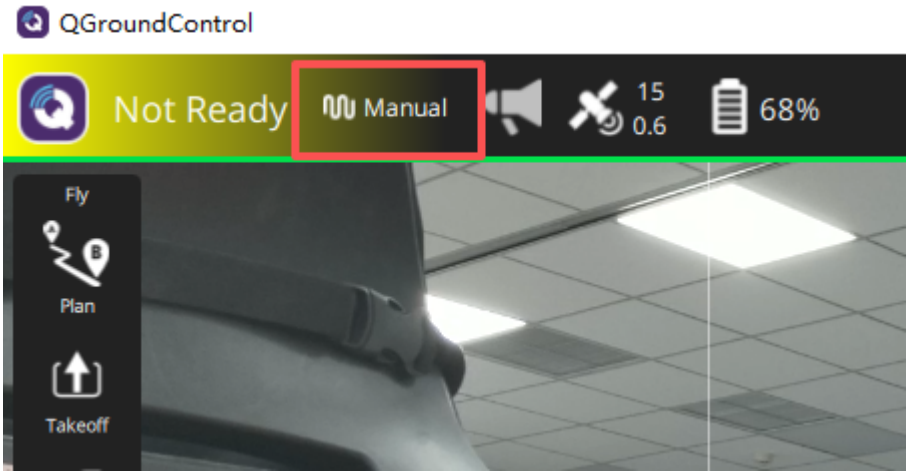
示例

```
import time
import sys
from pymavlink import mavutil
mode_map = {
    "MANUAL": 0x00010000,
    "POSCTL": 0x00030000,
    "AUTO_RTL": 0x05040000,
    "AUTO_MISSION": 0x04040000,
}# 仅列举部分模式
connection = mavutil.mavlink_connection('udpin:0.0.0.0:14550')
connection .wait_heartbeat()
target_system = 1
target_component = 1
custom_mode = mode_map["MANUAL"]  # 任务模式
# 发送SET_MODE消息
connection.mav.set_mode_send(
    target_system,
    target_component,
```

```
    custom_mode,
    0
)
```

执行结果





降落命令

方法1：发送以下降落命令

MAV_CMD_NAV_LAND 21

返回消息：COMMAND_ACK
返回值 command 21 执行命令id
返回值 result 0或4 0：执行成功，4：执行失败

方法2：使用切换飞行模式将飞行模式切换至LAND

详见切换飞行模式页

取消降落命令

方法1：使用切换飞行模式将飞行模式切换至POSCTL
详见切换飞行模式页

方法2：发送以下COMMAND_LONG命令

MAV_CMD_DO_REPOSITION 192

command = MAV_CMD_DO_REPOSITION
param4，param5，param6 = NaN
param7 = 0

返回消息：COMMAND_ACK
返回值 command 192 执行命令id
返回值 result 0或4 0：执行成功，4：执行失败

示例

发送降落命令（方法1）

```
import time
from pymavlink import mavutil

timestamp = int(time.time() * 1000)
millis = int(time.time() * 1000)

master = mavutil.mavlink_connection('udpin:0.0.0.0:14550')
master.wait_heartbeat()
boot_time = time.time()

def land():
    master.mav.command_long_send(
        master.target_system,
        master.target_component,
        mavutil.mavlink.MAV_CMD_NAV_LAND,
        0,
        0, 0, 0, 0, 0, 0, 0
    )

land()
```

发送取消降落命令（方法2）

```
import time
from pymavlink import mavutil
import math


def get_current_millis():
    """获取当前时间（毫秒级）"""
```

```
    return int(time.time() * 1000)


# MAVLink 命令常量
MAV_CMD_DO_REPOSITION = 192
DEFAULT_TIMEOUT = 5.0


def send_reposition_command(master, ground_speed, yaw, flags=0):
    """发送重新定位命令到无人机

    Args:
        master: MAVLink 连接对象
        ground_speed: 地速 (单位: m/s)
        yaw: 偏航角 (单位: 度, 0-360)
        flags: 标志位 (保留, 默认为0)
    """
    nan_value = math.nan
    send_time = get_current_millis()
    print(f"[{send_time}] 发送重新定位命令")
    print(f"  地速 = {ground_speed} m/s, 偏航角 = {yaw}°")

    master.mav.command_long_send(
        master.target_system,
        master.target_component,
        mavutil.mavlink.MAV_CMD_DO_REPOSITION,
        0,  # 确认标志
        ground_speed,    # param1: 地速 (单位: m/s)
        yaw,             # param2: 偏航角 (单位: 度)
        0,               # param3: 保留
        nan_value,       # param4: NaN
        nan_value,       # param5: NaN
        nan_value,       # param6: NaN
        0                # param7: 0
    )

    return send_time


def wait_for_ack(master, send_time, timeout=DEFAULT_TIMEOUT):
    """等待命令确认响应"""
    print("等待重新定位命令响应...")

    ack = master.recv_match(
        type='COMMAND_ACK',
```

```
        blocking=True,
        timeout=timeout
    )

    if ack:
        ack_time = get_current_millis()
        delay = ack_time - send_time
        print(f"[{ack_time}] 收到命令应答 (MAV_CMD_DO_REPOSITION)")
        print(f" 命令ID: {ack.command}, 结果: {ack.result}, 延迟: {de
lay}ms")
    else:
        print(f"[{get_current_millis()}] 超时：未收到重新定位命令应答")


def main():
    """主函数：连接无人机并发送重新定位命令"""
    # 连接无人机（UDP端口14550）
    master = mavutil.mavlink_connection('udpin:0.0.0.0:14550')
    print("等待无人机心跳...")
    master.wait_heartbeat()
    print("无人机已连接，开始发送重新定位命令")

    # 发送重新定位命令（示例参数，请根据实际情况修改）
    ground_speed = 5.0    # 地速 5 m/s
    yaw = 90.0            # 偏航角 90°（东向）
    send_time = send_reposition_command(master, ground_speed, yaw)

    # 等待响应
    wait_for_ack(master, send_time)

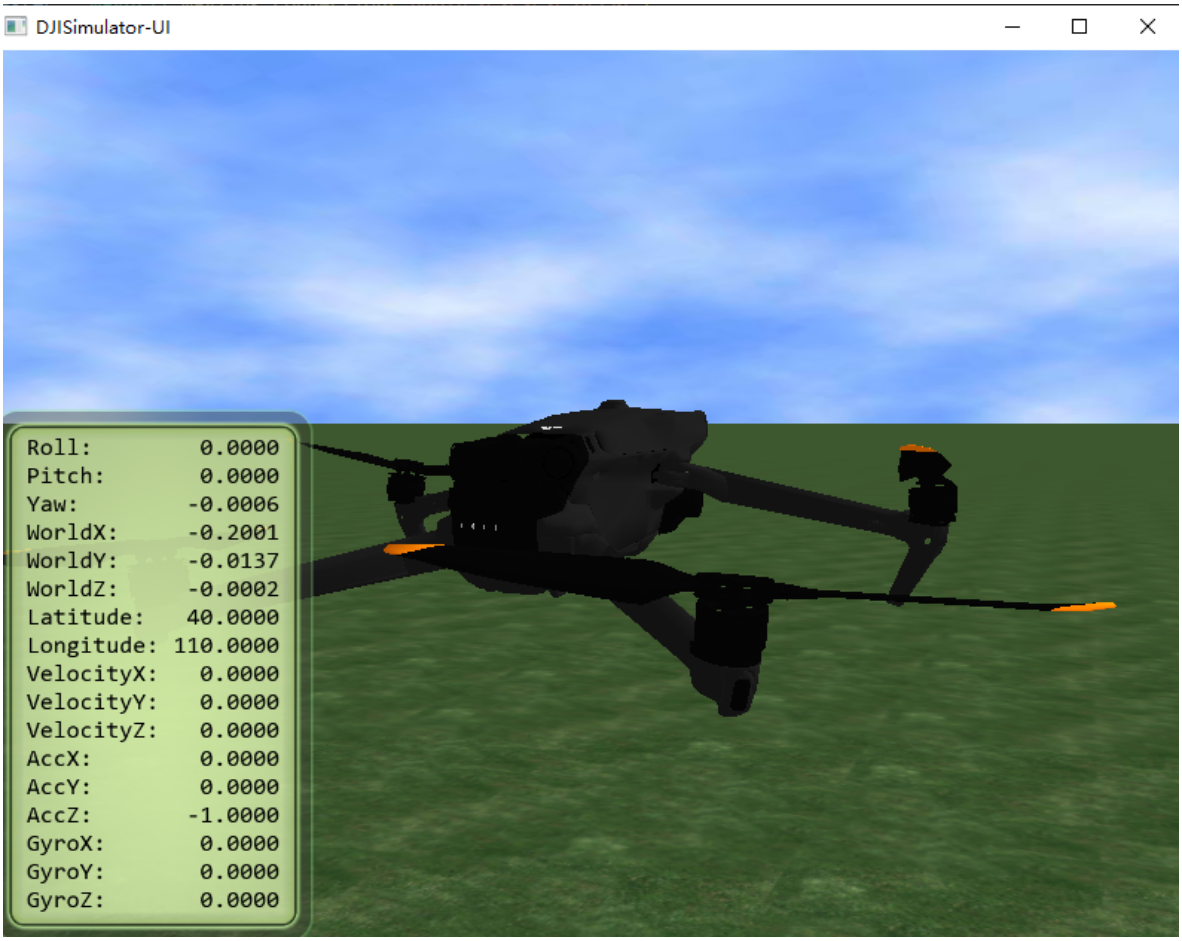
    print("重新定位命令发送完成")


if __name__ == "__main__":
    main()
```

执行

```
发送 CAMERA_CAPTURE_STATUS: video_status=0, recording_time_ms=0
video stop..
```

结果



返航命令

方法1：发送以下COMMAND_LONG命令

MAV_CMD_NAV_RETURN_TO_LAUNCH 20

param1 = 0 (0为返航) 非0为取消返航
返回消息 COMMAND_LONG
返回值 command = MAV_CMD_REQUEST_MESSAGE
返回值 param1 20 执行命令id
返回值 param2 0或1 是否执行成功
返回值 param3 失败代码
失败代码列表
0 无异常
1 无法初始化无人机飞行控制
2 返航失败
3 取消返航失败

方法2：使用切换飞行模式将飞行模式切换至RTL
详见切换飞行模式页

取消返航命令

方法1：发送以下COMMAND_LONG命令

MAV_CMD_NAV_RETURN_TO_LAUNCH 20

param1 = 1 (非0为取消返航)
返回消息同上

方法2：使用切换飞行模式将飞行模式切换至POSCTL
详见切换飞行模式页

方法3：

示例

```
import time
from pymavlink import mavutil

timestamp = int(time.time() * 1000)
millis = int(time.time() * 1000)

master = mavutil.mavlink_connection('udpin:0.0.0.0:14550')
master.wait_heartbeat()
boot_time = time.time()

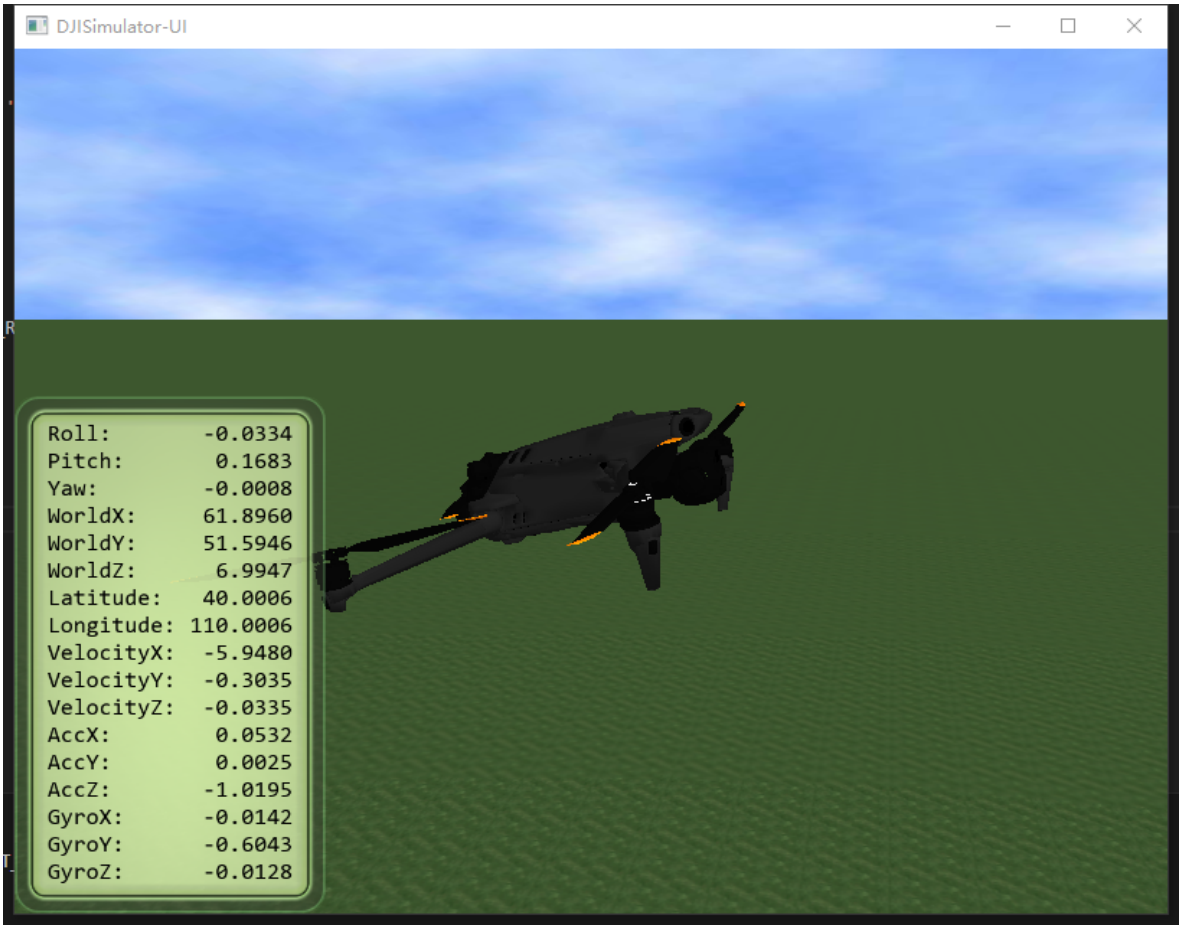
def return_to_base(cancleReturn):
    master.mav.command_long_send(
        master.target_system,
        master.target_component,
        mavutil.mavlink.MAV_CMD_NAV_RETURN_TO_LAUNCH,
        0,
        cancleReturn, 0, 0, 0, 0, 0, 0
    )

return_to_base(0)
time.sleep(10)
return_to_base(1)
```

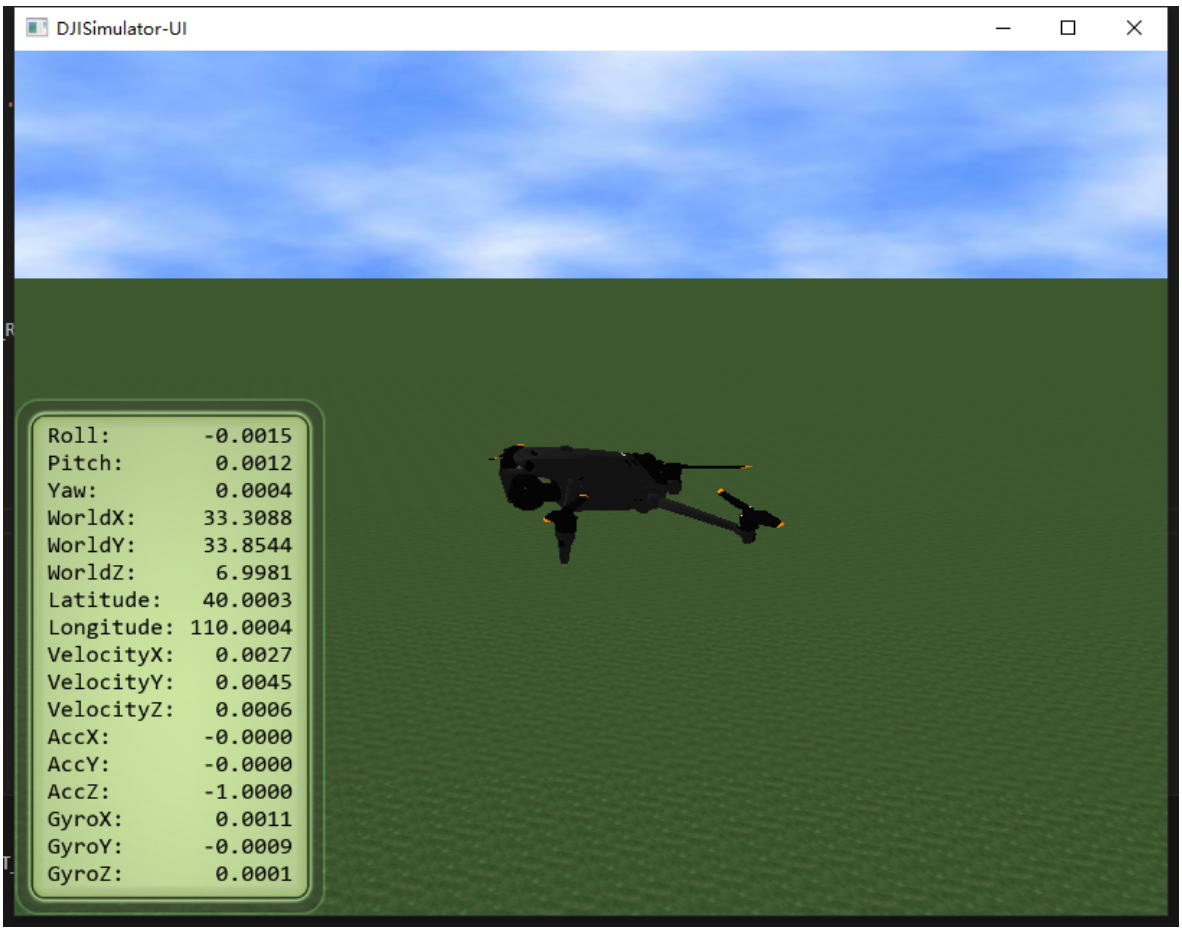
执行

```
PS D:\python> & C:\Users\Administrator\AppData\Local\Python\pythoncore-3.14-64\python.exe d:/python/tmp.py
PS D:\python>
```

开始返航



中途停止返航



GLOBAL_POSITION_INT 33

Field Name	Type	Units	Description
lat	int32_t	degE7	纬度int*10^7
lon	int32_t	degE7	经度int*10^7
alt	int32_t	mm	高度

返回值 param1 33 执行命令id
返回值 param2 0或1 是否执行成功
返回值 param3 失败代码

失败代码列表
0 无异常
1 无法初始化无人机飞行控制
2 获取无人机控制权失败
3 创建飞行线程失败

当到达目标点后，或通过**手动控制**传入非摇杆中立值的参数，导致无人机自动切换到POSCTL模式后，将自动退出经纬度位置控制。

示例

```
import time
from pymavlink import mavutil

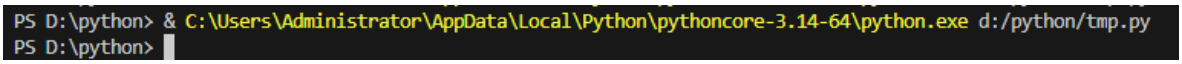
timestamp = int(time.time() * 1000)
millis = int(time.time() * 1000)

master = mavutil.mavlink_connection('udpin:0.0.0.0:14550')
master.wait_heartbeat()

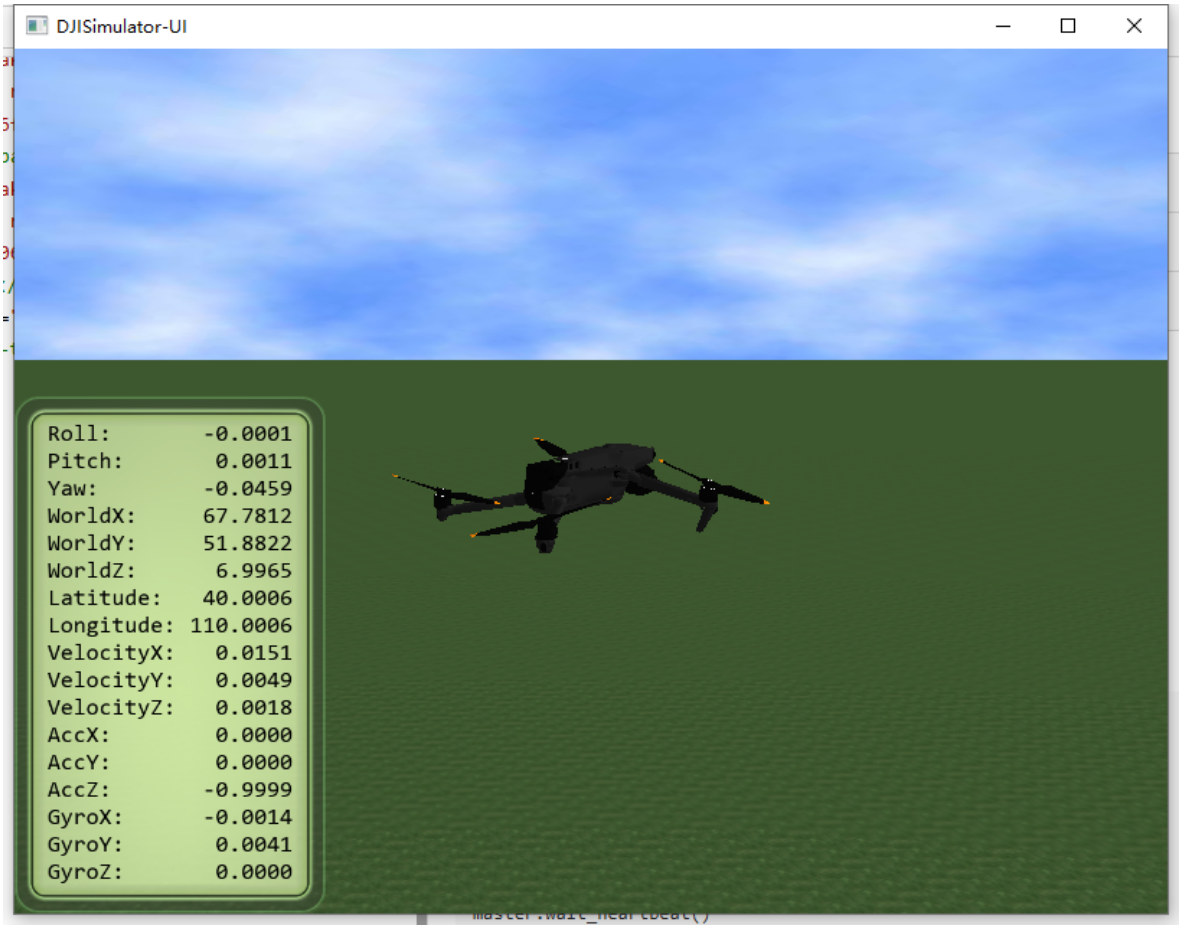
boot_time = time.time()
def fly():
    master.mav.global_position_int_send(
        int(1e3 * (time.time() - boot_time)),  # ms since boot
        lat=400006100, lon=1100006100, alt=700,
        relative_alt=0, vx=0, vy=0, vz=0, hdg=0
    )

fly()
```

执行



结果



MANUAL_CONTROL (69)

Field Name	Type	Description
target	uint8_t	控制模式选择，0、1: x,y,z轴为速度控制，r轴为偏航角速率；2：x，y轴为姿态角控制，z轴为上升速度，r轴为偏航角速率。
x	int16_t	X轴，范围已归一化为[-1000,1000]。 INT16_MAX的值表示该轴无效。对应于操纵杆的前进（1000），后退（-1000）移动以及飞行器的俯仰。
y	int16_t	Y轴，范围已归一化为[-1000,1000]。 INT16_MAX的值表示该轴无效。对应于操纵杆的左移（-1000），右移（1000）以及飞行器的滚转。
z	int16_t	Z轴，范围已归一化为[0,1000]。INT16_MAX的值表示该轴无效。对应于飞行器的推力。摇杆中立时值为500，大于500表示上升，小于于500表示下降。

r	int16_t	R轴，范围已归一化为[-1000,1000]。 INT16_MAX的值表示该轴无效。对应于飞行器的偏航，逆时针旋转（-1000），顺时针旋转（1000）。

target设置为0时摇杆轴取值的缩放：
x为control.x /200
y为control.y /200
z为(control.z /100 -5) / 2
r为control.r /16
target设置为1时摇杆轴取值的缩放：
x为control.x /50
y为control.y /50
z为(control.z - 500) /50
r为control.r /10
target设置为时摇杆轴取值的缩放：
x为control.x /10
y为control.y /10
z为control.z /10
r为control.r /10

当无人机为Orbit模式时，轴取值含义改为下表

X轴	更改盘旋半径，缩小半径（1000），增大半径（-1000）。最小半径是 1 米。 最大半径是 100 米。
Y轴	更改盘旋速度。与当前盘旋方向相同时则增加速度，反之则减少速度。 速度小于0时将改变盘旋方向。最大速度为 10 m/s， 进一步的限制是将向心加速度保持在 2 m/s^2 以下。
Z轴	更改盘旋时的高度。摇杆中立时值为500，大于500表示上升， 小于于500表示下降。

```
# Import mavutil
import time
from pymavlink import mavutil

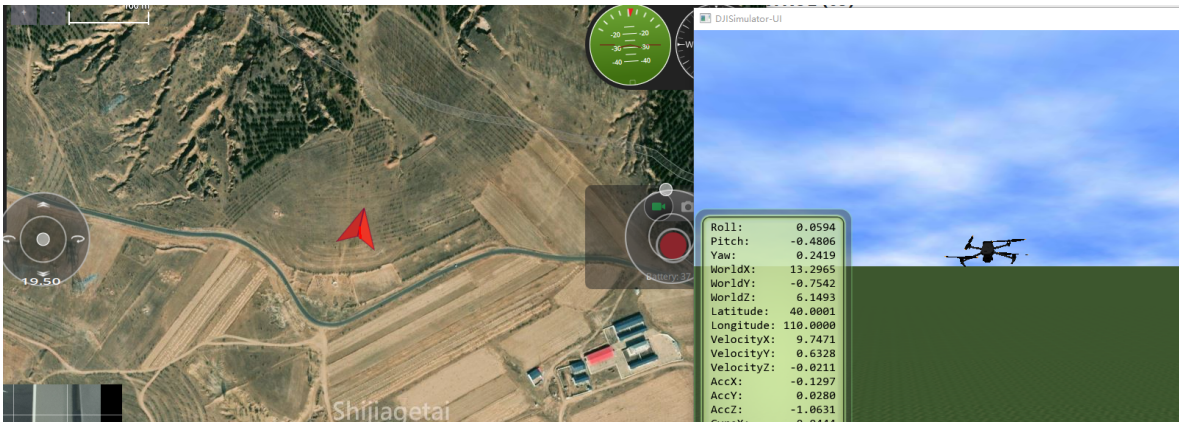
master = mavutil.mavlink_connection('udpin:0.0.0.0:14550')
master.wait_heartbeat()

def ready(x,y,z,r):
    master.mav.manual_control_send(
        1,
```

```
        x,
        y,
        z,
        r,
        0)


x = 0
while x < 1000:
    ready(10,10,10,10)
    x += 1
    time.sleep(0.05)
```

使用QGC虚拟摇杆发送操控指令遥控无人机运动



根据命令设置不同飞行状态以及目标

SET_POSITION_TARGET_LOCAL_NED 84

Field Name	Type	Units	Control Type	Descript
time_boot_ms	uint32_t	ms		Timestar boot).
target_system	uint8_t			System I
target_component	uint8_t			Compon
coordinate_frame	uint8_t		MAV_FRAME	坐标系配 支持机体
type_mask	uint16_t		POSITION_TARGET_TYPEMASK	Bitmap决
x	float	m		X坐标(N
y	float	m		Y坐标(N
z	float	m		Z坐标(NI
vx	float	m/s		X速度(N
vy	float	m/s		Y速度(N
vz	float	m/s		Z速度(NI
afx	float	m/s/s		0
afy	float	m/s/s		0
afz	float	m/s/s		0
yaw	float	rad		角度
yaw_rate	float	rad/s		角速率

type_mask参数说明

Value	Name	De
1	POSITION_TARGET_TYPEMASK_X_IGNORE	Ign
2	POSITION_TARGET_TYPEMASK_Y_IGNORE	Ign
4	POSITION_TARGET_TYPEMASK_Z_IGNORE	Ign

8	POSITION_TARGET_TYPEMASK_VX_IGNORE	Ign
16	POSITION_TARGET_TYPEMASK_VY_IGNORE	Ign
32	POSITION_TARGET_TYPEMASK_VZ_IGNORE	Ign
64	POSITION_TARGET_TYPEMASK_AX_IGNORE	Ign
128	POSITION_TARGET_TYPEMASK_AY_IGNORE	Ign
256	POSITION_TARGET_TYPEMASK_AZ_IGNORE	Ign
512	POSITION_TARGET_TYPEMASK_FORCE_SET	Us ac
1024	POSITION_TARGET_TYPEMASK_YAW_IGNORE	Ign
2048	POSITION_TARGET_TYPEMASK_YAW_RATE_IGNORE	Ign

当不忽略x,y,z时 此命令为位置飞行
当不忽略vx,vy,vz时 此命令为速度飞行
当不忽略yaw时 此命令为角度设置
当不忽略yaw_rate时 此命令为角速率命令

coordinate_frame 为机体坐标系参数
目前设置为两个参数
MAV_FRAME_LOCAL_NED 大地坐标系(NED坐标)
MAV_FRAME_BODY_FRD 机体坐标系(遵循右手定则)

返回command = MAV_CMD_REQUEST_MESSAGE
返回值 param1 84 执行命令id
返回值 param2 0或1 是否执行成功
返回值 param3 失败代码
失败代码列表
0 无异常
1 无法初始化无人机飞行控制
2 获取无人机控制权失败
3 未知坐标系
4 销毁FRU坐标系移动线程失败
5 销毁NED坐标系移动线程失败
6 创建FRU坐标系移动线程失败
7 创建NED坐标系移动线程失败

示例

```
import time
from pymavlink import mavutil

timestamp = int(time.time() * 1000)
millis = int(time.time() * 1000)

master = mavutil.mavlink_connection('udpin:0.0.0.0:14550')
master.wait_heartbeat()
boot_time = time.time()


def ready():
    master.mav.command_long_send(
        master.target_system,
        master.target_component,
        mavutil.mavlink.MAV_CMD_NAV_TAKEOFF,
        0,
        0, 0, 0, 0, 0, 0, 0
    )


def fly():
    # test pos mod use ned frame
    print('test pos mod use ned frame')
    master.mav.set_position_target_local_ned_send(
        int(1e3 * (time.time() - boot_time)),  # ms since boot
        master.target_system,
        master.target_component,
        coordinate_frame=mavutil.mavlink.MAV_FRAME_LOCAL_NED,
        type_mask=(
            # mavutil.mavlink.POSITION_TARGET_TYPEMASK_X_IGNORE |
            # mavutil.mavlink.POSITION_TARGET_TYPEMASK_Y_IGNORE |
            # mavutil.mavlink.POSITION_TARGET_TYPEMASK_Z_IGNORE |
            mavutil.mavlink.POSITION_TARGET_TYPEMASK_VX_IGNORE |
            mavutil.mavlink.POSITION_TARGET_TYPEMASK_VY_IGNORE |
            mavutil.mavlink.POSITION_TARGET_TYPEMASK_VZ_IGNORE |
            mavutil.mavlink.POSITION_TARGET_TYPEMASK_AX_IGNORE |
            mavutil.mavlink.POSITION_TARGET_TYPEMASK_AY_IGNORE |
            mavutil.mavlink.POSITION_TARGET_TYPEMASK_AZ_IGNORE |
            mavutil.mavlink.POSITION_TARGET_TYPEMASK_FORCE_SET |
            mavutil.mavlink.POSITION_TARGET_TYPEMASK_YAW_IGNORE |
            mavutil.mavlink.POSITION_TARGET_TYPEMASK_YAW_RATE_IGNOR
E
        ),
        x=10, y=10, z=-10,
```

```
        vx=0, vy=0, vz=0,
        afx=0, afy=0, afz=0,
        yaw=0, yaw_rate=0
    )

    # test yaw rate mod
    time.sleep(10)
    print('test yaw rate mod')
    master.mav.set_position_target_local_ned_send(
        int(1e3 * (time.time() - boot_time)),  # ms since boot
        master.target_system,
        master.target_component,
        coordinate_frame=mavutil.mavlink.MAV_FRAME_LOCAL_NED,
        type_mask=(
            mavutil.mavlink.POSITION_TARGET_TYPEMASK_X_IGNORE |
            mavutil.mavlink.POSITION_TARGET_TYPEMASK_Y_IGNORE |
            mavutil.mavlink.POSITION_TARGET_TYPEMASK_Z_IGNORE |
            mavutil.mavlink.POSITION_TARGET_TYPEMASK_VX_IGNORE |
            mavutil.mavlink.POSITION_TARGET_TYPEMASK_VY_IGNORE |
            mavutil.mavlink.POSITION_TARGET_TYPEMASK_VZ_IGNORE |
            mavutil.mavlink.POSITION_TARGET_TYPEMASK_AX_IGNORE |
            mavutil.mavlink.POSITION_TARGET_TYPEMASK_AY_IGNORE |
            mavutil.mavlink.POSITION_TARGET_TYPEMASK_AZ_IGNORE |
            mavutil.mavlink.POSITION_TARGET_TYPEMASK_FORCE_SET |
            mavutil.mavlink.POSITION_TARGET_TYPEMASK_YAW_IGNORE
            # mavutil.mavlink.POSITION_TARGET_TYPEMASK_YAW_RATE_IGN
ORE
        ),
        x=0, y=0, z=0,
        vx=0, vy=0, vz=0,
        afx=0, afy=0, afz=0,
        yaw=0, yaw_rate=90
    )

    # test velocity mod fru mod
    time.sleep(10)
    print('test velocity mod fru mod')
    x = 0
    while x < 200:
        master.mav.set_position_target_local_ned_send(
            int(1e3 * (time.time() - boot_time)),  # ms since boot
            master.target_system,
            master.target_component,
            coordinate_frame=mavutil.mavlink.MAV_FRAME_BODY_FRD,
            type_mask=(
```

```
                mavutil.mavlink.POSITION_TARGET_TYPEMASK_X_IGNORE |
                mavutil.mavlink.POSITION_TARGET_TYPEMASK_Y_IGNORE |
                mavutil.mavlink.POSITION_TARGET_TYPEMASK_Z_IGNORE |
                # mavutil.mavlink.POSITION_TARGET_TYPEMASK_VX_IGNOR
E |
                # mavutil.mavlink.POSITION_TARGET_TYPEMASK_VY_IGNOR
E |
                # mavutil.mavlink.POSITION_TARGET_TYPEMASK_VZ_IGNOR
E |
                mavutil.mavlink.POSITION_TARGET_TYPEMASK_AX_IGNORE
|
                mavutil.mavlink.POSITION_TARGET_TYPEMASK_AY_IGNORE
|
                mavutil.mavlink.POSITION_TARGET_TYPEMASK_AZ_IGNORE
|
                mavutil.mavlink.POSITION_TARGET_TYPEMASK_FORCE_SET
|
                mavutil.mavlink.POSITION_TARGET_TYPEMASK_YAW_IGNORE
|
                mavutil.mavlink.POSITION_TARGET_TYPEMASK_YAW_RATE_I
GNORE
            ),
            x=0, y=0, z=0,
            vx=5, vy=0, vz=-5,
            afx=0, afy=0, afz=0,
            yaw=0, yaw_rate=0)
        x += 1
        time.sleep(0.25)
        print('----------------' + str(x))

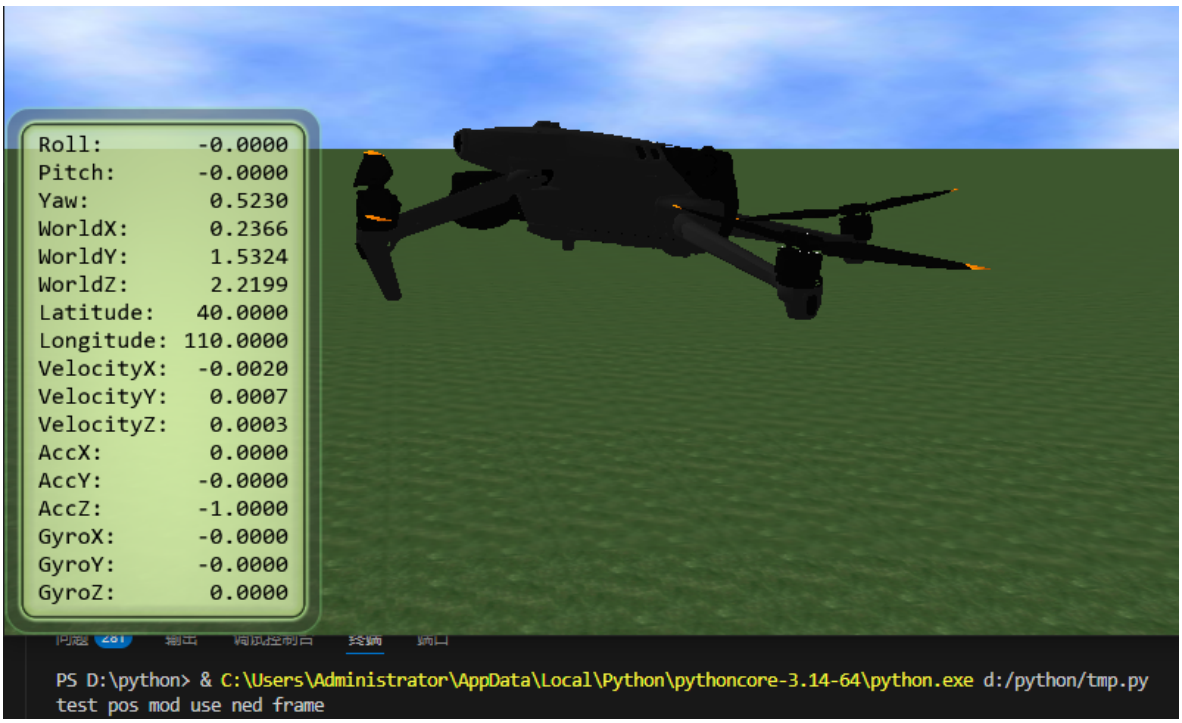
    # test yaw mod
    time.sleep(10)
    print('test yaw mod')
    master.mav.set_position_target_local_ned_send(
        int(1e3 * (time.time() - boot_time)),  # ms since boot
        master.target_system,
        master.target_component,
        coordinate_frame=mavutil.mavlink.MAV_FRAME_LOCAL_NED,
        type_mask=(
            mavutil.mavlink.POSITION_TARGET_TYPEMASK_X_IGNORE |
            mavutil.mavlink.POSITION_TARGET_TYPEMASK_Y_IGNORE |
            mavutil.mavlink.POSITION_TARGET_TYPEMASK_Z_IGNORE |
            mavutil.mavlink.POSITION_TARGET_TYPEMASK_VX_IGNORE |
            mavutil.mavlink.POSITION_TARGET_TYPEMASK_VY_IGNORE |
            mavutil.mavlink.POSITION_TARGET_TYPEMASK_VZ_IGNORE |
```

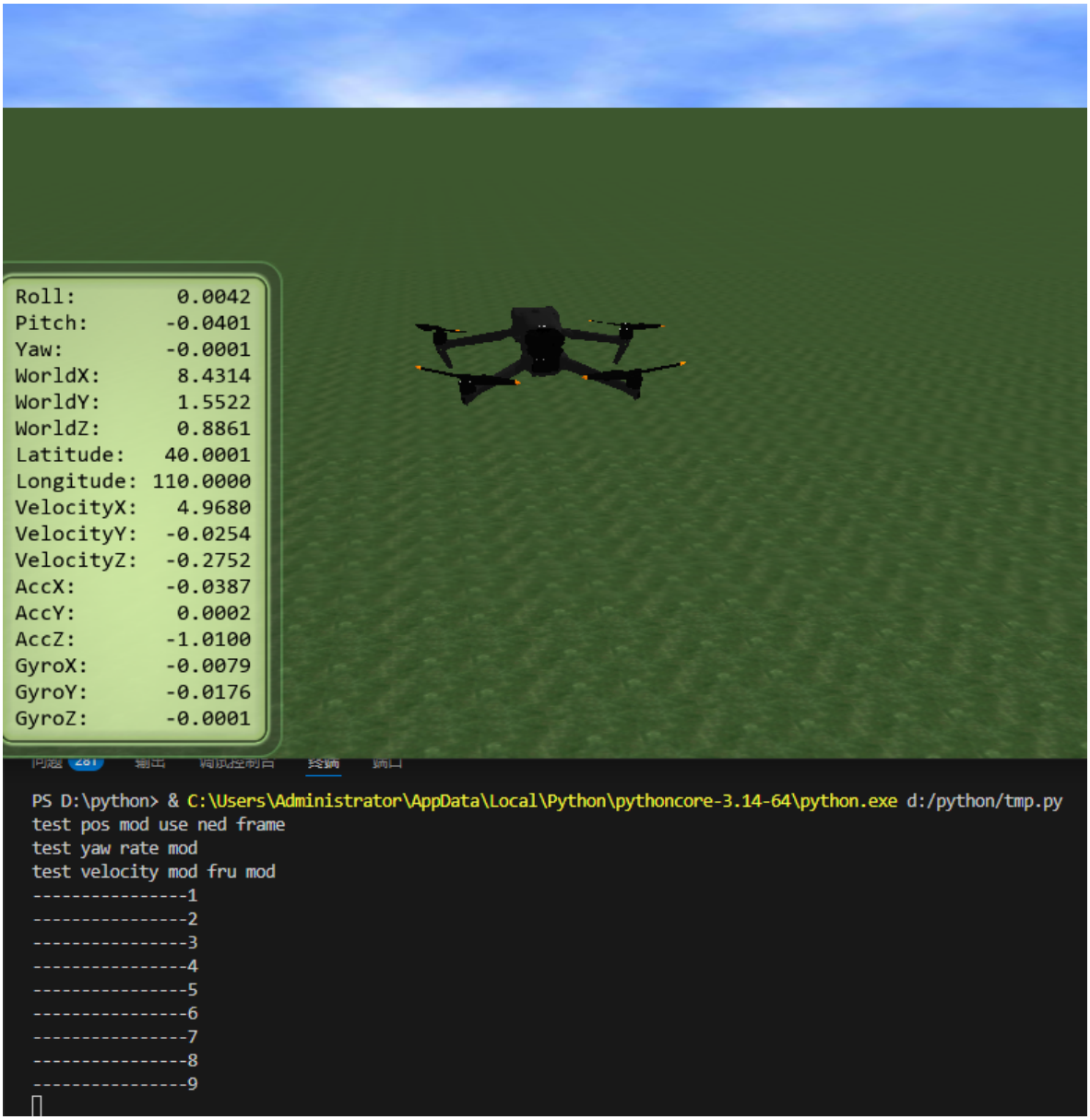
```
            mavutil.mavlink.POSITION_TARGET_TYPEMASK_AX_IGNORE |
            mavutil.mavlink.POSITION_TARGET_TYPEMASK_AY_IGNORE |
            mavutil.mavlink.POSITION_TARGET_TYPEMASK_AZ_IGNORE |
            mavutil.mavlink.POSITION_TARGET_TYPEMASK_FORCE_SET |
            # mavutil.mavlink.POSITION_TARGET_TYPEMASK_YAW_IGNORE |
            mavutil.mavlink.POSITION_TARGET_TYPEMASK_YAW_RATE_IGNOR
E
        ),
        x=0, y=0, z=0,
        vx=0, vy=0, vz=0,
        afx=0, afy=0, afz=0,
        yaw=-110, yaw_rate=0
    )


def land():
    master.mav.command_long_send(
        master.target_system,
        master.target_component,
        mavutil.mavlink.MAV_CMD_NAV_LAND,
        0,
        0, 0, 0, 0, 0, 0, 0
    )


ready()
time.sleep(10)
fly()
time.sleep(10)
land()
```

执行结果


Roll: -0.0000
Pitch: -0.0000
Yaw: 0.5230
WorldX: 0.2366
WorldY: 1.5324
WorldZ: 2.2199
Latitude: 40.0000
Longitude: 110.0000
VelocityX: -0.0020
VelocityY: 0.0007
VelocityZ: 0.0003
AccX: 0.0000
AccY: -0.0000
AccZ: -1.0000
GyroX: -0.0000
GyroY: -0.0000
GyroZ: 0.0000
PS D:\python> & C:\Users\Administrator\AppData\Local\Python\pythoncore-3.14-64\python.exe d:/python/tmp.py
test pos mod use ned frame



飞控全向避障行为

ALL_AVOID_ENABL

设置

mavlink消息：

接收PARAM_SET
发送PARAM_VALUE

获取

mavlink消息

接收PARAM_REQUEST_READ
发送PARAM_VALUE

参数：

枚举值	值	说明
DJI_FLIGHT_CONTROLLER_DISABLE_ALL_AVIOD	0	禁用全部避障功能
DJI_FLIGHT_CONTROLLER_ENABLE_ALL_AVIOD	1	启用全部避障功能

示例

```
import time
import sys
from pymavlink import mavutil

def connect_to_vehicle():
    """建立 UDP 监听连接（默认监听所有接口的 14550 端口）"""
    connection_string = "udpin:0.0.0.0:14550"
    print(f"Listening for MAVLink on: {connection_string}")
    vehicle = mavutil.mavlink_connection(connection_string)
    print("Waiting for HEARTBEAT (timeout: 10s)...")
    try:
        vehicle.wait_heartbeat(timeout=10)
        print(f"  Heartbeat received from system {vehicle.target_sy
stem}, component {vehicle.target_component}")
        return vehicle
    except Exception as e:
        print(f"  Connection failed: {str(e)}")
        print("Ensure your vehicle is sending MAVLink messages to 1
4550 port")
        print("Example QGroundControl setup: Settings > General > M
AVLink > Listening Port = 14550")
        sys.exit(1)

def format_param_id(param_id):
    """格式化16字节参数ID（符合MAVLink协议要求）"""
    if len(param_id) > 16:
        raise ValueError("param_id cannot exceed 16 characters")

    param_id_bytes = bytearray(16)
    for i in range(min(len(param_id), 16)):
        param_id_bytes[i] = ord(param_id[i])
```

```
    # 长度<16时添加NULL终止符
    if len(param_id) < 16:
        param_id_bytes[len(param_id)] = 0

    return param_id_bytes

def send_param_set(vehicle, param_id, param_value, param_type):
    """发送参数设置请求"""
    vehicle.mav.param_set_send(
        vehicle.target_system,
        vehicle.target_component,
        format_param_id(param_id),
        param_value,
        param_type
    )
    print(f"  Sent PARAM_SET for '{param_id}' = {param_value} (typ
e: {param_type})")

def send_param_request_read(vehicle, param_id=None, param_index=-
1):
    """发送参数读取请求"""
    param_id_bytes = format_param_id(param_id) if param_id else byt
earray(16)

    vehicle.mav.param_request_read_send(
        vehicle.target_system,
        vehicle.target_component,
        param_id_bytes,
        param_index
    )

    target = f"index {param_index}" if param_index >= 0 else f"ID
'{param_id}'"
    print(f"  Sent PARAM_REQUEST_READ for {target}")

def send_param_request_list(vehicle):
    """发送请求所有参数列表"""
    vehicle.mav.param_request_list_send(
        vehicle.target_system,
        vehicle.target_component
    )
    print(f"  Sent PARAM_REQUEST_LIST to system {vehicle.target_sys
tem}, component {vehicle.target_component}")

def receive_param_value(vehicle, timeout=5.0):
```

```
    """接收参数值响应"""
    print(f"  Waiting for PARAM_VALUE ({timeout}s timeout)...")
    msg = vehicle.recv_match(type='PARAM_VALUE', blocking=True, tim
eout=timeout)

    if not msg:
        print("  Timeout: No PARAM_VALUE received")
        return None

    # 解析参数ID
    param_id = msg.param_id

    # 解析参数值
    param_value = msg.param_value

    # 参数类型映射
    type_names = {
        0: "INT8", 1: "UINT8", 2: "INT16", 3: "UINT16", 4: "INT32",
        5: "UINT32", 6: "INT64", 7: "UINT64", 8: "FLOAT", 9: "DOUBL
E"
    }
    param_type = type_names.get(msg.param_type, f"UNKNOWN({msg.para
m_type})")

    print("\n  Received PARAM_VALUE:")
    print(f"    ID: {param_id}")
    print(f"    Value: {param_value}")
    print(f"    Type: {param_type}")
    print(f"    Index/Total: {msg.param_index}/{msg.param_count}")

    return {
        'id': param_id,
        'value': param_value,
        'type': param_type,
        'index': msg.param_index,
        'count': msg.param_count
    }

def receive_all_param_values(vehicle, timeout=10.0):
    """接收所有参数值，直到收到 param_count 条消息后退出循环"""
    received_params = []
    expected_count = None
    print(f"  Waiting for all PARAM_VALUE messages...")

    while True:
```

```
        msg = vehicle.recv_match(type='PARAM_VALUE', blocking=True,
timeout=timeout)

        if not msg:
            print(f"  Timeout: Received {len(received_params)} para
meters (expected {expected_count if expected_count else 'unknow
n'})")
            break

        param_id = msg.param_id
        param_value = msg.param_value

        # 参数类型映射
        type_names = {
            0: "INT8", 1: "UINT8", 2: "INT16", 3: "UINT16", 4: "INT
32",
            5: "UINT32", 6: "INT64", 7: "UINT64", 8: "FLOAT", 9: "D
OUBLE"
        }
        param_type = type_names.get(msg.param_type, f"UNKNOWN({msg.
param_type})")

        # 首次收到消息时记录总参数数量
        if expected_count is None:
            expected_count = msg.param_count
            print(f"  Total parameters to receive: {expected_coun
t}")

        print(f"  [{msg.param_index}/{expected_count}] {param_id} =
{param_value} ({param_type})")

        received_params.append({
            'id': param_id,
            'value': param_value,
            'type': param_type,
            'index': msg.param_index,
            'count': msg.param_count
        })

        # 当已接收数量达到预期总数时退出循环
        if len(received_params) >= expected_count:
            print(f"\n  Successfully received all {expected_count}
parameters")
            break
```

```
    return received_params

def main():
    print("  Starting MAVLink PARAM_SET Tool (UDP Listening Mode)")
    print("-------------------------------------------------------")
    print("This script listens for MAVLink on ALL interfaces:1455
0")
    print("Ensure your vehicle is configured to send to this port")
    print("-------------------------------------------------------
\n")

    vehicle = connect_to_vehicle()

    # 操作选择
    print("\nAvailable operations:")
    print("1. Set parameter value")
    print("2. Read parameter value (by ID or index)")
    print("3. Request all parameters (PARAM_REQUEST_LIST)")
    choice = input("\nSelect operation [1-3]: ").strip()

    if choice == '1':
        param_id = input("Enter parameter ID to set: ").strip()
        param_value = float(input("Enter new value: ").strip())

        print("\nParameter types:")
        print("  8 = FLOAT (most common)")
        print("  9 = DOUBLE")
        print("  4 = INT32")
        print("  1 = UINT8")
        param_type = int(input("Enter type number [8]: ").strip() o
r 8)

        send_param_set(vehicle, param_id, param_value, param_type)
        receive_param_value(vehicle)

    elif choice == '2':
        # mode = input("Read by [id/index]: ").lower()
        mode = 'id'
        if mode == 'id':
            param_id = input("Enter parameter ID: ").strip()
            send_param_request_read(vehicle, param_id=param_id)
            receive_param_value(vehicle)
        elif mode == 'index':
            param_index = int(input("Enter parameter index: "))
            send_param_request_read(vehicle, param_index=param_inde
```

```
x)
            receive_param_value(vehicle)
        else:
            print("Invalid mode. Use 'id' or 'index'")

    elif choice == '3':
        print("\n  Requesting all parameters...")
        send_param_request_list(vehicle)
        all_params = receive_all_param_values(vehicle)
        print(f"\n  Total parameters received: {len(all_params)}")

    else:
        print("Invalid choice. Select 1, 2, or 3.")


if __name__ == "__main__":
    main()
```

示例使用说明

获取

运行后，成功建立连接

```
🚀 Starting MAVLink PARAM_SET Tool (UDP Listening Mode)
------------------------------------------------------
This script listens for MAVLink on ALL interfaces:14550
Ensure your vehicle is configured to send to this port
------------------------------------------------------

Listening for MAVLink on: udpin:0.0.0.0:14550
Waiting for HEARTBEAT (timeout: 10s)...
✅ Heartbeat received from system 3, component 0

Available operations:
1. Set parameter value
2. Read parameter value (by ID or index)
3. Request all parameters (PARAM_REQUEST_LIST)
```

输入“2”选择读取参数

```
Select operation [1-3]: 2
```

输入对应参数“ALL_AVOID_ENABL”

```
Enter parameter ID: ALL_AVOID_ENABL
```

成功返回该参数对应的数值

```
📩 Received PARAM_VALUE:
  🆔 ID: ALL_AVOID_ENABL
  🔢 Value: 0.0
```

设置

运行后，成功建立连接

```
🚀 Starting MAVLink PARAM_SET Tool (UDP Listening Mode)
--------------------------------------------------
This script listens for MAVLink on ALL interfaces:14550
Ensure your vehicle is configured to send to this port
--------------------------------------------------

Listening for MAVLink on: udpin:0.0.0.0:14550
Waiting for HEARTBEAT (timeout: 10s)...
✅ Heartbeat received from system 3, component 0

Available operations:
1. Set parameter value
2. Read parameter value (by ID or index)
3. Request all parameters (PARAM_REQUEST_LIST)
```

输入“1”选择设置参数

```
Select operation [1-3]: 1
```

输入对应参数“ALL_AVOID_ENABL”

```
Enter parameter ID to set: ALL_AVOID_ENABL
```

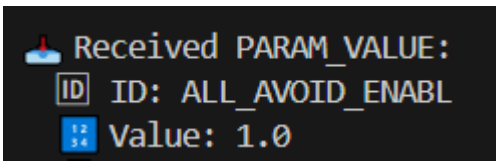
输入要设置的数值

```
Enter new value: 1
```

选择参数类型“1”

```
Parameter types:
  9 = DOUBLE
  4 = INT32
  1 = UINT8
Enter type number [8]: 1
```

成功返回设置值



最大速度

MAX_VELOCITY

设置

mavlink消息：

接收PARAM_SET
发送PARAM_VALUE

获取

mavlink消息

接收PARAM_REQUEST_READ
发送PARAM_VALUE

参数：

类型	取值范围	单位	说明
uint8_t	1 ~ 15	m/s	最大飞行速度

示例

```
import time
import sys
from pymavlink import mavutil

def connect_to_vehicle():
    """建立 UDP 监听连接（默认监听所有接口的 14550 端口）"""
    connection_string = "udpin:0.0.0.0:14550"
    print(f"Listening for MAVLink on: {connection_string}")
    vehicle = mavutil.mavlink_connection(connection_string)
    print("Waiting for HEARTBEAT (timeout: 10s)...")
    try:
        vehicle.wait_heartbeat(timeout=10)
        print(f"  Heartbeat received from system {vehicle.target_sy
```

```
stem}, component {vehicle.target_component}")
        return vehicle
    except Exception as e:
        print(f"  Connection failed: {str(e)}")
        print("Ensure your vehicle is sending MAVLink messages to 1
4550 port")
        print("Example QGroundControl setup: Settings > General > M
AVLink > Listening Port = 14550")
        sys.exit(1)

def format_param_id(param_id):
    """格式化16字节参数ID"""
    if len(param_id) > 16:
        raise ValueError("param_id cannot exceed 16 characters")

    param_id_bytes = bytearray(16)
    for i in range(min(len(param_id), 16)):
        param_id_bytes[i] = ord(param_id[i])

    # 长度<16时添加NULL终止符
    if len(param_id) < 16:
        param_id_bytes[len(param_id)] = 0

    return param_id_bytes

def send_param_set(vehicle, param_id, param_value, param_type):
    """发送参数设置请求"""
    vehicle.mav.param_set_send(
        vehicle.target_system,
        vehicle.target_component,
        format_param_id(param_id),
        param_value,
        param_type
    )
    print(f"  Sent PARAM_SET for '{param_id}' = {param_value} (typ
e: {param_type})")

def send_param_request_read(vehicle, param_id=None, param_index=-
1):
    """发送参数读取请求"""
    param_id_bytes = format_param_id(param_id) if param_id else byt
earray(16)

    vehicle.mav.param_request_read_send(
        vehicle.target_system,
```

```
        vehicle.target_component,
        param_id_bytes,
        param_index
    )

    target = f"index {param_index}" if param_index >= 0 else f"ID
'{param_id}'"
    print(f"  Sent PARAM_REQUEST_READ for {target}")

def send_param_request_list(vehicle):
    """发送请求所有参数列表"""
    vehicle.mav.param_request_list_send(
        vehicle.target_system,
        vehicle.target_component
    )
    print(f"  Sent PARAM_REQUEST_LIST to system {vehicle.target_sys
tem}, component {vehicle.target_component}")

def receive_param_value(vehicle, timeout=5.0):
    """接收参数值响应"""
    print(f"  Waiting for PARAM_VALUE ({timeout}s timeout)...")
    msg = vehicle.recv_match(type='PARAM_VALUE', blocking=True, tim
eout=timeout)

    if not msg:
        print("  Timeout: No PARAM_VALUE received")
        return None

    # 解析参数ID
    param_id = msg.param_id

    # 解析参数值
    param_value = msg.param_value

    # 参数类型映射
    type_names = {
        0: "INT8", 1: "UINT8", 2: "INT16", 3: "UINT16", 4: "INT32",
        5: "UINT32", 6: "INT64", 7: "UINT64", 8: "FLOAT", 9: "DOUBL
E"
    }
    param_type = type_names.get(msg.param_type, f"UNKNOWN({msg.para
m_type})")

    print("\n  Received PARAM_VALUE:")
    print(f"    ID: {param_id}")
```

```
    print(f"    Value: {param_value}")
    print(f"    Type: {param_type}")
    print(f"    Index/Total: {msg.param_index}/{msg.param_count}")

    return {
        'id': param_id,
        'value': param_value,
        'type': param_type,
        'index': msg.param_index,
        'count': msg.param_count
    }

def receive_all_param_values(vehicle, timeout=10.0):
    """接收所有参数值，直到收到 param_count 条消息后退出循环"""
    received_params = []
    expected_count = None
    print(f"  Waiting for all PARAM_VALUE messages...")

    while True:
        msg = vehicle.recv_match(type='PARAM_VALUE', blocking=True,
timeout=timeout)

        if not msg:
            print(f"  Timeout: Received {len(received_params)} para
meters (expected {expected_count if expected_count else 'unknow
n'})")
            break

        param_id = msg.param_id
        param_value = msg.param_value

        # 参数类型映射
        type_names = {
            0: "INT8", 1: "UINT8", 2: "INT16", 3: "UINT16", 4: "INT
32",
            5: "UINT32", 6: "INT64", 7: "UINT64", 8: "FLOAT", 9: "D
OUBLE"
        }
        param_type = type_names.get(msg.param_type, f"UNKNOWN({msg.
param_type})")

        # 首次收到消息时记录总参数数量
        if expected_count is None:
            expected_count = msg.param_count
            print(f"  Total parameters to receive: {expected_coun
```

```
t}")

        print(f"  [{msg.param_index}/{expected_count}] {param_id} =
{param_value} ({param_type})")

        received_params.append({
            'id': param_id,
            'value': param_value,
            'type': param_type,
            'index': msg.param_index,
            'count': msg.param_count
        })

        # 当已接收数量达到预期总数时退出循环
        if len(received_params) >= expected_count:
            print(f"\n  Successfully received all {expected_count}
parameters")
            break

    return received_params

def main():
    print("  Starting MAVLink PARAM_SET Tool (UDP Listening Mode)")
    print("------------------------------------------------------")
    print("This script listens for MAVLink on ALL interfaces:1455
0")
    print("Ensure your vehicle is configured to send to this port")
    print("------------------------------------------------------
\n")

    vehicle = connect_to_vehicle()

    # 操作选择
    print("\nAvailable operations:")
    print("1. Set parameter value")
    print("2. Read parameter value (by ID or index)")
    print("3. Request all parameters (PARAM_REQUEST_LIST)")
    choice = input("\nSelect operation [1-3]: ").strip()

    if choice == '1':
        param_id = input("Enter parameter ID to set: ").strip()
        param_value = float(input("Enter new value: ").strip())

        print("\nParameter types:")
        print("  8 = FLOAT (most common)")
```

```
        print("  9 = DOUBLE")
        print("  4 = INT32")
        print("  1 = UINT8")
        param_type = int(input("Enter type number [8]: ").strip() o
r 8)

        send_param_set(vehicle, param_id, param_value, param_type)
        receive_param_value(vehicle)

    elif choice == '2':
        # mode = input("Read by [id/index]: ").lower()
        mode = 'id'
        if mode == 'id':
            param_id = input("Enter parameter ID: ").strip()
            send_param_request_read(vehicle, param_id=param_id)
            receive_param_value(vehicle)
        elif mode == 'index':
            param_index = int(input("Enter parameter index: "))
            send_param_request_read(vehicle, param_index=param_inde
x)
            receive_param_value(vehicle)
        else:
            print("Invalid mode. Use 'id' or 'index'")

    elif choice == '3':
        print("\n  Requesting all parameters...")
        send_param_request_list(vehicle)
        all_params = receive_all_param_values(vehicle)
        print(f"\n  Total parameters received: {len(all_params)}")

    else:
        print("Invalid choice. Select 1, 2, or 3.")


if __name__ == "__main__":
    main()
```

示例使用说明

获取

运行后，成功建立连接

```
🚀 Starting MAVLink PARAM_SET Tool (UDP Listening Mode)
--------------------------------------------------
This script listens for MAVLink on ALL interfaces:14550
Ensure your vehicle is configured to send to this port
--------------------------------------------------

Listening for MAVLink on: udpin:0.0.0.0:14550
Waiting for HEARTBEAT (timeout: 10s)...
✅ Heartbeat received from system 3, component 0

Available operations:
1. Set parameter value
2. Read parameter value (by ID or index)
3. Request all parameters (PARAM_REQUEST_LIST)
```

输入“2”选择读取参数

```
Select operation [1-3]: 2
```

输入对应参数“MAX_VELOCITY”

```
Enter parameter ID: MAX_VELOCITY
```

成功返回该参数对应的数值

```
📥 Received PARAM_VALUE:
  🆔 ID: MAX_VELOCITY
  🔢 Value: 10.0
```

设置

运行后，成功建立连接

```
🚀 Starting MAVLink PARAM_SET Tool (UDP Listening Mode)
--------------------------------------------------
This script listens for MAVLink on ALL interfaces:14550
Ensure your vehicle is configured to send to this port
--------------------------------------------------

Listening for MAVLink on: udpin:0.0.0.0:14550
Waiting for HEARTBEAT (timeout: 10s)...
✅ Heartbeat received from system 3, component 0

Available operations:
1. Set parameter value
2. Read parameter value (by ID or index)
3. Request all parameters (PARAM_REQUEST_LIST)
```

输入“1”选择设置参数

```
Select operation [1-3]: 1
```

输入对应参数“MAX_VELOCITY”

```
Enter parameter ID to set: MAX_VELOCITY
```

输入要设置的数值

```
Enter new value: 12
```

选择参数类型“1”

```
Parameter types:
  9 = DOUBLE
  4 = INT32
  1 = UINT8
Enter type number [8]: 1
```

成功返回设置值

```
📩 Received PARAM_VALUE:
  🆔 ID: MAX_VELOCITY
  🔢 Value: 12.0
```

设置最低高度

MAX_HEIGHT

设置

mavlink消息：

接收PARAM_SET
发送PARAM_VALUE

获取

mavlink消息

接收PARAM_REQUEST_READ
发送PARAM_VALUE

参数：

参数名	类型	单位	描述
value	float	m	最小飞行高度。取值范围：1.0 ~ 3000.0

示例

```
import time
import sys
from pymavlink import mavutil

def connect_to_vehicle():
    """建立 UDP 监听连接（默认监听所有接口的 14550 端口）"""
    connection_string = "udpin:0.0.0.0:14550"
    print(f"Listening for MAVLink on: {connection_string}")
    vehicle = mavutil.mavlink_connection(connection_string)
    print("Waiting for HEARTBEAT (timeout: 10s)...")
    try:
        vehicle.wait_heartbeat(timeout=10)
        print(f"  Heartbeat received from system {vehicle.target_system}, component {vehicle.target_component}")
        return vehicle
    except Exception as e:
        print(f"  Connection failed: {str(e)}")
        print("Ensure your vehicle is sending MAVLink messages to 14550 port")
        print("Example QGroundControl setup: Settings > General > MAVLink > Listening Port = 14550")
        sys.exit(1)
```

```
def format_param_id(param_id):
    """格式化16字节参数ID（符合MAVLink协议要求）"""
    if len(param_id) > 16:
        raise ValueError("param_id cannot exceed 16 characters")

    param_id_bytes = bytearray(16)
    for i in range(min(len(param_id), 16)):
        param_id_bytes[i] = ord(param_id[i])

    # 长度<16时添加NULL终止符
    if len(param_id) < 16:
        param_id_bytes[len(param_id)] = 0

    return param_id_bytes

def send_param_set(vehicle, param_id, param_value, param_type):
    """发送参数设置请求"""
    vehicle.mav.param_set_send(
        vehicle.target_system,
        vehicle.target_component,
        format_param_id(param_id),
        param_value,
        param_type
    )
    print(f"  Sent PARAM_SET for '{param_id}' = {param_value} (type: {param_type})")

def send_param_request_read(vehicle, param_id=None, param_index=-1):
    """发送参数读取请求"""
    param_id_bytes = format_param_id(param_id) if param_id else bytearray(16)

    vehicle.mav.param_request_read_send(
        vehicle.target_system,
        vehicle.target_component,
        param_id_bytes,
        param_index
    )

    target = f"index {param_index}" if param_index >= 0 else f"ID '{param_id}'"
    print(f"  Sent PARAM_REQUEST_READ for {target}")

def send_param_request_list(vehicle):
```

```
    """发送请求所有参数列表"""
    vehicle.mav.param_request_list_send(
        vehicle.target_system,
        vehicle.target_component
    )
    print(f"  Sent PARAM_REQUEST_LIST to system {vehicle.target_system}, component {vehicle.target_component}")

def receive_param_value(vehicle, timeout=5.0):
    """接收参数值响应"""
    print(f"  Waiting for PARAM_VALUE ({timeout}s timeout)...")
    msg = vehicle.recv_match(type='PARAM_VALUE', blocking=True, timeout=timeout)

    if not msg:
        print("  Timeout: No PARAM_VALUE received")
        return None

    # 解析参数ID
    param_id = msg.param_id

    # 解析参数值
    param_value = msg.param_value

    # 参数类型映射
    type_names = {
        0: "INT8", 1: "UINT8", 2: "INT16", 3: "UINT16", 4: "INT32",
        5: "UINT32", 6: "INT64", 7: "UINT64", 8: "FLOAT", 9: "DOUBLE"
    }
    param_type = type_names.get(msg.param_type, f"UNKNOWN({msg.param_type})")

    print("\n  Received PARAM_VALUE:")
    print(f"    ID: {param_id}")
    print(f"    Value: {param_value}")
    print(f"    Type: {param_type}")
    print(f"    Index/Total: {msg.param_index}/{msg.param_count}")

    return {
        'id': param_id,
        'value': param_value,
        'type': param_type,
        'index': msg.param_index,
        'count': msg.param_count
```

```
    }

def receive_all_param_values(vehicle, timeout=10.0):
    """接收所有参数值，直到收到 param_count 条消息后退出循环"""
    received_params = []
    expected_count = None
    print(f"  Waiting for all PARAM_VALUE messages...")

    while True:
        msg = vehicle.recv_match(type='PARAM_VALUE', blocking=True,
timeout=timeout)

        if not msg:
            print(f"  Timeout: Received {len(received_params)} para
meters (expected {expected_count if expected_count else 'unknow
n'})")
            break

        param_id = msg.param_id
        param_value = msg.param_value

        # 参数类型映射
        type_names = {
            0: "INT8", 1: "UINT8", 2: "INT16", 3: "UINT16", 4: "INT
32",
            5: "UINT32", 6: "INT64", 7: "UINT64", 8: "FLOAT", 9: "D
OUBLE"
        }
        param_type = type_names.get(msg.param_type, f"UNKNOWN({msg.
param_type})")

        # 首次收到消息时记录总参数数量
        if expected_count is None:
            expected_count = msg.param_count
            print(f"  Total parameters to receive: {expected_coun
t}")

        print(f"  [{msg.param_index}/{expected_count}] {param_id} =
{param_value} ({param_type})")

        received_params.append({
            'id': param_id,
            'value': param_value,
            'type': param_type,
            'index': msg.param_index,
```

```
            'count': msg.param_count
        })

        # 当已接收数量达到预期总数时退出循环
        if len(received_params) >= expected_count:
            print(f"\n  Successfully received all {expected_count} parameters")
            break

    return received_params

def main():
    print("  Starting MAVLink PARAM_SET Tool (UDP Listening Mode)")
    print("------------------------------------------------------")
    print("This script listens for MAVLink on ALL interfaces:14550")
    print("Ensure your vehicle is configured to send to this port")
    print("------------------------------------------------------\n")

    vehicle = connect_to_vehicle()

    # 操作选择
    print("\nAvailable operations:")
    print("1. Set parameter value")
    print("2. Read parameter value (by ID or index)")
    print("3. Request all parameters (PARAM_REQUEST_LIST)")
    choice = input("\nSelect operation [1-3]: ").strip()

    if choice == '1':
        param_id = input("Enter parameter ID to set: ").strip()
        param_value = float(input("Enter new value: ").strip())

        print("\nParameter types:")
        print("  8 = FLOAT (most common)")
        print("  9 = DOUBLE")
        print("  4 = INT32")
        print("  1 = UINT8")
        param_type = int(input("Enter type number [8]: ").strip() or 8)

        send_param_set(vehicle, param_id, param_value, param_type)
        receive_param_value(vehicle)

    elif choice == '2':
```

```
        # mode = input("Read by [id/index]: ").lower()
        mode = 'id'
        if mode == 'id':
            param_id = input("Enter parameter ID: ").strip()
            send_param_request_read(vehicle, param_id=param_id)
            receive_param_value(vehicle)
        elif mode == 'index':
            param_index = int(input("Enter parameter index: "))
            send_param_request_read(vehicle, param_index=param_inde
x)
            receive_param_value(vehicle)
        else:
            print("Invalid mode. Use 'id' or 'index'")

    elif choice == '3':
        print("\n  Requesting all parameters...")
        send_param_request_list(vehicle)
        all_params = receive_all_param_values(vehicle)
        print(f"\n  Total parameters received: {len(all_params)}")

    else:
        print("Invalid choice. Select 1, 2, or 3.")


if __name__ == "__main__":
    main()
```

示例使用说明

获取

运行后，成功建立连接

```
🚀 Starting MAVLink PARAM_SET Tool (UDP Listening Mode)
--------------------------------------------------
This script listens for MAVLink on ALL interfaces:14550
Ensure your vehicle is configured to send to this port
--------------------------------------------------

Listening for MAVLink on: udpin:0.0.0.0:14550
Waiting for HEARTBEAT (timeout: 10s)...
✅ Heartbeat received from system 3, component 0

Available operations:
1. Set parameter value
2. Read parameter value (by ID or index)
3. Request all parameters (PARAM_REQUEST_LIST)
```

输入“2”选择读取参数

```
Select operation [1-3]: 2
```

输入对应参数“MAX_HEIGHT”

```
Enter parameter ID: MAX_HEIGHT
```

成功返回该参数对应的数值

```
📥 Received PARAM_VALUE:
  🆔 ID: MAX_HEIGHT
  🔢 Value: 20.0
```

设置

运行后，成功建立连接

```
🚀 Starting MAVLink PARAM_SET Tool (UDP Listening Mode)
--------------------------------------------------
This script listens for MAVLink on ALL interfaces:14550
Ensure your vehicle is configured to send to this port
--------------------------------------------------

Listening for MAVLink on: udpin:0.0.0.0:14550
Waiting for HEARTBEAT (timeout: 10s)...
✅ Heartbeat received from system 3, component 0

Available operations:
1. Set parameter value
2. Read parameter value (by ID or index)
3. Request all parameters (PARAM_REQUEST_LIST)
```

输入“1”选择设置参数

```
Available operations:
1. Set parameter value
2. Read parameter value (by ID or index)
3. Request all parameters (PARAM_REQUEST_LIST)

Select operation [1-3]: 1
```

输入对应参数“MAX_HEIGHT”

```
Enter parameter ID to set: MAX_HEIGHT
```

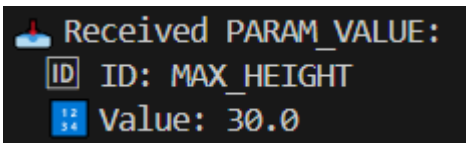
输入要设置的数值

```
Enter new value: 30
```

选择参数类型“8”

```
Parameter types:
  8 = FLOAT (most common)
  9 = DOUBLE
  4 = INT32
  1 = UINT8
Enter type number [8]: 8
```

成功返回设置值



RTK位置功能

RTK_POS_ENABLE

设置

mavlink消息：

接收PARAM_SET
发送PARAM_VALUE

获取

mavlink消息

接收PARAM_REQUEST_READ
发送PARAM_VALUE

参数：

枚举值	值	说明
DJI_FLIGHT_CONTROLLER_DISABLE_RTK_POSITION	0	禁用 RTK 定位，飞机使 GPS 数据执行需要位置信息的 （如航点飞行、返航等）
DJI_FLIGHT_CONTROLLER_ENABLE_RTK_POSITION	1	启用 RTK 定位，飞机使 RTK 数据执行需要位置信息的 （如航点飞行、返航等）

示例

```
import time
import sys
from pymavlink import mavutil

def connect_to_vehicle():
```

```
    """建立 UDP 监听连接（默认监听所有接口的 14550 端口）"""
    connection_string = "udpin:0.0.0.0:14550"
    print(f"Listening for MAVLink on: {connection_string}")
    vehicle = mavutil.mavlink_connection(connection_string)
    print("Waiting for HEARTBEAT (timeout: 10s)...")
    try:
        vehicle.wait_heartbeat(timeout=10)
        print(f"  Heartbeat received from system {vehicle.target_sy
stem}, component {vehicle.target_component}")
        return vehicle
    except Exception as e:
        print(f"  Connection failed: {str(e)}")
        print("Ensure your vehicle is sending MAVLink messages to 1
4550 port")
        print("Example QGroundControl setup: Settings > General > M
AVLink > Listening Port = 14550")
        sys.exit(1)

def format_param_id(param_id):
    """格式化16字节参数ID（符合MAVLink协议要求）"""
    if len(param_id) > 16:
        raise ValueError("param_id cannot exceed 16 characters")

    param_id_bytes = bytearray(16)
    for i in range(min(len(param_id), 16)):
        param_id_bytes[i] = ord(param_id[i])

    # 长度<16时添加NULL终止符
    if len(param_id) < 16:
        param_id_bytes[len(param_id)] = 0

    return param_id_bytes

def send_param_set(vehicle, param_id, param_value, param_type):
    """发送参数设置请求"""
    vehicle.mav.param_set_send(
        vehicle.target_system,
        vehicle.target_component,
        format_param_id(param_id),
        param_value,
        param_type
    )
    print(f"  Sent PARAM_SET for '{param_id}' = {param_value} (typ
e: {param_type})")
```

```
def send_param_request_read(vehicle, param_id=None, param_index=-
1):
    """发送参数读取请求"""
    param_id_bytes = format_param_id(param_id) if param_id else byt
earray(16)

    vehicle.mav.param_request_read_send(
        vehicle.target_system,
        vehicle.target_component,
        param_id_bytes,
        param_index
    )

    target = f"index {param_index}" if param_index >= 0 else f"ID
'{param_id}'"
    print(f"  Sent PARAM_REQUEST_READ for {target}")

def send_param_request_list(vehicle):
    """发送请求所有参数列表"""
    vehicle.mav.param_request_list_send(
        vehicle.target_system,
        vehicle.target_component
    )
    print(f"  Sent PARAM_REQUEST_LIST to system {vehicle.target_sys
tem}, component {vehicle.target_component}")

def receive_param_value(vehicle, timeout=5.0):
    """接收参数值响应"""
    print(f"  Waiting for PARAM_VALUE ({timeout}s timeout)...")
    msg = vehicle.recv_match(type='PARAM_VALUE', blocking=True, tim
eout=timeout)

    if not msg:
        print("  Timeout: No PARAM_VALUE received")
        return None

    # 解析参数ID
    param_id = msg.param_id

    # 解析参数值
    param_value = msg.param_value

    # 参数类型映射
    type_names = {
        0: "INT8", 1: "UINT8", 2: "INT16", 3: "UINT16", 4: "INT32",
```

```
        5: "UINT32", 6: "INT64", 7: "UINT64", 8: "FLOAT", 9: "DOUBL
E"
    }
    param_type = type_names.get(msg.param_type, f"UNKNOWN({msg.para
m_type})")

    print("\n  Received PARAM_VALUE:")
    print(f"    ID: {param_id}")
    print(f"    Value: {param_value}")
    print(f"    Type: {param_type}")
    print(f"    Index/Total: {msg.param_index}/{msg.param_count}")

    return {
        'id': param_id,
        'value': param_value,
        'type': param_type,
        'index': msg.param_index,
        'count': msg.param_count
    }

def receive_all_param_values(vehicle, timeout=10.0):
    """接收所有参数值，直到收到 param_count 条消息后退出循环"""
    received_params = []
    expected_count = None
    print(f"  Waiting for all PARAM_VALUE messages...")

    while True:
        msg = vehicle.recv_match(type='PARAM_VALUE', blocking=True,
timeout=timeout)

        if not msg:
            print(f"  Timeout: Received {len(received_params)} para
meters (expected {expected_count if expected_count else 'unknow
n'})")
            break

        param_id = msg.param_id
        param_value = msg.param_value

        # 参数类型映射
        type_names = {
            0: "INT8", 1: "UINT8", 2: "INT16", 3: "UINT16", 4: "INT
32",
            5: "UINT32", 6: "INT64", 7: "UINT64", 8: "FLOAT", 9: "D
OUBLE"
```

```
        }
        param_type = type_names.get(msg.param_type, f"UNKNOWN({msg.
param_type})")

        # 首次收到消息时记录总参数数量
        if expected_count is None:
            expected_count = msg.param_count
            print(f"  Total parameters to receive: {expected_coun
t}")

        print(f"  [{msg.param_index}/{expected_count}] {param_id} =
{param_value} ({param_type})")

        received_params.append({
            'id': param_id,
            'value': param_value,
            'type': param_type,
            'index': msg.param_index,
            'count': msg.param_count
        })

        # 当已接收数量达到预期总数时退出循环
        if len(received_params) >= expected_count:
            print(f"\n  Successfully received all {expected_count}
parameters")
            break

    return received_params

def main():
    print("  Starting MAVLink PARAM_SET Tool (UDP Listening Mode)")
    print("------------------------------------------------------")
    print("This script listens for MAVLink on ALL interfaces:1455
0")
    print("Ensure your vehicle is configured to send to this port")
    print("------------------------------------------------------
\n")

    vehicle = connect_to_vehicle()

    # 操作选择
    print("\nAvailable operations:")
    print("1. Set parameter value")
    print("2. Read parameter value (by ID or index)")
    print("3. Request all parameters (PARAM_REQUEST_LIST)")
```

```
    choice = input("\nSelect operation [1-3]: ").strip()

    if choice == '1':
        param_id = input("Enter parameter ID to set: ").strip()
        param_value = float(input("Enter new value: ").strip())

        print("\nParameter types:")
        print("  8 = FLOAT (most common)")
        print("  9 = DOUBLE")
        print("  4 = INT32")
        print("  1 = UINT8")
        param_type = int(input("Enter type number [8]: ").strip() o
r 8)

        send_param_set(vehicle, param_id, param_value, param_type)
        receive_param_value(vehicle)

    elif choice == '2':
        # mode = input("Read by [id/index]: ").lower()
        mode = 'id'
        if mode == 'id':
            param_id = input("Enter parameter ID: ").strip()
            send_param_request_read(vehicle, param_id=param_id)
            receive_param_value(vehicle)
        elif mode == 'index':
            param_index = int(input("Enter parameter index: "))
            send_param_request_read(vehicle, param_index=param_inde
x)
            receive_param_value(vehicle)
        else:
            print("Invalid mode. Use 'id' or 'index'")

    elif choice == '3':
        print("\n  Requesting all parameters...")
        send_param_request_list(vehicle)
        all_params = receive_all_param_values(vehicle)
        print(f"\n  Total parameters received: {len(all_params)}")

    else:
        print("Invalid choice. Select 1, 2, or 3.")


if __name__ == "__main__":
    main()
```

示例使用说明

获取

运行后，成功建立连接

```
🚀 Starting MAVLink PARAM_SET Tool (UDP Listening Mode)
------------------------------------------------------
This script listens for MAVLink on ALL interfaces:14550
Ensure your vehicle is configured to send to this port
------------------------------------------------------

Listening for MAVLink on: udpin:0.0.0.0:14550
Waiting for HEARTBEAT (timeout: 10s)...
✅ Heartbeat received from system 3, component 0

Available operations:
1. Set parameter value
2. Read parameter value (by ID or index)
3. Request all parameters (PARAM_REQUEST_LIST)
```

输入“2”选择读取参数

```
Select operation [1-3]: 2
```

输入对应参数“RTK_POS_ENABLE”

```
Enter parameter ID: RTK_POS_ENABLE
```

成功返回该参数对应的数值

```
📥 Received PARAM_VALUE:
  🆔 ID: RTK_POS_ENABLE
  🔢 Value: 1.0
```

设置

运行后，成功建立连接

```
🚀 Starting MAVLink PARAM_SET Tool (UDP Listening Mode)
--------------------------------------------------
This script listens for MAVLink on ALL interfaces:14550
Ensure your vehicle is configured to send to this port
--------------------------------------------------

Listening for MAVLink on: udpin:0.0.0.0:14550
Waiting for HEARTBEAT (timeout: 10s)...
✅ Heartbeat received from system 3, component 0

Available operations:
1. Set parameter value
2. Read parameter value (by ID or index)
3. Request all parameters (PARAM_REQUEST_LIST)
```

输入“1”选择设置参数

```
Select operation [1-3]: 1
```

输入对应参数“RTK_POS_ENABLE”

```
Enter parameter ID to set: RTK_POS_ENABLE
```

输入要设置的数值

```
Enter new value: 0
```

选择参数类型“1”

```
Parameter types:
  9 = DOUBLE
  4 = INT32
  1 = UINT8
Enter type number [8]: 1
```

成功返回设置值

```
📥 Received PARAM_VALUE:
  🆔 ID: RTK_POS_ENABLE
  🔢 Value: 0.0
```

遥控器失联动作

LOST_ACTION

设置

mavlink消息：

接收PARAM_SET
发送PARAM_VALUE

获取

mavlink消息

接收PARAM_REQUEST_READ
发送PARAM_VALUE

参数：

枚举值	值	说明
DJI_FLIGHT_CONTROLLER_RC_LOST_ACTION_HOVER	0	遥控器失联后，飞机执行悬停动作（Hover）
DJI_FLIGHT_CONTROLLER_RC_LOST_ACTION_LANDING	1	遥控器失联后，飞机执行降落动作（Landing）
DJI_FLIGHT_CONTROLLER_RC_LOST_ACTION_GOHOME	2	遥控器失联后，飞机执行返航动作（Go Home）

示例

```
import time
import sys
from pymavlink import mavutil

def connect_to_vehicle():
    """建立 UDP 监听连接（默认监听所有接口的 14550 端口）"""
    connection_string = "udpin:0.0.0.0:14550"
    print(f"Listening for MAVLink on: {connection_string}")
    vehicle = mavutil.mavlink_connection(connection_string)
    print("Waiting for HEARTBEAT (timeout: 10s)...")
    try:
        vehicle.wait_heartbeat(timeout=10)
        print(f"  Heartbeat received from system {vehicle.target_sy
```

```
stem}, component {vehicle.target_component}")
        return vehicle
    except Exception as e:
        print(f"  Connection failed: {str(e)}")
        print("Ensure your vehicle is sending MAVLink messages to 1
4550 port")
        print("Example QGroundControl setup: Settings > General > M
AVLink > Listening Port = 14550")
        sys.exit(1)

def format_param_id(param_id):
    """格式化16字节参数ID（符合MAVLink协议要求）"""
    if len(param_id) > 16:
        raise ValueError("param_id cannot exceed 16 characters")

    param_id_bytes = bytearray(16)
    for i in range(min(len(param_id), 16)):
        param_id_bytes[i] = ord(param_id[i])

    # 长度<16时添加NULL终止符
    if len(param_id) < 16:
        param_id_bytes[len(param_id)] = 0

    return param_id_bytes

def send_param_set(vehicle, param_id, param_value, param_type):
    """发送参数设置请求"""
    vehicle.mav.param_set_send(
        vehicle.target_system,
        vehicle.target_component,
        format_param_id(param_id),
        param_value,
        param_type
    )
    print(f"  Sent PARAM_SET for '{param_id}' = {param_value} (typ
e: {param_type})")

def send_param_request_read(vehicle, param_id=None, param_index=-
1):
    """发送参数读取请求"""
    param_id_bytes = format_param_id(param_id) if param_id else byt
earray(16)

    vehicle.mav.param_request_read_send(
        vehicle.target_system,
```

```
        vehicle.target_component,
        param_id_bytes,
        param_index
    )

    target = f"index {param_index}" if param_index >= 0 else f"ID
'{param_id}'"
    print(f"  Sent PARAM_REQUEST_READ for {target}")

def send_param_request_list(vehicle):
    """发送请求所有参数列表"""
    vehicle.mav.param_request_list_send(
        vehicle.target_system,
        vehicle.target_component
    )
    print(f"  Sent PARAM_REQUEST_LIST to system {vehicle.target_sys
tem}, component {vehicle.target_component}")

def receive_param_value(vehicle, timeout=5.0):
    """接收参数值响应"""
    print(f"  Waiting for PARAM_VALUE ({timeout}s timeout)...")
    msg = vehicle.recv_match(type='PARAM_VALUE', blocking=True, tim
eout=timeout)

    if not msg:
        print("  Timeout: No PARAM_VALUE received")
        return None

    # 解析参数ID
    param_id = msg.param_id

    # 解析参数值
    param_value = msg.param_value

    # 参数类型映射
    type_names = {
        0: "INT8", 1: "UINT8", 2: "INT16", 3: "UINT16", 4: "INT32",
        5: "UINT32", 6: "INT64", 7: "UINT64", 8: "FLOAT", 9: "DOUBL
E"
    }
    param_type = type_names.get(msg.param_type, f"UNKNOWN({msg.para
m_type})")

    print("\n  Received PARAM_VALUE:")
    print(f"    ID: {param_id}")
```

```
    print(f"    Value: {param_value}")
    print(f"    Type: {param_type}")
    print(f"    Index/Total: {msg.param_index}/{msg.param_count}")

    return {
        'id': param_id,
        'value': param_value,
        'type': param_type,
        'index': msg.param_index,
        'count': msg.param_count
    }

def receive_all_param_values(vehicle, timeout=10.0):
    """接收所有参数值，直到收到 param_count 条消息后退出循环"""
    received_params = []
    expected_count = None
    print(f"  Waiting for all PARAM_VALUE messages...")

    while True:
        msg = vehicle.recv_match(type='PARAM_VALUE', blocking=True,
timeout=timeout)

        if not msg:
            print(f"  Timeout: Received {len(received_params)} para
meters (expected {expected_count if expected_count else 'unknow
n'})")
            break

        param_id = msg.param_id
        param_value = msg.param_value

        # 参数类型映射
        type_names = {
            0: "INT8", 1: "UINT8", 2: "INT16", 3: "UINT16", 4: "INT
32",
            5: "UINT32", 6: "INT64", 7: "UINT64", 8: "FLOAT", 9: "D
OUBLE"
        }
        param_type = type_names.get(msg.param_type, f"UNKNOWN({msg.
param_type})")

        # 首次收到消息时记录总参数数量
        if expected_count is None:
            expected_count = msg.param_count
            print(f"  Total parameters to receive: {expected_coun
```

```
t}")

        print(f"  [{msg.param_index}/{expected_count}] {param_id} =
{param_value} ({param_type})")

        received_params.append({
            'id': param_id,
            'value': param_value,
            'type': param_type,
            'index': msg.param_index,
            'count': msg.param_count
        })

        # 当已接收数量达到预期总数时退出循环
        if len(received_params) >= expected_count:
            print(f"\n  Successfully received all {expected_count}
parameters")
            break

    return received_params

def main():
    print("  Starting MAVLink PARAM_SET Tool (UDP Listening Mode)")
    print("------------------------------------------------------")
    print("This script listens for MAVLink on ALL interfaces:1455
0")
    print("Ensure your vehicle is configured to send to this port")
    print("------------------------------------------------------
\n")

    vehicle = connect_to_vehicle()

    # 操作选择
    print("\nAvailable operations:")
    print("1. Set parameter value")
    print("2. Read parameter value (by ID or index)")
    print("3. Request all parameters (PARAM_REQUEST_LIST)")
    choice = input("\nSelect operation [1-3]: ").strip()

    if choice == '1':
        param_id = input("Enter parameter ID to set: ").strip()
        param_value = float(input("Enter new value: ").strip())

        print("\nParameter types:")
        print("  8 = FLOAT (most common)")
```

```
        print("  9 = DOUBLE")
        print("  4 = INT32")
        print("  1 = UINT8")
        param_type = int(input("Enter type number [8]: ").strip() o
r 8)

        send_param_set(vehicle, param_id, param_value, param_type)
        receive_param_value(vehicle)

    elif choice == '2':
        # mode = input("Read by [id/index]: ").lower()
        mode = 'id'
        if mode == 'id':
            param_id = input("Enter parameter ID: ").strip()
            send_param_request_read(vehicle, param_id=param_id)
            receive_param_value(vehicle)
        elif mode == 'index':
            param_index = int(input("Enter parameter index: "))
            send_param_request_read(vehicle, param_index=param_inde
x)
            receive_param_value(vehicle)
        else:
            print("Invalid mode. Use 'id' or 'index'")

    elif choice == '3':
        print("\n  Requesting all parameters...")
        send_param_request_list(vehicle)
        all_params = receive_all_param_values(vehicle)
        print(f"\n  Total parameters received: {len(all_params)}")

    else:
        print("Invalid choice. Select 1, 2, or 3.")


if __name__ == "__main__":
    main()
```

示例使用说明

获取

运行后，成功建立连接

```
🚀 Starting MAVLink PARAM_SET Tool (UDP Listening Mode)
--------------------------------------------------
This script listens for MAVLink on ALL interfaces:14550
Ensure your vehicle is configured to send to this port
--------------------------------------------------

Listening for MAVLink on: udpin:0.0.0.0:14550
Waiting for HEARTBEAT (timeout: 10s)...
✅ Heartbeat received from system 3, component 0

Available operations:
1. Set parameter value
2. Read parameter value (by ID or index)
3. Request all parameters (PARAM_REQUEST_LIST)
```

输入“2”选择读取参数

```
Select operation [1-3]: 2
```

输入对应参数“LOST_ACTION”

```
Enter parameter ID:
📤 Sent PARAM_REQUES
⏳ Waiting for PARAM
```

成功返回该参数对应的数值

```
📥 Received PARAM_VALUE:
  🆔 ID: LOST_ACTION
  🔢 Value: 2.0
```

设置

运行后，成功建立连接

```
🚀 Starting MAVLink PARAM_SET Tool (UDP Listening Mode)
--------------------------------------------------
This script listens for MAVLink on ALL interfaces:14550
Ensure your vehicle is configured to send to this port
--------------------------------------------------

Listening for MAVLink on: udpin:0.0.0.0:14550
Waiting for HEARTBEAT (timeout: 10s)...
✅ Heartbeat received from system 3, component 0

Available operations:
1. Set parameter value
2. Read parameter value (by ID or index)
3. Request all parameters (PARAM_REQUEST_LIST)
```

输入“1”选择设置参数

```
Select operation [1-3]: 1
```

输入对应参数“LOST_ACTION”

```
Enter parameter ID to set: LOST_ACTION
```

输入要设置的数值

```
Enter new value: 1
```

选择参数类型“1”

```
Parameter types:
  9 = DOUBLE
  4 = INT32
  1 = UINT8
Enter type number [8]: 1
```

成功返回设置值

```
📥 Received PARAM_VALUE:
  🆔 ID: LOST_ACTION
  🔢 Value: 1.0
```

水平视觉障碍物避让的开关状态

HOR_VIS_AVO_ENA

设置

mavlink消息：

接收PARAM_SET
发送PARAM_VALUE

获取

mavlink消息

接收PARAM_REQUEST_READ
发送PARAM_VALUE

参数：

枚举值	值	说明
DJI_FLIGHT_CONTROLLER_DISABLE_OBSTACLE_AVOIDANCE	0	禁用指定方向
DJI_FLIGHT_CONTROLLER_ENABLE_OBSTACLE_AVOIDANCE	1	启用指定方向

示例

```
import time
import sys
from pymavlink import mavutil

def connect_to_vehicle():
    """建立 UDP 监听连接（默认监听所有接口的 14550 端口）"""
    connection_string = "udpin:0.0.0.0:14550"
    print(f"Listening for MAVLink on: {connection_string}")
    vehicle = mavutil.mavlink_connection(connection_string)
    print("Waiting for HEARTBEAT (timeout: 10s)...")
    try:
        vehicle.wait_heartbeat(timeout=10)
        print(f"  Heartbeat received from system {vehicle.target_sy
stem}, component {vehicle.target_component}")
        return vehicle
    except Exception as e:
        print(f"  Connection failed: {str(e)}")
        print("Ensure your vehicle is sending MAVLink messages to 1
4550 port")
        print("Example QGroundControl setup: Settings > General > M
AVLink > Listening Port = 14550")
```

```
        sys.exit(1)

def format_param_id(param_id):
    """格式化16字节参数ID（符合MAVLink协议要求）"""
    if len(param_id) > 16:
        raise ValueError("param_id cannot exceed 16 characters")

    param_id_bytes = bytearray(16)
    for i in range(min(len(param_id), 16)):
        param_id_bytes[i] = ord(param_id[i])

    # 长度<16时添加NULL终止符
    if len(param_id) < 16:
        param_id_bytes[len(param_id)] = 0

    return param_id_bytes

def send_param_set(vehicle, param_id, param_value, param_type):
    """发送参数设置请求"""
    vehicle.mav.param_set_send(
        vehicle.target_system,
        vehicle.target_component,
        format_param_id(param_id),
        param_value,
        param_type
    )
    print(f"  Sent PARAM_SET for '{param_id}' = {param_value} (typ
e: {param_type})")

def send_param_request_read(vehicle, param_id=None, param_index=-
1):
    """发送参数读取请求"""
    param_id_bytes = format_param_id(param_id) if param_id else byt
earray(16)

    vehicle.mav.param_request_read_send(
        vehicle.target_system,
        vehicle.target_component,
        param_id_bytes,
        param_index
    )

    target = f"index {param_index}" if param_index >= 0 else f"ID
'{param_id}'"
    print(f"  Sent PARAM_REQUEST_READ for {target}")
```

```
def send_param_request_list(vehicle):
    """发送请求所有参数列表"""
    vehicle.mav.param_request_list_send(
        vehicle.target_system,
        vehicle.target_component
    )
    print(f"  Sent PARAM_REQUEST_LIST to system {vehicle.target_sys
tem}, component {vehicle.target_component}")

def receive_param_value(vehicle, timeout=5.0):
    """接收参数值响应"""
    print(f"  Waiting for PARAM_VALUE ({timeout}s timeout)...")
    msg = vehicle.recv_match(type='PARAM_VALUE', blocking=True, tim
eout=timeout)

    if not msg:
        print("  Timeout: No PARAM_VALUE received")
        return None

    # 解析参数ID
    param_id = msg.param_id

    # 解析参数值
    param_value = msg.param_value

    # 参数类型映射
    type_names = {
        0: "INT8", 1: "UINT8", 2: "INT16", 3: "UINT16", 4: "INT32",
        5: "UINT32", 6: "INT64", 7: "UINT64", 8: "FLOAT", 9: "DOUBL
E"
    }
    param_type = type_names.get(msg.param_type, f"UNKNOWN({msg.para
m_type})")

    print("\n  Received PARAM_VALUE:")
    print(f"    ID: {param_id}")
    print(f"    Value: {param_value}")
    print(f"    Type: {param_type}")
    print(f"    Index/Total: {msg.param_index}/{msg.param_count}")

    return {
        'id': param_id,
        'value': param_value,
        'type': param_type,
```

```
        'index': msg.param_index,
        'count': msg.param_count
    }

def receive_all_param_values(vehicle, timeout=10.0):
    """接收所有参数值，直到收到 param_count 条消息后退出循环"""
    received_params = []
    expected_count = None
    print(f"  Waiting for all PARAM_VALUE messages...")

    while True:
        msg = vehicle.recv_match(type='PARAM_VALUE', blocking=True,
timeout=timeout)

        if not msg:
            print(f"  Timeout: Received {len(received_params)} para
meters (expected {expected_count if expected_count else 'unknow
n'})")
            break

        param_id = msg.param_id
        param_value = msg.param_value

        # 参数类型映射
        type_names = {
            0: "INT8", 1: "UINT8", 2: "INT16", 3: "UINT16", 4: "INT
32",
            5: "UINT32", 6: "INT64", 7: "UINT64", 8: "FLOAT", 9: "D
OUBLE"
        }
        param_type = type_names.get(msg.param_type, f"UNKNOWN({msg.
param_type})")

        # 首次收到消息时记录总参数数量
        if expected_count is None:
            expected_count = msg.param_count
            print(f"  Total parameters to receive: {expected_coun
t}")

        print(f"  [{msg.param_index}/{expected_count}] {param_id} =
{param_value} ({param_type})")

        received_params.append({
            'id': param_id,
            'value': param_value,
```

```
            'type': param_type,
            'index': msg.param_index,
            'count': msg.param_count
        })

        # 当已接收数量达到预期总数时退出循环
        if len(received_params) >= expected_count:
            print(f"\n  Successfully received all {expected_count} parameters")
            break

    return received_params

def main():
    print("  Starting MAVLink PARAM_SET Tool (UDP Listening Mode)")
    print("------------------------------------------------------")
    print("This script listens for MAVLink on ALL interfaces:14550")
    print("Ensure your vehicle is configured to send to this port")
    print("------------------------------------------------------\n")

    vehicle = connect_to_vehicle()

    # 操作选择
    print("\nAvailable operations:")
    print("1. Set parameter value")
    print("2. Read parameter value (by ID or index)")
    print("3. Request all parameters (PARAM_REQUEST_LIST)")
    choice = input("\nSelect operation [1-3]: ").strip()

    if choice == '1':
        param_id = input("Enter parameter ID to set: ").strip()
        param_value = float(input("Enter new value: ").strip())

        print("\nParameter types:")
        print("  8 = FLOAT (most common)")
        print("  9 = DOUBLE")
        print("  4 = INT32")
        print("  1 = UINT8")
        param_type = int(input("Enter type number [8]: ").strip() or 8)

        send_param_set(vehicle, param_id, param_value, param_type)
        receive_param_value(vehicle)
```

```
    elif choice == '2':
        # mode = input("Read by [id/index]: ").lower()
        mode = 'id'
        if mode == 'id':
            param_id = input("Enter parameter ID: ").strip()
            send_param_request_read(vehicle, param_id=param_id)
            receive_param_value(vehicle)
        elif mode == 'index':
            param_index = int(input("Enter parameter index: "))
            send_param_request_read(vehicle, param_index=param_inde
x)
            receive_param_value(vehicle)
        else:
            print("Invalid mode. Use 'id' or 'index'")

    elif choice == '3':
        print("\n  Requesting all parameters...")
        send_param_request_list(vehicle)
        all_params = receive_all_param_values(vehicle)
        print(f"\n  Total parameters received: {len(all_params)}")

    else:
        print("Invalid choice. Select 1, 2, or 3.")


if __name__ == "__main__":
    main()
```

示例使用说明

获取

运行后，成功建立连接

```
🚀 Starting MAVLink PARAM_SET Tool (UDP Listening Mode)
--------------------------------------------------
This script listens for MAVLink on ALL interfaces:14550
Ensure your vehicle is configured to send to this port
--------------------------------------------------

Listening for MAVLink on: udpin:0.0.0.0:14550
Waiting for HEARTBEAT (timeout: 10s)...
✅ Heartbeat received from system 3, component 0

Available operations:
1. Set parameter value
2. Read parameter value (by ID or index)
3. Request all parameters (PARAM_REQUEST_LIST)
```

输入“2”选择读取参数

```
Select operation [1-3]: 2
```

输入对应参数“HOR_VIS_AVO_ENA”

```
Enter parameter ID: HOR_VIS_AVO_ENA
```

成功返回该参数对应的数值

```
📥 Received PARAM_VALUE:
  🆔 ID: HOR_VIS_AVO_ENA
  🔢 Value: 1.0
```

设置

运行后，成功建立连接

```
🚀 Starting MAVLink PARAM_SET Tool (UDP Listening Mode)
------------------------------------------------------
This script listens for MAVLink on ALL interfaces:14550
Ensure your vehicle is configured to send to this port
------------------------------------------------------

Listening for MAVLink on: udpin:0.0.0.0:14550
Waiting for HEARTBEAT (timeout: 10s)...
✅ Heartbeat received from system 3, component 0

Available operations:
1. Set parameter value
2. Read parameter value (by ID or index)
3. Request all parameters (PARAM_REQUEST_LIST)
```

输入“1”选择设置参数

```
Select operation [1-3]: 1
```

输入对应参数“HOR_VIS_AVO_ENA”

```
Enter parameter ID to set: HOR_VIS_AVO_ENA
```

输入要设置的数值

```
Enter new value: 0
```

选择参数类型“1”

```
Parameter types:
  9 = DOUBLE
  4 = INT32
  1 = UINT8
Enter type number [8]: 1
```

成功返回设置值

```
📥 Received PARAM_VALUE:
  🆔 ID: HOR_VIS_AVO_ENA
  🔢 Value: 0.0
```

上视避障功能

UP_VIS_AVO_ENA

设置

mavlink消息：

接收PARAM_SET
发送PARAM_VALUE

获取

mavlink消息

接收PARAM_REQUEST_READ
发送PARAM_VALUE

参数：

枚举值	值	说明
DJI_FLIGHT_CONTROLLER_DISABLE_OBSTACLE_AVOIDANCE	0	禁用指定方向
DJI_FLIGHT_CONTROLLER_ENABLE_OBSTACLE_AVOIDANCE	1	启用指定方向

示例

```
import time
import sys
from pymavlink import mavutil

def connect_to_vehicle():
    """建立 UDP 监听连接（默认监听所有接口的 14550 端口）"""
    connection_string = "udpin:0.0.0.0:14550"
    print(f"Listening for MAVLink on: {connection_string}")
    vehicle = mavutil.mavlink_connection(connection_string)
    print("Waiting for HEARTBEAT (timeout: 10s)...")
    try:
        vehicle.wait_heartbeat(timeout=10)
        print(f"  Heartbeat received from system {vehicle.target_sy
stem}, component {vehicle.target_component}")
        return vehicle
    except Exception as e:
        print(f"  Connection failed: {str(e)}")
        print("Ensure your vehicle is sending MAVLink messages to 1
4550 port")
        print("Example QGroundControl setup: Settings > General > M
AVLink > Listening Port = 14550")
```

```
        sys.exit(1)

def format_param_id(param_id):
    """格式化16字节参数ID（符合MAVLink协议要求）"""
    if len(param_id) > 16:
        raise ValueError("param_id cannot exceed 16 characters")

    param_id_bytes = bytearray(16)
    for i in range(min(len(param_id), 16)):
        param_id_bytes[i] = ord(param_id[i])

    # 长度<16时添加NULL终止符
    if len(param_id) < 16:
        param_id_bytes[len(param_id)] = 0

    return param_id_bytes

def send_param_set(vehicle, param_id, param_value, param_type):
    """发送参数设置请求"""
    vehicle.mav.param_set_send(
        vehicle.target_system,
        vehicle.target_component,
        format_param_id(param_id),
        param_value,
        param_type
    )
    print(f"  Sent PARAM_SET for '{param_id}' = {param_value} (type: {param_type})")

def send_param_request_read(vehicle, param_id=None, param_index=-1):
    """发送参数读取请求"""
    param_id_bytes = format_param_id(param_id) if param_id else bytearray(16)

    vehicle.mav.param_request_read_send(
        vehicle.target_system,
        vehicle.target_component,
        param_id_bytes,
        param_index
    )

    target = f"index {param_index}" if param_index >= 0 else f"ID '{param_id}'"
    print(f"  Sent PARAM_REQUEST_READ for {target}")
```

```
def send_param_request_list(vehicle):
    """发送请求所有参数列表"""
    vehicle.mav.param_request_list_send(
        vehicle.target_system,
        vehicle.target_component
    )
    print(f"  Sent PARAM_REQUEST_LIST to system {vehicle.target_sys
tem}, component {vehicle.target_component}")

def receive_param_value(vehicle, timeout=5.0):
    """接收参数值响应"""
    print(f"  Waiting for PARAM_VALUE ({timeout}s timeout)...")
    msg = vehicle.recv_match(type='PARAM_VALUE', blocking=True, tim
eout=timeout)

    if not msg:
        print("  Timeout: No PARAM_VALUE received")
        return None

    # 解析参数ID
    param_id = msg.param_id

    # 解析参数值
    param_value = msg.param_value

    # 参数类型映射
    type_names = {
        0: "INT8", 1: "UINT8", 2: "INT16", 3: "UINT16", 4: "INT32",
        5: "UINT32", 6: "INT64", 7: "UINT64", 8: "FLOAT", 9: "DOUBL
E"
    }
    param_type = type_names.get(msg.param_type, f"UNKNOWN({msg.para
m_type})")

    print("\n  Received PARAM_VALUE:")
    print(f"    ID: {param_id}")
    print(f"    Value: {param_value}")
    print(f"    Type: {param_type}")
    print(f"    Index/Total: {msg.param_index}/{msg.param_count}")

    return {
        'id': param_id,
        'value': param_value,
        'type': param_type,
```

```
        'index': msg.param_index,
        'count': msg.param_count
    }

def receive_all_param_values(vehicle, timeout=10.0):
    """接收所有参数值，直到收到 param_count 条消息后退出循环"""
    received_params = []
    expected_count = None
    print(f"  Waiting for all PARAM_VALUE messages...")

    while True:
        msg = vehicle.recv_match(type='PARAM_VALUE', blocking=True,
timeout=timeout)

        if not msg:
            print(f"  Timeout: Received {len(received_params)} para
meters (expected {expected_count if expected_count else 'unknow
n'})")
            break

        param_id = msg.param_id
        param_value = msg.param_value

        # 参数类型映射
        type_names = {
            0: "INT8", 1: "UINT8", 2: "INT16", 3: "UINT16", 4: "INT
32",
            5: "UINT32", 6: "INT64", 7: "UINT64", 8: "FLOAT", 9: "D
OUBLE"
        }
        param_type = type_names.get(msg.param_type, f"UNKNOWN({msg.
param_type})")

        # 首次收到消息时记录总参数数量
        if expected_count is None:
            expected_count = msg.param_count
            print(f"  Total parameters to receive: {expected_coun
t}")

        print(f"  [{msg.param_index}/{expected_count}] {param_id} =
{param_value} ({param_type})")

        received_params.append({
            'id': param_id,
            'value': param_value,
```

```
            'type': param_type,
            'index': msg.param_index,
            'count': msg.param_count
        })

        # 当已接收数量达到预期总数时退出循环
        if len(received_params) >= expected_count:
            print(f"\n  Successfully received all {expected_count} parameters")
            break

    return received_params

def main():
    print("  Starting MAVLink PARAM_SET Tool (UDP Listening Mode)")
    print("------------------------------------------------------")
    print("This script listens for MAVLink on ALL interfaces:14550")
    print("Ensure your vehicle is configured to send to this port")
    print("------------------------------------------------------\n")

    vehicle = connect_to_vehicle()

    # 操作选择
    print("\nAvailable operations:")
    print("1. Set parameter value")
    print("2. Read parameter value (by ID or index)")
    print("3. Request all parameters (PARAM_REQUEST_LIST)")
    choice = input("\nSelect operation [1-3]: ").strip()

    if choice == '1':
        param_id = input("Enter parameter ID to set: ").strip()
        param_value = float(input("Enter new value: ").strip())

        print("\nParameter types:")
        print("  8 = FLOAT (most common)")
        print("  9 = DOUBLE")
        print("  4 = INT32")
        print("  1 = UINT8")
        param_type = int(input("Enter type number [8]: ").strip() or 8)

        send_param_set(vehicle, param_id, param_value, param_type)
        receive_param_value(vehicle)
```

```
    elif choice == '2':
        # mode = input("Read by [id/index]: ").lower()
        mode = 'id'
        if mode == 'id':
            param_id = input("Enter parameter ID: ").strip()
            send_param_request_read(vehicle, param_id=param_id)
            receive_param_value(vehicle)
        elif mode == 'index':
            param_index = int(input("Enter parameter index: "))
            send_param_request_read(vehicle, param_index=param_inde
x)
            receive_param_value(vehicle)
        else:
            print("Invalid mode. Use 'id' or 'index'")

    elif choice == '3':
        print("\n  Requesting all parameters...")
        send_param_request_list(vehicle)
        all_params = receive_all_param_values(vehicle)
        print(f"\n  Total parameters received: {len(all_params)}")

    else:
        print("Invalid choice. Select 1, 2, or 3.")


if __name__ == "__main__":
    main()
```

示例使用说明

获取

运行后，成功建立连接

```
🚀 Starting MAVLink PARAM_SET Tool (UDP Listening Mode)
--------------------------------------------------
This script listens for MAVLink on ALL interfaces:14550
Ensure your vehicle is configured to send to this port
--------------------------------------------------

Listening for MAVLink on: udpin:0.0.0.0:14550
Waiting for HEARTBEAT (timeout: 10s)...
✅ Heartbeat received from system 3, component 0

Available operations:
1. Set parameter value
2. Read parameter value (by ID or index)
3. Request all parameters (PARAM_REQUEST_LIST)
```

输入“2”选择读取参数

```
Select operation [1-3]: 2
```

输入对应参数“UP_VIS_AVO_ENA”

```
Enter parameter ID: UP_VIS_AVO_ENA
```

成功返回该参数对应的数值

```
📥 Received PARAM_VALUE:
  🆔 ID: UP_VIS_AVO_ENA
  🔢 Value: 1.0
```

设置

运行后，成功建立连接

```
🚀 Starting MAVLink PARAM_SET Tool (UDP Listening Mode)
------------------------------------------------------
This script listens for MAVLink on ALL interfaces:14550
Ensure your vehicle is configured to send to this port
------------------------------------------------------

Listening for MAVLink on: udpin:0.0.0.0:14550
Waiting for HEARTBEAT (timeout: 10s)...
✅ Heartbeat received from system 3, component 0

Available operations:
1. Set parameter value
2. Read parameter value (by ID or index)
3. Request all parameters (PARAM_REQUEST_LIST)
```

输入“1”选择设置参数

```
Select operation [1-3]: 1
```

输入对应参数“UP_VIS_AVO_ENA”

```
Enter parameter ID to set: UP_VIS_AVO_ENA
```

输入要设置的数值

```
Enter new value: 0
```

选择参数类型“1”

```
Parameter types:
  9 = DOUBLE
  4 = INT32
  1 = UINT8
Enter type number [8]: 1
```

成功返回设置值

```
📥 Received PARAM_VALUE:
  🆔 ID: UP_VIS_AVO_ENA
  🔢 Value: 0.0
```

向下视觉避障功能

DOW_VIS_AVO_ENA

设置

mavlink消息：

接收PARAM_SET
发送PARAM_VALUE

获取

mavlink消息

接收PARAM_REQUEST_READ
发送PARAM_VALUE

参数：

枚举值	值	说明
DJI_FLIGHT_CONTROLLER_DISABLE_OBSTACLE_AVOIDANCE	0	禁用指定方向
DJI_FLIGHT_CONTROLLER_ENABLE_OBSTACLE_AVOIDANCE	1	启用指定方向

示例

```
import time
import sys
from pymavlink import mavutil

def connect_to_vehicle():
    """建立 UDP 监听连接（默认监听所有接口的 14550 端口）"""
    connection_string = "udpin:0.0.0.0:14550"
    print(f"Listening for MAVLink on: {connection_string}")
    vehicle = mavutil.mavlink_connection(connection_string)
    print("Waiting for HEARTBEAT (timeout: 10s)...")
    try:
        vehicle.wait_heartbeat(timeout=10)
        print(f"  Heartbeat received from system {vehicle.target_system}, component {vehicle.target_component}")
        return vehicle
    except Exception as e:
        print(f"  Connection failed: {str(e)}")
        print("Ensure your vehicle is sending MAVLink messages to 14550 port")
        print("Example QGroundControl setup: Settings > General > MAVLink > Listening Port = 14550")
```

```
        sys.exit(1)

def format_param_id(param_id):
    """格式化16字节参数ID（符合MAVLink协议要求）"""
    if len(param_id) > 16:
        raise ValueError("param_id cannot exceed 16 characters")

    param_id_bytes = bytearray(16)
    for i in range(min(len(param_id), 16)):
        param_id_bytes[i] = ord(param_id[i])

    # 长度<16时添加NULL终止符
    if len(param_id) < 16:
        param_id_bytes[len(param_id)] = 0

    return param_id_bytes

def send_param_set(vehicle, param_id, param_value, param_type):
    """发送参数设置请求"""
    vehicle.mav.param_set_send(
        vehicle.target_system,
        vehicle.target_component,
        format_param_id(param_id),
        param_value,
        param_type
    )
    print(f"  Sent PARAM_SET for '{param_id}' = {param_value} (type: {param_type})")

def send_param_request_read(vehicle, param_id=None, param_index=-1):
    """发送参数读取请求"""
    param_id_bytes = format_param_id(param_id) if param_id else bytearray(16)

    vehicle.mav.param_request_read_send(
        vehicle.target_system,
        vehicle.target_component,
        param_id_bytes,
        param_index
    )

    target = f"index {param_index}" if param_index >= 0 else f"ID '{param_id}'"
    print(f"  Sent PARAM_REQUEST_READ for {target}")
```

```
def send_param_request_list(vehicle):
    """发送请求所有参数列表"""
    vehicle.mav.param_request_list_send(
        vehicle.target_system,
        vehicle.target_component
    )
    print(f"  Sent PARAM_REQUEST_LIST to system {vehicle.target_sys
tem}, component {vehicle.target_component}")

def receive_param_value(vehicle, timeout=5.0):
    """接收参数值响应"""
    print(f"  Waiting for PARAM_VALUE ({timeout}s timeout)...")
    msg = vehicle.recv_match(type='PARAM_VALUE', blocking=True, tim
eout=timeout)

    if not msg:
        print("  Timeout: No PARAM_VALUE received")
        return None

    # 解析参数ID
    param_id = msg.param_id

    # 解析参数值
    param_value = msg.param_value

    # 参数类型映射
    type_names = {
        0: "INT8", 1: "UINT8", 2: "INT16", 3: "UINT16", 4: "INT32",
        5: "UINT32", 6: "INT64", 7: "UINT64", 8: "FLOAT", 9: "DOUBL
E"
    }
    param_type = type_names.get(msg.param_type, f"UNKNOWN({msg.para
m_type})")

    print("\n  Received PARAM_VALUE:")
    print(f"    ID: {param_id}")
    print(f"    Value: {param_value}")
    print(f"    Type: {param_type}")
    print(f"    Index/Total: {msg.param_index}/{msg.param_count}")

    return {
        'id': param_id,
        'value': param_value,
        'type': param_type,
```

```
        'index': msg.param_index,
        'count': msg.param_count
    }

def receive_all_param_values(vehicle, timeout=10.0):
    """接收所有参数值，直到收到 param_count 条消息后退出循环"""
    received_params = []
    expected_count = None
    print(f"  Waiting for all PARAM_VALUE messages...")

    while True:
        msg = vehicle.recv_match(type='PARAM_VALUE', blocking=True,
timeout=timeout)

        if not msg:
            print(f"  Timeout: Received {len(received_params)} para
meters (expected {expected_count if expected_count else 'unknow
n'})")
            break

        param_id = msg.param_id
        param_value = msg.param_value

        # 参数类型映射
        type_names = {
            0: "INT8", 1: "UINT8", 2: "INT16", 3: "UINT16", 4: "INT
32",
            5: "UINT32", 6: "INT64", 7: "UINT64", 8: "FLOAT", 9: "D
OUBLE"
        }
        param_type = type_names.get(msg.param_type, f"UNKNOWN({msg.
param_type})")

        # 首次收到消息时记录总参数数量
        if expected_count is None:
            expected_count = msg.param_count
            print(f"  Total parameters to receive: {expected_coun
t}")

        print(f"  [{msg.param_index}/{expected_count}] {param_id} =
{param_value} ({param_type})")

        received_params.append({
            'id': param_id,
            'value': param_value,
```

```
            'type': param_type,
            'index': msg.param_index,
            'count': msg.param_count
        })

        # 当已接收数量达到预期总数时退出循环
        if len(received_params) >= expected_count:
            print(f"\n  Successfully received all {expected_count} parameters")
            break

    return received_params

def main():
    print("  Starting MAVLink PARAM_SET Tool (UDP Listening Mode)")
    print("------------------------------------------------------")
    print("This script listens for MAVLink on ALL interfaces:14550")
    print("Ensure your vehicle is configured to send to this port")
    print("------------------------------------------------------\n")

    vehicle = connect_to_vehicle()

    # 操作选择
    print("\nAvailable operations:")
    print("1. Set parameter value")
    print("2. Read parameter value (by ID or index)")
    print("3. Request all parameters (PARAM_REQUEST_LIST)")
    choice = input("\nSelect operation [1-3]: ").strip()

    if choice == '1':
        param_id = input("Enter parameter ID to set: ").strip()
        param_value = float(input("Enter new value: ").strip())

        print("\nParameter types:")
        print("  8 = FLOAT (most common)")
        print("  9 = DOUBLE")
        print("  4 = INT32")
        print("  1 = UINT8")
        param_type = int(input("Enter type number [8]: ").strip() or 8)

        send_param_set(vehicle, param_id, param_value, param_type)
        receive_param_value(vehicle)
```

```
    elif choice == '2':
        # mode = input("Read by [id/index]: ").lower()
        mode = 'id'
        if mode == 'id':
            param_id = input("Enter parameter ID: ").strip()
            send_param_request_read(vehicle, param_id=param_id)
            receive_param_value(vehicle)
        elif mode == 'index':
            param_index = int(input("Enter parameter index: "))
            send_param_request_read(vehicle, param_index=param_inde
x)
            receive_param_value(vehicle)
        else:
            print("Invalid mode. Use 'id' or 'index'")

    elif choice == '3':
        print("\n  Requesting all parameters...")
        send_param_request_list(vehicle)
        all_params = receive_all_param_values(vehicle)
        print(f"\n  Total parameters received: {len(all_params)}")

    else:
        print("Invalid choice. Select 1, 2, or 3.")


if __name__ == "__main__":
    main()
```

示例使用说明

获取

运行后，成功建立连接

```
🚀 Starting MAVLink PARAM_SET Tool (UDP Listening Mode)
--------------------------------------------------
This script listens for MAVLink on ALL interfaces:14550
Ensure your vehicle is configured to send to this port
--------------------------------------------------

Listening for MAVLink on: udpin:0.0.0.0:14550
Waiting for HEARTBEAT (timeout: 10s)...
✅ Heartbeat received from system 3, component 0

Available operations:
1. Set parameter value
2. Read parameter value (by ID or index)
3. Request all parameters (PARAM_REQUEST_LIST)
```

输入“2”选择读取参数

```
Select operation [1-3]: 2
```

输入对应参数“DOW_VIS_AVO_ENA”

```
Enter parameter ID: DOW_VIS_AVO_ENA
```

成功返回该参数对应的数值

```
📥 Received PARAM_VALUE:
  🆔 ID: DOW_VIS_AVO_ENA
  🔢 Value: 1.0
```

设置

运行后，成功建立连接

```
🚀 Starting MAVLink PARAM_SET Tool (UDP Listening Mode)
--------------------------------------------------
This script listens for MAVLink on ALL interfaces:14550
Ensure your vehicle is configured to send to this port
--------------------------------------------------

Listening for MAVLink on: udpin:0.0.0.0:14550
Waiting for HEARTBEAT (timeout: 10s)...
✅ Heartbeat received from system 3, component 0

Available operations:
1. Set parameter value
2. Read parameter value (by ID or index)
3. Request all parameters (PARAM_REQUEST_LIST)
```

输入“1”选择设置参数

```
Available operations:
1. Set parameter value
2. Read parameter value (by ID or index)
3. Request all parameters (PARAM_REQUEST_LIST)

Select operation [1-3]: 1
```

输入对应参数“DOW_VIS_AVO_ENA”

```
Enter parameter ID to set: DOW_VIS_AVO_ENA
```

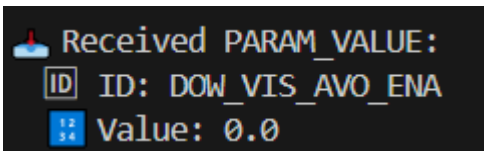
输入要设置的数值

```
Enter new value: 0
```

选择参数类型“1”

```
Parameter types:
  8 = FLOAT (most common)
  9 = DOUBLE
  4 = INT32
  1 = UINT8
Enter type number [8]: 1
```

成功返回设置值



紧急制动飞行

MAV_DJI_CMD_ARREST_FLYING

示例

```
import time
from pymavlink import mavutil
from enum import IntEnum


# MAV_CMD 执行结果映射
MAV_RESULT_NAMES = {
    mavutil.mavlink.MAV_RESULT_ACCEPTED: "ACCEPTED",
    mavutil.mavlink.MAV_RESULT_TEMPORARILY_REJECTED: "TEMPORARILY_R
EJECTED",
    mavutil.mavlink.MAV_RESULT_DENIED: "DENIED",
    mavutil.mavlink.MAV_RESULT_UNSUPPORTED: "UNSUPPORTED",
    mavutil.mavlink.MAV_RESULT_FAILED: "FAILED",
    mavutil.mavlink.MAV_RESULT_IN_PROGRESS: "IN_PROGRESS",
}


def recv_command_ack(timeout=5):
    """接收并显示 command_ack 消息

    Args:
        timeout: 超时时间（秒）

    Returns:
        ack 消息对象，None 表示超时
    """
    print(f"等待 command_ack 响应 (超时 {timeout}s)...")

    ack = master.recv_match(type='COMMAND_ACK', blocking=True, time
out=timeout)

    if ack is None:
        print("错误: 命令响应超时")
```

```
        return None

    # 获取命令名称
    cmd_name = getattr(ack, 'command_name', None)
    if not cmd_name:
        # 尝试通过枚举查找命令名称
        try:
            cmd_name = mavutil.mavlink.enums.get('MAV_CMD', {}).get
(ack.command, {}).name
        except:
            cmd_name = str(ack.command)

    # 获取结果名称
    result_name = MAV_RESULT_NAMES.get(ack.result, f"UNKNOWN({ack.r
esult})")

    print("=" * 50)
    print(f"命令: {cmd_name} ({ack.command})")
    print(f"结果: {result_name}")
    print(f"进度: {ack.progress * 100:.0f}%" if ack.progress > 0 els
e "")
    # print(f"参数2: {ack.result_param2}")
    print(f"系统ID: {ack.target_system}")
    print(f"组件ID: {ack.target_component}")
    print("=" * 50)

    return ack


class MavDjiCmd(IntEnum):
    MAV_CMD_NAV_LAND = 21
    MAV_DJI_CMD_TRIGGER_FFC=40
    MAV_DJI_CMD_START_RECORD_POINT_CLOUD=41
    MAV_DJI_CMD_STOP_RECORD_POINT_CLOUD=42
    MAV_DJI_CMD_RESTORE_FACTORY_SETTINGS=43
    MAV_DJI_CMD_APPLY_HIGH_POWER_SYNC=44
    MAV_DJI_CMD_APPLY_HIGH_POWER_SYNCV2=45
    MAV_DJI_CMD_START_CONFIRM_LANDING=46
    MAV_DJI_CMD_START_FORCE_LANDING=47
    MAV_DJI_CMD_EXECUTE_EMERGENCY_BRAKE_ACTION=48
    MAV_DJI_CMD_CANCEL_EMERGENCY_BRAKE_ACTION=49
    MAV_DJI_CMD_START_SLOW_ROTATE_MOTOR=50
    MAV_DJI_CMD_STOP_SLOW_ROTATE_MOTOR=51
    MAV_DJI_CMD_ARREST_FLYING=52
    MAV_DJI_CMD_CANCEL_ARREST_FLYING=53
```

```
    MAV_DJI_CMD_OUTPUT_HIGH_POWER=54
    TUNNEL_PROTOCOL_ID_INJECT_HMS_ERROR=32770

timestamp = int(time.time() * 1000)
millis = int(time.time() * 1000)

master = mavutil.mavlink_connection('udpin:0.0.0.0:14550')
print("waiting for heartbeat")
master.wait_heartbeat()
print("heartbeat received")
boot_time = time.time()

def send_command(command, param1=0, param2=0, param3=0, param4=0, p
aram5=0, param6=0, param7=0, wait_ack=True, timeout=5):
    """发送 command_long 并可选接收 command_ack

    Args:
        command: 命令ID (如 MavDjiCmd.TRIGGER_FFC)
        param1~param7: 命令参数
        wait_ack: 是否等待并显示 ACK
        timeout: ACK 超时时间(秒)

    Returns:
        ack 消息对象,None 表示超时或未等待
    """
    print(f"发送命令: {command} ({int(command)})")
    master.mav.command_long_send(
        master.target_system,
        master.target_component,
        int(command),
        0,  # confirmation
        param1, param2, param3, param4, param5, param6, param7
    )

    if wait_ack:
        return recv_command_ack(timeout=timeout)
    return None


def ready():
    send_command(MavDjiCmd.MAV_DJI_CMD_ARREST_FLYING)

ready()
```

示例使用说明

运行后，成功建立连接并发送指令

```
waiting for heartbeat
heartbeat received
发送命令: 52 (52)
等待 command_ack 响应 (超时 5s)...
================================================
命令: 52 (52)
结果: ACCEPTED
```

退出紧急制动状态

MAV_DJI_CMD_CANCEL_ARREST_FLYING

示例

```
import time
from pymavlink import mavutil
from enum import IntEnum


# MAV_CMD 执行结果映射
MAV_RESULT_NAMES = {
    mavutil.mavlink.MAV_RESULT_ACCEPTED: "ACCEPTED",
    mavutil.mavlink.MAV_RESULT_TEMPORARILY_REJECTED: "TEMPORARILY_R
EJECTED",
    mavutil.mavlink.MAV_RESULT_DENIED: "DENIED",
    mavutil.mavlink.MAV_RESULT_UNSUPPORTED: "UNSUPPORTED",
    mavutil.mavlink.MAV_RESULT_FAILED: "FAILED",
    mavutil.mavlink.MAV_RESULT_IN_PROGRESS: "IN_PROGRESS",
}


def recv_command_ack(timeout=5):
    """接收并显示 command_ack 消息

    Args:
        timeout: 超时时间（秒）

    Returns:
        ack 消息对象，None 表示超时
    """
    print(f"等待 command_ack 响应 (超时 {timeout}s)...")

    ack = master.recv_match(type='COMMAND_ACK', blocking=True, time
```

```
out=timeout)

    if ack is None:
        print("错误: 命令响应超时")
        return None

    # 获取命令名称
    cmd_name = getattr(ack, 'command_name', None)
    if not cmd_name:
        # 尝试通过枚举查找命令名称
        try:
            cmd_name = mavutil.mavlink.enums.get('MAV_CMD', {}).get
(ack.command, {}).name
        except:
            cmd_name = str(ack.command)

    # 获取结果名称
    result_name = MAV_RESULT_NAMES.get(ack.result, f"UNKNOWN({ack.r
esult})")

    print("=" * 50)
    print(f"命令: {cmd_name} ({ack.command})")
    print(f"结果: {result_name}")
    print(f"进度: {ack.progress * 100:.0f}%" if ack.progress > 0 els
e "")
    # print(f"参数2: {ack.result_param2}")
    print(f"系统ID: {ack.target_system}")
    print(f"组件ID: {ack.target_component}")
    print("=" * 50)

    return ack


class MavDjiCmd(IntEnum):
    MAV_CMD_NAV_LAND = 21
    MAV_DJI_CMD_TRIGGER_FFC=40
    MAV_DJI_CMD_START_RECORD_POINT_CLOUD=41
    MAV_DJI_CMD_STOP_RECORD_POINT_CLOUD=42
    MAV_DJI_CMD_RESTORE_FACTORY_SETTINGS=43
    MAV_DJI_CMD_APPLY_HIGH_POWER_SYNC=44
    MAV_DJI_CMD_APPLY_HIGH_POWER_SYNCV2=45
    MAV_DJI_CMD_START_CONFIRM_LANDING=46
    MAV_DJI_CMD_START_FORCE_LANDING=47
    MAV_DJI_CMD_EXECUTE_EMERGENCY_BRAKE_ACTION=48
    MAV_DJI_CMD_CANCEL_EMERGENCY_BRAKE_ACTION=49
```

```
    MAV_DJI_CMD_START_SLOW_ROTATE_MOTOR=50
    MAV_DJI_CMD_STOP_SLOW_ROTATE_MOTOR=51
    MAV_DJI_CMD_ARREST_FLYING=52
    MAV_DJI_CMD_CANCEL_ARREST_FLYING=53
    MAV_DJI_CMD_OUTPUT_HIGH_POWER=54
    TUNNEL_PROTOCOL_ID_INJECT_HMS_ERROR=32770

timestamp = int(time.time() * 1000)
millis = int(time.time() * 1000)

master = mavutil.mavlink_connection('udpin:0.0.0.0:14550')
print("waiting for heartbeat")
master.wait_heartbeat()
print("heartbeat received")
boot_time = time.time()

def send_command(command, param1=0, param2=0, param3=0, param4=0, p
aram5=0, param6=0, param7=0, wait_ack=True, timeout=5):
    """发送 command_long 并可选接收 command_ack

    Args:
        command: 命令ID (如 MavDjiCmd.TRIGGER_FFC)
        param1~param7: 命令参数
        wait_ack: 是否等待并显示 ACK
        timeout: ACK 超时时间(秒)

    Returns:
        ack 消息对象,None 表示超时或未等待
    """
    print(f"发送命令: {command} ({int(command)})")
    master.mav.command_long_send(
        master.target_system,
        master.target_component,
        int(command),
        0,  # confirmation
        param1, param2, param3, param4, param5, param6, param7
    )

    if wait_ack:
        return recv_command_ack(timeout=timeout)
    return None

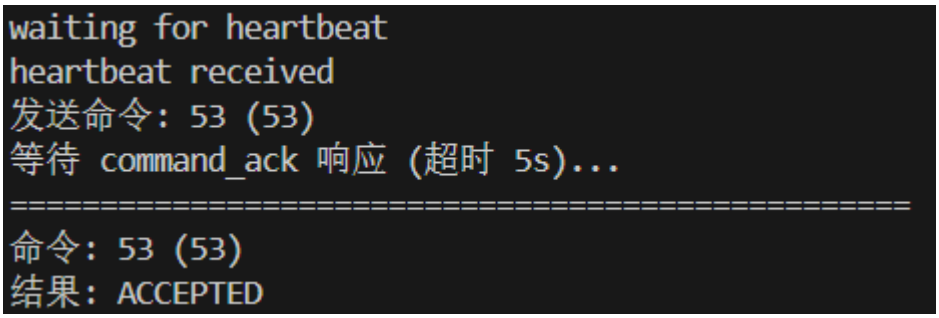

def ready():
    send_command(MavDjiCmd.MAV_DJI_CMD_CANCEL_ARREST_FLYING)
```

```
ready()
```

示例使用说明

获取

运行后，成功建立连接并发送指令



在任何情况下紧急停止电机

MAV_CMD_COMPONENT_ARM_DISARM

示例

```
import time
from pymavlink import mavutil
from enum import IntEnum


# MAV_CMD 执行结果映射
MAV_RESULT_NAMES = {
    mavutil.mavlink.MAV_RESULT_ACCEPTED: "ACCEPTED",
    mavutil.mavlink.MAV_RESULT_TEMPORARILY_REJECTED: "TEMPORARILY_R
EJECTED",
    mavutil.mavlink.MAV_RESULT_DENIED: "DENIED",
    mavutil.mavlink.MAV_RESULT_UNSUPPORTED: "UNSUPPORTED",
    mavutil.mavlink.MAV_RESULT_FAILED: "FAILED",
    mavutil.mavlink.MAV_RESULT_IN_PROGRESS: "IN_PROGRESS",
}


def recv_command_ack(timeout=5):
    """接收并显示 command_ack 消息

    Args:
        timeout: 超时时间（秒）
```

```
    Returns:
        ack 消息对象，None 表示超时
    """
    print(f"等待 command_ack 响应 (超时 {timeout}s)...")

    ack = master.recv_match(type='COMMAND_ACK', blocking=True, timeout=timeout)

    if ack is None:
        print("错误: 命令响应超时")
        return None

    # 获取命令名称
    cmd_name = getattr(ack, 'command_name', None)
    if not cmd_name:
        # 尝试通过枚举查找命令名称
        try:
            cmd_name = mavutil.mavlink.enums.get('MAV_CMD', {}).get(ack.command, {}).name
        except:
            cmd_name = str(ack.command)

    # 获取结果名称
    result_name = MAV_RESULT_NAMES.get(ack.result, f"UNKNOWN({ack.result})")

    print("=" * 50)
    print(f"命令: {cmd_name} ({ack.command})")
    print(f"结果: {result_name}")
    print(f"进度: {ack.progress * 100:.0f}%" if ack.progress > 0 else "")
    # print(f"参数2: {ack.result_param2}")
    print(f"系统ID: {ack.target_system}")
    print(f"组件ID: {ack.target_component}")
    print("=" * 50)

    return ack


class MavDjiCmd(IntEnum):
    MAV_CMD_NAV_LAND = 21
    MAV_DJI_CMD_TRIGGER_FFC=40
    MAV_DJI_CMD_START_RECORD_POINT_CLOUD=41
    MAV_DJI_CMD_STOP_RECORD_POINT_CLOUD=42
    MAV_DJI_CMD_RESTORE_FACTORY_SETTINGS=43
```

```
    MAV_DJI_CMD_APPLY_HIGH_POWER_SYNC=44
    MAV_DJI_CMD_APPLY_HIGH_POWER_SYNCV2=45
    MAV_DJI_CMD_START_CONFIRM_LANDING=46
    MAV_DJI_CMD_START_FORCE_LANDING=47
    MAV_DJI_CMD_EXECUTE_EMERGENCY_BRAKE_ACTION=48
    MAV_DJI_CMD_CANCEL_EMERGENCY_BRAKE_ACTION=49
    MAV_DJI_CMD_START_SLOW_ROTATE_MOTOR=50
    MAV_DJI_CMD_STOP_SLOW_ROTATE_MOTOR=51
    MAV_DJI_CMD_ARREST_FLYING=52
    MAV_DJI_CMD_CANCEL_ARREST_FLYING=53
    MAV_DJI_CMD_OUTPUT_HIGH_POWER=54
    TUNNEL_PROTOCOL_ID_INJECT_HMS_ERROR=32770

timestamp = int(time.time() * 1000)
millis = int(time.time() * 1000)

master = mavutil.mavlink_connection('udpin:0.0.0.0:14550')
print("waiting for heartbeat")
master.wait_heartbeat()
print("heartbeat received")
boot_time = time.time()

def send_command(command, param1=0, param2=0, param3=0, param4=0, p
aram5=0, param6=0, param7=0, wait_ack=True, timeout=5):
    """发送 command_long 并可选接收 command_ack

    Args:
        command: 命令ID (如 MavDjiCmd.TRIGGER_FFC)
        param1~param7: 命令参数
        wait_ack: 是否等待并显示 ACK
        timeout: ACK 超时时间(秒)

    Returns:
        ack 消息对象,None 表示超时或未等待
    """
    print(f"发送命令: {command} ({int(command)})")
    master.mav.command_long_send(
        master.target_system,
        master.target_component,
        int(command),
        0,  # confirmation
        param1, param2, param3, param4, param5, param6, param7
    )

    if wait_ack:
```

```
        return recv_command_ack(timeout=timeout)
    return None

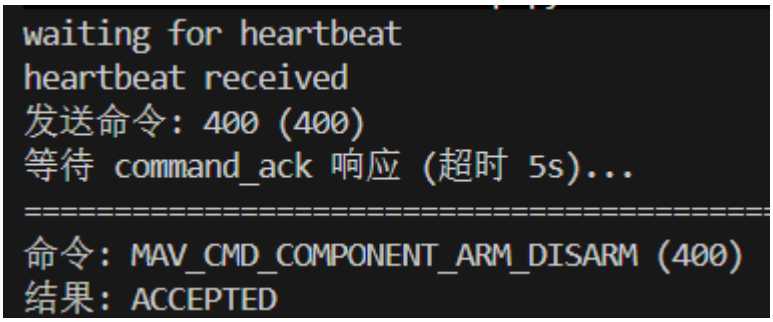

def ready():
    send_command(400,0,21196)

ready()
```

示例使用说明

获取

运行后，成功建立连接并发送指令



注意：紧急停机后电机锁定，必须重启无人机！

无人机在地面时请求起飞

MAV_CMD_NAV_TAKEOFF 22
返回消息：COMMAND_ACK
返回值 command 22 执行命令id
返回值 result 0或4 0：执行成功，4：执行失败

示例

```
import time
from pymavlink import mavutil

timestamp = int(time.time() * 1000)
millis = int(time.time() * 1000)

master = mavutil.mavlink_connection('udpin:0.0.0.0:14550')
master.wait_heartbeat()
boot_time = time.time()

def ready():
    master.mav.command_long_send(
```

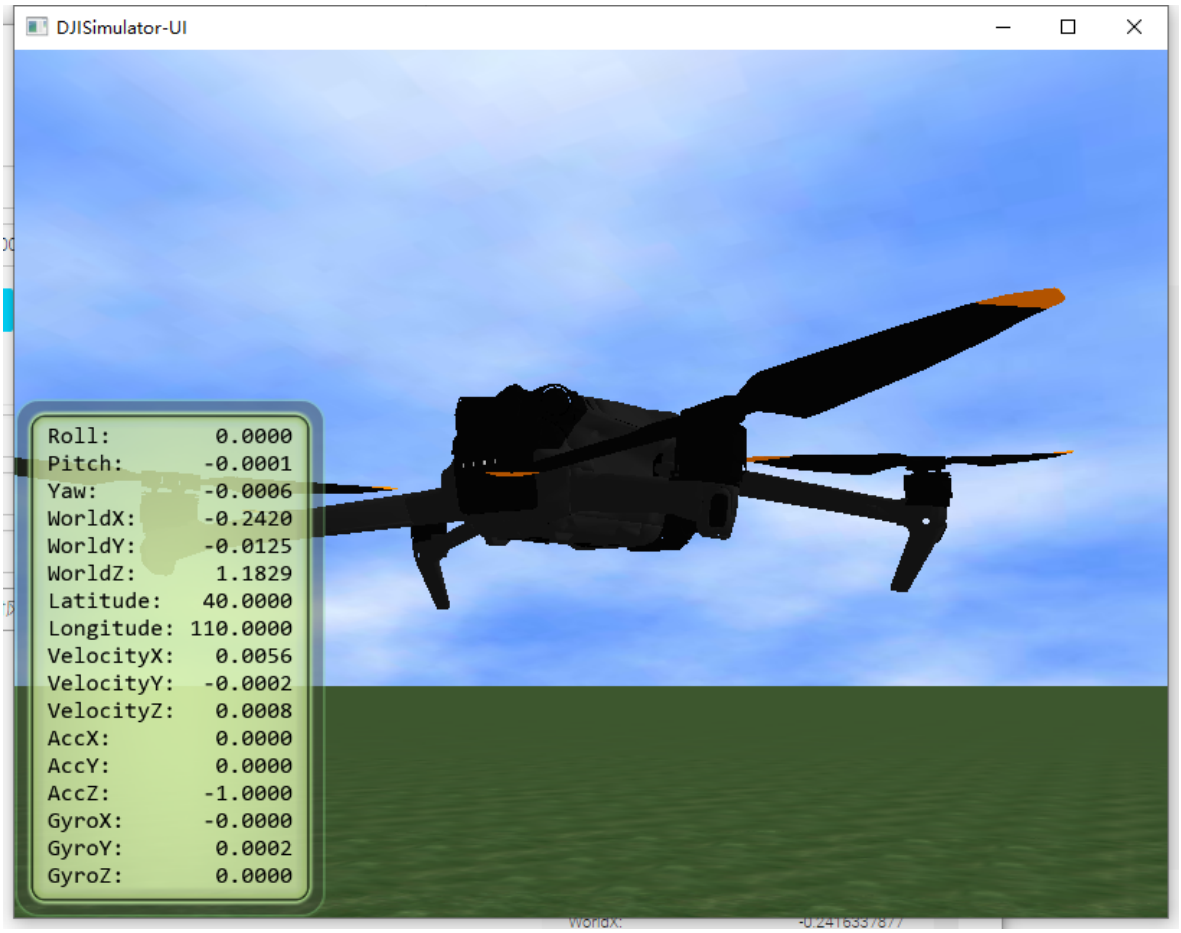
```
        master.target_system,
        master.target_component,
        mavutil.mavlink.MAV_CMD_NAV_TAKEOFF,
        0,
        0, 0, 0, 0, 0, 0, 0
    )

ready()
```

执行示例

```
PS D:\python> & C:\Users\Administrator\AppData\Local\Python\pythoncore-3.14-64\python.exe d:/python/tmp.py
PS D:\python>
```

结果



降落命令

方法1：发送以下降落命令

MAV_CMD_NAV_LAND 21

返回消息：COMMAND_ACK
返回值 command 21 执行命令id
返回值 result 0或4 0：执行成功，4：执行失败

方法2：使用切换飞行模式将飞行模式切换至LAND

详见切换飞行模式页

取消降落命令

方法1：使用切换飞行模式将飞行模式切换至POSCTL
详见切换飞行模式页

方法2：发送以下COMMAND_LONG命令

MAV_CMD_DO_REPOSITION 192

command = MAV_CMD_DO_REPOSITION
param4，param5，param6 = NaN
param7 = 0

返回消息：COMMAND_ACK
返回值 command 192 执行命令id
返回值 result 0或4 0：执行成功，4：执行失败

示例

发送降落命令（方法1）

```
import time
from pymavlink import mavutil

timestamp = int(time.time() * 1000)
millis = int(time.time() * 1000)

master = mavutil.mavlink_connection('udpin:0.0.0.0:14550')
master.wait_heartbeat()
boot_time = time.time()

def land():
    master.mav.command_long_send(
        master.target_system,
```

```
        master.target_component,
        mavutil.mavlink.MAV_CMD_NAV_LAND,
        0,
        0, 0, 0, 0, 0, 0, 0
    )

land()
```

发送取消降落命令（方法2）

```
import time
from pymavlink import mavutil
import math


def get_current_millis():
    """获取当前时间（毫秒级）"""
    return int(time.time() * 1000)


# MAVLink 命令常量
MAV_CMD_DO_REPOSITION = 192
DEFAULT_TIMEOUT = 5.0


def send_reposition_command(master, ground_speed, yaw, flags=0):
    """发送重新定位命令到无人机

    Args:
        master: MAVLink 连接对象
        ground_speed: 地速 (单位: m/s)
        yaw: 偏航角 (单位: 度, 0-360)
        flags: 标志位 (保留, 默认为0)
    """
    nan_value = math.nan
    send_time = get_current_millis()
    print(f"[{send_time}] 发送重新定位命令")
    print(f"  地速 = {ground_speed} m/s, 偏航角 = {yaw}°")

    master.mav.command_long_send(
        master.target_system,
        master.target_component,
        mavutil.mavlink.MAV_CMD_DO_REPOSITION,
        0,  # 确认标志
        ground_speed,   # param1: 地速 (单位: m/s)
```

```
        yaw,            # param2: 偏航角（单位: 度）
        0,              # param3: 保留
        nan_value,      # param4: NaN
        nan_value,      # param5: NaN
        nan_value,      # param6: NaN
        0               # param7: 0
    )

    return send_time


def wait_for_ack(master, send_time, timeout=DEFAULT_TIMEOUT):
    """等待命令确认响应"""
    print("等待重新定位命令响应...")

    ack = master.recv_match(
        type='COMMAND_ACK',
        blocking=True,
        timeout=timeout
    )

    if ack:
        ack_time = get_current_millis()
        delay = ack_time - send_time
        print(f"[{ack_time}] 收到命令应答 (MAV_CMD_DO_REPOSITION)")
        print(f" 命令ID: {ack.command}, 结果: {ack.result}, 延迟: {de
lay}ms")
    else:
        print(f"[{get_current_millis()}] 超时：未收到重新定位命令应答")


def main():
    """主函数：连接无人机并发送重新定位命令"""
    # 连接无人机（UDP端口14550）
    master = mavutil.mavlink_connection('udpin:0.0.0.0:14550')
    print("等待无人机心跳...")
    master.wait_heartbeat()
    print("无人机已连接，开始发送重新定位命令")

    # 发送重新定位命令（示例参数，请根据实际情况修改）
    ground_speed = 5.0   # 地速 5 m/s
    yaw = 90.0           # 偏航角 90°（东向）
    send_time = send_reposition_command(master, ground_speed, yaw)

    # 等待响应
```

```
    wait_for_ack(master, send_time)

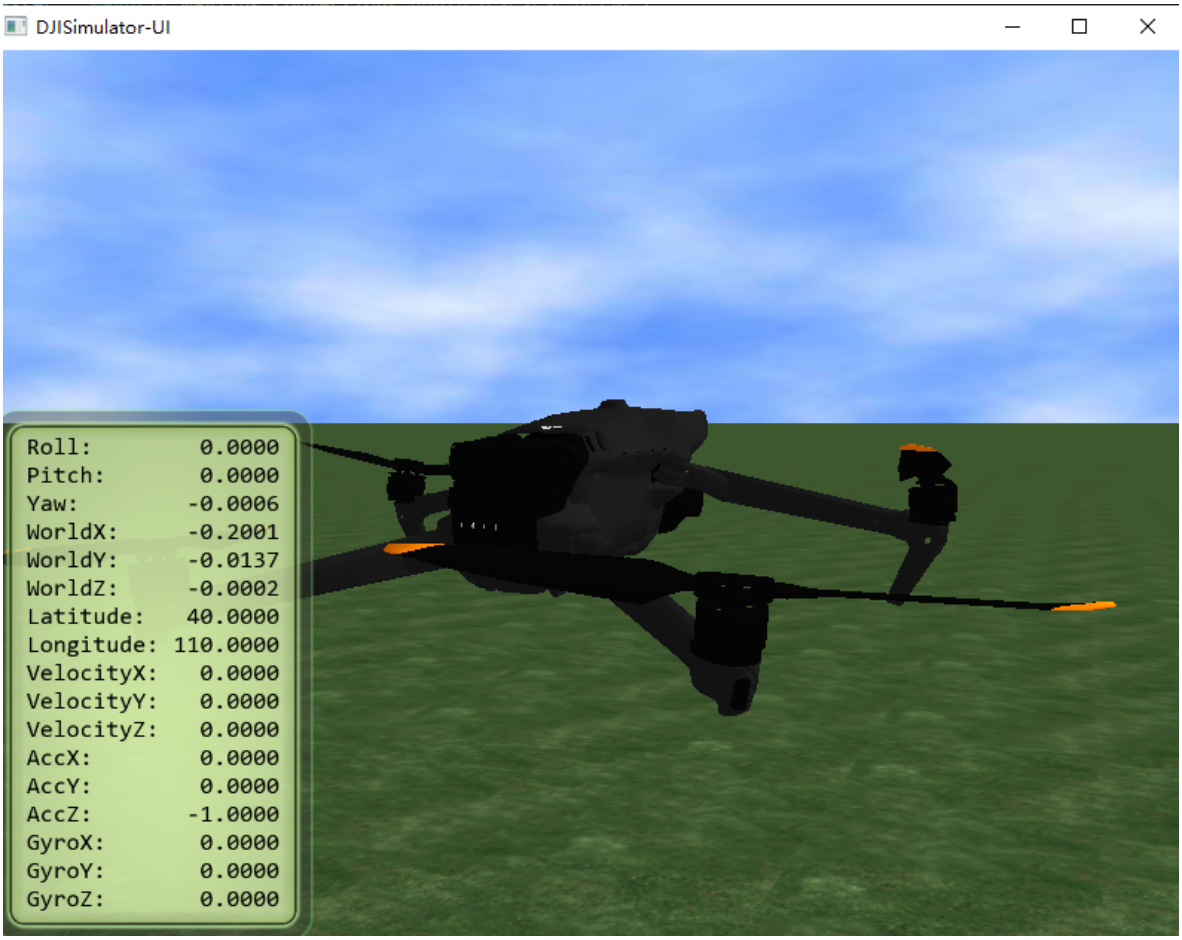
    print("重新定位命令发送完成")


if __name__ == "__main__":
    main()
```

执行降落示例

```
发送 CAMERA_CAPTURE_STATUS: video_status=0, recording_time_ms=0
video stop..
```

结果



当无人机距离地面0.7米时确认着陆

MAV_DJI_CMD_START_CONFIRM_LANDING

示例

```
import time
from pymavlink import mavutil
from enum import IntEnum


# MAV_CMD 执行结果映射
MAV_RESULT_NAMES = {
    mavutil.mavlink.MAV_RESULT_ACCEPTED: "ACCEPTED",
    mavutil.mavlink.MAV_RESULT_TEMPORARILY_REJECTED: "TEMPORARILY_R
EJECTED",
    mavutil.mavlink.MAV_RESULT_DENIED: "DENIED",
    mavutil.mavlink.MAV_RESULT_UNSUPPORTED: "UNSUPPORTED",
    mavutil.mavlink.MAV_RESULT_FAILED: "FAILED",
    mavutil.mavlink.MAV_RESULT_IN_PROGRESS: "IN_PROGRESS",
}


def recv_command_ack(timeout=5):
    """接收并显示 command_ack 消息

    Args:
        timeout: 超时时间（秒）

    Returns:
        ack 消息对象，None 表示超时
    """
    print(f"等待 command_ack 响应 (超时 {timeout}s)...")

    ack = master.recv_match(type='COMMAND_ACK', blocking=True, time
out=timeout)

    if ack is None:
        print("错误: 命令响应超时")
        return None

    # 获取命令名称
    cmd_name = getattr(ack, 'command_name', None)
    if not cmd_name:
        # 尝试通过枚举查找命令名称
        try:
            cmd_name = mavutil.mavlink.enums.get('MAV_CMD', {}).get
(ack.command, {}).name
        except:
            cmd_name = str(ack.command)
```

```
    # 获取结果名称
    result_name = MAV_RESULT_NAMES.get(ack.result, f"UNKNOWN({ack.r
esult})")

    print("=" * 50)
    print(f"命令: {cmd_name} ({ack.command})")
    print(f"结果: {result_name}")
    print(f"进度: {ack.progress * 100:.0f}%" if ack.progress > 0 els
e "")
    # print(f"参数2: {ack.result_param2}")
    print(f"系统ID: {ack.target_system}")
    print(f"组件ID: {ack.target_component}")
    print("=" * 50)

    return ack


class MavDjiCmd(IntEnum):
    MAV_CMD_NAV_LAND = 21
    MAV_DJI_CMD_TRIGGER_FFC=40
    MAV_DJI_CMD_START_RECORD_POINT_CLOUD=41
    MAV_DJI_CMD_STOP_RECORD_POINT_CLOUD=42
    MAV_DJI_CMD_RESTORE_FACTORY_SETTINGS=43
    MAV_DJI_CMD_APPLY_HIGH_POWER_SYNC=44
    MAV_DJI_CMD_APPLY_HIGH_POWER_SYNCV2=45
    MAV_DJI_CMD_START_CONFIRM_LANDING=46
    MAV_DJI_CMD_START_FORCE_LANDING=47
    MAV_DJI_CMD_EXECUTE_EMERGENCY_BRAKE_ACTION=48
    MAV_DJI_CMD_CANCEL_EMERGENCY_BRAKE_ACTION=49
    MAV_DJI_CMD_START_SLOW_ROTATE_MOTOR=50
    MAV_DJI_CMD_STOP_SLOW_ROTATE_MOTOR=51
    MAV_DJI_CMD_ARREST_FLYING=52
    MAV_DJI_CMD_CANCEL_ARREST_FLYING=53
    MAV_DJI_CMD_OUTPUT_HIGH_POWER=54
    TUNNEL_PROTOCOL_ID_INJECT_HMS_ERROR=32770

timestamp = int(time.time() * 1000)
millis = int(time.time() * 1000)

master = mavutil.mavlink_connection('udpin:0.0.0.0:14550')
print("waiting for heartbeat")
master.wait_heartbeat()
print("heartbeat received")
boot_time = time.time()
```

```
def send_command(command, param1=0, param2=0, param3=0, param4=0, p
aram5=0, param6=0, param7=0, wait_ack=True, timeout=5):
    """发送 command_long 并可选接收 command_ack

    Args:
        command: 命令ID (如 MavDjiCmd.TRIGGER_FFC)
        param1~param7: 命令参数
        wait_ack: 是否等待并显示 ACK
        timeout: ACK 超时时间(秒)

    Returns:
        ack 消息对象，None 表示超时或未等待
    """
    print(f"发送命令: {command} ({int(command)})")
    master.mav.command_long_send(
        master.target_system,
        master.target_component,
        int(command),
        0,  # confirmation
        param1, param2, param3, param4, param5, param6, param7
    )

    if wait_ack:
        return recv_command_ack(timeout=timeout)
    return None


def ready():
    send_command(MavDjiCmd.MAV_DJI_CMD_START_CONFIRM_LANDING)

ready()
```

示例使用说明

获取

运行后，成功建立连接并发送指令

```
waiting for heartbeat
heartbeat received
发送命令: 46 (46)
等待 command_ack 响应 (超时 5s)...
==================================================
命令: 46 (46)
结果: ACCEPTED
```

注：仅在下避障开启时有效！

在任何情况下都强制着陆

MAV_DJI_CMD_START_FORCE_LANDING

示例

```
import time
from pymavlink import mavutil
from enum import IntEnum


# MAV_CMD 执行结果映射
MAV_RESULT_NAMES = {
    mavutil.mavlink.MAV_RESULT_ACCEPTED: "ACCEPTED",
    mavutil.mavlink.MAV_RESULT_TEMPORARILY_REJECTED: "TEMPORARILY_R
EJECTED",
    mavutil.mavlink.MAV_RESULT_DENIED: "DENIED",
    mavutil.mavlink.MAV_RESULT_UNSUPPORTED: "UNSUPPORTED",
    mavutil.mavlink.MAV_RESULT_FAILED: "FAILED",
    mavutil.mavlink.MAV_RESULT_IN_PROGRESS: "IN_PROGRESS",
}


def recv_command_ack(timeout=5):
    """接收并显示 command_ack 消息

    Args:
        timeout: 超时时间（秒）

    Returns:
        ack 消息对象，None 表示超时
    """
    print(f"等待 command_ack 响应 (超时 {timeout}s)...")

    ack = master.recv_match(type='COMMAND_ACK', blocking=True, time
out=timeout)

    if ack is None:
        print("错误: 命令响应超时")
        return None

    # 获取命令名称
    cmd_name = getattr(ack, 'command_name', None)
    if not cmd_name:
        # 尝试通过枚举查找命令名称
```

```
        try:
            cmd_name = mavutil.mavlink.enums.get('MAV_CMD', {}).get
(ack.command, {}).name
        except:
            cmd_name = str(ack.command)

    # 获取结果名称
    result_name = MAV_RESULT_NAMES.get(ack.result, f"UNKNOWN({ack.r
esult})")

    print("=" * 50)
    print(f"命令: {cmd_name} ({ack.command})")
    print(f"结果: {result_name}")
    print(f"进度: {ack.progress * 100:.0f}%" if ack.progress > 0 els
e "")
    # print(f"参数2: {ack.result_param2}")
    print(f"系统ID: {ack.target_system}")
    print(f"组件ID: {ack.target_component}")
    print("=" * 50)

    return ack


class MavDjiCmd(IntEnum):
    MAV_CMD_NAV_LAND = 21
    MAV_DJI_CMD_TRIGGER_FFC=40
    MAV_DJI_CMD_START_RECORD_POINT_CLOUD=41
    MAV_DJI_CMD_STOP_RECORD_POINT_CLOUD=42
    MAV_DJI_CMD_RESTORE_FACTORY_SETTINGS=43
    MAV_DJI_CMD_APPLY_HIGH_POWER_SYNC=44
    MAV_DJI_CMD_APPLY_HIGH_POWER_SYNCV2=45
    MAV_DJI_CMD_START_CONFIRM_LANDING=46
    MAV_DJI_CMD_START_FORCE_LANDING=47
    MAV_DJI_CMD_EXECUTE_EMERGENCY_BRAKE_ACTION=48
    MAV_DJI_CMD_CANCEL_EMERGENCY_BRAKE_ACTION=49
    MAV_DJI_CMD_START_SLOW_ROTATE_MOTOR=50
    MAV_DJI_CMD_STOP_SLOW_ROTATE_MOTOR=51
    MAV_DJI_CMD_ARREST_FLYING=52
    MAV_DJI_CMD_CANCEL_ARREST_FLYING=53
    MAV_DJI_CMD_OUTPUT_HIGH_POWER=54
    TUNNEL_PROTOCOL_ID_INJECT_HMS_ERROR=32770

timestamp = int(time.time() * 1000)
millis = int(time.time() * 1000)
```

```
master = mavutil.mavlink_connection('udpin:0.0.0.0:14550')
print("waiting for heartbeat")
master.wait_heartbeat()
print("heartbeat received")
boot_time = time.time()

def send_command(command, param1=0, param2=0, param3=0, param4=0, p
aram5=0, param6=0, param7=0, wait_ack=True, timeout=5):
    """发送 command_long 并可选接收 command_ack

    Args:
        command: 命令ID (如 MavDjiCmd.TRIGGER_FFC)
        param1~param7: 命令参数
        wait_ack: 是否等待并显示 ACK
        timeout: ACK 超时时间(秒)

    Returns:
        ack 消息对象,None 表示超时或未等待
    """
    print(f"发送命令: {command} ({int(command)})")
    master.mav.command_long_send(
        master.target_system,
        master.target_component,
        int(command),
        0,  # confirmation
        param1, param2, param3, param4, param5, param6, param7
    )

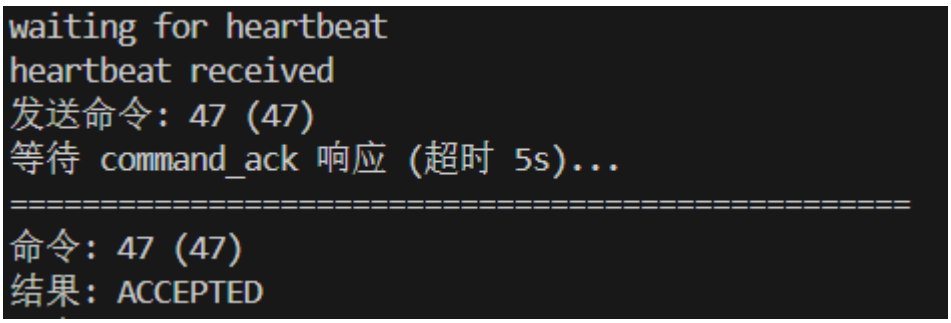
    if wait_ack:
        return recv_command_ack(timeout=timeout)
    return None


def ready():
    send_command(MavDjiCmd.MAV_DJI_CMD_START_FORCE_LANDING)

ready()
```

示例使用说明

运行后，成功建立连接并发送指令



注：该功能仅在无人机处于空中时可执行。

返航高度

RTL_RETURN_ALT

设置

mavlink消息：

接收PARAM_SET
发送PARAM_VALUE

获取

mavlink消息

接收PARAM_REQUEST_READ
发送PARAM_VALUE

参数：

名称	类型	单位	描述
GoHomeAltitude	uint16_t	m	返航高度。取值范围：20～500 米。

示例

```
import time
import sys
from pymavlink import mavutil

def connect_to_vehicle():
    """建立 UDP 监听连接（默认监听所有接口的 14550 端口）"""
    connection_string = "udpin:0.0.0.0:14550"
    print(f"Listening for MAVLink on: {connection_string}")
```

```
    vehicle = mavutil.mavlink_connection(connection_string)
    print("Waiting for HEARTBEAT (timeout: 10s)...")
    try:
        vehicle.wait_heartbeat(timeout=10)
        print(f"  Heartbeat received from system {vehicle.target_sy
stem}, component {vehicle.target_component}")
        return vehicle
    except Exception as e:
        print(f"  Connection failed: {str(e)}")
        print("Ensure your vehicle is sending MAVLink messages to 1
4550 port")
        print("Example QGroundControl setup: Settings > General > M
AVLink > Listening Port = 14550")
        sys.exit(1)

def format_param_id(param_id):
    """格式化16字节参数ID（符合MAVLink协议要求）"""
    if len(param_id) > 16:
        raise ValueError("param_id cannot exceed 16 characters")

    param_id_bytes = bytearray(16)
    for i in range(min(len(param_id), 16)):
        param_id_bytes[i] = ord(param_id[i])

    # 长度<16时添加NULL终止符
    if len(param_id) < 16:
        param_id_bytes[len(param_id)] = 0

    return param_id_bytes

def send_param_set(vehicle, param_id, param_value, param_type):
    """发送参数设置请求"""
    vehicle.mav.param_set_send(
        vehicle.target_system,
        vehicle.target_component,
        format_param_id(param_id),
        param_value,
        param_type
    )
    print(f"  Sent PARAM_SET for '{param_id}' = {param_value} (typ
e: {param_type})")

def send_param_request_read(vehicle, param_id=None, param_index=-
1):
    """发送参数读取请求"""
```

```
    param_id_bytes = format_param_id(param_id) if param_id else byt
earray(16)

    vehicle.mav.param_request_read_send(
        vehicle.target_system,
        vehicle.target_component,
        param_id_bytes,
        param_index
    )

    target = f"index {param_index}" if param_index >= 0 else f"ID
'{param_id}'"
    print(f"  Sent PARAM_REQUEST_READ for {target}")

def send_param_request_list(vehicle):
    """发送请求所有参数列表"""
    vehicle.mav.param_request_list_send(
        vehicle.target_system,
        vehicle.target_component
    )
    print(f"  Sent PARAM_REQUEST_LIST to system {vehicle.target_sys
tem}, component {vehicle.target_component}")

def receive_param_value(vehicle, timeout=5.0):
    """接收参数值响应"""
    print(f"  Waiting for PARAM_VALUE ({timeout}s timeout)...")
    msg = vehicle.recv_match(type='PARAM_VALUE', blocking=True, tim
eout=timeout)

    if not msg:
        print("  Timeout: No PARAM_VALUE received")
        return None

    # 解析参数ID
    param_id = msg.param_id

    # 解析参数值
    param_value = msg.param_value

    # 参数类型映射
    type_names = {
        0: "INT8", 1: "UINT8", 2: "INT16", 3: "UINT16", 4: "INT32",
        5: "UINT32", 6: "INT64", 7: "UINT64", 8: "FLOAT", 9: "DOUBL
E"
    }
```

```
    param_type = type_names.get(msg.param_type, f"UNKNOWN({msg.para
m_type})")

    print("\n  Received PARAM_VALUE:")
    print(f"    ID: {param_id}")
    print(f"    Value: {param_value}")
    print(f"    Type: {param_type}")
    print(f"    Index/Total: {msg.param_index}/{msg.param_count}")

    return {
        'id': param_id,
        'value': param_value,
        'type': param_type,
        'index': msg.param_index,
        'count': msg.param_count
    }

def receive_all_param_values(vehicle, timeout=10.0):
    """接收所有参数值，直到收到 param_count 条消息后退出循环"""
    received_params = []
    expected_count = None
    print(f"  Waiting for all PARAM_VALUE messages...")

    while True:
        msg = vehicle.recv_match(type='PARAM_VALUE', blocking=True,
timeout=timeout)

        if not msg:
            print(f"  Timeout: Received {len(received_params)} para
meters (expected {expected_count if expected_count else 'unknow
n'})")
            break

        param_id = msg.param_id
        param_value = msg.param_value

        # 参数类型映射
        type_names = {
            0: "INT8", 1: "UINT8", 2: "INT16", 3: "UINT16", 4: "INT
32",
            5: "UINT32", 6: "INT64", 7: "UINT64", 8: "FLOAT", 9: "D
OUBLE"
        }
        param_type = type_names.get(msg.param_type, f"UNKNOWN({msg.
param_type})")
```

```
            # 首次收到消息时记录总参数数量
            if expected_count is None:
                expected_count = msg.param_count
                print(f"  Total parameters to receive: {expected_coun
t}")

            print(f"  [{msg.param_index}/{expected_count}] {param_id} =
{param_value} ({param_type})")

            received_params.append({
                'id': param_id,
                'value': param_value,
                'type': param_type,
                'index': msg.param_index,
                'count': msg.param_count
            })

            # 当已接收数量达到预期总数时退出循环
            if len(received_params) >= expected_count:
                print(f"\n  Successfully received all {expected_count}
parameters")
                break

    return received_params

def main():
    print("  Starting MAVLink PARAM_SET Tool (UDP Listening Mode)")
    print("------------------------------------------------------------")
    print("This script listens for MAVLink on ALL interfaces:1455
0")
    print("Ensure your vehicle is configured to send to this port")
    print("------------------------------------------------------------
\n")

    vehicle = connect_to_vehicle()

    # 操作选择
    print("\nAvailable operations:")
    print("1. Set parameter value")
    print("2. Read parameter value (by ID or index)")
    print("3. Request all parameters (PARAM_REQUEST_LIST)")
    choice = input("\nSelect operation [1-3]: ").strip()

    if choice == '1':
```

```
        param_id = input("Enter parameter ID to set: ").strip()
        param_value = float(input("Enter new value: ").strip())

        print("\nParameter types:")
        print("  8 = FLOAT (most common)")
        print("  9 = DOUBLE")
        print("  4 = INT32")
        print("  1 = UINT8")
        param_type = int(input("Enter type number [8]: ").strip() or 8)

        send_param_set(vehicle, param_id, param_value, param_type)
        receive_param_value(vehicle)

    elif choice == '2':
        # mode = input("Read by [id/index]: ").lower()
        mode = 'id'
        if mode == 'id':
            param_id = input("Enter parameter ID: ").strip()
            send_param_request_read(vehicle, param_id=param_id)
            receive_param_value(vehicle)
        elif mode == 'index':
            param_index = int(input("Enter parameter index: "))
            send_param_request_read(vehicle, param_index=param_index)
            receive_param_value(vehicle)
        else:
            print("Invalid mode. Use 'id' or 'index'")

    elif choice == '3':
        print("\n  Requesting all parameters...")
        send_param_request_list(vehicle)
        all_params = receive_all_param_values(vehicle)
        print(f"\n  Total parameters received: {len(all_params)}")

    else:
        print("Invalid choice. Select 1, 2, or 3.")


if __name__ == "__main__":
    main()
```

示例使用说明

获取

运行后，成功建立连接

```
🚀 Starting MAVLink PARAM_SET Tool (UDP Listening Mode)
--------------------------------------------------
This script listens for MAVLink on ALL interfaces:14550
Ensure your vehicle is configured to send to this port
--------------------------------------------------

Listening for MAVLink on: udpin:0.0.0.0:14550
Waiting for HEARTBEAT (timeout: 10s)...
✅ Heartbeat received from system 3, component 0

Available operations:
1. Set parameter value
2. Read parameter value (by ID or index)
3. Request all parameters (PARAM_REQUEST_LIST)
```

输入“2”选择读取参数

```
Select operation [1-3]: 2
```

输入对应参数“RTL_RETURN_ALT”

```
Enter parameter ID: RTL_RETURN_ALT
```

成功返回该参数对应的数值

```
📥 Received PARAM_VALUE:
  🆔 ID: RTL_RETURN_ALT
  🔢 Value: 20.0
```

设置

运行后，成功建立连接

```
🚀 Starting MAVLink PARAM_SET Tool (UDP Listening Mode)
--------------------------------------------------
This script listens for MAVLink on ALL interfaces:14550
Ensure your vehicle is configured to send to this port
--------------------------------------------------

Listening for MAVLink on: udpin:0.0.0.0:14550
Waiting for HEARTBEAT (timeout: 10s)...
✅ Heartbeat received from system 3, component 0

Available operations:
1. Set parameter value
2. Read parameter value (by ID or index)
3. Request all parameters (PARAM_REQUEST_LIST)
```

输入“1”选择设置参数

```
Available operations:
1. Set parameter value
2. Read parameter value (by ID or index)
3. Request all parameters (PARAM_REQUEST_LIST)

Select operation [1-3]: 1
```

输入对应参数“RTL_RETURN_ALT”

```
Enter parameter ID to set: RTL_RETURN_ALT
```

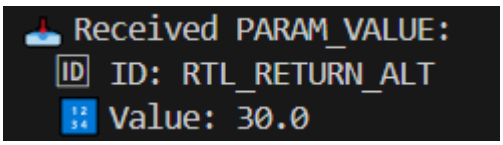
输入要设置的数值

```
Enter new value: 30
```

选择参数类型“1”

```
Parameter types:
  8 = FLOAT (most common)
  9 = DOUBLE
  4 = INT32
  1 = UINT8
Enter type number [8]: 1
```

成功返回设置值



获取国家码

COUNTRY_CODE

获取

mavlink消息

接收PARAM_REQUEST_READ
发送PARAM_VALUE

参数：

参数名	类型	描述
countryCode	uint16_t	返回国家/地区代码。

表示飞行器当前所在的国家或地区，具体含义请参考 ISO 3166-1 国家代码标准。

示例

```
import time
import sys
from pymavlink import mavutil

def connect_to_vehicle():
    """建立 UDP 监听连接（默认监听所有接口的 14550 端口）"""
    connection_string = "udpin:0.0.0.0:14550"
    print(f"Listening for MAVLink on: {connection_string}")
    vehicle = mavutil.mavlink_connection(connection_string)
    print("Waiting for HEARTBEAT (timeout: 10s)...")
    try:
        vehicle.wait_heartbeat(timeout=10)
        print(f"  Heartbeat received from system {vehicle.target_sy
stem}, component {vehicle.target_component}")
        return vehicle
    except Exception as e:
        print(f"  Connection failed: {str(e)}")
        print("Ensure your vehicle is sending MAVLink messages to 1
```

```
4550 port")
        print("Example QGroundControl setup: Settings > General > M
AVLink > Listening Port = 14550")
        sys.exit(1)

def format_param_id(param_id):
    """格式化16字节参数ID（符合MAVLink协议要求）"""
    if len(param_id) > 16:
        raise ValueError("param_id cannot exceed 16 characters")

    param_id_bytes = bytearray(16)
    for i in range(min(len(param_id), 16)):
        param_id_bytes[i] = ord(param_id[i])

    # 长度<16时添加NULL终止符
    if len(param_id) < 16:
        param_id_bytes[len(param_id)] = 0

    return param_id_bytes

def send_param_set(vehicle, param_id, param_value, param_type):
    """发送参数设置请求"""
    vehicle.mav.param_set_send(
        vehicle.target_system,
        vehicle.target_component,
        format_param_id(param_id),
        param_value,
        param_type
    )
    print(f"  Sent PARAM_SET for '{param_id}' = {param_value} (typ
e: {param_type})")

def send_param_request_read(vehicle, param_id=None, param_index=-
1):
    """发送参数读取请求"""
    param_id_bytes = format_param_id(param_id) if param_id else byt
earray(16)

    vehicle.mav.param_request_read_send(
        vehicle.target_system,
        vehicle.target_component,
        param_id_bytes,
        param_index
    )
```

```
    target = f"index {param_index}" if param_index >= 0 else f"ID
'{param_id}'"
    print(f"  Sent PARAM_REQUEST_READ for {target}")

def send_param_request_list(vehicle):
    """发送请求所有参数列表"""
    vehicle.mav.param_request_list_send(
        vehicle.target_system,
        vehicle.target_component
    )
    print(f"  Sent PARAM_REQUEST_LIST to system {vehicle.target_sys
tem}, component {vehicle.target_component}")

def receive_param_value(vehicle, timeout=5.0):
    """接收参数值响应"""
    print(f"  Waiting for PARAM_VALUE ({timeout}s timeout)...")
    msg = vehicle.recv_match(type='PARAM_VALUE', blocking=True, tim
eout=timeout)

    if not msg:
        print("  Timeout: No PARAM_VALUE received")
        return None

    # 解析参数ID
    param_id = msg.param_id

    # 解析参数值
    param_value = msg.param_value

    # 参数类型映射
    type_names = {
        0: "INT8", 1: "UINT8", 2: "INT16", 3: "UINT16", 4: "INT32",
        5: "UINT32", 6: "INT64", 7: "UINT64", 8: "FLOAT", 9: "DOUBL
E"
    }
    param_type = type_names.get(msg.param_type, f"UNKNOWN({msg.para
m_type})")

    print("\n  Received PARAM_VALUE:")
    print(f"    ID: {param_id}")
    print(f"    Value: {param_value}")
    print(f"    Type: {param_type}")
    print(f"    Index/Total: {msg.param_index}/{msg.param_count}")

    return {
```

```
        'id': param_id,
        'value': param_value,
        'type': param_type,
        'index': msg.param_index,
        'count': msg.param_count
    }

def receive_all_param_values(vehicle, timeout=10.0):
    """接收所有参数值，直到收到 param_count 条消息后退出循环"""
    received_params = []
    expected_count = None
    print(f"  Waiting for all PARAM_VALUE messages...")

    while True:
        msg = vehicle.recv_match(type='PARAM_VALUE', blocking=True,
timeout=timeout)

        if not msg:
            print(f"  Timeout: Received {len(received_params)} para
meters (expected {expected_count if expected_count else 'unknow
n'})")
            break

        param_id = msg.param_id
        param_value = msg.param_value

        # 参数类型映射
        type_names = {
            0: "INT8", 1: "UINT8", 2: "INT16", 3: "UINT16", 4: "INT
32",
            5: "UINT32", 6: "INT64", 7: "UINT64", 8: "FLOAT", 9: "D
OUBLE"
        }
        param_type = type_names.get(msg.param_type, f"UNKNOWN({msg.
param_type})")

        # 首次收到消息时记录总参数数量
        if expected_count is None:
            expected_count = msg.param_count
            print(f"  Total parameters to receive: {expected_coun
t}")

        print(f"  [{msg.param_index}/{expected_count}] {param_id} =
{param_value} ({param_type})")
```

```
            received_params.append({
                'id': param_id,
                'value': param_value,
                'type': param_type,
                'index': msg.param_index,
                'count': msg.param_count
            })

            # 当已接收数量达到预期总数时退出循环
            if len(received_params) >= expected_count:
                print(f"\n  Successfully received all {expected_count} parameters")
                break

    return received_params

def main():
    print("  Starting MAVLink PARAM_SET Tool (UDP Listening Mode)")
    print("------------------------------------------------------")
    print("This script listens for MAVLink on ALL interfaces:14550")
    print("Ensure your vehicle is configured to send to this port")
    print("------------------------------------------------------\n")

    vehicle = connect_to_vehicle()

    # 操作选择
    print("\nAvailable operations:")
    print("1. Set parameter value")
    print("2. Read parameter value (by ID or index)")
    print("3. Request all parameters (PARAM_REQUEST_LIST)")
    choice = input("\nSelect operation [1-3]: ").strip()

    if choice == '1':
        param_id = input("Enter parameter ID to set: ").strip()
        param_value = float(input("Enter new value: ").strip())

        print("\nParameter types:")
        print("  8 = FLOAT (most common)")
        print("  9 = DOUBLE")
        print("  4 = INT32")
        print("  1 = UINT8")
        param_type = int(input("Enter type number [8]: ").strip() or 8)
```

```
        send_param_set(vehicle, param_id, param_value, param_type)
        receive_param_value(vehicle)

    elif choice == '2':
        # mode = input("Read by [id/index]: ").lower()
        mode = 'id'
        if mode == 'id':
            param_id = input("Enter parameter ID: ").strip()
            send_param_request_read(vehicle, param_id=param_id)
            receive_param_value(vehicle)
        elif mode == 'index':
            param_index = int(input("Enter parameter index: "))
            send_param_request_read(vehicle, param_index=param_inde
x)
            receive_param_value(vehicle)
        else:
            print("Invalid mode. Use 'id' or 'index'")

    elif choice == '3':
        print("\n  Requesting all parameters...")
        send_param_request_list(vehicle)
        all_params = receive_all_param_values(vehicle)
        print(f"\n  Total parameters received: {len(all_params)}")

    else:
        print("Invalid choice. Select 1, 2, or 3.")


if __name__ == "__main__":
    main()
```

示例使用说明

获取

运行后，成功建立连接

```
🚀 Starting MAVLink PARAM_SET Tool (UDP Listening Mode)
--------------------------------------------------
This script listens for MAVLink on ALL interfaces:14550
Ensure your vehicle is configured to send to this port
--------------------------------------------------

Listening for MAVLink on: udpin:0.0.0.0:14550
Waiting for HEARTBEAT (timeout: 10s)...
✅ Heartbeat received from system 3, component 0

Available operations:
1. Set parameter value
2. Read parameter value (by ID or index)
3. Request all parameters (PARAM_REQUEST_LIST)
```

输入“2”选择读取参数

```
Select operation [1-3]: 2
```

输入对应参数“COUNTRY_CODE”

```
Enter parameter ID: COUNTRY_CODE
```

成功返回该参数对应的数值

```
📥 Received PARAM_VALUE:
  🆔 ID: COUNTRY_CODE
  🔢 Value: 156.0
```

请求紧急制动动作

MAV_DJI_CMD_EXECUTE_EMERGENCY_BRAKE_ACTION

示例

```python
import time
from pymavlink import mavutil
from enum import IntEnum


# MAV_CMD 执行结果映射
MAV_RESULT_NAMES = {
    mavutil.mavlink.MAV_RESULT_ACCEPTED: "ACCEPTED",
```

```
    mavutil.mavlink.MAV_RESULT_TEMPORARILY_REJECTED: "TEMPORARILY_R
EJECTED",
    mavutil.mavlink.MAV_RESULT_DENIED: "DENIED",
    mavutil.mavlink.MAV_RESULT_UNSUPPORTED: "UNSUPPORTED",
    mavutil.mavlink.MAV_RESULT_FAILED: "FAILED",
    mavutil.mavlink.MAV_RESULT_IN_PROGRESS: "IN_PROGRESS",
}


def recv_command_ack(timeout=5):
    """接收并显示 command_ack 消息

    Args:
        timeout: 超时时间（秒）

    Returns:
        ack 消息对象，None 表示超时
    """
    print(f"等待 command_ack 响应 (超时 {timeout}s)...")

    ack = master.recv_match(type='COMMAND_ACK', blocking=True, time
out=timeout)

    if ack is None:
        print("错误: 命令响应超时")
        return None

    # 获取命令名称
    cmd_name = getattr(ack, 'command_name', None)
    if not cmd_name:
        # 尝试通过枚举查找命令名称
        try:
            cmd_name = mavutil.mavlink.enums.get('MAV_CMD', {}).get
(ack.command, {}).name
        except:
            cmd_name = str(ack.command)

    # 获取结果名称
    result_name = MAV_RESULT_NAMES.get(ack.result, f"UNKNOWN({ack.r
esult})")

    print("=" * 50)
    print(f"命令: {cmd_name} ({ack.command})")
    print(f"结果: {result_name}")
    print(f"进度: {ack.progress * 100:.0f}%" if ack.progress > 0 els
```

```
e "")
    # print(f"参数2: {ack.result_param2}")
    print(f"系统ID: {ack.target_system}")
    print(f"组件ID: {ack.target_component}")
    print("=" * 50)

    return ack


class MavDjiCmd(IntEnum):
    MAV_CMD_NAV_LAND = 21
    MAV_DJI_CMD_TRIGGER_FFC=40
    MAV_DJI_CMD_START_RECORD_POINT_CLOUD=41
    MAV_DJI_CMD_STOP_RECORD_POINT_CLOUD=42
    MAV_DJI_CMD_RESTORE_FACTORY_SETTINGS=43
    MAV_DJI_CMD_APPLY_HIGH_POWER_SYNC=44
    MAV_DJI_CMD_APPLY_HIGH_POWER_SYNCV2=45
    MAV_DJI_CMD_START_CONFIRM_LANDING=46
    MAV_DJI_CMD_START_FORCE_LANDING=47
    MAV_DJI_CMD_EXECUTE_EMERGENCY_BRAKE_ACTION=48
    MAV_DJI_CMD_CANCEL_EMERGENCY_BRAKE_ACTION=49
    MAV_DJI_CMD_START_SLOW_ROTATE_MOTOR=50
    MAV_DJI_CMD_STOP_SLOW_ROTATE_MOTOR=51
    MAV_DJI_CMD_ARREST_FLYING=52
    MAV_DJI_CMD_CANCEL_ARREST_FLYING=53
    MAV_DJI_CMD_OUTPUT_HIGH_POWER=54
    TUNNEL_PROTOCOL_ID_INJECT_HMS_ERROR=32770

timestamp = int(time.time() * 1000)
millis = int(time.time() * 1000)

master = mavutil.mavlink_connection('udpin:0.0.0.0:14550')
print("waiting for heartbeat")
master.wait_heartbeat()
print("heartbeat received")
boot_time = time.time()

def send_command(command, param1=0, param2=0, param3=0, param4=0, p
aram5=0, param6=0, param7=0, wait_ack=True, timeout=5):
    """发送 command_long 并可选接收 command_ack

    Args:
        command: 命令ID (如 MavDjiCmd.TRIGGER_FFC)
        param1~param7: 命令参数
        wait_ack: 是否等待并显示 ACK
```

```
        timeout: ACK 超时时间（秒）

    Returns:
        ack 消息对象，None 表示超时或未等待
    """
    print(f"发送命令: {command} ({int(command)})")
    master.mav.command_long_send(
        master.target_system,
        master.target_component,
        int(command),
        0,  # confirmation
        param1, param2, param3, param4, param5, param6, param7
    )

    if wait_ack:
        return recv_command_ack(timeout=timeout)
    return None


def ready():
    send_command(MavDjiCmd.MAV_DJI_CMD_EXECUTE_EMERGENCY_BRAKE_ACTI
ON)

ready()
```

示例使用说明

获取

运行后，成功建立连接并发送指令

```
waiting for heartbeat
heartbeat received
发送命令: 48 (48)
等待 command_ack 响应 (超时 5s)...
==================================================
命令: 48 (48)
结果: ACCEPTED
```

请求取消紧急制动动作

MAV_DJI_CMD_CANCEL_EMERGENCY_BRAKE_ACTION

示例

```
import time
from pymavlink import mavutil
from enum import IntEnum


# MAV_CMD 执行结果映射
MAV_RESULT_NAMES = {
    mavutil.mavlink.MAV_RESULT_ACCEPTED: "ACCEPTED",
    mavutil.mavlink.MAV_RESULT_TEMPORARILY_REJECTED: "TEMPORARILY_R
EJECTED",
    mavutil.mavlink.MAV_RESULT_DENIED: "DENIED",
    mavutil.mavlink.MAV_RESULT_UNSUPPORTED: "UNSUPPORTED",
    mavutil.mavlink.MAV_RESULT_FAILED: "FAILED",
    mavutil.mavlink.MAV_RESULT_IN_PROGRESS: "IN_PROGRESS",
}


def recv_command_ack(timeout=5):
    """接收并显示 command_ack 消息

    Args:
        timeout: 超时时间（秒）

    Returns:
        ack 消息对象，None 表示超时
    """
    print(f"等待 command_ack 响应 (超时 {timeout}s)...")

    ack = master.recv_match(type='COMMAND_ACK', blocking=True, time
out=timeout)

    if ack is None:
        print("错误: 命令响应超时")
        return None

    # 获取命令名称
    cmd_name = getattr(ack, 'command_name', None)
    if not cmd_name:
        # 尝试通过枚举查找命令名称
        try:
            cmd_name = mavutil.mavlink.enums.get('MAV_CMD', {}).get
(ack.command, {}).name
        except:
            cmd_name = str(ack.command)
```

```
    # 获取结果名称
    result_name = MAV_RESULT_NAMES.get(ack.result, f"UNKNOWN({ack.r
esult})")

    print("=" * 50)
    print(f"命令: {cmd_name} ({ack.command})")
    print(f"结果: {result_name}")
    print(f"进度: {ack.progress * 100:.0f}%" if ack.progress > 0 els
e "")
    # print(f"参数2: {ack.result_param2}")
    print(f"系统ID: {ack.target_system}")
    print(f"组件ID: {ack.target_component}")
    print("=" * 50)

    return ack


class MavDjiCmd(IntEnum):
    MAV_CMD_NAV_LAND = 21
    MAV_DJI_CMD_TRIGGER_FFC=40
    MAV_DJI_CMD_START_RECORD_POINT_CLOUD=41
    MAV_DJI_CMD_STOP_RECORD_POINT_CLOUD=42
    MAV_DJI_CMD_RESTORE_FACTORY_SETTINGS=43
    MAV_DJI_CMD_APPLY_HIGH_POWER_SYNC=44
    MAV_DJI_CMD_APPLY_HIGH_POWER_SYNCV2=45
    MAV_DJI_CMD_START_CONFIRM_LANDING=46
    MAV_DJI_CMD_START_FORCE_LANDING=47
    MAV_DJI_CMD_EXECUTE_EMERGENCY_BRAKE_ACTION=48
    MAV_DJI_CMD_CANCEL_EMERGENCY_BRAKE_ACTION=49
    MAV_DJI_CMD_START_SLOW_ROTATE_MOTOR=50
    MAV_DJI_CMD_STOP_SLOW_ROTATE_MOTOR=51
    MAV_DJI_CMD_ARREST_FLYING=52
    MAV_DJI_CMD_CANCEL_ARREST_FLYING=53
    MAV_DJI_CMD_OUTPUT_HIGH_POWER=54
    TUNNEL_PROTOCOL_ID_INJECT_HMS_ERROR=32770

timestamp = int(time.time() * 1000)
millis = int(time.time() * 1000)

master = mavutil.mavlink_connection('udpin:0.0.0.0:14550')
print("waiting for heartbeat")
master.wait_heartbeat()
print("heartbeat received")
boot_time = time.time()
```

```
def send_command(command, param1=0, param2=0, param3=0, param4=0, p
aram5=0, param6=0, param7=0, wait_ack=True, timeout=5):
    """发送 command_long 并可选接收 command_ack

    Args:
        command: 命令ID (如 MavDjiCmd.TRIGGER_FFC)
        param1~param7: 命令参数
        wait_ack: 是否等待并显示 ACK
        timeout: ACK 超时时间(秒)

    Returns:
        ack 消息对象,None 表示超时或未等待
    """
    print(f"发送命令: {command} ({int(command)})")
    master.mav.command_long_send(
        master.target_system,
        master.target_component,
        int(command),
        0,  # confirmation
        param1, param2, param3, param4, param5, param6, param7
    )

    if wait_ack:
        return recv_command_ack(timeout=timeout)
    return None


def ready():
    send_command(MavDjiCmd.MAV_DJI_CMD_CANCEL_EMERGENCY_BRAKE_ACTIO
N)

ready()
```

示例使用说明

获取

运行后，成功建立连接并发送指令

```
waiting for heartbeat
heartbeat received
发送命令: 49 (49)
等待 command_ack 响应 (超时 5s)...
================================================
命令: 49 (49)
结果: ACCEPTED
```

遥控器失联动作使能

RC_LOST_ENABLE

设置

mavlink消息：

接收PARAM_SET
发送PARAM_VALUE

获取

mavlink消息

接收PARAM_REQUEST_READ
发送PARAM_VALUE

参数：

名称	枚举值	描述
DJI_FLIGHT_CONTROLLER_ENABLE_RC_LOST_ACTION	0	启用失控保护
DJI_FLIGHT_CONTROLLER_DISABLE_RC_LOST_ACTION	1	禁用失控保护

示例

```
import time
import sys
from pymavlink import mavutil

def connect_to_vehicle():
    """建立 UDP 监听连接（默认监听所有接口的 14550 端口）"""
    connection_string = "udpin:0.0.0.0:14550"
    print(f"Listening for MAVLink on: {connection_string}")
    vehicle = mavutil.mavlink_connection(connection_string)
    print("Waiting for HEARTBEAT (timeout: 10s)...")
    try:
        vehicle.wait_heartbeat(timeout=10)
        print(f"  Heartbeat received from system {vehicle.target_sy
stem}, component {vehicle.target_component}")
        return vehicle
    except Exception as e:
        print(f"  Connection failed: {str(e)}")
```

```
        print("Ensure your vehicle is sending MAVLink messages to 1
4550 port")
        print("Example QGroundControl setup: Settings > General > M
AVLink > Listening Port = 14550")
        sys.exit(1)

def format_param_id(param_id):
    """格式化16字节参数ID（符合MAVLink协议要求）"""
    if len(param_id) > 16:
        raise ValueError("param_id cannot exceed 16 characters")

    param_id_bytes = bytearray(16)
    for i in range(min(len(param_id), 16)):
        param_id_bytes[i] = ord(param_id[i])

    # 长度<16时添加NULL终止符
    if len(param_id) < 16:
        param_id_bytes[len(param_id)] = 0

    return param_id_bytes

def send_param_set(vehicle, param_id, param_value, param_type):
    """发送参数设置请求"""
    vehicle.mav.param_set_send(
        vehicle.target_system,
        vehicle.target_component,
        format_param_id(param_id),
        param_value,
        param_type
    )
    print(f"  Sent PARAM_SET for '{param_id}' = {param_value} (typ
e: {param_type})")

def send_param_request_read(vehicle, param_id=None, param_index=-
1):
    """发送参数读取请求"""
    param_id_bytes = format_param_id(param_id) if param_id else byt
earray(16)

    vehicle.mav.param_request_read_send(
        vehicle.target_system,
        vehicle.target_component,
        param_id_bytes,
        param_index
    )
```

```
    target = f"index {param_index}" if param_index >= 0 else f"ID
'{param_id}'"
    print(f"  Sent PARAM_REQUEST_READ for {target}")

def send_param_request_list(vehicle):
    """发送请求所有参数列表"""
    vehicle.mav.param_request_list_send(
        vehicle.target_system,
        vehicle.target_component
    )
    print(f"  Sent PARAM_REQUEST_LIST to system {vehicle.target_sys
tem}, component {vehicle.target_component}")

def receive_param_value(vehicle, timeout=5.0):
    """接收参数值响应"""
    print(f"  Waiting for PARAM_VALUE ({timeout}s timeout)...")
    msg = vehicle.recv_match(type='PARAM_VALUE', blocking=True, tim
eout=timeout)

    if not msg:
        print("  Timeout: No PARAM_VALUE received")
        return None

    # 解析参数ID
    param_id = msg.param_id

    # 解析参数值
    param_value = msg.param_value

    # 参数类型映射
    type_names = {
        0: "INT8", 1: "UINT8", 2: "INT16", 3: "UINT16", 4: "INT32",
        5: "UINT32", 6: "INT64", 7: "UINT64", 8: "FLOAT", 9: "DOUBL
E"
    }
    param_type = type_names.get(msg.param_type, f"UNKNOWN({msg.para
m_type})")

    print("\n  Received PARAM_VALUE:")
    print(f"    ID: {param_id}")
    print(f"    Value: {param_value}")
    print(f"    Type: {param_type}")
    print(f"    Index/Total: {msg.param_index}/{msg.param_count}")
```

```
    return {
        'id': param_id,
        'value': param_value,
        'type': param_type,
        'index': msg.param_index,
        'count': msg.param_count
    }

def receive_all_param_values(vehicle, timeout=10.0):
    """接收所有参数值，直到收到 param_count 条消息后退出循环"""
    received_params = []
    expected_count = None
    print(f"  Waiting for all PARAM_VALUE messages...")

    while True:
        msg = vehicle.recv_match(type='PARAM_VALUE', blocking=True,
timeout=timeout)

        if not msg:
            print(f"  Timeout: Received {len(received_params)} para
meters (expected {expected_count if expected_count else 'unknow
n'})")
            break

        param_id = msg.param_id
        param_value = msg.param_value

        # 参数类型映射
        type_names = {
            0: "INT8", 1: "UINT8", 2: "INT16", 3: "UINT16", 4: "INT
32",
            5: "UINT32", 6: "INT64", 7: "UINT64", 8: "FLOAT", 9: "D
OUBLE"
        }
        param_type = type_names.get(msg.param_type, f"UNKNOWN({msg.
param_type})")

        # 首次收到消息时记录总参数数量
        if expected_count is None:
            expected_count = msg.param_count
            print(f"  Total parameters to receive: {expected_coun
t}")

        print(f"  [{msg.param_index}/{expected_count}] {param_id} =
{param_value} ({param_type})")
```

```
        received_params.append({
            'id': param_id,
            'value': param_value,
            'type': param_type,
            'index': msg.param_index,
            'count': msg.param_count
        })

        # 当已接收数量达到预期总数时退出循环
        if len(received_params) >= expected_count:
            print(f"\n  Successfully received all {expected_count}
parameters")
            break

    return received_params

def main():
    print("  Starting MAVLink PARAM_SET Tool (UDP Listening Mode)")
    print("------------------------------------------------------")
    print("This script listens for MAVLink on ALL interfaces:1455
0")
    print("Ensure your vehicle is configured to send to this port")
    print("------------------------------------------------------
\n")

    vehicle = connect_to_vehicle()

    # 操作选择
    print("\nAvailable operations:")
    print("1. Set parameter value")
    print("2. Read parameter value (by ID or index)")
    print("3. Request all parameters (PARAM_REQUEST_LIST)")
    choice = input("\nSelect operation [1-3]: ").strip()

    if choice == '1':
        param_id = input("Enter parameter ID to set: ").strip()
        param_value = float(input("Enter new value: ").strip())

        print("\nParameter types:")
        print("  8 = FLOAT (most common)")
        print("  9 = DOUBLE")
        print("  4 = INT32")
        print("  1 = UINT8")
        param_type = int(input("Enter type number [8]: ").strip() o
```

```
r 8)

        send_param_set(vehicle, param_id, param_value, param_type)
        receive_param_value(vehicle)

    elif choice == '2':
        # mode = input("Read by [id/index]: ").lower()
        mode = 'id'
        if mode == 'id':
            param_id = input("Enter parameter ID: ").strip()
            send_param_request_read(vehicle, param_id=param_id)
            receive_param_value(vehicle)
        elif mode == 'index':
            param_index = int(input("Enter parameter index: "))
            send_param_request_read(vehicle, param_index=param_inde
x)
            receive_param_value(vehicle)
        else:
            print("Invalid mode. Use 'id' or 'index'")

    elif choice == '3':
        print("\n  Requesting all parameters...")
        send_param_request_list(vehicle)
        all_params = receive_all_param_values(vehicle)
        print(f"\n  Total parameters received: {len(all_params)}")

    else:
        print("Invalid choice. Select 1, 2, or 3.")


if __name__ == "__main__":
    main()
```

示例使用说明

获取

运行后，成功建立连接

```
🚀 Starting MAVLink PARAM_SET Tool (UDP Listening Mode)
--------------------------------------------------
This script listens for MAVLink on ALL interfaces:14550
Ensure your vehicle is configured to send to this port
--------------------------------------------------

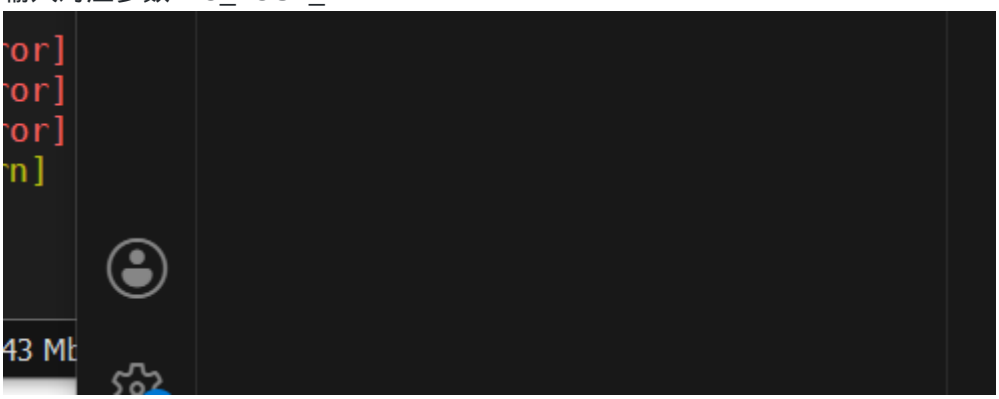
Listening for MAVLink on: udpin:0.0.0.0:14550
Waiting for HEARTBEAT (timeout: 10s)...
✅ Heartbeat received from system 3, component 0

Available operations:
1. Set parameter value
2. Read parameter value (by ID or index)
3. Request all parameters (PARAM_REQUEST_LIST)
```

输入“2”选择读取参数

```
Select operation [1-3]: 2
```

输入对应参数“RC_LOST_ENABLE”



成功返回该参数对应的数值

```
📥 Received PARAM_VALUE:
  🆔 ID: RC_LOST_ENABLE
  🔢 Value: 1.0
```

设置

运行后，成功建立连接

```
🚀 Starting MAVLink PARAM_SET Tool (UDP Listening Mode)
--------------------------------------------------
This script listens for MAVLink on ALL interfaces:14550
Ensure your vehicle is configured to send to this port
--------------------------------------------------

Listening for MAVLink on: udpin:0.0.0.0:14550
Waiting for HEARTBEAT (timeout: 10s)...
✅ Heartbeat received from system 3, component 0

Available operations:
1. Set parameter value
2. Read parameter value (by ID or index)
3. Request all parameters (PARAM_REQUEST_LIST)
```

输入“1”选择设置参数

```
Available operations:
1. Set parameter value
2. Read parameter value (by ID or index)
3. Request all parameters (PARAM_REQUEST_LIST)

Select operation [1-3]: 1
```

输入对应参数“RC_LOST_ENABLE”

```
Enter parameter ID to set: RC_LOST_ENABLE
```

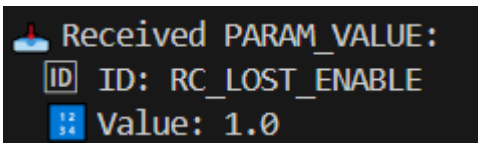
输入要设置的数值

```
Enter new value: 1
```

选择参数类型“1”

```
Parameter types:
  8 = FLOAT (most common)
  9 = DOUBLE
  4 = INT32
  1 = UINT8
Enter type number [8]: 1
```

成功返回设置值



机臂灯状态

ARM_LIGHT

设置

mavlink消息：

接收PARAM_SET
发送PARAM_VALUE

获取

mavlink消息

接收PARAM_REQUEST_READ
发送PARAM_VALUE

参数：

名称	枚举值	描述
DJI_FLIGHT_CONTROLLER_LIGHT_OFF	0	关闭机臂灯。
DJI_FLIGHT_CONTROLLER_LIGHT_ON	1	开启机臂灯。

示例

```
import time
import sys
from pymavlink import mavutil

def connect_to_vehicle():
    """建立 UDP 监听连接（默认监听所有接口的 14550 端口）"""
    connection_string = "udpin:0.0.0.0:14550"
    print(f"Listening for MAVLink on: {connection_string}")
    vehicle = mavutil.mavlink_connection(connection_string)
    print("Waiting for HEARTBEAT (timeout: 10s)...")
    try:
        vehicle.wait_heartbeat(timeout=10)
```

```
        print(f"  Heartbeat received from system {vehicle.target_system}, component {vehicle.target_component}")
        return vehicle
    except Exception as e:
        print(f"  Connection failed: {str(e)}")
        print("Ensure your vehicle is sending MAVLink messages to 14550 port")
        print("Example QGroundControl setup: Settings > General > MAVLink > Listening Port = 14550")
        sys.exit(1)

def format_param_id(param_id):
    """格式化16字节参数ID（符合MAVLink协议要求）"""
    if len(param_id) > 16:
        raise ValueError("param_id cannot exceed 16 characters")

    param_id_bytes = bytearray(16)
    for i in range(min(len(param_id), 16)):
        param_id_bytes[i] = ord(param_id[i])

    # 长度<16时添加NULL终止符
    if len(param_id) < 16:
        param_id_bytes[len(param_id)] = 0

    return param_id_bytes

def send_param_set(vehicle, param_id, param_value, param_type):
    """发送参数设置请求"""
    vehicle.mav.param_set_send(
        vehicle.target_system,
        vehicle.target_component,
        format_param_id(param_id),
        param_value,
        param_type
    )
    print(f"  Sent PARAM_SET for '{param_id}' = {param_value} (type: {param_type})")

def send_param_request_read(vehicle, param_id=None, param_index=-1):
    """发送参数读取请求"""
    param_id_bytes = format_param_id(param_id) if param_id else bytearray(16)

    vehicle.mav.param_request_read_send(
```

```
        vehicle.target_system,
        vehicle.target_component,
        param_id_bytes,
        param_index
    )

    target = f"index {param_index}" if param_index >= 0 else f"ID
'{param_id}'"
    print(f"  Sent PARAM_REQUEST_READ for {target}")

def send_param_request_list(vehicle):
    """发送请求所有参数列表"""
    vehicle.mav.param_request_list_send(
        vehicle.target_system,
        vehicle.target_component
    )
    print(f"  Sent PARAM_REQUEST_LIST to system {vehicle.target_sys
tem}, component {vehicle.target_component}")

def receive_param_value(vehicle, timeout=5.0):
    """接收参数值响应"""
    print(f"  Waiting for PARAM_VALUE ({timeout}s timeout)...")
    msg = vehicle.recv_match(type='PARAM_VALUE', blocking=True, tim
eout=timeout)

    if not msg:
        print("  Timeout: No PARAM_VALUE received")
        return None

    # 解析参数ID
    param_id = msg.param_id

    # 解析参数值
    param_value = msg.param_value

    # 参数类型映射
    type_names = {
        0: "INT8", 1: "UINT8", 2: "INT16", 3: "UINT16", 4: "INT32",
        5: "UINT32", 6: "INT64", 7: "UINT64", 8: "FLOAT", 9: "DOUBL
E"
    }
    param_type = type_names.get(msg.param_type, f"UNKNOWN({msg.para
m_type})")

    print("\n  Received PARAM_VALUE:")
```

```
    print(f"    ID: {param_id}")
    print(f"    Value: {param_value}")
    print(f"    Type: {param_type}")
    print(f"    Index/Total: {msg.param_index}/{msg.param_count}")

    return {
        'id': param_id,
        'value': param_value,
        'type': param_type,
        'index': msg.param_index,
        'count': msg.param_count
    }

def receive_all_param_values(vehicle, timeout=10.0):
    """接收所有参数值，直到收到 param_count 条消息后退出循环"""
    received_params = []
    expected_count = None
    print(f"  Waiting for all PARAM_VALUE messages...")

    while True:
        msg = vehicle.recv_match(type='PARAM_VALUE', blocking=True,
timeout=timeout)

        if not msg:
            print(f"  Timeout: Received {len(received_params)} para
meters (expected {expected_count if expected_count else 'unknow
n'})")
            break

        param_id = msg.param_id
        param_value = msg.param_value

        # 参数类型映射
        type_names = {
            0: "INT8", 1: "UINT8", 2: "INT16", 3: "UINT16", 4: "INT
32",
            5: "UINT32", 6: "INT64", 7: "UINT64", 8: "FLOAT", 9: "D
OUBLE"
        }
        param_type = type_names.get(msg.param_type, f"UNKNOWN({msg.
param_type})")

        # 首次收到消息时记录总参数数量
        if expected_count is None:
            expected_count = msg.param_count
```

```
            print(f"  Total parameters to receive: {expected_coun
t}")

        print(f"  [{msg.param_index}/{expected_count}] {param_id} =
{param_value} ({param_type})")

        received_params.append({
            'id': param_id,
            'value': param_value,
            'type': param_type,
            'index': msg.param_index,
            'count': msg.param_count
        })

        # 当已接收数量达到预期总数时退出循环
        if len(received_params) >= expected_count:
            print(f"\n  Successfully received all {expected_count}
parameters")
            break

    return received_params

def main():
    print("  Starting MAVLink PARAM_SET Tool (UDP Listening Mode)")
    print("------------------------------------------------------")
    print("This script listens for MAVLink on ALL interfaces:1455
0")
    print("Ensure your vehicle is configured to send to this port")
    print("------------------------------------------------------
\n")

    vehicle = connect_to_vehicle()

    # 操作选择
    print("\nAvailable operations:")
    print("1. Set parameter value")
    print("2. Read parameter value (by ID or index)")
    print("3. Request all parameters (PARAM_REQUEST_LIST)")
    choice = input("\nSelect operation [1-3]: ").strip()

    if choice == '1':
        param_id = input("Enter parameter ID to set: ").strip()
        param_value = float(input("Enter new value: ").strip())

        print("\nParameter types:")
```

```
        print("  8 = FLOAT (most common)")
        print("  9 = DOUBLE")
        print("  4 = INT32")
        print("  1 = UINT8")
        param_type = int(input("Enter type number [8]: ").strip() o
r 8)

        send_param_set(vehicle, param_id, param_value, param_type)
        receive_param_value(vehicle)

    elif choice == '2':
        # mode = input("Read by [id/index]: ").lower()
        mode = 'id'
        if mode == 'id':
            param_id = input("Enter parameter ID: ").strip()
            send_param_request_read(vehicle, param_id=param_id)
            receive_param_value(vehicle)
        elif mode == 'index':
            param_index = int(input("Enter parameter index: "))
            send_param_request_read(vehicle, param_index=param_inde
x)
            receive_param_value(vehicle)
        else:
            print("Invalid mode. Use 'id' or 'index'")

    elif choice == '3':
        print("\n  Requesting all parameters...")
        send_param_request_list(vehicle)
        all_params = receive_all_param_values(vehicle)
        print(f"\n  Total parameters received: {len(all_params)}")

    else:
        print("Invalid choice. Select 1, 2, or 3.")


if __name__ == "__main__":
    main()
```

示例使用说明

获取

运行后，成功建立连接

```
🚀 Starting MAVLink PARAM_SET Tool (UDP Listening Mode)
--------------------------------------------------
This script listens for MAVLink on ALL interfaces:14550
Ensure your vehicle is configured to send to this port
--------------------------------------------------

Listening for MAVLink on: udpin:0.0.0.0:14550
Waiting for HEARTBEAT (timeout: 10s)...
✅ Heartbeat received from system 3, component 0

Available operations:
1. Set parameter value
2. Read parameter value (by ID or index)
3. Request all parameters (PARAM_REQUEST_LIST)
```

输入“2”选择读取参数

```
Select operation [1-3]: 2
```

输入对应参数“ARM_LIGHT”

```
Enter parameter ID: ARM_LIGHT
```

成功返回该参数对应的数值

```
📩 Received PARAM_VALUE:
  🆔 ID: ARM_LIGHT
  🔢 Value: 1.0
```

设置

运行后，成功建立连接

```
🚀 Starting MAVLink PARAM_SET Tool (UDP Listening Mode)
--------------------------------------------------
This script listens for MAVLink on ALL interfaces:14550
Ensure your vehicle is configured to send to this port
--------------------------------------------------

Listening for MAVLink on: udpin:0.0.0.0:14550
Waiting for HEARTBEAT (timeout: 10s)...
✅ Heartbeat received from system 3, component 0

Available operations:
1. Set parameter value
2. Read parameter value (by ID or index)
3. Request all parameters (PARAM_REQUEST_LIST)
```

输入“1”选择设置参数

```
Available operations:
1. Set parameter value
2. Read parameter value (by ID or index)
3. Request all parameters (PARAM_REQUEST_LIST)

Select operation [1-3]: 1
```

输入对应参数“ARM_LIGHT”

```
Enter parameter ID to set: ARM_LIGHT
```

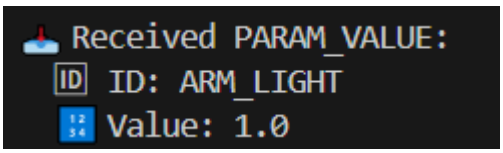
输入要设置的数值

```
Enter new value: 1
```

选择参数类型“1”

```
Parameter types:
  8 = FLOAT (most common)
  9 = DOUBLE
  4 = INT32
  1 = UINT8
Enter type number [8]: 1
```

成功返回设置值



返回command = MAV_CMD_REQUEST_MESSAGE

码流 AI 识别

码流AI识别

功能说明

LiveView AI 功能用于开启 DJI PSDK LiveView 图像流，使用示例中的 YOLOvFastest 目标检测模块处理图像，并通过 MAVLink `TUNNEL` 消息输出识别框信息。

该功能同时会将 AI meta 信息回传给 DJI Pilot，并将检测结果转换为 MAVLink 自定义二进制 payload，方便地面端或伴随计算机解析。

注意：

当前仅在 Matrice 400 无人机挂载 H30 相机和 DJI 妙算3 场景下支持该功能。

编解码回调函数中均不允许进行阻塞耗时操作，推荐创建独立线程进行目标识别等功能。

相关文件

文件	说明
`src/dji_mavlink/dji_liveview_ai_mavlink.hpp`	LiveVie 接口和 定义
`src/dji_mavlink/dji_liveview_ai_mavlink.cpp`	LiveVie 目标检 TUNNE
`src/dji_mavlink/mavlink_connector.cpp`	`MAV_C` 命令入口
`samples/sample_c++/module_sample/liveview/image_processor_yolovfastest.*`	YOLOv 示例处理

启停命令

使用 MAVLink `COMMAND_LONG` 发送 `MAV_CMD_USER_1` 。

参数	说明
`param1`	`1` 启动 LiveView AI；`0` 停止 LiveView AI
`param2`	DJI LiveView camera position，默认 `DJI_LIVEVIEW_CAMERA_POSITION_NO_1`
`param3`	DJI LiveView camera source，默认 `DJI_LIVEVIEW_CAMERA_SOURCE_DEFAULT`

命令执行后，程序返回 `COMMAND_ACK` 。

返回值	说明
`MAV_RESULT_ACCEPTED`	PSDK 接口调用成功
`MAV_RESULT_FAILED`	PSDK 接口调用失败，或未编译 OpenCV 支持

MAVLink 输出

LiveView AI 识别结果通过 `MAVLink TUNNEL` 输出。

字段	值
`payload_type`	`32790`
`payload_length`	根据当前包内识别框数量变化
`payload`	自定义二进制结构

Payload 格式

每个 TUNNEL payload 由一个 header 加若干个 box 组成。单包最多携带 8 个目标框；超过 8 个目标时自动拆成多包。

Header

字段	类型	说明
`version`	`uint8_t`	当前为 `1`

字段	类型	说明
`packet_index`	`uint8_t`	当前分包序号，从 0 开始
`packet_count`	`uint8_t`	当前帧总分包数
`box_count`	`uint8_t`	当前包内目标框数量
`frame_id`	`uint32_t`	DJI LiveView 图像帧 ID
`width`	`uint16_t`	图像宽度
`height`	`uint16_t`	图像高度

Box

字段	类型	说明
`id`	`uint16_t`	DJI 目标 ID
`type`	`uint8_t`	目标类别 ID
`state`	`uint8_t`	目标状态
`cx`	`uint16_t`	目标框中心 X，DJI 归一化坐标，范围 0 到 10000
`cy`	`uint16_t`	目标框中心 Y，DJI 归一化坐标，范围 0 到 10000
`w`	`uint16_t`	目标框宽度，DJI 归一化坐标，范围 0 到 10000
`h`	`uint16_t`	目标框高度，DJI 归一化坐标，范围 0 到 10000
`distance_mm`	`uint32_t`	目标距离，单位 mm；无距离时按 DJI 数据填充

使用流程

1. 确认工程编译时开启 `OPEN_CV_INSTALLED`。
2. 地面端发送 `MAV_CMD_USER_1`，`param1=1`。
3. 程序开启 LiveView 图像流并启动 AI 处理线程。
4. 地面端监听 `MAVLink TUNNEL payload_type=32790`。
5. 停止时发送 `MAV_CMD_USER_1`，`param1=0`。

示例

启动默认相机源：

```
COMMAND_LONG.command = MAV_CMD_USER_1
COMMAND_LONG.param1 = 1
COMMAND_LONG.param2 = 0
COMMAND_LONG.param3 = 0
```

停止：

```
COMMAND_LONG.command = MAV_CMD_USER_1
COMMAND_LONG.param1 = 0
```

Python 例程

以下例程使用 `pymavlink` 启动 LiveView AI，监听 `TUNNEL payload_type=32790` ，解析识别框并打印目标信息，最后发送停止命令。

```
import struct
import time
from pymavlink import mavutil


LIVEVIEW_AI_PAYLOAD_TYPE = 32790
MAV_CMD_USER_1 = mavutil.mavlink.MAV_CMD_USER_1


def get_current_millis():
    return int(time.time() * 1000)


def send_liveview_ai_command(master, enable, camera_position=0, cam
era_source=0):
    send_time = get_current_millis()
    print(
        f"[{send_time}] 发送 LiveView AI 命令: "
        f"enable={enable}, position={camera_position}, source={came
ra_source}"
    )

    master.mav.command_long_send(
        master.target_system,
        master.target_component,
        MAV_CMD_USER_1,
```

```
        0,
        1 if enable else 0,    # param1: 1=start, 0=stop
        camera_position,       # param2: DJI LiveView camera positio
n，0 使用默认值
        camera_source,         # param3: DJI LiveView camera sourc
e，0 使用默认值
        0, 0, 0, 0
    )

    ack = master.recv_match(type="COMMAND_ACK", blocking=True, time
out=5.0)
    if ack:
        ack_time = get_current_millis()
        print(
            f"[{ack_time}] 收到 COMMAND_ACK: "
            f"command={ack.command}, result={ack.result}, delay={ac
k_time - send_time}ms"
        )
        return ack.result == mavutil.mavlink.MAV_RESULT_ACCEPTED

    print(f"[{get_current_millis()}] 超时：未收到 LiveView AI 命令应
答")
    return False


def parse_liveview_ai_payload(payload):
    # Header: version, packet_index, packet_count, box_count, frame
_id, width, height
    header_fmt = "<BBBBIHH"
    box_fmt = "<HBBHHHHI"
    header_size = struct.calcsize(header_fmt)
    box_size = struct.calcsize(box_fmt)

    if len(payload) < header_size:
        return None, []

    version, packet_index, packet_count, box_count, frame_id, widt
h, height = struct.unpack_from(
        header_fmt, payload, 0
    )

    header = {
        "version": version,
        "packet_index": packet_index,
        "packet_count": packet_count,
```

```
        "box_count": box_count,
        "frame_id": frame_id,
        "width": width,
        "height": height,
    }

    boxes = []
    offset = header_size
    for _ in range(box_count):
        if offset + box_size > len(payload):
            break
        target_id, target_type, state, cx, cy, w, h, distance_mm =
struct.unpack_from(
            box_fmt, payload, offset
        )
        boxes.append({
            "id": target_id,
            "type": target_type,
            "state": state,
            "cx": cx,
            "cy": cy,
            "w": w,
            "h": h,
            "distance_mm": distance_mm,
        })
        offset += box_size

    return header, boxes


# 连接无人机（UDP 端口 14550）
master = mavutil.mavlink_connection("udpin:0.0.0.0:14550")
print("等待无人机心跳...")
master.wait_heartbeat()
print(f"无人机已连接: system={master.target_system}, component={maste
r.target_component}")

if not send_liveview_ai_command(master, enable=True):
    raise SystemExit("LiveView AI 启动失败")

print("开始监听 LiveView AI 识别结果，按 Ctrl+C 停止")

try:
    while True:
        msg = master.recv_match(type="TUNNEL", blocking=True, timeo
```

```
ut=5.0)
        now_ms = get_current_millis()

        if msg is None:
            print(f"[{now_ms}] 超时：未收到 LiveView AI TUNNEL 数据")
            continue

        if msg.payload_type != LIVEVIEW_AI_PAYLOAD_TYPE:
            continue

        payload = bytes(msg.payload[:msg.payload_length])
        header, boxes = parse_liveview_ai_payload(payload)
        if header is None:
            print(f"[{now_ms}] 收到无效 LiveView AI payload")
            continue

        print(
            f"[{now_ms}] frame={header['frame_id']} "
            f"packet={header['packet_index'] + 1}/{header['packet_c
ount']} "
            f"size={header['width']}x{header['height']} boxes={len
(boxes)}"
        )

        for box in boxes:
            print(
                f"  id={box['id']} type={box['type']} state={box['s
tate']} "
                f"cx={box['cx']} cy={box['cy']} w={box['w']} h={box
['h']} "
                f"distance={box['distance_mm']}mm"
            )

except KeyboardInterrupt:
    print("准备停止 LiveView AI")
finally:
    send_liveview_ai_command(master, enable=False)
    print("LiveView AI 例程结束")
```

注意事项

- 如果未定义 `OPEN_CV_INSTALLED`，启动接口会返回 `DJI_ERROR_SYSTEM_MODULE_CODE_NONSUPPORT`，MAVLink ACK 为

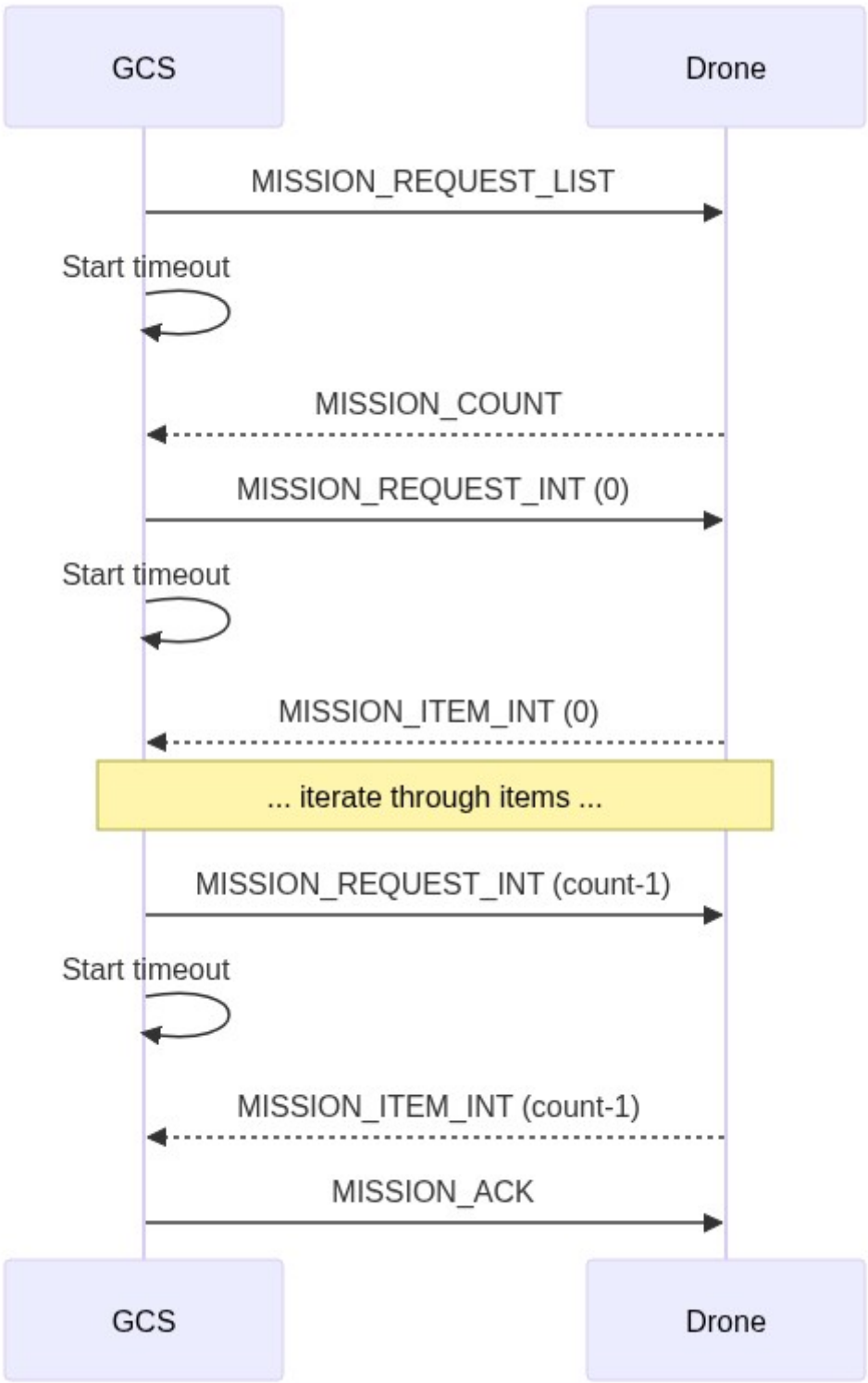
`MAV_RESULT_FAILED`。

- `cx/cy/w/h` 是 DJI 归一化坐标，不是像素坐标。地面端如需像素坐标，需要结合 `width/height` 自行换算。
- LiveView AI 使用 LiveView 图像流和编码回调。若系统同时使用普通视频流，需要联调确认 LiveView 初始化和回调不会互相抢占。
- 自定义 payload 为二进制结构，不是 JSON；地面端解析时需按小端结构体读取。

运动规划

下载任务

下图展示了从无人机下载任务的通信序列（假设所有作成功）。

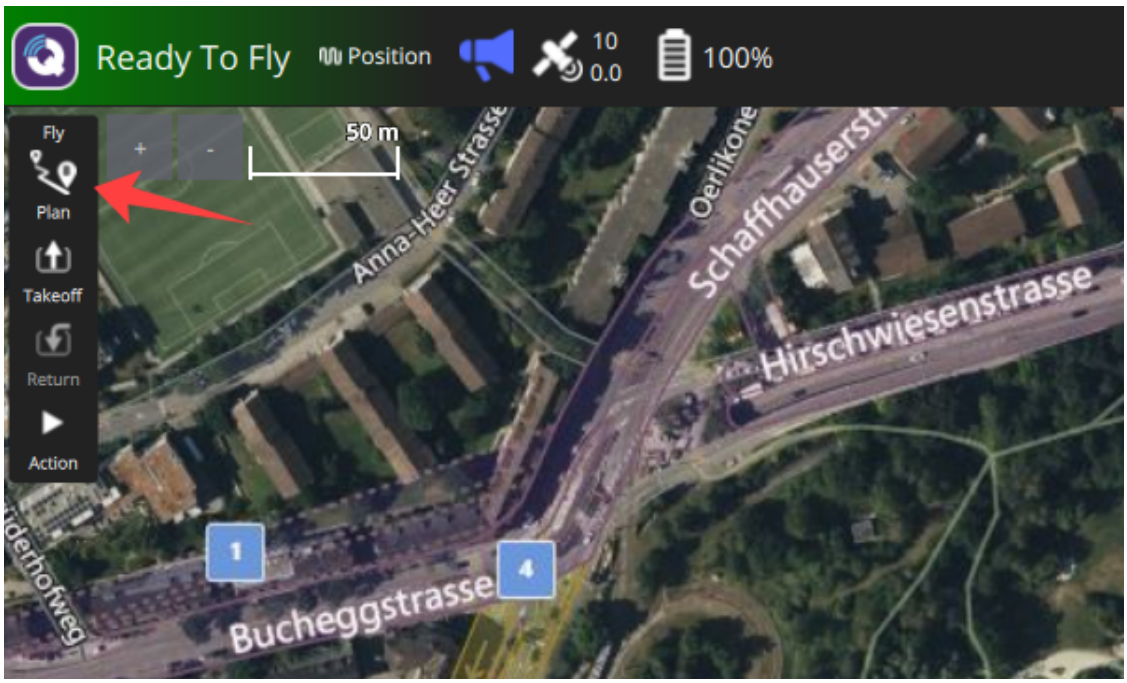


这个流程和上传任务时类似。主要区别是客户端（例如 GCS）发送 MISSION_REQUEST_LIST，触发无人机负载计算机响应当前任务项数量（MISSION_COUNT）。开启一个新循环，GCS 请求任务项，无人机提供。

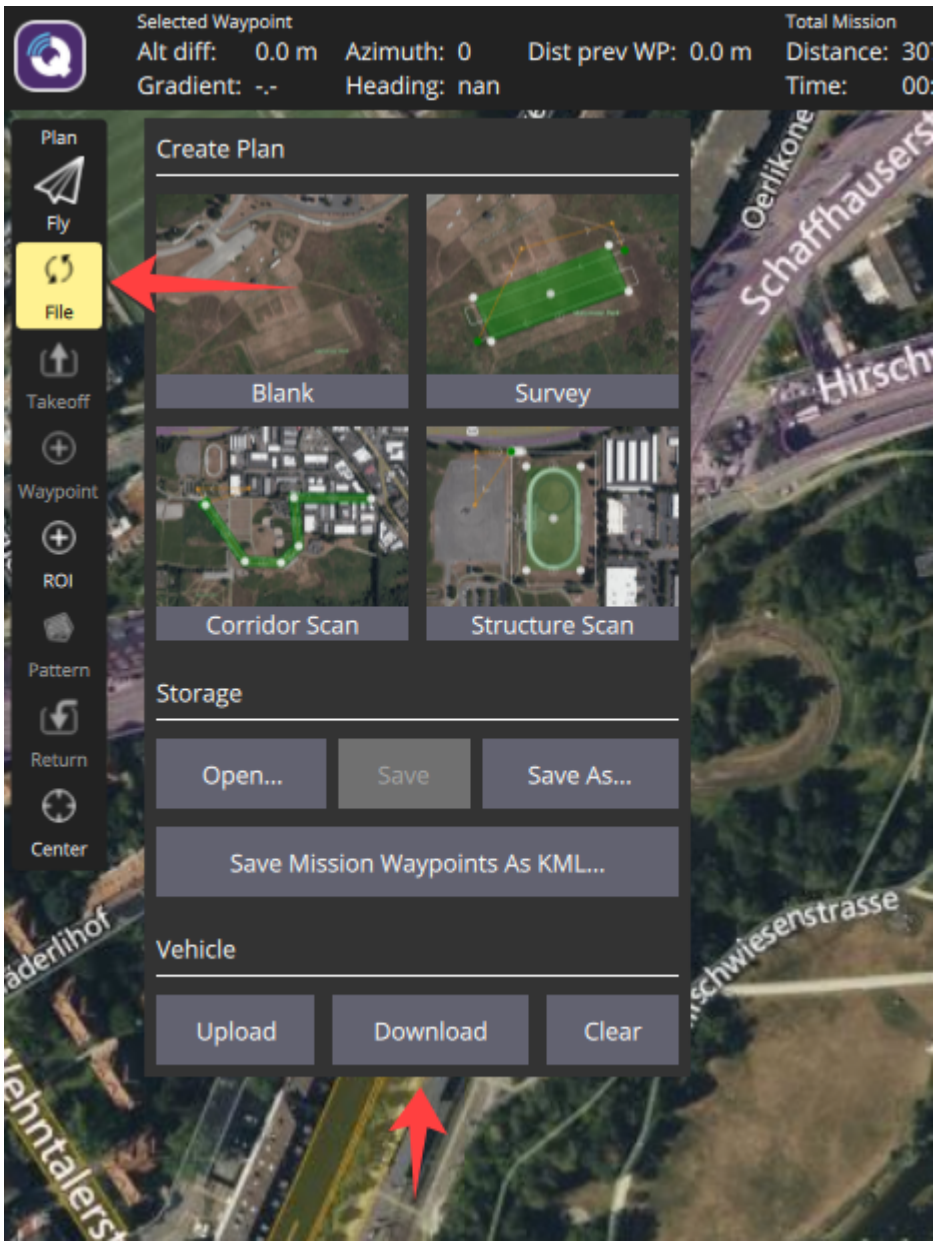
注：
每个需要响应的消息都会设置超时（例如 MISSION_REQUEST_INT）。如果超时结束且未收到回复，则必须重新发送请求。
任务物品必须按顺序接收。如果项目未按顺序接收，GCS 应重新请求预期项目（该不按顺序的项目被丢弃）。
协议实现也必须以上传任务页中同样的方式支持 MISSION_ITEM 和 MISSION_REQUEST。

使用QGC下载无人机上的任务：

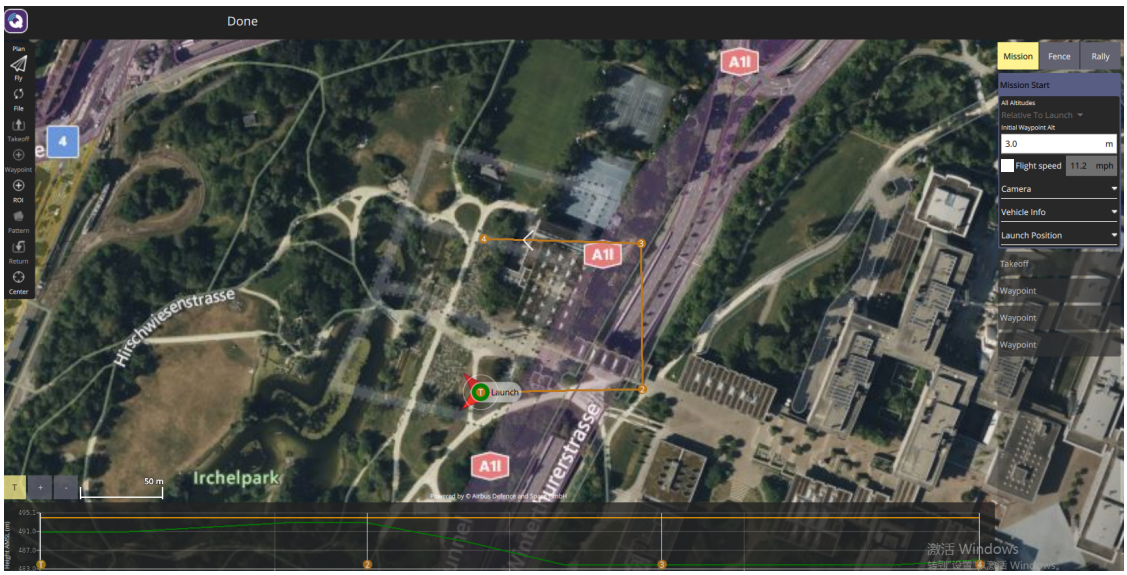
1. 点击左侧plan按钮进入飞行计划页面



2. 点击左侧File->Download按钮开始下载流程

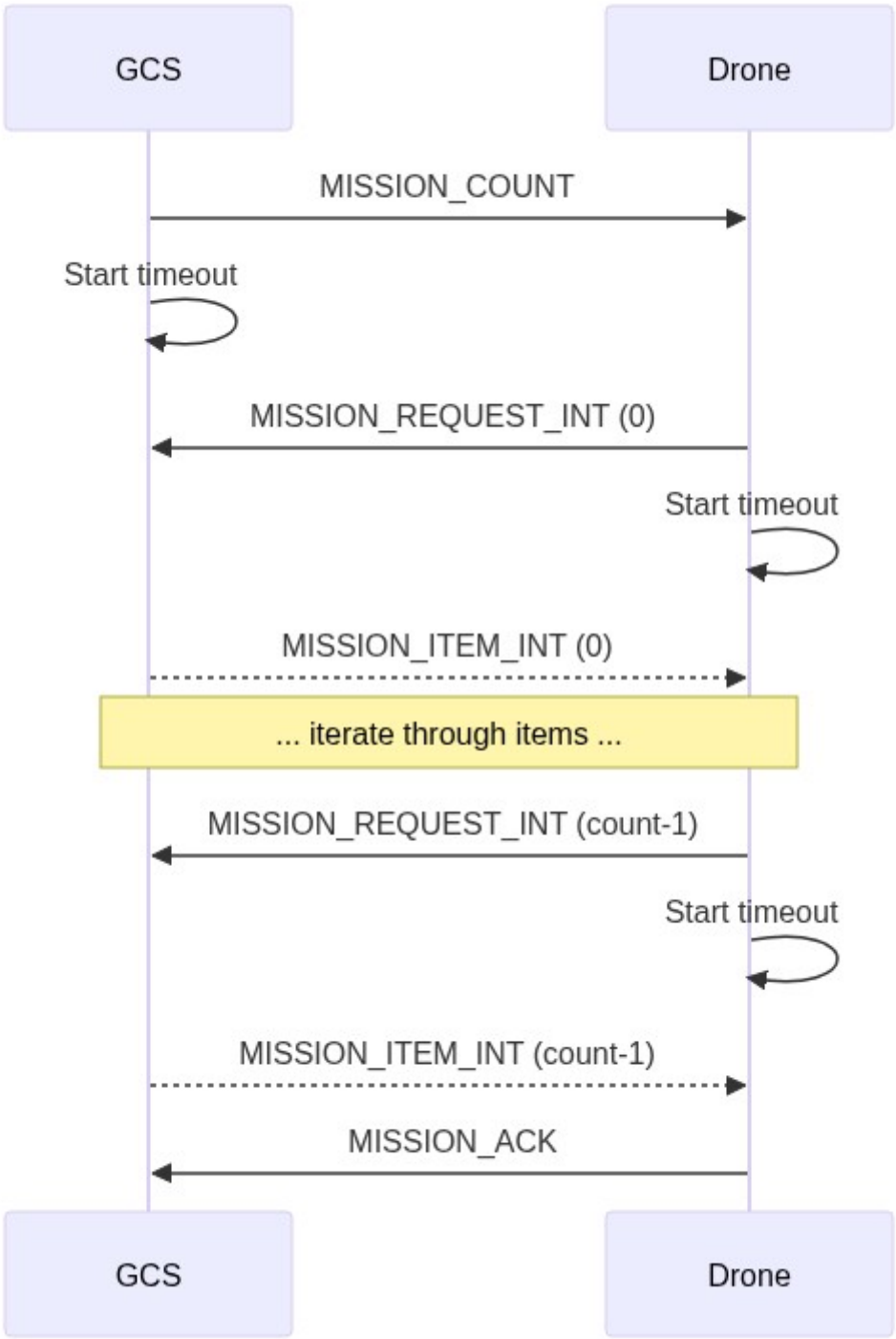


3. 任务成功下载到QGC并显示



上传任务

下图展示了将任务上传到无人机的通信序列（假设所有作都成功）。



更详细地说，作业顺序如下：

1. GCS 会发送 MISSION_COUNT 包括需要上传的任务项总数。
   · GCS启动一个超时等待 ，等待无人机的响应（MISSION_REQUEST_INT）。
2. 无人机收到消息并回复 MISSION_REQUEST_INT 请求第一个任务项（seq==0）。
   · 无人机启动一个超时等待 ，等待 GCS 的 MISSION_ITEM_INT 响应。
3. GCS 接收 MISSION_REQUEST_INT 并以 MISSION_ITEM_INT 消息回复请求的任务项目。

4. 无人机和 GCS 重复 MISSION_REQUEST_INT/MISSION_ITEM_INT 循环，迭代序列直到所有项目上传（seq==count-1）。
5. 收到最后一项任务物品后，无人机会回复 MISSION_ACK，并发出指示任务上传完成/成功的 MAV_MISSION_ACCEPTED 类型。
   · 无人机应将新任务设置为当前任务，丢弃原始数据。
   · 无人机认为上传完成。
6. GCS 接收包含 MAV_MISSION_ACCEPTED 的 MISSION_ACK 表示作已完成。

进入/退出任务功能见“切换飞行模式”页。

mavlink id 44 MISSION_COUNT

字段名	类型	单位	描
target_system	uint8_t		目
target_component	uint8_t		目
count	uint16_t		任
mission_type	uint8_t	MAV_MISSION_TYPE	任

mavlink id 51 MISSION_REQUEST_INT

字段名	类型	单位	描
target_system	uint8_t		目
target_component	uint8_t		目
seq	uint16_t		请
mission_type	uint8_t	MAV_MISSION_TYPE	任

MAVLink ID 73: MISSION_ITEM_INT

字段名	类型	值	描述
target_system	uint8_t		系统ID
target_component	uint8_t		组件ID

seq	uint16_t		航点ID（序列号 每个航点的编号 序列中无间隔（
frame	uint8_t	MAV_FRAME	航点的坐标系。
command	uint16_t	MAV_CMD	航点的计划动作
current	uint8_t	bool	是否为当前执行
autocontinue	uint8_t		自动继续到下一 1：true。 设置为false可在
param1	float		PARAM1，参见
param2	float		PARAM2，参见
param3	float		PARAM3，参见
param4	float		PARAM4，参见
x	int32_t		PARAM5 / 本地 1e4，全球：以
y	int32_t		PARAM6 / y位置 单位为米 * 1e4 单位为度 *10^7
z	float		PARAM7 / z位置 单位为米（取决
mission_type	uint8_t		任务类型。固定

mavlink id 47 MISSION_ACK

字段名	类型	单位	描
target_system	uint8_t		目
target_component	uint8_t		目
type	uint8_t	MAV_MISSION_RESULT	任
mission_type	uint8_t	MAV_MISSION_TYPE	任

MAV_MISSION_RESULT

Value	Field Name	Description
0	MAV_MISSION_ACCEPTED	任务成功
1	MAV_MISSION_ERROR	通用错误/目前完
2	MAV_MISSION_UNSUPPORTED_FRAME	不支持坐标系。
3	MAV_MISSION_UNSUPPORTED	不支持该命令。
4	MAV_MISSION_NO_SPACE	任务项超出存储空
5	MAV_MISSION_INVALID	其中一个参数的值
6	MAV_MISSION_INVALID_PARAM1	参数1的值无效。
7	MAV_MISSION_INVALID_PARAM2	参数2的值无效。
8	MAV_MISSION_INVALID_PARAM3	参数3的值无效。
9	MAV_MISSION_INVALID_PARAM4	param4 的值无效

10	MAV_MISSION_INVALID_PARAM5_X	x / param5 的值无
11	MAV_MISSION_INVALID_PARAM6_Y	y / param6 的值无
12	MAV_MISSION_INVALID_PARAM7	z / param7 的值无
13	MAV_MISSION_INVALID_SEQUENCE	任务项接收顺序钅
14	MAV_MISSION_DENIED	不接受来自此通信
15	MAV_MISSION_OPERATION_CANCELLED	当前任务操作已取 任务下载）。

示例

发送任务到无人机：1、起飞，2、移动至指定航点

```
import time
import sys
from pymavlink import mavutil

def convert_lat_lon_to_int(lat, lon):
    """将纬度和经度转换为int32_t * 1E7的格式"""
    lat_int = int(lat * 1e7)
    lon_int = int(lon * 1e7)
    return lat_int, lon_int

def send_mission_upload(connection, target_system, target_componen
t, mission_items):
    """
    上传任务到飞行器（符合Mavlink文档规范）
    :param connection: MAVLink连接对象
    :param target_system: 目标系统ID
    :param target_component: 目标组件ID
    :param mission_items: 任务项列表，每个任务项是一个包含参数的字典
    """
    # 发送MISSION_COUNT（任务项数量）
```

```
    count = len(mission_items)
    print(f"发送MISSION_COUNT: {count}个任务项")
    connection.mav.mission_count_send(
        target_system,
        target_component,
        count,
        mavutil.mavlink.MAV_MISSION_TYPE_MISSION  # 任务类型：飞行计划
    )

    # 等待并处理每个任务项
    for seq in range(count):
        print(f"等待MISSION_REQUEST_INT for seq={seq}")
        retries = 0
        max_retries = 5  # 根据文档：Retries (max): 5

        while retries < max_retries:
            # 根据文档：Timeout (default): 1500 ms
            msg = connection.recv_match(
                type='MISSION_REQUEST_INT',
                blocking=True,
                timeout=1.5  # 1500 ms
            )

            if msg is not None:
                # 确保请求的是正确的序列号
                if msg.seq != seq:
                    print(f"错误: 请求的序列号不匹配，期望{seq}，收到{ms
g.seq}")
                    return False
                break
            else:
                print(f"超时，未收到MISSION_REQUEST_INT (尝试 {retries
+1}/{max_retries})")
                retries += 1

        if retries == max_retries:
            print("达到最大重试次数，任务上传失败")
            return False

        # 发送MISSION_ITEM_INT
        item = mission_items[seq]
        print(f"准备发送MISSION_ITEM_INT for seq={seq},command={item
['command']}")
        connection.mav.mission_item_int_send(
            target_system,
```

```
            target_component,
            seq,
            item['frame'],
            item['command'],
            item['current'],
            item['autocontinue'],
            item['param1'],
            item['param2'],
            item['param3'],
            item['param4'],
            item['param5'],
            item['param6'],
            item['param7'],
            mavutil.mavlink.MAV_MISSION_TYPE_MISSION
        )

    # 等待MISSION_ACK
    print("等待MISSION_ACK")
    max_retries = 5
    retries = 0
    while retries < max_retries:
        msg = connection.recv_match(
            type='MISSION_ACK',
            blocking=True,
            timeout=1.5
        )

        if msg is not None:
            if msg.type == mavutil.mavlink.MAV_MISSION_ACCEPTED:
                print("任务上传成功!")
                return True
            else:
                print(f"任务上传失败，错误代码: {msg.type}")
                return False
        else:
            print(f"超时，未收到MISSION_ACK (尝试 {retries+1}/{max_ret
ries})")
            retries += 1

    print("达到最大重试次数，任务上传失败")
    return False

def main():
    # 配置连接 UDP连接 (QGroundControl默认)
    connection = mavutil.mavlink_connection('udpin:0.0.0.0:14550')
```

```
    print("等待心跳信号...")
    try:
        # 等待心跳包（最多等待10秒）
        msg = connection.wait_heartbeat(timeout=10)
        print(f"心跳包已收到！系统ID: {msg.get_srcSystem()}, 组件ID: {m
sg.get_srcComponent()}")

        target_system = 1
        target_component = 1

        # 创建简单任务：一个航点
        # 纬度: 47.3667, 经度: 8.5500, 高度: 100.0米
        lat = 41.3667
        lon = 111.5500
        altitude = 50.0

        # 转换坐标
        lat_int, lon_int = convert_lat_lon_to_int(lat, lon)

        # 任务项
        mission_item = {
            'frame': mavutil.mavlink.MAV_FRAME_GLOBAL,
            'command': mavutil.mavlink.MAV_CMD_NAV_TAKEOFF, # 起飞
            'current': 0,  # 0表示不是当前任务项
            'autocontinue': 1,  # 1表示自动继续到下一个任务项
            'param1': 0,
            'param2': 0,
            'param3': 0,
            'param4': 0,
            'param5': 0,
            'param6': 0,
            'param7': 0,
        }
        mission_item2 = {
            'frame': mavutil.mavlink.MAV_FRAME_GLOBAL,
            'command': mavutil.mavlink.MAV_CMD_NAV_WAYPOINT,
# 导航到指定航点
            'current': 0,  # 0表示不是当前任务项
            'autocontinue': 1,  # 1表示自动继续到下一个任务项
            'param1': 0,  # 0秒停留
            'param2': 0,  # 0秒停留
            'param3': 0,  # 0秒停留
            'param4': 0,  # 航向
            'param5': lat_int,  # 纬度
```

```
            'param6': lon_int,  # 经度
            'param7': altitude,  # 高度
        }

        mission_items = [mission_item]

        mission_items.append(mission_item2)
        # 上传任务
        success = send_mission_upload(connection, target_system, ta
rget_component, mission_items)
        if success:
            print("任务上传成功!")
        else:
            print("任务上传失败!")

    except Exception as e:
        print(f"连接错误: {e}")
        sys.exit(1)

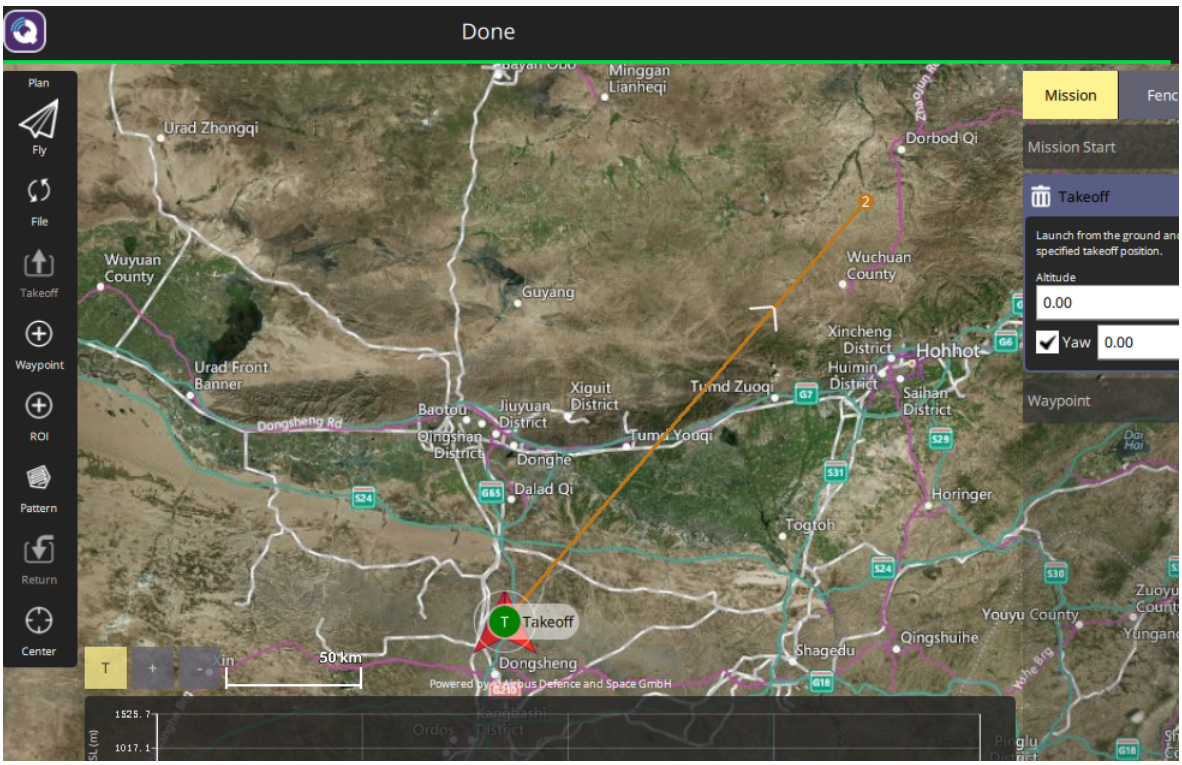
    # 关闭连接
    connection.close()
    print("连接已关闭")

if __name__ == '__main__':
    main()
```

执行结果

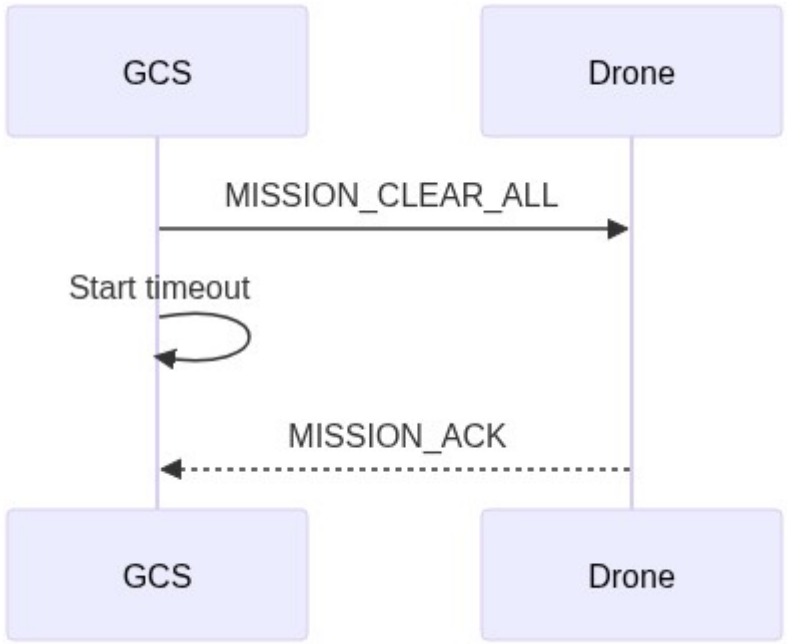
```
PS D:\python> & C:\Users\Administrator\AppData\Local\Python\pythoncore-3.14-64\python.exe d:/python/upload_mission.py
等待心跳信号...
心跳包已收到! 系统ID: 1, 组件ID: 1
发送MISSION_COUNT: 2个任务项
等待MISSION_REQUEST_INT for seq=0
准备发送MISSION_ITEM_INT for seq=0,command=22
等待MISSION_REQUEST_INT for seq=1
准备发送MISSION_ITEM_INT for seq=1,command=16
等待MISSION_ACK
任务上传成功!
任务上传成功!
连接已关闭
PS D:\python>
```

于QGC下载任务后查看任务计划



删除任务

下图展示了无人机清除任务的通信序列（假设所有行动成功）。



更详细地说，作顺序如下：

1. GCS发送 MISSION_CLEAR_ALL
   • GCS 暂停等待无人机的 MISSION_ACK。

2. 无人机收到消息，并从存储中清除任务。
3. 无人机会以 MISSION_ACK 结果 MAV_MISSION_ACCEPTED 表示成功 。
4. GCS 接收 MISSION_ACK 并清理其自身存储的任务信息。手术现已完成。

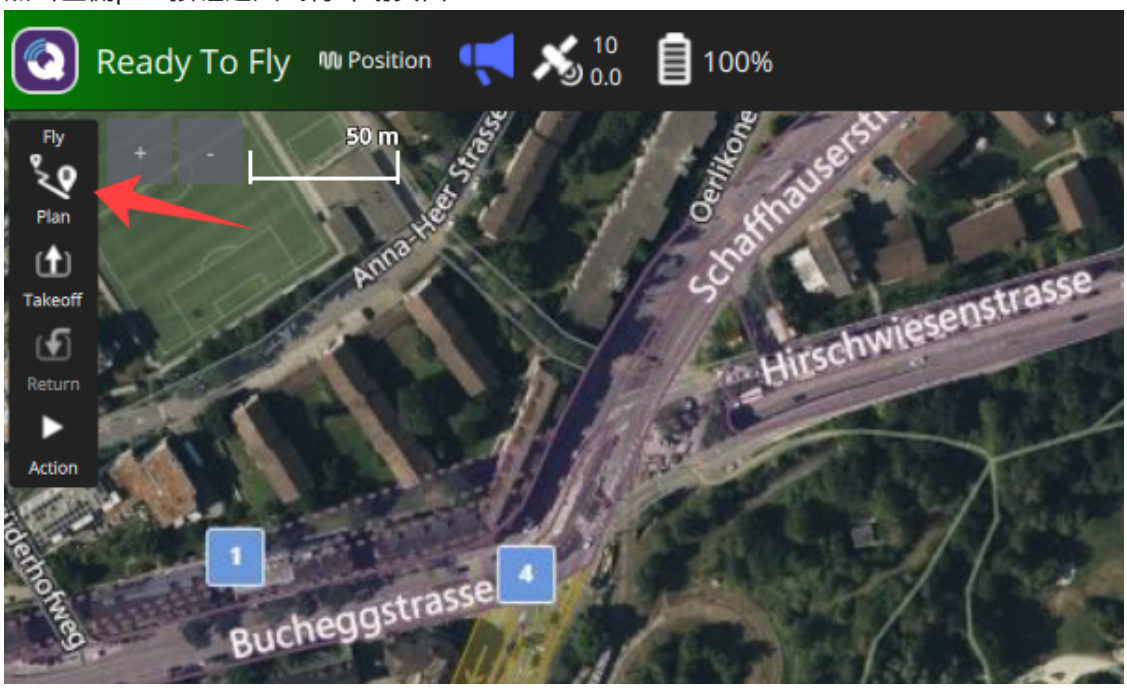
注：
每个需要响应的消息（例如 MISSION_CLEAR_ALL ）都会设置超时 。如果超时结束且未收到回复，则必须重新发送请求。
MISSION_ACK对MISSION_CLEAR_ALL响应时的mission_type字段值为0。

mavlink id 45 MISSION_CLEAR_ALL

Field Name	Type	Description
target_system	uint8_t	系统 ID
target_component	uint8_t	组件 ID

使用QGC删除无人机上的任务：

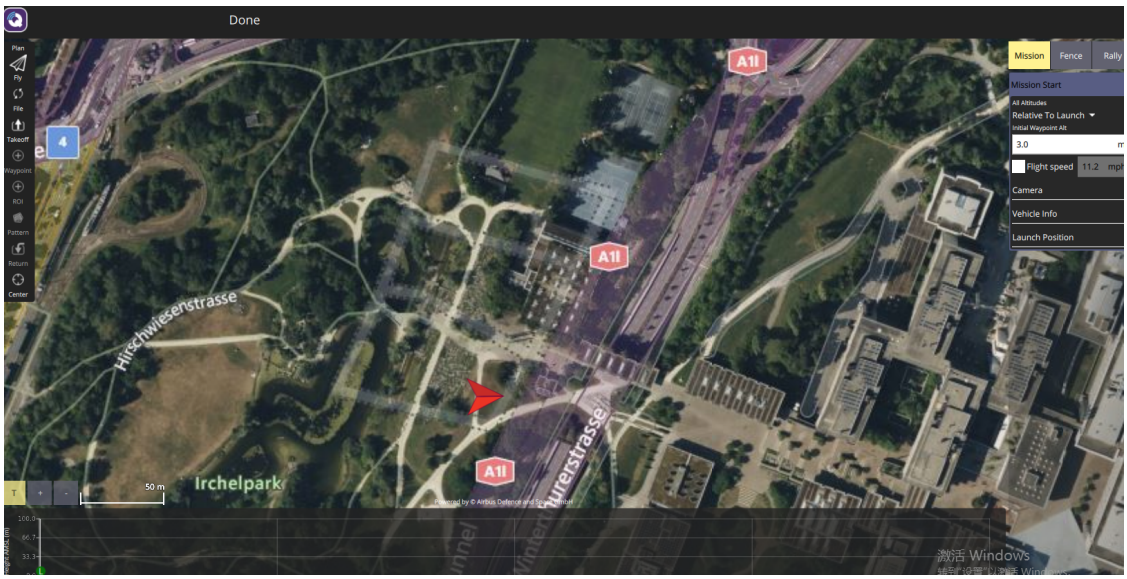
1. 点击左侧plan按钮进入飞行计划页面



2. 点击左侧File->Clear按钮开始删除流程



3. 无人机任务已清空并正确响应QGC，QGC清空本地任务列表



环绕模式

MAV_CMD_DO_ORBIT（34）

开始沿着参数定义的圆周绕行。将值设置为 NaN（视情况而定）会使用默认值。

Param (Label)	Description	Values	Units
1（半径）	圆的半径。正值：顺时针轨道。负值：逆时针绕轨道。NaN：使用飞行器默认半径。		m
2（速度）	切线速度。NaN：使用飞行器默认速度。		m/s
3（偏航行为）	车辆的偏航行为。	ORBIT_YAW_BEHAVIOUR	
4（轨道）	绕中心点运行该弧度数（即设定为 270*Pi/180 的三分之三轨道）。0或NaN：一直环绕。	最少：0	弧度
5（纬度/X）	中心点纬度坐标。如果为 NaN：使用当前飞行器位置。		

Param (Label)	Description	Values	Units
6（经度/Y）	中心点经度坐标。如果为 NaN：使用当前飞行器位置。		
7（高度/Z）	中心点高度（MSL）坐标。 NaN： 使用当前飞行器的高度。		

ORBIT_YAW_BEHAVIOUR

环绕飞行中的偏航行为模式。

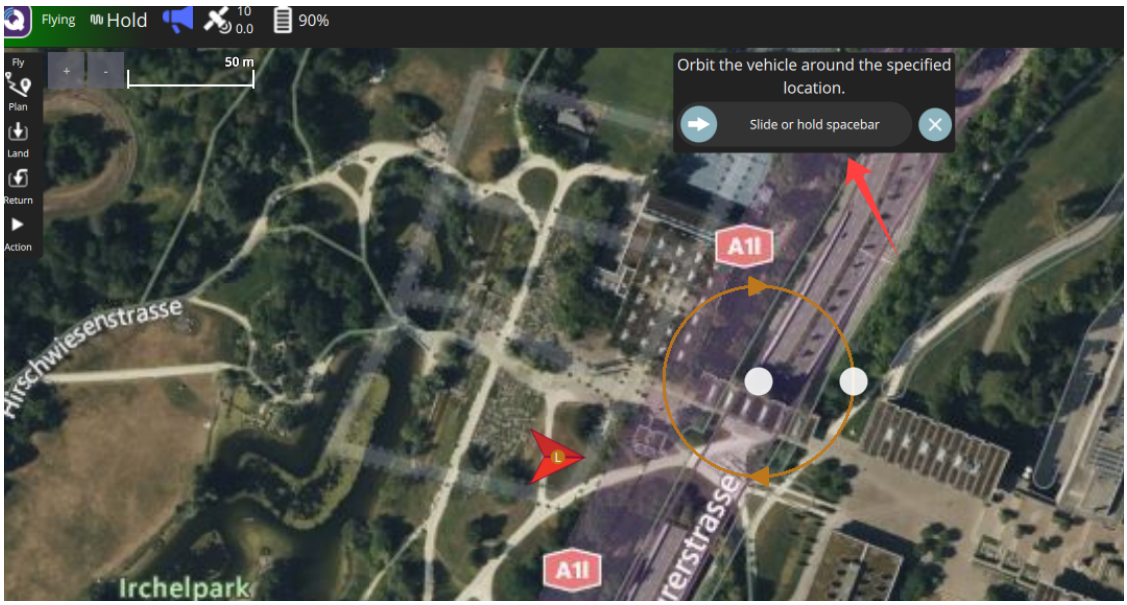
Value	Name	Descript
0	ORBIT_YAW_BEHAVIOUR_HOLD_FRONT_TO_CIRCLE_CENTER	无人机前
1	ORBIT_YAW_BEHAVIOUR_HOLD_INITIAL_HEADING	收到消息
3	ORBIT_YAW_BEHAVIOUR_HOLD_FRONT_TANGENT_TO_CIRCLE	无人机前（与圆相

使用QGC发送执行环绕模式命令：

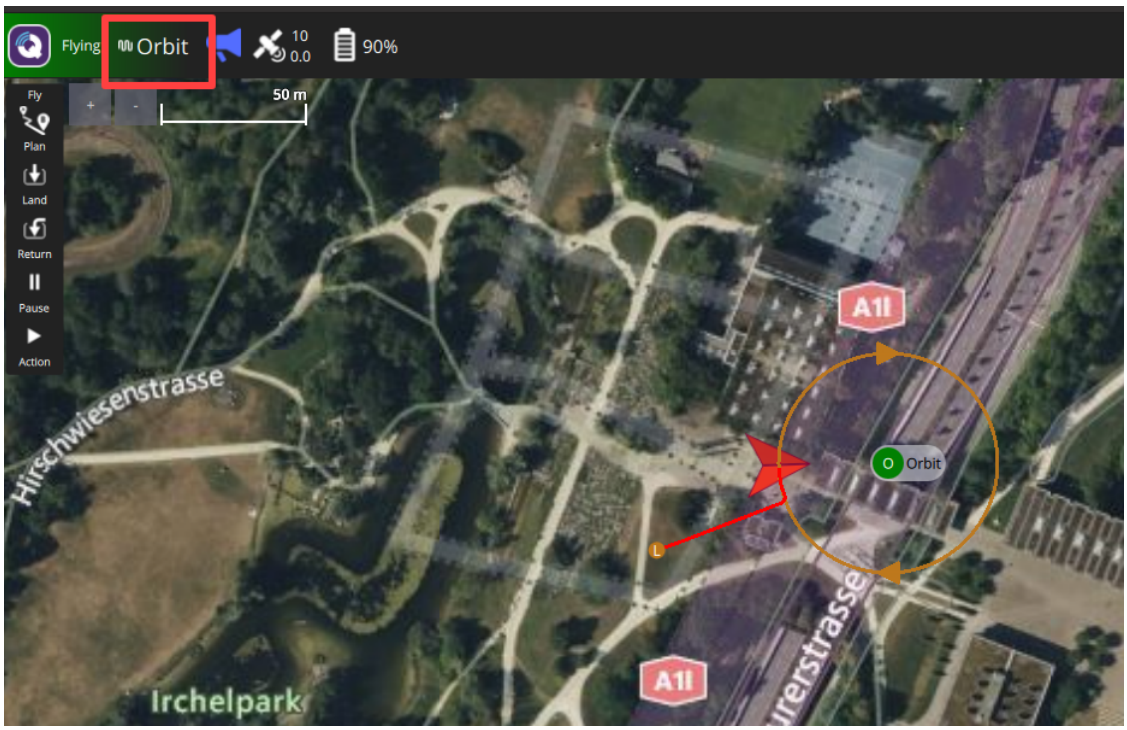
1. 鼠标左键点击地面作为环绕时的中心点，在弹出菜单中点击Orbit at location



2. 拖动上方滑条以发送环绕模式指令



3. 无人机进入环绕模式并沿规定轨道飞行



请求序列号为seq的任务项信息。系统对此消息的响应应为MISSION_ITEM_INT消息。 航点处理期间的确认消息。type字段指示此消息是正面确认（type=0）还是发生错误（type≠0）。 任务操作结果（在MISSION_ACK消息中）。

网络RTK

网络 RTK

概述

DJI 开放机载的网络 RTK 能力，结合无人机 4G 模块联网能力，可以使用网络 RTK 服务实现厘米级精度定位需求。开发者通过 PSDK 提供的网络 RTK 接口，可以配置 NTRIP 服务参数、启动和停止网络 RTK 服务，并通过回调函数实时监控 RTK 服务的运行状态。

本项目在 PSDK Network RTK 能力基础上增加 MAVLink 对接：地面端通过 MAVLink 下发 NTRIP 配置、启动/停止 Network RTK 服务，并通过 MAVLink `TUNNEL` 获取服务状态。

说明：

- 该功能适用于 Matrice 400 + 妙算 3 搭配使用。
- Network RTK 只负责建立 NTRIP/RTCM 数据链路，不直接输出飞行定位坐标。
- RTK 收敛后的高精度坐标由 RTK 精准定位功能输出到标准 MAVLink 定位消息。

接口总览

开发者需要先注册状态回调，再启动网络 RTK 服务。下表按功能分类列出网络 RTK 的主要接口和回调：

功能分类	接口 / 回调函数	功能说明
状态回调注册	`DjiNetworkRtk_RegReceiveNetworkRtkStateCallback()`	注册网络 RTK 服务状态接收 建议在启动服
启动服务	`DjiNetworkRtk_StartService()`	使用指定的 N 配置启动网络
停止服务	`DjiNetworkRtk_StopService()`	停止网络 RTK 断开高精度定

本项目 MAVLink 封装接口如下：

功能分类	MAVLink 接口	功能说明
配置下发	`TUNNEL payload_type=32791`	分片下发 `T_DjiNetworkRtkServiceConfig` 配置
状态回传	`TUNNEL payload_type=32792`	回传 Network RTK 状态、配置接收进度和最近一次 PSDK 返回码
启停控制	`COMMAND_LONG / MAV_CMD_USER_2`	启动、停止、查询 Network RTK 服务

相关文件

文件	说明
`src/dji_mavlink/dji_network_rtk_mavlink.hpp`	Network RTK MAVLink 接口和 payload type 定义
`src/dji_mavlink/dji_network_rtk_mavlink.cpp`	配置分片重组、参数校验、Network RTK 启停、状态回传
`src/dji_mavlink/mavlink_connector.cpp`	`MAV_CMD_USER_2` 和 `TUNNEL` 入口
`psdk_lib/include/dji_network_rtk.h`	DJI PSDK Network RTK 官方接口

基础概念

网络 RTK 服务状态

网络 RTK 服务运行过程中会通过状态回调通知当前的服务状态，各状态含义如下：

状态枚举	值	说明
`DJI_LOG_IN`	`0`	正在登录 NTRIP 服务
`DJI_CONNECTING`	`1`	正在连接远端服务器
`DJI_TRANSFER`	`2`	数据传输中，服务已成功启动。注意：此状态不代表 RTK 解算已收敛，需另行订阅 RTK 定位数据验证精度
`DJI_RECONNECT`	`3`	正在尝试重新连接

状态枚举	值	说明
`DJI_BROKEN`	`4`	连接断开
`DJI_STOPPING`	`5`	服务正在停止
`DJI_REBOOTING`	`6`	需要重启飞机以切换 RTK 源。飞机不会自动重启，需手动重启后再次调用启动接口

NTRIP 服务配置参数

启动网络 RTK 服务时需要配置以下 NTRIP 参数：

参数	最大长度	结构体长度	说明
`account`	15 字符	16 bytes	NTRIP 用户名
`password`	63 字符	64 bytes	NTRIP 密码
`host`	63 字符	64 bytes	NTRIP Caster 地址，仅支持 IP 地址，不支持域名
`port`	5 字符	6 bytes	NTRIP 端口号，范围 `1` 到 `65535`
`mountPoint`	31 字符	32 bytes	NTRIP 挂载点，仅允许字母、数字、`-`、`_`、`.`

完整 `T_DjiNetworkRtkServiceConfig` 长度为 182 bytes，超过 MAVLink `TUNNEL` 单包 128 bytes，因此本项目使用分片协议下发配置。

功能介绍

注意：

- 以下代码均为参考示例，旨在帮助开发者理解各功能的调用流程和实现思路。开发者可根据自身负载设备的实际硬件特性和应用场景进行适配和调整。
- 以下流程中，主动调用的接口由 MAVLink 命令或 TUNNEL 消息触发；状态回调由 SDK 运行时自动触发。

网络 RTK 推荐启动流程：

1. 注册 Network RTK 状态回调。

2. 接收地面端通过 MAVLink TUNNEL 下发的 NTRIP 配置。
3. 地面端发送 `MAV_CMD_USER_2 param1=1`。
4. 机载端调用 `DjiNetworkRtk_StartService()`。
5. 通过回调和 `TUNNEL payload_type=32792` 回传状态。
6. 若状态为 `DJI_REBOOTING`，手动重启飞机，重启后再次启动服务。
7. 若状态为 `DJI_TRANSFER`，继续通过 RTK 定位数据确认是否收敛到 Float/Fixed。

状态回调注册

建议在启动 RTK 服务之前先注册状态回调函数，以便及时获取服务状态变化。各状态的含义参见“基础概念 - 网络 RTK 服务状态”。

```
static void ReceiveNetworkRtkStateCallback(E_DjiNetworkRtkOnboardSt
ate state)
{
    switch (state) {
        case DJI_LOG_IN:
            USER_LOG_INFO("Network RTK state: Logging in");
            break;
        case DJI_CONNECTING:
            USER_LOG_INFO("Network RTK state: Connecting");
            break;
        case DJI_TRANSFER:
            USER_LOG_INFO("Network RTK state: Data transfer in prog
ress, service started successfully");
            break;
        case DJI_RECONNECT:
            USER_LOG_WARN("Network RTK state: Reconnecting");
            break;
        case DJI_BROKEN:
            USER_LOG_ERROR("Network RTK state: Connection broken");
            break;
        case DJI_STOPPING:
            USER_LOG_INFO("Network RTK state: Stopping");
            break;
        case DJI_REBOOTING:
            USER_LOG_WARN("Network RTK state: Rebooting required, p
lease manually reboot the aircraft");
            break;
        default:
            USER_LOG_WARN("Network RTK state: Unknown state %d", st
ate);
            break;
    }
```

```
}

returnCode = DjiNetworkRtk_RegReceiveNetworkRtkStateCallback(ReceiveNetworkRtkStateCallback);
if (returnCode != DJI_ERROR_SYSTEM_MODULE_CODE_SUCCESS) {
    USER_LOG_ERROR("Register network RTK state callback failed, error code: 0x%08X", returnCode);
}
```

本项目中，回调注册由 `DjiNetworkRtkMavlink_Start()` 在首次启动服务前自动完成。状态变化会同时通过 `STATUSTEXT` 和 `TUNNEL payload_type=32792` 回传到 MAVLink 端。

启动服务

通过 `DjiNetworkRtk_StartService()` 接口配置 NTRIP 参数并启动 RTK 服务。启动后需监控状态回调：

- 如果状态为 `DJI_REBOOTING`：表示当前 RTK 源不是自定义网络源，需要手动重启飞机完成源切换，重启后需再次调用此接口启动服务。
- 如果状态为 `DJI_TRANSFER`：表示网络 RTK 服务已连接成功。

```
T_DjiNetworkRtkServiceConfig config = {0};

// 配置 NTRIP 服务参数
strncpy((char *)config.account, "your_account", sizeof(config.account) - 1);
strncpy((char *)config.password, "your_password", sizeof(config.password) - 1);
strncpy((char *)config.host, "192.168.1.100", sizeof(config.host) - 1);  // 仅支持 IP 地址，不支持域名
strncpy((char *)config.port, "8001", sizeof(config.port) - 1);
strncpy((char *)config.mountPoint, "RTCM32_GGB", sizeof(config.mountPoint) - 1);

returnCode = DjiNetworkRtk_StartService(&config);
if (returnCode != DJI_ERROR_SYSTEM_MODULE_CODE_SUCCESS) {
    USER_LOG_ERROR("Start network RTK service failed, error code: 0x%08X", returnCode);
}
```

推荐的调用流程：

1. 调用 `DjiNetworkRtk_StartService()`。

2. 如果状态回调报告 `DJI_REBOOTING`，手动重启飞机。
3. 飞机重启后，再次调用 `DjiNetworkRtk_StartService()`。
4. 监控状态回调等待 `DJI_TRANSFER` 状态，并订阅 RTK 定位数据验证收敛。

本项目中，启动服务由 MAVLink 触发：

```
COMMAND_LONG.command = MAV_CMD_USER_2
COMMAND_LONG.param1 = 1
```

前提条件：地面端必须先通过 `TUNNEL payload_type=32791` 下发有效配置。未配置时启动会失败，并通过 `COMMAND_ACK=MAV_RESULT_FAILED` 和状态 payload 回传错误码。

停止服务

当不再需要使用网络 RTK 服务时，调用以下接口停止服务。停止后可通过回调收到 `DJI_STOPPING` 状态。

```
returnCode = DjiNetworkRtk_StopService();
if (returnCode != DJI_ERROR_SYSTEM_MODULE_CODE_SUCCESS) {
    USER_LOG_ERROR("Stop network RTK service failed, error code: 0
x%08X", returnCode);
}
```

本项目中，停止服务由 MAVLink 触发：

```
COMMAND_LONG.command = MAV_CMD_USER_2
COMMAND_LONG.param1 = 0
```

查询当前状态：

```
COMMAND_LONG.command = MAV_CMD_USER_2
COMMAND_LONG.param1 = 2
```

MAVLink 配置下发

Network RTK 配置通过 `MAVLink TUNNEL` 分片下发。

字段	值
`payload_type`	`32791`

字段	值
payload	NetworkRtkConfigChunkHeader + data

分片 Header

字段	类型	说明
version	uint8_t	当前为 1
config_id	uint8_t	配置批次 ID；新配置使用新的 ID
total_len	uint16_t	固定为 sizeof(T_DjiNetworkRtkServiceConfig)，当前为 182
offset	uint16_t	当前 data 在完整配置结构体中的偏移

每个 TUNNEL payload 最大 128 bytes。Header 为 6 bytes，因此单包最多携带 122 bytes 配置数据。

配置数据体必须按 DJI PSDK T_DjiNetworkRtkServiceConfig 的内存布局发送：

```
account[16] + password[64] + host[64] + port[6] + mountPoint[32]
```

MAVLink 状态回传

Network RTK 状态通过 MAVLink TUNNEL 回传。

字段	值
payload_type	32792
payload	NetworkRtkStatePayload

状态 Payload

字段	类型	说明
version	uint8_t	当前为 1
network_state	uint8_t	DJI Network RTK 状态

字段	类型	说明
`configured`	`uint8_t`	`1` 表示已有有效配置
`running`	`uint8_t`	`1` 表示服务处于登录、连接、传输或重连状态；`DJI_REBOOTING`、`DJI_BRO` 时为 `0`
`config_bytes_received`	`uint16_t`	当前配置已接收字节数
`config_total_len`	`uint16_t`	配置总长度
`last_return_code`	`uint64_t`	最近一次 DJI PSDK 返回码

MAVLink 使用流程

1. 地面端构造 `T_DjiNetworkRtkServiceConfig` 二进制结构体。
2. 使用 `MAVLink TUNNEL payload_type=32791` 分片发送完整配置。
3. 等待 `payload_type=32792` 状态回传，确认 `configured=1`。
4. 发送 `MAV_CMD_USER_2`，`param1=1` 启动 Network RTK。
5. 监听 `payload_type=32792` 状态。若收到 `DJI_REBOOTING`，需要手动重启飞机，重启后再次发送启动命令。
6. RTK 服务进入 `DJI_TRANSFER` 后，继续通过标准 MAVLink 定位消息确认 RTK 是否 Float/Fixed。
7. 停止服务时发送 `MAV_CMD_USER_2`，`param1=0`。

Python 例程

以下例程使用 `pymavlink` 构造 `T_DjiNetworkRtkServiceConfig`，通过 `TUNNEL payload_type=32791` 分片下发配置，然后使用 `MAV_CMD_USER_2` 启动 Network RTK，并监听 `payload_type=32792` 状态回传。

```
import struct
import time
from pymavlink import mavutil


NETWORK_RTK_CONFIG_PAYLOAD_TYPE = 32791
NETWORK_RTK_STATE_PAYLOAD_TYPE = 32792
MAV_CMD_USER_2 = mavutil.mavlink.MAV_CMD_USER_2

CONFIG_TOTAL_LEN = 16 + 64 + 64 + 6 + 32
```

```
CHUNK_HEADER_FMT = "<BBHH"      # version, config_id, total_len, off
set
STATE_PAYLOAD_FMT = "<BBBBHHQ" # version, state, configured, runnin
g, rx_bytes, total_len, last_return_code
CHUNK_HEADER_LEN = struct.calcsize(CHUNK_HEADER_FMT)
MAX_TUNNEL_PAYLOAD_LEN = 128
MAX_CHUNK_DATA_LEN = MAX_TUNNEL_PAYLOAD_LEN - CHUNK_HEADER_LEN

NETWORK_RTK_STATES = {
    0: "DJI_LOG_IN",
    1: "DJI_CONNECTING",
    2: "DJI_TRANSFER",
    3: "DJI_RECONNECT",
    4: "DJI_BROKEN",
    5: "DJI_STOPPING",
    6: "DJI_REBOOTING",
    255: "UNKNOWN",
}


def get_current_millis():
    return int(time.time() * 1000)


def fixed_c_string(text, size):
    raw = text.encode("ascii")
    if len(raw) > size - 1:
        raise ValueError(f"字段过长: {text}, 最大长度 {size - 1}")
    return raw + b"\x00" * (size - len(raw))


def build_network_rtk_config(account, password, host, port, mount_p
oint):
    # 必须与 T_DjiNetworkRtkServiceConfig 内存布局一致：
    # account[16] + password[64] + host[64] + port[6] + mountPoint
[32]
    return b"".join([
        fixed_c_string(account, 16),
        fixed_c_string(password, 64),
        fixed_c_string(host, 64),
        fixed_c_string(str(port), 6),
        fixed_c_string(mount_point, 32),
    ])
```

```
def send_tunnel(master, payload_type, payload):
    tunnel_payload = list(payload) + [0] * (128 - len(payload))
    master.mav.tunnel_send(
        master.target_system,
        master.target_component,
        payload_type,
        len(payload),
        tunnel_payload,
    )


def send_network_rtk_config(master, config_bytes, config_id=1):
    if len(config_bytes) != CONFIG_TOTAL_LEN:
        raise ValueError(f"配置长度错误: {len(config_bytes)} != {CONF
IG_TOTAL_LEN}")

    print(f"开始下发 Network RTK 配置，总长度 {len(config_bytes)} byte
s")

    for offset in range(0, len(config_bytes), MAX_CHUNK_DATA_LEN):
        chunk = config_bytes[offset:offset + MAX_CHUNK_DATA_LEN]
        header = struct.pack(CHUNK_HEADER_FMT, 1, config_id, CONFIG
_TOTAL_LEN, offset)
        payload = header + chunk

        send_time = get_current_millis()
        send_tunnel(master, NETWORK_RTK_CONFIG_PAYLOAD_TYPE, payloa
d)
        print(f"[{send_time}] 发送配置分片: config_id={config_id}, of
fset={offset}, size={len(chunk)}")

        # 等待一次状态回传，确认机载端接收进度
        deadline = time.time() + 2.0
        while time.time() < deadline:
            msg = master.recv_match(type="TUNNEL", blocking=True, t
imeout=0.5)
            if msg and msg.payload_type == NETWORK_RTK_STATE_PAYLOA
D_TYPE:
                state = parse_network_rtk_state(msg)
                if state:
                    print_state(state)
                    break

        time.sleep(0.05)
```

```
def parse_network_rtk_state(msg):
    payload = bytes(msg.payload[:msg.payload_length])
    if len(payload) < struct.calcsize(STATE_PAYLOAD_FMT):
        return None

    version, state, configured, running, rx_bytes, total_len, last_
return_code = struct.unpack_from(
        STATE_PAYLOAD_FMT, payload, 0
    )
    return {
        "version": version,
        "state": state,
        "configured": configured,
        "running": running,
        "rx_bytes": rx_bytes,
        "total_len": total_len,
        "last_return_code": last_return_code,
    }


def print_state(state):
    now_ms = get_current_millis()
    state_name = NETWORK_RTK_STATES.get(state["state"], f"UNKNOWN
({state['state']})")
    print(
        f"[{now_ms}] Network RTK 状态: {state_name}, "
        f"configured={state['configured']}, running={state['runnin
g']}, "
        f"rx={state['rx_bytes']}/{state['total_len']}, "
        f"last_return_code=0x{state['last_return_code']:016X}"
    )

    if state["state"] == 6:
        print("  提示：收到 DJI_REBOOTING，需要手动重启飞机，重启后再次发
送启动命令")
    elif state["state"] == 2:
        print("  提示：收到 DJI_TRANSFER，网络 RTK 数据链路已建立，但仍需
查看 GPS_RAW_INT.fix_type 确认是否收敛")


def send_network_rtk_command(master, action):
    # action: 1=start, 0=stop, 2=status
    send_time = get_current_millis()
    print(f"[{send_time}] 发送 Network RTK 命令: action={action}")
```

```
    master.mav.command_long_send(
        master.target_system,
        master.target_component,
        MAV_CMD_USER_2,
        0,
        action,
        0, 0, 0, 0, 0, 0
    )

    ack = master.recv_match(type="COMMAND_ACK", blocking=True, timeout=5.0)
    if ack:
        ack_time = get_current_millis()
        print(
            f"[{ack_time}] 收到 COMMAND_ACK: "
            f"command={ack.command}, result={ack.result}, delay={ack_time - send_time}ms"
        )
        return ack.result == mavutil.mavlink.MAV_RESULT_ACCEPTED

    print(f"[{get_current_millis()}] 超时：未收到 Network RTK 命令应答")
    return False


# 连接无人机（UDP 端口 14550）
master = mavutil.mavlink_connection("udpin:0.0.0.0:14550")
print("等待无人机心跳...")
master.wait_heartbeat()
print(f"无人机已连接: system={master.target_system}, component={master.target_component}")

# 按实际 NTRIP 服务修改以下参数。host 必须是 IPv4/IPv6 地址，不支持域名。
config = build_network_rtk_config(
    account="your_account",
    password="your_password",
    host="192.168.1.100",
    port="8001",
    mount_point="RTCM32_GGB",
)

send_network_rtk_config(master, config, config_id=1)

if not send_network_rtk_command(master, action=1):
```

```
    raise SystemExit("Network RTK 启动命令失败")

print("开始监听 Network RTK 状态，按 Ctrl+C 停止服务")

try:
    while True:
        msg = master.recv_match(type="TUNNEL", blocking=True, timeo
ut=5.0)
        if msg is None:
            print(f"[{get_current_millis()}] 超时：未收到 Network RTK
状态")
            continue

        if msg.payload_type != NETWORK_RTK_STATE_PAYLOAD_TYPE:
            continue

        state = parse_network_rtk_state(msg)
        if state:
            print_state(state)

except KeyboardInterrupt:
    print("准备停止 Network RTK")
finally:
    send_network_rtk_command(master, action=0)
    print("Network RTK 例程结束")
```

注意事项

- Network RTK 配置中不要使用域名，DJI PSDK 接口要求 `host` 为 IPv4 或 IPv6 地址。
- `DJI_TRANSFER` 只表示网络 RTK 数据链路建立，不表示 RTK 已经固定解。定位质量必须查看 `GPS_RAW_INT.fix_type` 或 RTK 订阅输出。
- 配置分片为二进制协议，不是 JSON；地面端需要按小端格式写入 header。
- 密码不会通过日志和状态文本输出，地面端也应避免在 UI 或日志中明文保存。
- 如果未先发送有效配置就启动，机载端会返回失败，并通过状态 payload 回传错误码。

Mavros功能

MAVROS 使用说明

本文说明本项目将 DJI PSDK 能力转换为 MAVLink 后，在 MAVROS 中通常如何查看、控制和理解这些数据。

- PSDK 侧输出的 MAVLink 数据在 MAVROS 中对应哪些 topic 或服务。
- MAVROS 可以通过哪些入口发送位置、速度、姿态、航点、相机和录像相关命令。
- 哪些能力已经在当前项目代码中有对应的 MAVLink 处理。
- 哪些 topic 名称可能受 MAVROS 版本、插件配置和命名空间影响。

1. 系统链路

当前项目的典型链路如下：

```
DJI PSDK / M4TD
  -> 项目内 MAVLink 转换模块
  -> UDP 14550
  -> MAVROS
  -> ROS2 topic / service
```

如果 PSDK 程序和 MAVROS 都运行在妙算 3 本机，常见配置为：

```
PSDK MAVLink 目标：127.0.0.1:14550
MAVROS 监听地址：udp://:14550@
```

如果 MAVROS 运行在另一台设备上，PSDK 目标地址不能使用 `127.0.0.1`，应改为 MAVROS 所在设备的实际 IP。

2. MAVROS 启动示例

```
ros2 run mavros mavros_node \
  --ros-args \
  -r __ns:=/mavros \
  -p "fcu_url:=udp://:14550@" \
  -p "target_system_id:=3" \
```

```
  -p "target_component_id:=1" \
  -p "fcu_protocol:=v2.0"
```

参数说明：

参数	说明
`fcu_url`	MAVROS 监听 MAVLink UDP 数据的地址。
`target_system_id`	MAVROS 期望通信的飞行器 system id，应与项目 `mavlink_config.json` 中的 system id 保持一致。
`target_component_id`	MAVROS 期望通信的飞控 component id，通常为 `1`。
`fcu_protocol`	使用 MAVLink 2.0。

如果项目中修改了 `system_id`、`component_id` 或 camera component id，MAVROS 和其它地面站也应按相同 ID 配置或识别。

3. 快速查看方式

建议优先查看与功能直接相关的 topic，而不是只依赖完整的 `ros2 topic list`。

```
ros2 topic echo /mavros/state
ros2 topic echo /mavros/imu/data
ros2 topic echo /mavros/global_position/global
ros2 topic echo /mavros/local_position/pose
ros2 topic echo /mavros/battery
ros2 topic echo /mavros/rc/in
```

按类别查找扩展 topic：

```
ros2 topic list | grep -E "setpoint|local_position|global_position|
imu|battery|rc"
ros2 topic list | grep -E "camera|image|video|storage|mount|gimbal"
ros2 topic list | grep -E "mission|home|geofence|fence"
ros2 topic list | grep -E "distance|range|obstacle"
ros2 topic list | grep -E "debug|statustext|param"
```

不同 MAVROS 版本和插件配置会导致 topic 名称略有差异。文档中的 topic 名称以常见 MAVROS 命名为准，现场应以实际 `ros2 topic list` 输出为准。

4. 飞行器状态与遥测显示

PSDK / MAVLink 能力	项目 MAVLink 消息	MAVROS 常见
链路与飞行模式	`HEARTBEAT`	`/mavros/state`
扩展飞行状态	`EXTENDED_SYS_STATE`	`/mavros/exter`
系统健康与电源	`SYS_STATUS`	`/mavros/sys_`
文本提示	`STATUSTEXT`	`/mavros/statu`
系统时间	`SYSTEM_TIME`、`TIMESYNC`	时间同步相关或内部使用
飞控能力	`AUTOPILOT_VERSION`、`PROTOCOL_VERSION`	能力协商相关

5. 姿态、IMU、本地位置

PSDK / MAVLink 能力	项目 MAVLink 消息	MAVROS 常见显示
四元数姿态	`ATTITUDE_QUATERNION`	`/mavros/imu/data`
欧拉角姿态	`ATTITUDE`	`/mavros/imu/data` 或姿态相关输出
原始 IMU	`SCALED_IMU`	`/mavros/imu/data_raw`、`/mavros/imu/m`
高精度 IMU	`HIGHRES_IMU`	`/mavros/imu/data_raw` 或 IMU 相关输出
本地位置	`LOCAL_POSITION_NED`	`/mavros/local_position/pose`

PSDK / MAVLink 能力	项目 MAVLink 消息	MAVROS 常见显示
本地速度	LOCAL_POSITION_NED	/mavros/local_position/velocity_local
里程计	ODOMETRY	/mavros/odometry/out 或 odometry 相关 topic
高度	ALTITUDE	/mavros/altitude
飞行综合信息	VFR_HUD	/mavros/vfr_hud

6. GPS、全局位置与 Home 点

PSDK / MAVLink 能力	项目 MAVLink 消息	MAVROS 常见显示	说
原始 GPS	GPS_RAW_INT	/mavros/global_position/raw/fix	定 高 精
第二路 GPS	GPS2_RAW	GPS2 相关 topic	是 MA 版
全局位置	GLOBAL_POSITION_INT	/mavros/global_position/global	经 相
相对高度	GLOBAL_POSITION_INT	/mavros/global_position/rel_alt	相 点
航向	GLOBAL_POSITION_INT	/mavros/global_position/compass_hdg	指
GPS 原点	GPS_GLOBAL_ORIGIN	/mavros/global_position/gp_origin	全
Home 点	HOME_POSITION	/mavros/home_position/home	返 点

PSDK / MAVLink 能力	项目 MAVLink 消息	MAVROS 常见显示	说
RTK 信息	`GPS_RTK`、`GPS2_RTK`	RTK/GPS 相关输出	当 相 MA 显

7. 电池、遥控器、动力与避障

PSDK / MAVLink 能力	项目 MAVLink 消息	MAVROS 常见显示
电池状态	`BATTERY_STATUS`	`/mavros/battery`
遥控器通道	`RC_CHANNELS`	`/mavros/rc/in`
遥控覆盖输入	`RC_CHANNELS_OVERRIDE`	MAVROS RC override 相关入口
手动控制输入	`MANUAL_CONTROL`	MAVROS manual control 相关服务 topic
执行器目标	`ACTUATOR_CONTROL_TARGET`	actuator/setpoint 相关输出
电调信息	`ESC_INFO`、`ESC_STATUS`	`/mavros/esc_info`、`/mavros/esc` 或相关 topic
舵机/PWM 输出	`SERVO_OUTPUT_RAW`	`/mavros/servo_output_raw`
距离传感器	`DISTANCE_SENSOR`	`/mavros/distance_sensor/*`
障碍物距离	`OBSTACLE_DISTANCE`	obstacle/distance 相关输出

如果 MAVROS 日志出现：

```
DS: Plugin not configured!
```

表示 MAVLink 距离类消息可能已经进入 MAVROS，但 MAVROS 的 distance_sensor 插件尚未配置，因此不会按预期发布距离 topic。

8. MAVROS 位置、速度和姿态控制

本项目不仅输出遥测，也接收 MAVROS 发送的控制 setpoint。代码中已经处理以下 MAVLink 输入：

MAVROS 控制入口	MAVLink 输入消息	项目处理
/mavros/setpoint_raw/local	SET_POSITION_TARGET_LOCAL_NED	解码后保 MavrosS
/mavros/setpoint_raw/global	SET_POSITION_TARGET_GLOBAL_INT	解码后保 MavrosS
/mavros/setpoint_raw/attitude	SET_ATTITUDE_TARGET	解码后保 MavrosS
manual control 相关入口	MANUAL_CONTROL	进入异步
RC override 相关入口	RC_CHANNELS_OVERRIDE	解码并保

项目也会将收到的 setpoint 进行 MAVLink 回显：

回显消息	MAVROS 常见显示	说明
POSITION_TARGET_LOCAL_NED	/mavros/setpoint_raw/target_local	本地位置 速度目标
POSITION_TARGET_GLOBAL_INT	/mavros/setpoint_raw/target_global	全球位置 速度目标

回显消息	MAVROS 常见显示	说明
ATTITUDE_TARGET	/mavros/setpoint_raw/target_attitude	姿态目标

8.1 使用条件

位置、速度和姿态 setpoint 类控制通常需要满足以下条件：

- 飞行器已进入项目支持的外部控制或 Offboard 类模式。
- MAVROS 持续发布 setpoint，频率不能过低。
- 项目内 setpoint rate checker 认为输入有效。
- 飞行器当前状态允许 PSDK 控制，例如未处于飞控接管的返航、低电量降落、失控保护等状态。
- 若启用地理围栏或安全区域判断，目标点需要位于允许区域内。

8.2 本地速度控制示例

MAVROS ROS2 中常见消息类型为 `mavros_msgs/msg/PositionTarget`。下面示例表达“只控制本地 NED 速度，不控制位置和加速度”：

```
ros2 topic pub /mavros/setpoint_raw/local mavros_msgs/msg/PositionTarget "{
  header: {frame_id: 'map'},
  coordinate_frame: 1,
  type_mask: 3527,
  velocity: {x: 1.0, y: 0.0, z: 0.0},
  yaw_rate: 0.0
}"
```

说明：

- `coordinate_frame: 1` 常见含义为 `FRAME_LOCAL_NED`。
- `type_mask` 用于指定忽略哪些字段。不同控制需求应按 MAVROS/MAVLink 常量组合。
- NED 坐标中 `z` 方向向下，使用时要注意与 ROS ENU 坐标的方向差异。

8.3 本地位置控制示例

```
ros2 topic pub /mavros/setpoint_raw/local mavros_msgs/msg/PositionTarget "{
```

```
  header: {frame_id: 'map'},
  coordinate_frame: 1,
  type_mask: 3576,
  position: {x: 2.0, y: 0.0, z: -1.5},
  yaw: 0.0
}"
```

说明：

- 示例表达本地 NED 目标位置。
- `z: -1.5` 表示相对本地坐标向上 1.5 m。
- 实际飞行速度、加减速和到点行为由项目飞控控制逻辑和 DJI 飞控接口共同决定。

8.4 全球位置控制

全球位置控制通过 `/mavros/setpoint_raw/global` 进入，MAVLink 对应 `SET_POSITION_TARGET_GLOBAL_INT`。常见用途是按经纬度和高度发送目标点。

MAVROS 常见消息类型为 `mavros_msgs/msg/GlobalPositionTarget`，字段包括：

- `latitude`
- `longitude`
- `altitude`
- `velocity`
- `acceleration_or_force`
- `yaw`
- `yaw_rate`
- `type_mask`
- `coordinate_frame`

项目接收后会保存为 global setpoint，并可回显到 `/mavros/setpoint_raw/target_global`。

8.5 姿态目标控制

姿态目标通过 `/mavros/setpoint_raw/attitude` 进入，MAVLink 对应 `SET_ATTITUDE_TARGET`。项目代码中已有解码、缓存、频率检测和目标回显。

需要注意：

- DJI 飞控可开放的姿态控制能力和 PX4 Offboard 不完全一致。

- 当前项目中更建议优先使用位置/速度 setpoint 做无人机移动控制。
- 姿态目标适合做高级控制接口预留或特定场景调试，使用前需要确认飞行模式和安全条件。

9. 起飞、降落、返航、任务与模式

MAVROS 可通过 command 或 mode 相关服务发送标准 MAVLink 命令。项目代码中已处理多种常用命令：

MAVROS 常见入口	MAVLink 命令
`/mavros/cmd/takeoff` 或 `/mavros/cmd/command`	`MAV_CMD_NAV_TAKEOFF`
`/mavros/cmd/land` 或 `/mavros/cmd/command`	`MAV_CMD_NAV_LAND`
`/mavros/cmd/arming`	`MAV_CMD_COMPONENT_ARM_DISARM`
`/mavros/set_mode` 或 command long	`MAV_CMD_DO_SET_MODE` / `SET_MODE`
`/mavros/cmd/command`	`MAV_CMD_NAV_RETURN_TO_LAUNCH`
mission 相关服务	`MISSION_COUNT`、`MISSION_ITEM_INT`、`MISSION_ACK`
home position 相关服务	`MAV_CMD_DO_SET_HOME`、`MAV_CMD_GET_HOME_PO`
reposition	`MAV_CMD_DO_REPOSITION`
orbit	`MAV_CMD_DO_ORBIT`

任务状态输出：

项目 MAVLink 消息	MAVROS 常见显示	说明
`MISSION_CURRENT`	mission 相关 topic	当前任务航点或任务进度
`MISSION_ITEM_REACHED`	`/mavros/mission/reached`	航点到达事件。
`NAV_CONTROLLER_OUTPUT`	nav controller 相关 topic	导航控制输出， 具体显示取决于 MAVRO

项目 MAVLink 消息	MAVROS 常见显示	说明
		版本。

10. 云台、相机、拍照、录像与视频

当前项目代码中已经处理了多类相机和云台 MAVLink 能力。

10.1 云台控制与状态

MAVROS 常见入口或显示	MAVLink 消息/命令
mount control 相关服务/topic	`MAV_CMD_DO_MOUNT_CONFIGURE`
mount control 相关服务/topic	`MAV_CMD_DO_MOUNT_CONTROL`
gimbal manager 相关服务/topic	`MAV_CMD_DO_GIMBAL_MANAGER_PITCHYAW`
gimbal 状态相关 topic	`MOUNT_ORIENTATION`
gimbal manager 状态相关 topic	`GIMBAL_MANAGER_INFORMATION`、`GIMBAL_MANAGER_STATU`
gimbal device 状态相关 topic	`GIMBAL_DEVICE_ATTITUDE_STATUS`

10.2 相机信息与状态

项目 MAVLink 消息	MAVROS 常见显示	说明
`CAMERA_INFORMATION`	camera information 相关 topic 或服务响应	相机基础信息

项目 MAVLink 消息	MAVROS 常见显示	说明
`CAMERA_SETTINGS`	camera settings 相关 topic	相机模式、变焦等设置状
`CAMERA_CAPTURE_STATUS`	camera capture/status 相关 topic	拍照、录像状
`STORAGE_INFORMATION`	storage/camera 相关 topic	存储卡容量、剩余空间。
`CAMERA_TRIGGER`	camera trigger 相关 topic	拍照触发事件
`CAMERA_IMAGE_CAPTURED`	`/mavros/camera/image_captured` 或 camera 相关 topic	拍照完成事件
`VIDEO_STREAM_INFORMATION`	video stream 相关 topic 或服务响应	视频流地址、编码、分辨率等元信

说明：

- MAVROS 对 camera/video/storage topic 的支持依赖版本和插件配置。
- 如果 topic 名称不同，请使用 `ros2 topic list | grep -E "camera|image|video|storage"` 查找实际名称。
- `VIDEO_STREAM_INFORMATION` 只是视频流元信息，真正的 H264/RTP/UDP 视频流不一定通过 MAVROS 转发。

10.3 切换拍照和录像模式

项目已处理：

MAVROS 入口	MAVLink 命令	命令号	项目行为
`/mavros/cmd/command`	`MAV_CMD_SET_CAMERA_MODE`	`530`	切换拍照/录像工作模并向相机 c返回 `COMMAN`

`MAV_CMD_SET_CAMERA_MODE` 的关键参数：

参数	含义
param1	保留，通常为 0。
param2	camera mode。常见为 image/photo 或 video mode，具体值按 MAVLink camera mode 定义。

10.4 拍照

项目已处理两类拍照命令：

MAVROS 入口	MAVLink 命令	命令号	项目行
/mavros/cmd/command	MAV_CMD_IMAGE_START_CAPTURE	2000	启动拍 CAM
/mavros/cmd/command	MAV_CMD_DO_DIGICAM_CONTROL	203	兼容旧
/mavros/cmd/command	MAV_CMD_DO_TRIGGER_CONTROL	2003	按触发

单张拍照示例：

```
ros2 service call /mavros/cmd/command mavros_msgs/srv/CommandLong "
{
  broadcast: false,
  command: 2000,
  confirmation: 0,
  param1: 0.0,
  param2: 0.0,
  param3: 1.0,
  param4: 1.0,
  param5: .nan,
  param6: .nan,
  param7: .nan
}"
```

说明：

- param3: 1.0 表示拍 1 张。
- param4 可作为 capture sequence number。

- 命令成功后，可观察 camera capture/status、camera trigger、image captured 相关 topic。

10.5 开始录像与停止录像

项目已处理：

MAVROS 入口	MAVLink 命令	命令号	项目行
/mavros/cmd/command	MAV_CMD_VIDEO_START_CAPTURE	2500	切换/ACK，CAM
/mavros/cmd/command	MAV_CMD_VIDEO_STOP_CAPTURE	2501	停止录并发送

开始录像示例：

```
ros2 service call /mavros/cmd/command mavros_msgs/srv/CommandLong "
{
  broadcast: false,
  command: 2500,
  confirmation: 0,
  param1: 0.0,
  param2: 1.0,
  param3: .nan,
  param4: .nan,
  param5: .nan,
  param6: .nan,
  param7: .nan
}"
```

停止录像示例：

```
ros2 service call /mavros/cmd/command mavros_msgs/srv/CommandLong "
{
  broadcast: false,
  command: 2501,
  confirmation: 0,
  param1: 0.0,
  param2: .nan,
  param3: .nan,
```

```
  param4: .nan,
  param5: .nan,
  param6: .nan,
  param7: .nan
}"
```

10.6 请求相机状态消息

项目已处理 `MAV_CMD_REQUEST_MESSAGE` ，可请求以下消息：

请求消息	MAVLink message id	项目响应
`CAMERA_INFORMATION`	`259`	发送相机基础信息。
`CAMERA_SETTINGS`	`260`	发送当前相机设置。
`CAMERA_CAPTURE_STATUS`	`262`	发送拍照/录像状态。
`STORAGE_INFORMATION`	`261`	发送存储状态。
`VIDEO_STREAM_INFORMATION`	`269`	发送视频流元信息。

请求示例：

```
ros2 service call /mavros/cmd/command mavros_msgs/srv/CommandLong "
{
  broadcast: false,
  command: 512,
  confirmation: 0,
  param1: 262.0,
  param2: 0.0,
  param3: 0.0,
  param4: 0.0,
  param5: 0.0,
  param6: 0.0,
  param7: 0.0
}"
```

其中 `command: 512` 为 `MAV_CMD_REQUEST_MESSAGE` ， `param1` 为请求的 MAVLink message id。

11. 参数、日志、调试与扩展消息

PSDK / MAVLink 能力	项目 MAVLink 消息或命令	MAVR 常见显
参数列表	`PARAM_REQUEST_LIST`、`PARAM_VALUE`	MAVR param 相关服
单参数读写	`PARAM_REQUEST_READ`、`PARAM_SET`、`PARAM_VALUE`	MAVR param 相关服
日志列表/数据	`LOG_ENTRY`、`LOG_DATA`	log_tr 相关服 topic
debug 数值	`DEBUG`、`DEBUG_VECT`、`DEBUG_FLOAT_ARRAY`	debug 相关 t
命名数值	`NAMED_VALUE_FLOAT`、`NAMED_VALUE_INT`	debug 相关 t
串口控制	`SERIAL_CONTROL`	serial_ 相关人
自定义通道	`TUNNEL`	tunne topic

面向客户说明时，建议优先使用标准 MAVROS topic 和服务表达功能；`DEBUG`、`NAMED_VALUE_*` 更适合作为工程辅助信息。

12. 常见现象说明

12.1 `/mavros/state` 已连接，但某些 topic 不存在

可能原因：

- MAVROS 对应插件未启用。
- 当前 MAVROS 版本未发布该 topic。
- 项目只在特定设备状态下发送该类 MAVLink 消息。
- camera、gimbal、distance、video 等扩展 topic 名称与文档示例不同。

建议先按关键字搜索：

```
ros2 topic list | grep -E "camera|image|video|storage"
ros2 topic list | grep -E "mount|gimbal"
ros2 topic list | grep -E "distance|range|obstacle"
ros2 topic list | grep -E "setpoint|target"
```

12.2 setpoint 已发布，但无人机没有动作

可能原因：

- 当前模式不是项目支持的 Offboard/外部控制模式。
- setpoint 频率过低，被 rate checker 判定为无效。
- 飞控处于返航、低电量降落、失控保护等安全接管状态。
- 目标点不在允许区域内。
- `type_mask` 配置不符合预期，导致位置或速度字段被忽略。
- 坐标系理解错误，例如 ROS ENU 与 MAVLink NED 的方向差异。

12.3 相机命令有 ACK，但 MAVROS 中看不到相机 topic

可能原因：

- MAVROS camera 插件或当前版本不发布对应 topic。
- 相机状态只通过 `COMMAND_ACK`、`CAMERA_CAPTURE_STATUS` 或 `CAMERA_IMAGE_CAPTURED` 返回。
- 需要主动发送 `MAV_CMD_REQUEST_MESSAGE` 请求某些状态消息。
- 相机 component id 与 MAVROS/地面站识别的 component 不一致。

12.4 视频流与 MAVROS 的关系

`VIDEO_STREAM_INFORMATION` 可以通过 MAVLink/MAVROS 表达视频流元信息，但实际 H264/RTP/UDP 视频流通常直接由项目推送到指定 IP/端口，不一定经过 MAVROS topic。

如果 MAVROS 中能看到视频流信息，但播放器或 QGC 无画面，应检查：

- 视频目标 IP 是否为接收端实际 IP。
- UDP/RTP 端口是否一致。
- 防火墙是否允许 UDP。
- QGC 或播放器视频协议是否与项目输出一致。

- PSDK Liveview 是否成功启动。

13. 推荐给用户展示的 MAVROS 入口

功能	推荐入口
连接状态	`/mavros/state`
姿态	`/mavros/imu/data`
原始 IMU	`/mavros/imu/data_raw`
本地位置	`/mavros/local_position/pose`
本地速度	`/mavros/local_position/velocity_local`
GPS 原始定位	`/mavros/global_position/raw/fix`
全局位置	`/mavros/global_position/global`
相对高度	`/mavros/global_position/rel_alt`
航向	`/mavros/global_position/compass_hdg`
电池	`/mavros/battery`
遥控器	`/mavros/rc/in`
文字状态	`/mavros/statustext/recv`
本地位置/速度控制	`/mavros/setpoint_raw/local`
全球位置/速度控制	`/mavros/setpoint_raw/global`
姿态目标控制	`/mavros/setpoint_raw/attitude`
本地目标回显	`/mavros/setpoint_raw/target_local`
全球目标回显	`/mavros/setpoint_raw/target_global`
姿态目标回显	`/mavros/setpoint_raw/target_attitude`
拍照/录像命令	`/mavros/cmd/command`
相机状态	通过 `camera/image/video/storage` 关键字查找实际 topic

功能	推荐入口
云台状态	通过 `mount/gimbal` 关键字查找实际 topic
航点任务	通过 `mission` 关键字查找实际 topic/service

参考链接

项目参考链接

本文档整理 PSDK 转 MAVLink / MAVROS 项目开发、维护和问题定位时常用的官方参考资料入口。

1. DJI Payload SDK 文档

链接：

- https://developer.dji.com/doc/payload-sdk-tutorial/cn/

用途：

- 查看 DJI Payload SDK 的整体功能说明。
- 查询 PSDK 初始化、订阅飞控数据、相机管理、云台控制、Liveview、任务控制等接口。
- 确认不同飞机、负载、妙算平台对 PSDK 功能的支持情况。

建议重点查看：

- 基础功能
- 飞控数据订阅
- 相机管理
- 云台管理
- Liveview / 视频流
- 飞行控制
- HMS / 状态与错误码

2. 妙算 3 开发文档

链接：

- https://developer.dji.com/doc/payload-sdk-tutorial/cn/manifold-quick-start/manifold-development-introduction/manifold-product-overview.html

用途：

- 查看妙算 3 产品概览、系统环境和开发注意事项。
- 确认妙算 3 与飞机系统、PSDK、硬件接口、系统时间同步、视频编解码等能力的关系。
- 作为部署、环境确认和设备侧问题排查的参考入口。

建议结合项目文档查看：

- 妙算 3 开发环境与系统配置说明

3. MAVLink 官方文档

链接：

- https://mavlink.io/zh/

用途：

- 查看 MAVLink 协议、消息、命令、组件 ID、system ID、路由、心跳、任务、参数、相机、云台、Offboard 控制等标准定义。
- 确认项目中 PSDK 数据转换成 MAVLink 时，字段单位、坐标系、消息语义是否符合标准。
- 查询 QGC、MAVROS 或其它地面站对标准 MAVLink 消息的预期行为。

建议重点查看：

- Heartbeat / Connection Protocol
- Mission Protocol
- Parameter Protocol
- Command Protocol
- Manual Control Protocol
- Camera Protocol
- Gimbal Protocol v2
- Offboard Control
- Battery Protocol
- Time Synchronization
- Standard Messages / Commands
- MAVLink ID Assignment

4. MAVROS ROS2 源码

链接：

- https://github.com/mavlink/mavros/tree/ros2/mavros

用途：

- 查看 MAVROS ROS2 分支源码和插件实现。
- 确认 MAVLink 消息在 MAVROS 中转换成哪些 ROS2 topic、service 或参数接口。
- 排查某个 MAVLink 消息进入 MAVROS 后没有 topic 输出的问题。
- 查询 camera、gimbal、mission、setpoint、global_position、local_position、imu、battery 等插件行为。

建议重点查看：

- `mavros/src/plugins`
- `mavros/include`
- `mavros_msgs`
- setpoint raw 相关插件
- global position / local position 插件
- imu / sys status / battery 插件
- mission / command / param 插件
- camera / gimbal / mount 相关插件

建议结合项目文档查看：

- PSDK MAVLink 与 MAVROS 使用说明

5. 本项目相关说明文档

文档	用途
PSDK MAVLink 与 MAVROS 使用说明	说明项目中 MAVLink 输出、控制输入、相机录像、MAVROS topic/service 对应关系。
妙算 3 开发环境与系统配置说明	说明妙算 3 JetPack、CUDA、CMake、OpenCV、系统时间、硬件编解码等环境信息。

文档	用途
MAVLink 标准使用习惯与 PSDK 映射检查	记录项目中 MAVLink 标准化、PSDK 能力映射、后续优化方向。

6. 查询建议

开发和排查问题时建议按以下顺序查阅：

1. PSDK 功能是否支持：先查 DJI Payload SDK 文档。
2. 妙算 3 系统或硬件能力：查妙算 3 开发文档。
3. MAVLink 消息、命令和字段语义：查 MAVLink 官方文档。
4. MAVROS topic/service 是否支持：查 MAVROS ROS2 源码和插件。
5. 项目当前实现方式：查本项目 `src/` 代码和 `doc/` 文档。