目　录

- 免责声明
- 产品简介
  - 产品图片
  - 产品参数
- 物品清单
- 部件名称
- 固件刷机
- Mavlink
  - 接受数据
    - 心跳以及电池状态
    - 电池信息
    - GPS位置信息
    - GPS时间
    - 视频流信息
    - 无人机状态
    - 避障距离
    - 返航点信息
    - 无人机NED坐标 速度 加速度
    - 变焦倍数
    - 云台角度
    - 姿态四元数
  - 发送命令
    - 视频源切换并设置参数
    - 设置返航点以及返航高度
    - 设置相机焦点
    - 设置失控行为
    - 设置避障状态
    - 设置相机变焦
    - 执行任务命令
    - 录像命令
    - 停止录像命令

拍照命令
起飞命令
降落命令与取消降落
强制降落命令
返航命令以及取消返航命令
经纬度位置控制
经纬度飞行停止
飞行控制
设置云台角度
下视推流以及停止下视推流
视频流获取
视频流接收脚本示例
QGC遥控
三方遥控
QGC地面站
共享图传
通信技术
应用案例
参考文献

免责声明

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。请访问爱华芯官网以获取最新的产品信息。

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。

爱华芯（iHwasin）是爱华芯（大连）科技有限公司及其关联公司的商标。本文出现的产品名称、品牌等，均为其所属公司的商标或注册商标。本产品及文档为爱华芯（大连）科技有限公司版权所有。未经许可，不得以任何形式复制翻印。

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EC-OPi-AIpro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。
7. 连接至EC-OPi-AIpro的Typec 3.0 设备可能会对 GNSS、Wi-Fi 等信号产生干扰，必要时可采取电磁屏蔽措施以减小干扰。

产品简介

产品图片

















产品参数

EC-OPi-AIpro系列是Orange Pi AI Pro为 Onboard SDK及PSDK开发者打造的第二代微型计算机，内存提供了 12GB 和 24GB 两种版本；搭载 Atlas 200I 处理芯片，可更加快速地完成复杂的图形处理工作，其算力可达到 20TOPS，具备优秀的处理能力和响应速度;具备多种接口以连接不同的外部设备，适配多款 DJI 飞行平台、飞控系统等设备，拥有更强的灵活性与扩展性，同时为用户提供丰富便捷的开发途径，可以实现图像、视频等多种数据分析与推理计算等场景。

参数表如下：

NO	类别	参数
1	昇腾 AI 处理器	4 核 64 位 Arm 处理器 + AI 处理器
2	AI 算力	整数精度（INT8）：20 TOPS
3	视频编解码能力	支持H.264/H.265 硬件解码:40路1080P 30FPS，4路4K(3840x21
4	操作系统支持	Ubuntu 22.04 和 openEuler 22.03
5	存储	>板载 32MB 的 SPI Flash； Micro SD 卡插槽； eMMC 插座（可
6	视频显示	支持4K HDMI高清显示
7	软件接口	可支持DJI-PSDK/OSDK接口

8	外接接口	PSDK-TypeC安插接口
9		PCIe 2.5G 网口
10		3个USB3.0接口
11		Type-C USB3.0 OTG 接口
12		2 个 HDMI 接口
13		调试串口
14		耳机孔音频输入/输出
15		40 pin 扩展接口
16		TF(microSD)卡安装槽
17	内部接口	UART
18		百兆网
19		VCC-5V
20		VCC-12V
21	工作温度	商业版：-20℃~70℃；工业版：20℃~80℃；车规版：-40℃~85
22	尺寸	119mm*87mm*34mm
23	重量	259g

物品清单

EC-OPi-AIpro系列-M3系列物品清单

工控机 x 1



DJI-M3-EPort安装底座 x 1



M3x8mm 螺丝 x 4



2.5mm 六角扳手 x 1



EC-OPi-AIpro系列-M30系列物品清单

工控机 x 1



DJI-M30安装底座 x 1



数据线 x 1



M3X10mm 螺丝 x 6



2.5mm 六角扳手 x 1



EC-OPi-AIpro系列-M300物品清单

工控机 x 1



DJI-M300安装底座 x 1



数据线 x 1



M3X10mm 螺丝 x 6



2.5mm 六角扳手 x 1



EC-OPi-AIpro系列-M350物品清单

工控机 x 1



DJI-M350安装底座 x 1



数据线 x 1



M3X10mm 螺丝 x 6



2.5mm 六角扳手 x 1



EC-OPi-AIpro系列-FC30物品清单

工控机 x 1



DJI-FC30安装底座 x 1



供电线 x 1



数据线 x 1



天线 x 4



M3x12mm 螺丝 x 6

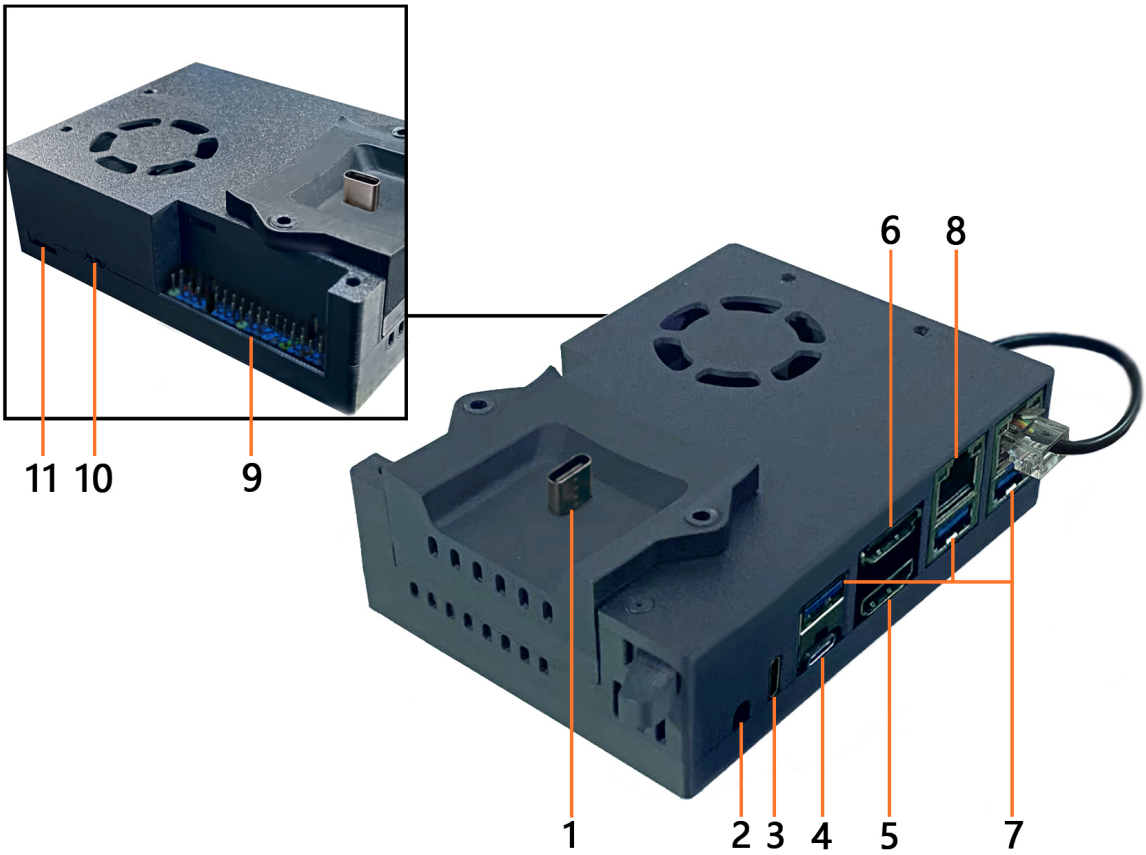


2.5mm 六角扳手 x 1



部件名称

EC-OPi-AIpro系列-M3系列



1：PSDK-TypeC安插接口

2：3.5mm耳机孔音频输入/输出

3：调试串口

4：TypeC3.0 OTG

5：HDMI1

6：HDMI0

7：USB3.0 x3

8：2.5G以太网口

9：40 pin 扩展接口

10：LED灯

11：TF(microSD)卡安装槽

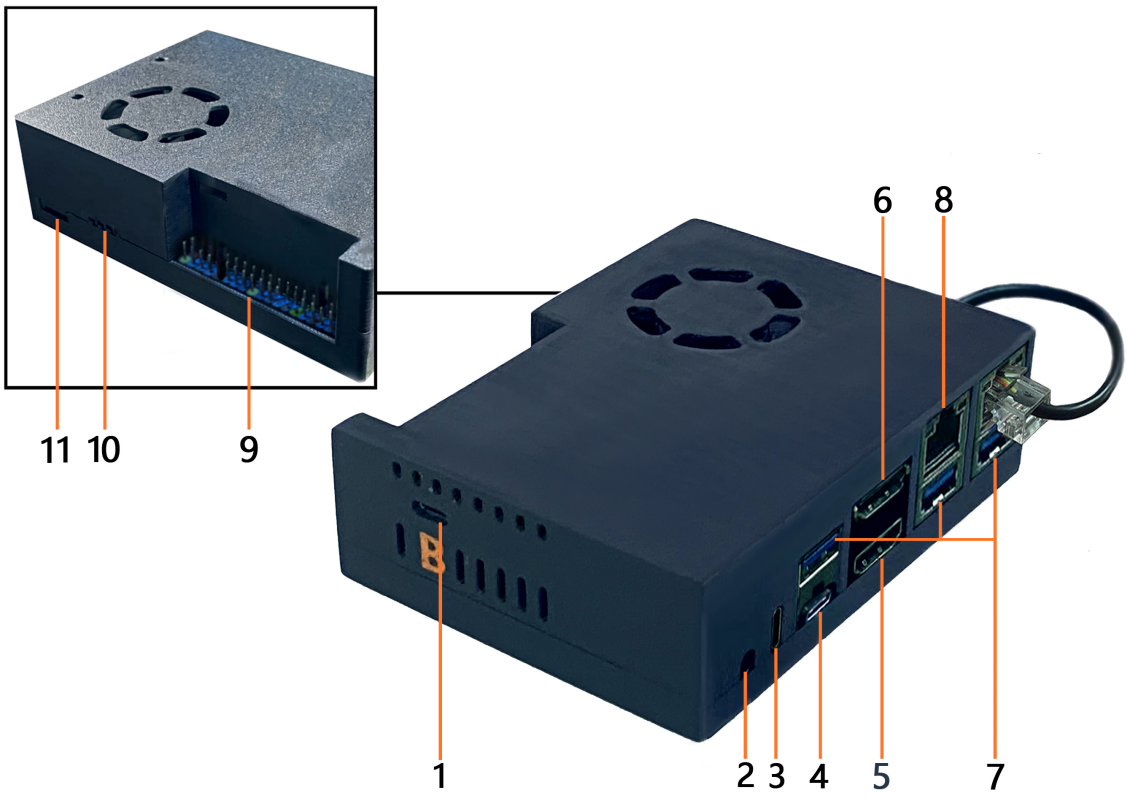
EC-OPi-AIpro系列-M3系列安装图



①：用2枚M3 x 8mm螺丝，将DJI-M3-EPort安装底座固定到M3无人机上。

②：沿箭头方向将EC-OPi-AIpro工控机插入DJI-M3-EPort按装底座。

EC-OPi-AIpro系列-M30系列



1：PSDK-TypeC安插接口

2：3.5mm耳机孔音频输入/输出

3：调试串口

4：TypeC3.0 OTG

5：HDMI1

6：HDMI0

7：USB3.0 x3

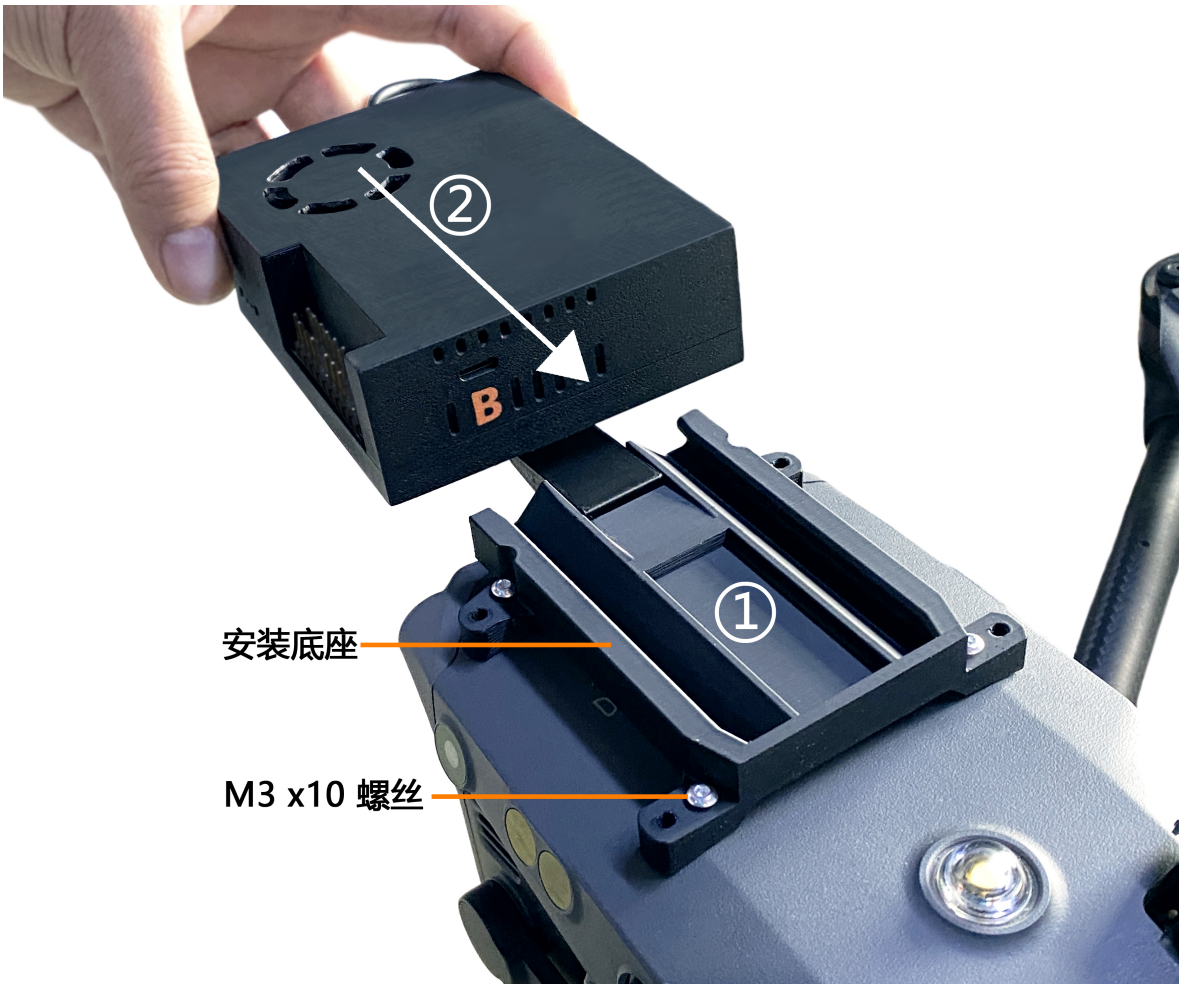
8：2.5G以太网口

9：40 pin 扩展接口

10：LED灯

11：TF(microSD)卡安装槽

EC-OPi-AIpro系列-M30系列安装图



①：用4枚M3 x 10mm螺丝，将安装底座固定到M30无人机上。

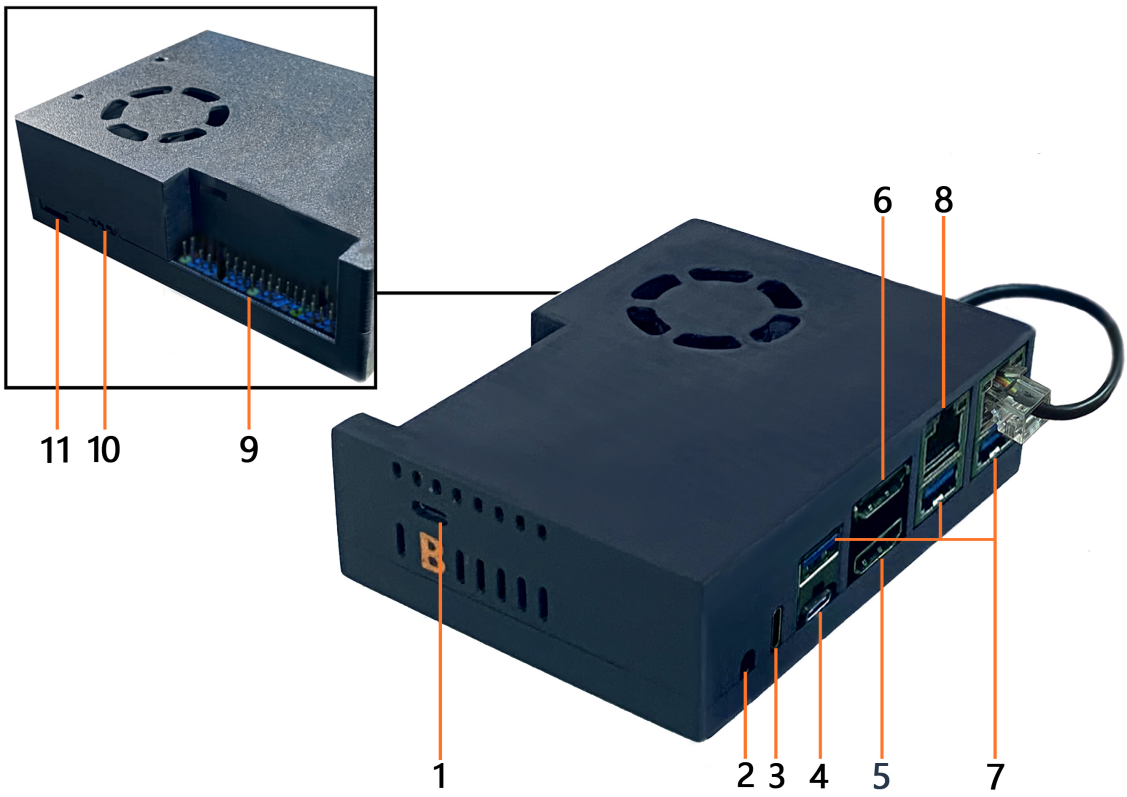
②：沿箭头方向将EC-OPi-AIpro工控机插入按装底座。



③：用数据线连接飞机与EC-OPi-AIpro工控机。

注：数据线Type-C插头B面朝下插入PSDK-TypeC接口，如安装图所示。

EC-OPi-AIpro系列-M300系列



1：PSDK-TypeC安插接口

2：3.5mm耳机孔音频输入/输出

3：调试串口

4：TypeC3.0 OTG

5：HDMI1

6：HDMI0

7：USB3.0 x3

8：2.5G以太网口

9：40 pin 扩展接口

10：LED灯

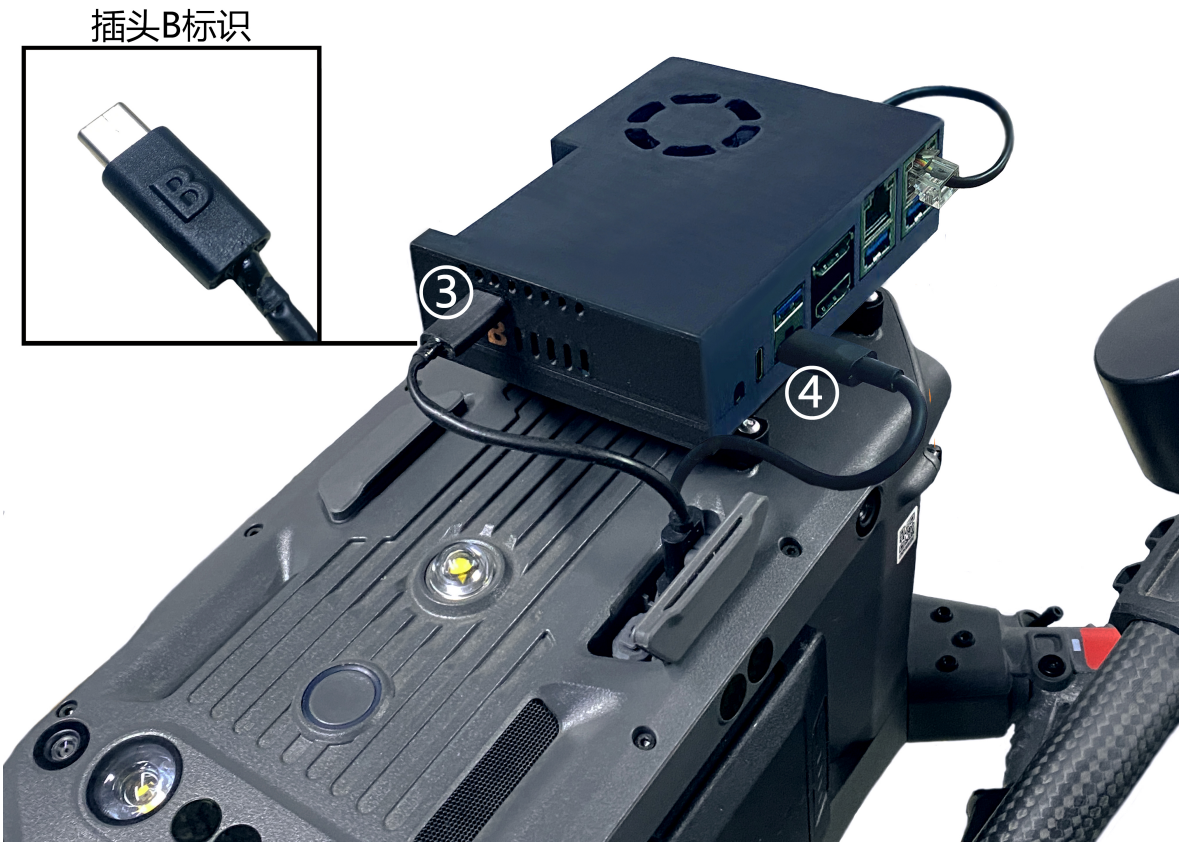
11：TF(microSD)卡安装槽

EC-OPi-AIpro系列-M300安装图



①：用4枚M3 x 10mm螺丝，将安装底座固定到M350无人机上。

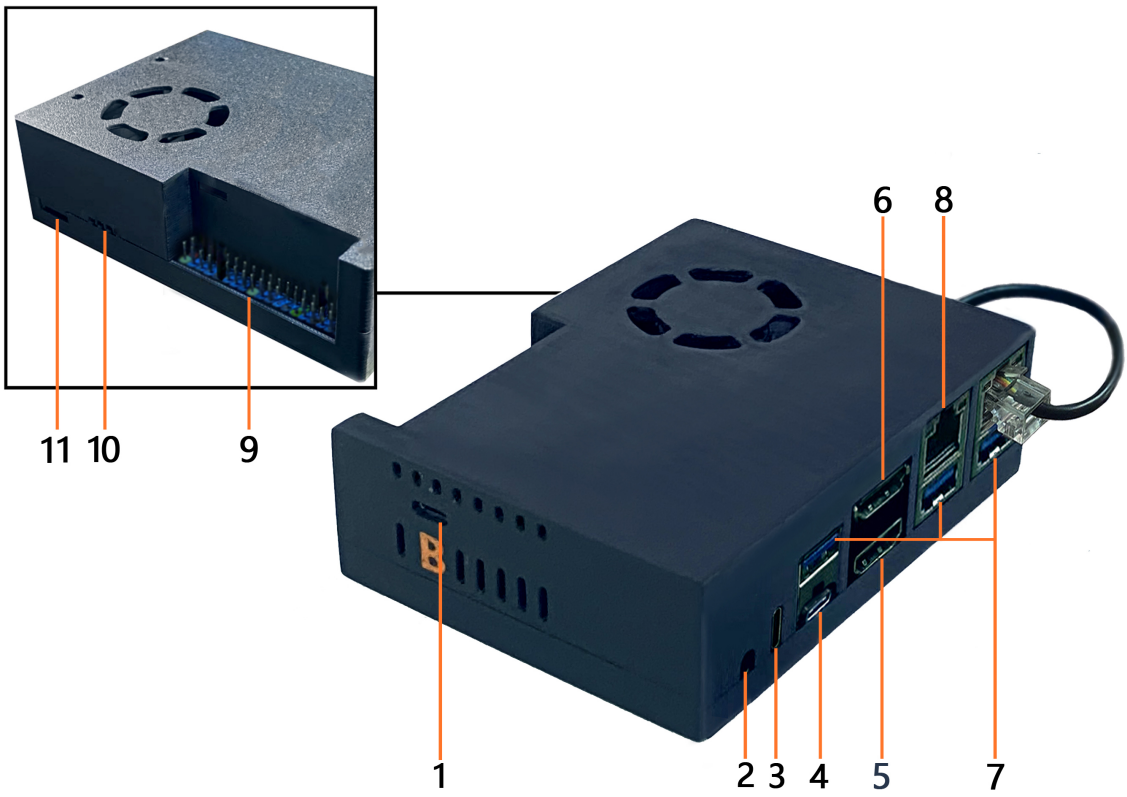
②：沿箭头方向将EC-OPi-AIpro工控机插入按装底座。



③：数据线有B标识的Type-C插头，B面向下插入EC-OPi-AIpro工控机PSDK-TypeC接口，如安装图所示。

④：数据线无标识的Type-C插头，插入USB3.0-TypeC插口，如安装图所示。

EC-OPi-AIpro系列-M350系列



1：PSDK-TypeC安插接口

2：3.5mm耳机孔音频输入/输出

3：调试串口

4：TypeC3.0 OTG

5：HDMI1

6：HDMI0

7：USB3.0 x3

8：2.5G以太网口

9：40 pin 扩展接口

10：LED灯

11：TF(microSD)卡安装槽

EC-OPi-AIpro系列-M350安装图



①：用4枚M3 x 10mm螺丝，将安装底座固定到M350无人机上。

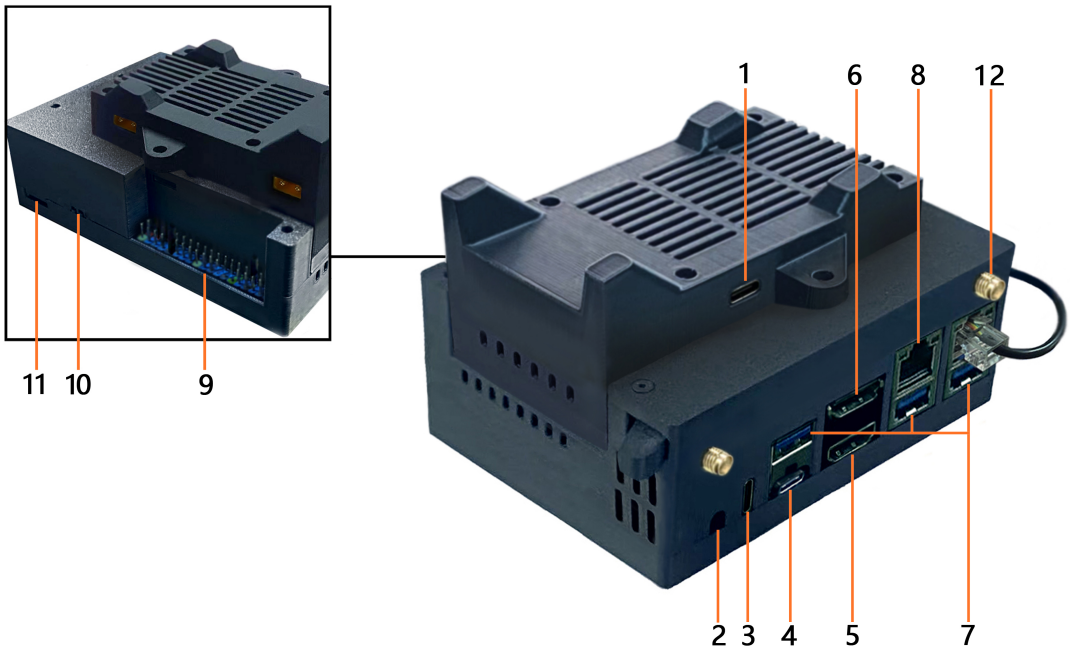
②：沿箭头方向将EC-OPi-AIpro工控机插入按装底座。



③：用数据线连接飞机与EC-OPi-AIpro工控机。

注：数据线Type-C插头B面朝下插入PSDK-TypeC接口，如安装图所示。

EC-OPi-AIpro系列-FC30系列



1：FC30-TypeC安插接口

2：3.5mm耳机孔音频输入/输出

3：调试串口

4：TypeC3.0 OTG

5：HDMI1

6：HDMI0

7：USB3.0 x3

8：2.5G以太网口

9：40 pin 扩展接口

10：LED灯

11：TF(microSD)卡安装槽

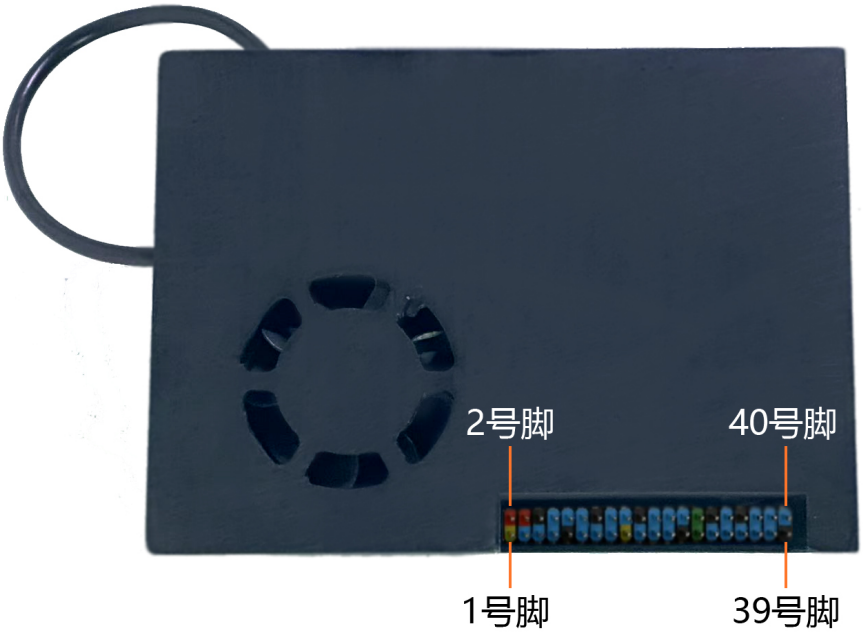
12：天线接口x4

EC-OPi-AIpro系列-FC30安装图





40pin 扩展接口线。


2号脚
40号脚
1号脚
39号脚

GPIO序号	GPIO	功能	引脚
		3.3V	1
76	GPIO2_12	SDA7	3
75	GPIO2_11	SCL7	5
226	GPIO7_02	UTXD7	7
		GND	9
82	GPIO2_18	URXD2	11
38	GPIO1_06		13
79	GPIO2_15		15
		3.3V	17
91	GPIO2_27	SPI0_SDO	19
92	GPIO2_28	SPI0_SDI	21
89	GPIO2_25	SPI0_SCLK	23
		GND	25
		SDA6	27
231	GPIO7_07	URXD7	29
84	GPIO2_20		31
128	GPIO4_00		33
228	GPIO7_04		35
3	GPIO0_03		37
		GND	39

引脚	功能	GPIO	GPIO序号
2	5V		
4	5V		
6	GND		
8	UTXD0	GPIO0_14	14
10	URXD0	GPIO0_15	15
12		GPIO7_03	227
14	GND		
16		GPIO2_16	80
18		GPIO0_25	25
20	GND		
22		GPIO0_02	2
24	SPI0_CS	GPIO2_26	90
26		GPIO2_19	83
28	SCL6		
30	GND		
32	PWM3	GPIO1_01	33
34	GND		
36	UTXD2	GPIO2_17	81
38		GPIO7_06	230
40		GPIO7_05	229

注：4号脚（5V），6号脚（GND），11号脚(URXD2），36号脚（UTXD2）已占用。

固件刷机

刷机准备需要的配件

1) TF 卡，最小 32GB 容量的 class10 级或以上的高速闪迪卡。强烈推荐使用 64GB 或以上容量的 TF 卡
2) TF 卡读卡器，用于读写 TF 卡
3) HDMI 转 HDMI 连接线，用于将开发板连接到 HDMI 显示器或者电视进行显示
4) Type-C转USB3.0 转接线，用于Type-C接口连接USB3.0 的存储设备
5) Type-C接口的数据线，用于调试串口、Type-C USB接口的Device等功能
6) HDMI 接口的显示器
7) 电源，Type-C 接口的 20V PD-65W 适配器

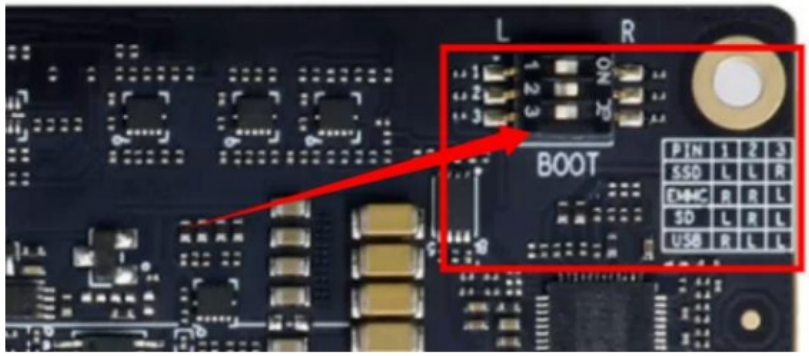
下载开发板的镜像和相关的资料

下载地址



控制启动设备的 3 个拨码开关的使用说明
开发板的 Linux 系统支持从 TF 卡、eMMC 和 SSD（支持 NVMe SSD 和 SATA SSD）启动，USB 启动不支持。具体从哪个设备启动是由开发板背面的 3 个拨码开

关来控制的



3 个拨码开关都支持左右 2 种设置状态，所以总共有 8 种设置状态，目前开发板只使用了其中的 3 种。不同的设置状态对应的启动设备如下表所示：

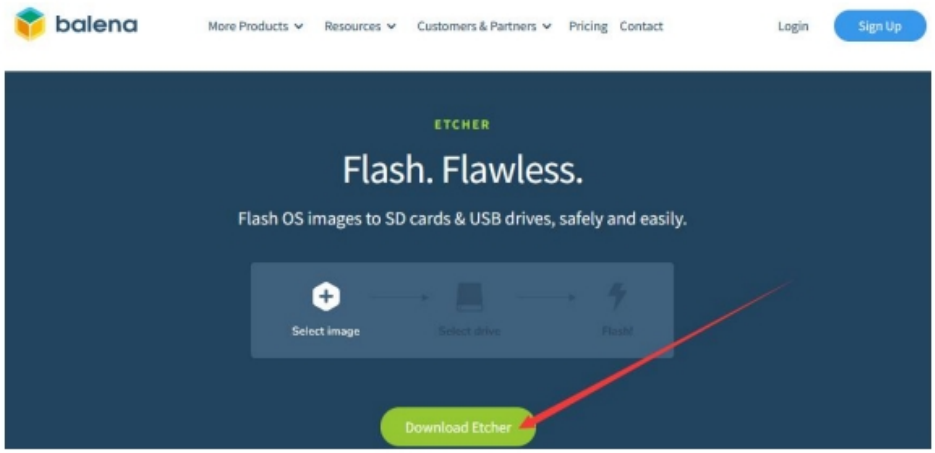
拨码开关 1	拨码开关 2	拨码开关 3	对应的启动设备
左	左	右	SATA SSD 和 NVMe SSD
右	右	左	eMMC
左	右	左	TF 卡

注意，SATA SSD和NVMe SSD的启动方式对应的拨码开关的设置状态是一样的。这两种启动方式是通过M2_TYPE引脚的电平来自动区分的。

另外请注意，切换拨码开关后必须重新拔插电源上下电才能让新的启动设备选项生效。通过开发板的复位按键来复位系统是不会让拨码开关新设置的配置生效的。

烧写 Linux 镜像到 TF 卡中的方法

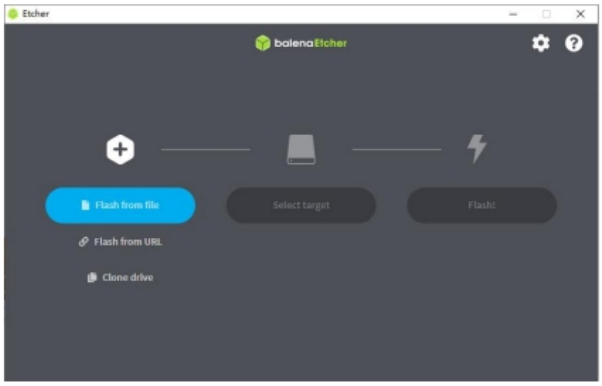
1) 首先准备一张 32GB 或更大容量的 TF 卡（推荐使用 64GB 或以上容量的 TF 卡），TF 卡的传输速度必须为 class10 级或 class10 级以上，建议使用闪迪等品牌的 TF 卡。
2) 然后把 TF 卡插入读卡器，再把读卡器插入电脑。
3) 从开发板的资料下载页面下载想要烧录的 Linux 镜像压缩包。
4) 然后下载用于烧录 Linux 镜像的软件——balenaEtcher，下载地址为：https://www.balena.io/etcher/
5) 进入 balenaEtcher 下载页面后，点击绿色的下载按钮会跳到软件下载的地方。



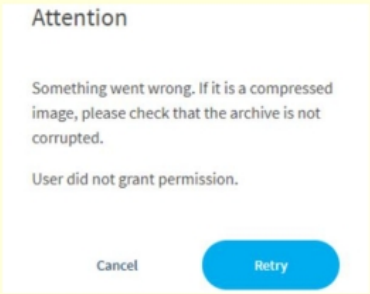
6) 然后可以选择下载 Portable 版本的 balenaEtcher，Portable 版本无需安装，双击打开就可以使用。



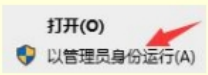
7) 打开后的 balenaEtcher 界面如下图所示:



打开 balenaEtcher 时如果提示下面的错误：



请选择 balenaEtcher 后点击右键，然后选择以管理员身份运行。

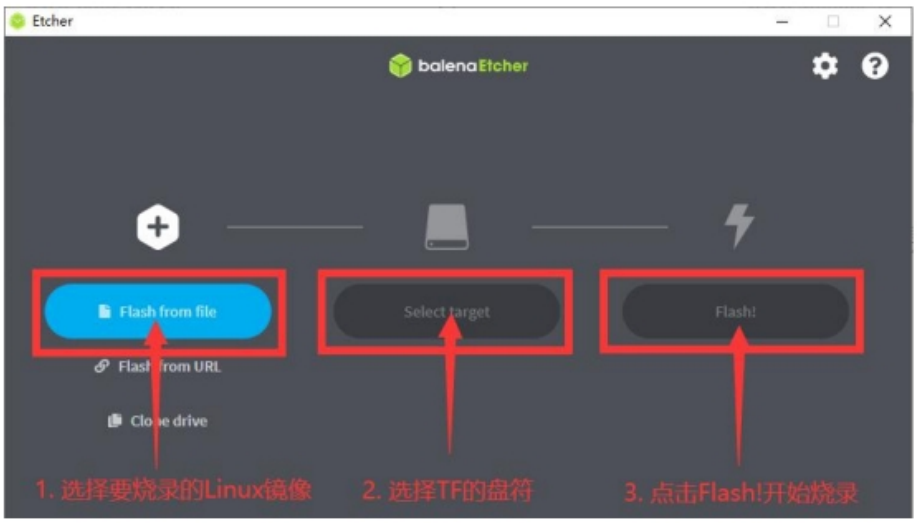


8) 使用 balenaEtcher 烧录 Linux 镜像的具体步骤如下所示：

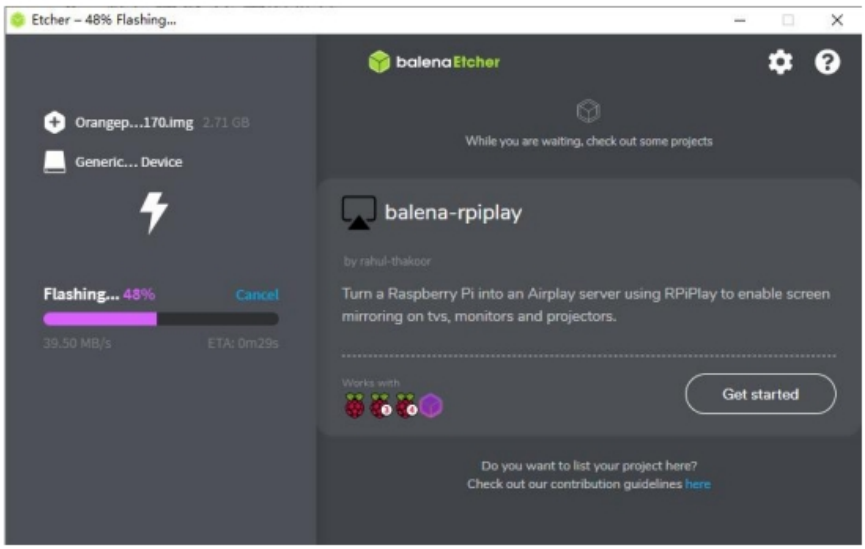
a. 首先选择要烧录的 Linux 镜像文件的路径。

b. 然后选择 TF 卡的盘符。

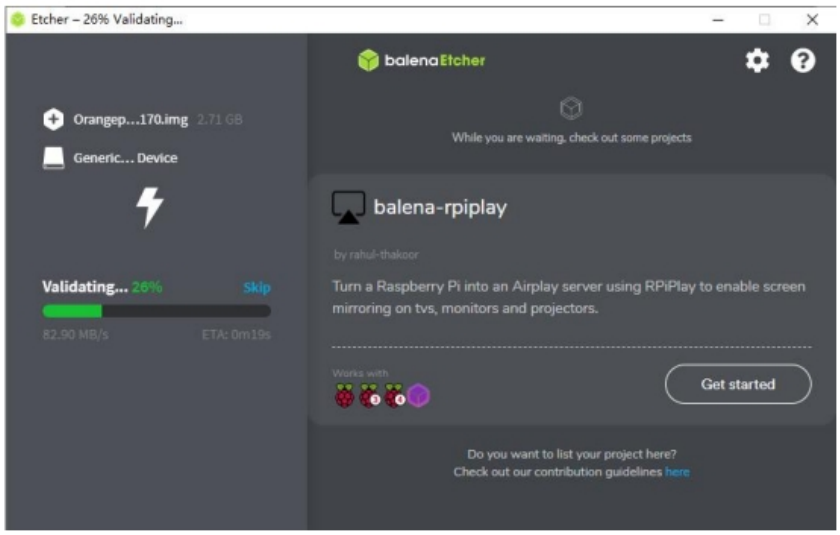
c. 最后点击 Flash 就会开始烧录 Linux 镜像到 TF 卡



9) balenaEtcher 烧录 Linux 镜像的过程显示的界面如下图所示，另外进度条显示紫色表示正在烧录 Linux 镜像到 TF 卡中。

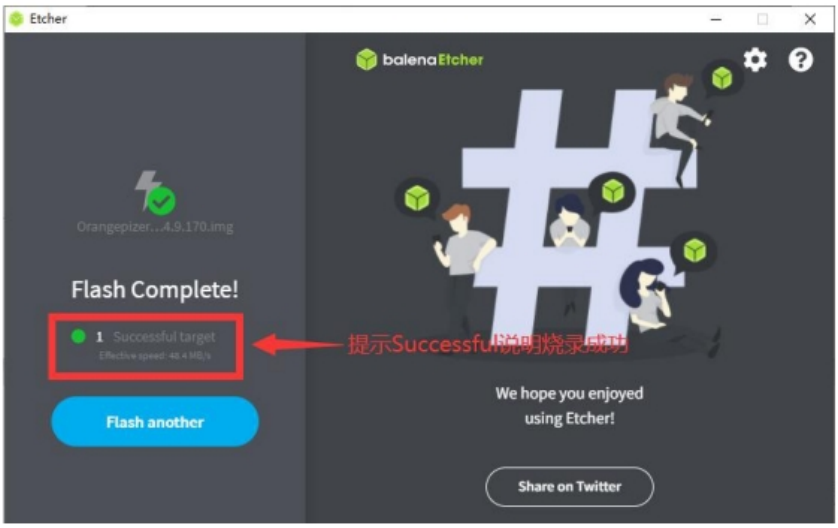


10) Linux 镜像烧录完后，balenaEtcher 默认还会对烧录到 TF 卡中的镜像进行校验，确保烧录过程没有出问题。如下图所示，显示绿色的进度条就表示镜像已经烧录完成，balenaEtcher 正在对烧录完成的镜像进行校验。



11) 成功烧录完成后 balenaEtcher 的显示界面如下图所示，如果显示绿色的指示图标说明镜像烧录成功，此时就可以退出 balenaEtcher，然后拔出 TF 卡插入到开发

板的 TF 卡槽中使用了。



注意，启动系统前请确保拨码开关拨到了TF卡启动的位置了。拨码开关的使用说明请参考控制启动设备的 3 个拨码开关的使用说明一小节的说明。

Mavlink

本模块实现大疆PSDK至MAVLink协议转换，促进设备间通信。兼容QGC地面站，实现无缝对接。提供定制开发服务，满足用户个性化需求。注意，MAVLink相关资料仅限内部团队使用，暂不对外开放。

首先安装软件所需环境

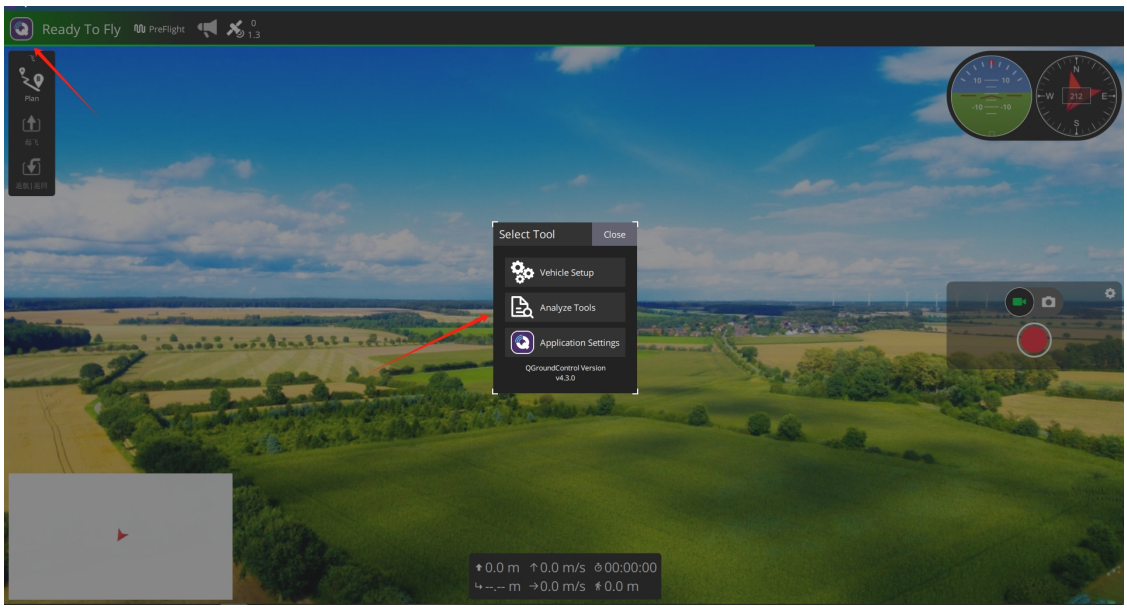
```
sudo apt update
sudo apt install libgstreamer1.0-dev libgstreamer-plugins-base1.0-dev
sudo apt-get install libgtk-3-dev
pkg-config --modversion gtk+-3.0
```

PC端需要安装QGC软件
下载地址 http://qgroundcontrol.com/

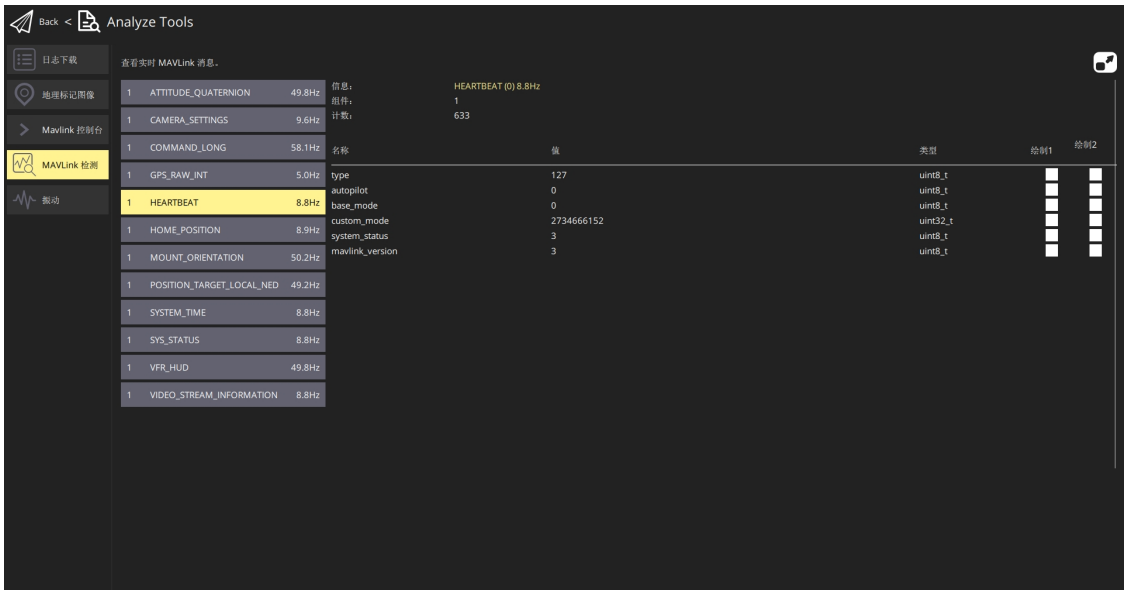
使用mavlink软件包

1. 将 psdk-ih.zip 程序包放到 /home/firefly 目录下
2. 执行 unzip psdk-ih.zip 解压程序文件
3. 执行 cd psdk-ih/build 进入程序文件
4. 执行 chmod 777 bin/mavlink_demo 因为大疆psdk需要较高权限修改网卡配置所以需要给程序一个权限
5. 修改文件夹中的mavlink_ip将其改PC端ip
6. 执行测试程序 sudo bin/mavlink_demo
7. 打开PC端QGC软件
8. 先点击左上角logo图标

9. 再点击Analyze Tools



10. 查看MAVLink检测页面



11. 查看MAVLink数据

12. 可以在这个页面输入测试命令控制psdk

输入命令后需要带空格作为结束符号

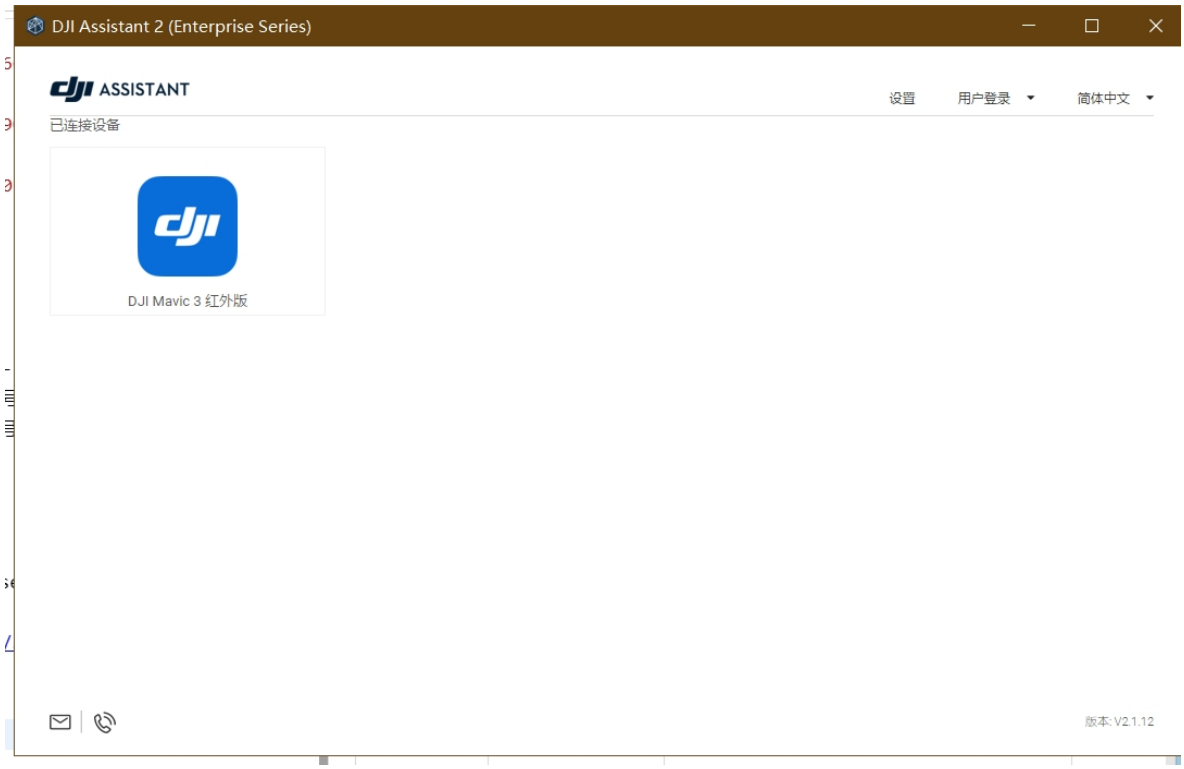
命令	参数说明	备注
zoomVideo		主视频流 开启后QGC可查看端口号为5600
IRVideo		红外视频流 开启后QGC可查看端口号为5600
gimbal 10	10 代表转动角度	云台转动

命令	参数说明	备注
zoom 5	5 代表缩放倍数	变焦倍数
mission		执行航线任务

注:主视频流需要遥控器调整到录像模式并且分辨率调到1920*1080
红外视频流需要遥控器关闭超分辨率选项

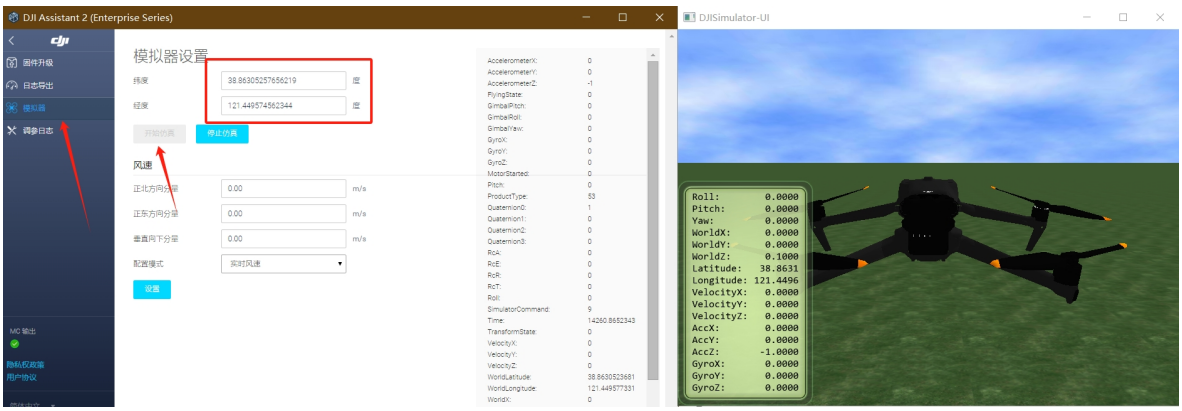
运行PSDK可使用大疆DJI Assistant 2 (Enterprise Series)运行模拟器看飞行
DJI Assistant 2 下载地址https://www.dji.com/cn/downloads/softwares/assistant-dji-2-for-matrice

下载后使用typec线连接pc与无人机
开启软件后



点击模拟器
如果需要测试航线任务需要修改经纬度为
纬度:38.86305257656219
经度:121.449574562344

然后点击开始仿真



发送测试命令

python测试例程请参考 https://www.ardusub.com/developers/pymavlink.html

mavlink message定义请查看 https://mavlink.io/zh/messages/common.html

c测试例程请参考 https://mavlink.io/en/mavgen_c/

接受数据

心跳以及电池状态

mavlink id 0 HEARTBEAT

返回字段名称	类型	单位	描
system_status	uint8_t	MAV_STATE 见下表	无

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

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	

电池信息

mavlink id 147 BATTERY_STATUS

返回字段名称	类型	单位	描
current_battery	int16_t	mA	当
battery_remaining	int8_t	%	剩

GPS位置信息

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	精
lon	int32_t	degE7	纬
alt	int32_t	mm	海
satellites_visible	uint16_t		卫

GPS时间

mavlink id 2 SYSTEM_TIME

返回字段名称	类型	单位	描
time_unix_usec	uint64_t		时

视频流信息

mavlink id 269 VIDEO_STREAM_INFORMATION

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

无人机状态

mavlink id 76 COMMAND_LONG command = MAV_CMD_NAV_LAST

Param (Label)	Description	Values
1	飞行状态	0 飞机在地面引擎停止 1 飞机在地面引擎已开启 2 飞机在空中
2	飞行模式	0 手动控制模式 1 姿态模式，飞机可以进行姿态和 并仅使用气压计进行定位以控制高 6 正常GPS模式，飞机可以自主定 9 热点模式 10 辅助起飞模式，用户可以推动 11 自动起飞模式，飞机将自主启 上升并最终悬停 12 自动降落模式，飞机可以自主 15 导航返航模式， 飞机可以自主返回最后记录的返航 17 SDK控制模式，飞机由SDK A 33 强制自动降落模式，可能是因 40 搜索模式， 无人机将搜索到最后一个遥控器未 41 每次用户解锁电机时，这将是
3	航线执行状态	0 航点v3任务处于空闲状态 16 航点v3任务处于准备状态 32 航点v3任务处于任务过渡状态 48 航点v3任务处于执行任务状态 64 航点v3任务处于中断状态 80 航点v3任务处于恢复状态 98 航点v3任务处于返回首个航点

避障距离

mavlink id 76 COMMAND_LONG command = MAV_CMD_CONDITION_DISTANCE;

Param (Label)	Description	Values	Ur
1	前向		m
2	后向		m
3	左向		m
4	右向		m
5	上方		m
6	下方		m

返航点信息

mavlink id 242 HOME_POSITION

返回字段名称	类型	单位	描
latitude	int32_t	degE7	La
longitude	int32_t	degE7	Lo
altitude	int32_t	m	返

无人机NED坐标 速度 加速度

mavlink id 85 POSITION_TARGET_LOCAL_NED

返回字段名称	类型	单位	描
time_boot_ms	uint32_t	ms	时
x	float	m	X坐
y	float	m	Y坐
z	float	m	对
vx	float	m/s	X速
vy	float	m/s	Y速
vz	float	m/s	Z速
afx	float	m/s/s	X加
afy	float	m/s/s	Y加
afz	float	m/s/s	Z加

变焦倍数

mavlink id 260 CAMERA_SETTINGS

返回字段名称	类型	单位	描
zoomLevel	float		变

云台角度

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

姿态四元数

航空坐标系中的姿态

mavlink id 31 ATTITUDE_QUATERNION

返回字段名称	类型	单位	描
time_boot_ms	uint32_t	ms	时
q1	float		姿
q2	float		姿
q3	float		姿
q4	float		姿
roll	float	rad	翻
pitch	float	rad	俯
yaw	float	rad	偏

发送命令

视频源切换并设置参数

MAV_CMD_DO_CONTROL_VIDEO 200

Param (:Label)	功能	Description	Values
param1	type	设置视频类型	M3T 1主视频 3 IR M30T 1wide 2
param2	width	视频输出宽度	
param3	height	视频输出高度	
param4	framerate	帧率	
param5	bitrate	码率	
param6	status	视频开关	0开始 1停止

返回command = MAV_CMD_REQUEST_MESSAGE
返回值 param1 200 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无法获取无人机信息)

python测试脚本

```
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 control(type, width, height, framerate, bitrate, startOrStop=
0):
    master.mav.command_long_send(
        master.target_system,
```

```
        master.target_component,
        mavutil.mavlink.MAV_CMD_DO_CONTROL_VIDEO,
        0,
        type, width, height, framerate, bitrate, startOrStop, 0
    )

def stop_streaming():
    master.mav.video_stream_status_send(
        1,
        1,
        0,0,0,0,0,0
    )

# type  1 VIS  3 IR
# width  1920
# height  1080
# framerate  30
# bitrate  4096
# start or stop  0 start  1 stop
control(1, 1920, 1080, 30, 4096)
time.sleep(10)
stop_streaming()
time.sleep(5)
control(3,640,480,20,4096)
time.sleep(30)
control(1, 1920, 1080, 30, 4096)
```

设置返航点以及返航高度

MAV_CMD_DO_SET_HOME 179

Param (:Label)	Description	Values
1: height	返航高度	范围20-200

如果height为0 为设置返航点模式
*

返回command = MAV_CMD_REQUEST_MESSAGE
返回值 param1 179 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 无法初始化无人机飞行控制 2 返航点设置失败 3 返航高度设置失败)

python测试脚本

```
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 set_home(height):
    master.mav.command_long_send(
        master.target_system,
        master.target_component,
        mavutil.mavlink.MAV_CMD_DO_SET_HOME,
        0,
        height, 0, 0, 0, 0, 0, 0
    )

set_home(70)
while True:
```

```
    msg = master.recv_match()
    if not msg:
        continue
    if msg.get_type() == 'COMMAND_LONG':
        print("\n\n*****Got message: %s*****" % msg.get_type())
        print("Message: %s" % msg)
        print("\nAs dictionary: %s" % msg.to_dict())
```

*如果height大于0 为设置返航高度模式

设置相机焦点

MAV_CMD_SET_CAMERA_FOCUS 532

Param (:Label)	Description	Values
1: FocusX	聚焦范围X	0 到 1
2: FocusY	聚焦范围Y	0 到 1

python测试脚本

```
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 focus():
    master.mav.command_long_send(
        master.target_system,
        master.target_component,
        mavutil.mavlink.MAV_CMD_SET_CAMERA_FOCUS,
        0.3,
        0.4,
        0, 0, 0, 0, 0, 0
    )

focus()
```

返回command = MAV_CMD_REQUEST_MESSAGE 返回值 param1 532 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init镜头控制 2无法设置相机模式 3无法设置焦点)

设置失控行为

MAV_CMD_NAV_GUIDED_ENABLE 92

Param (:Label)	Description	Values
2: status	失控行为	0 悬停 1 降

返回command = MAV_CMD_REQUEST_MESSAGE
返回值 param1 92 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无法获取无人机信息 2无法设置失控行为)

python测试脚本

```
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 set_rc_lost(status):
    master.mav.command_long_send(
        master.target_system,
        master.target_component,
        mavutil.mavlink.MAV_CMD_NAV_GUIDED_ENABLE,
        0,
        0, status, 0, 0, 0, 0, 0
    )
#0 hover 1 land 2 go home
set_rc_lost(0)

while True:
    msg = master.recv_match()
    if not msg:
        continue
    if msg.get_type() == 'COMMAND_LONG':
```

```
        print("\n\n*****Got message: %s*****" % msg.get_type())
        print("Message: %s" % msg)
        print("\nAs dictionary: %s" % msg.to_dict())
```

本接口大疆官方只适配M30T

设置避障状态

MAV_CMD_NAV_PATHPLANNING 81

Param (:Label)	Description	Values
1: status	避障状态	0 关闭 1 开

python测试脚本

```
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 set_avoid():
    master.mav.command_long_send(
        master.target_system,
        master.target_component,
        mavutil.mavlink.MAV_CMD_NAV_PATHPLANNING,
        0,
        0, 0, 0, 0, 0, 0, 0
    )
#param1 0 close 1 open
set_avoid()
while True:
    msg = master.recv_match()
    if not msg:
        continue
    if msg.get_type() == 'COMMAND_LONG':
        print("\n\n*****Got message: %s*****" % msg.get_type())
```

```
        print("Message: %s" % msg)
        print("\nAs dictionary: %s" % msg.to_dict())
```

返回command = MAV_CMD_REQUEST_MESSAGE返回值 param1 81 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 无法初始化无人机飞行控制 2 避障状态设置失败)

设置相机变焦

MAV_CMD_SET_CAMERA_ZOOM 531

Param (:Label)	Description	Values
1: Zoom Value	缩放倍数	
2: Zoom Type	缩放类型	0 聚焦变小

python测试脚本

```
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 zoom(type):
    master.mav.command_long_send(
        master.target_system,
        master.target_component,
        mavutil.mavlink.MAV_CMD_SET_CAMERA_ZOOM,
        type,
        3, 0, 0, 0, 0, 0, 0
    )

zoom(1)
time.sleep(5)
zoom(0)
```

返回command = MAV_CMD_REQUEST_MESSAGE返回值 param1 531 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 获取相机数据失败 3 获取相机版本失败 4 相机变焦执行失败)

执行任务命令

MAV_CMD_DO_SET_MISSION_CURRENT 224

任务文件名称以及路径为 ./psdk/build/temp.kmz
param1 为任务状态设置0 开始 1 停止 2 暂停 3 继续

返回command = MAV_CMD_REQUEST_MESSAGE
返回值 param1 224 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 无法初始化无人机飞行控制 2 创建无人机任务失败)

```
航线执行状态
command = MAV_CMD_NAV_LAST

param3 标记以下状态

0    航点v3任务处于空闲状态
16   航点v3任务处于准备状态
32   航点v3任务处于任务过渡状态
48   航点v3任务处于执行任务状态
64   航点v3任务处于中断状态
80   航点v3任务处于恢复状态
98   航点v3任务处于返回首个航点状态
```

python测试脚本

```
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 mission(action):
    master.mav.command_long_send(
        master.target_system,
        master.target_component,
```

```
        mavutil.mavlink.MAV_CMD_DO_SET_MISSION_CURRENT,
        0,
        action, 0, 0, 0, 0, 0
    )

#0 start 1 stop 2 pause 3 resume
mission(0)
time.sleep(10);
mission(2)
time.sleep(10);
mission(3)
time.sleep(10);
mission(1)
```

录像命令

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)

python测试脚本

```
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()
```

停止录像命令

MAV_CMD_VIDEO_STOP_CAPTURE 2501

python测试脚本

```
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()
```

拍照命令

MAV_CMD_DO_TRIGGER_CONTROL 2003

单张拍摄

python测试脚本

```
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_TRIGGER_CONTROL,
        0,
        0, 0, 0, 0, 0, 0, 0
    )

shoot()
```

起飞命令

MAV_CMD_NAV_TAKEOFF 22

返回command = MAV_CMD_REQUEST_MESSAGE
返回值 param1 22 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 无法获取无人机飞行控制权限
3 无法起飞
4 起飞失败并且螺旋桨未转
5 起飞失败并且螺旋桨在转
6 已起飞但发生飞控错误

python测试脚本

```
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()
```

降落命令与取消降落

MAV_CMD_NAV_LAND 21

param1 0 为降落
param1 1 为取消降落

返回command = MAV_CMD_REQUEST_MESSAGE
返回值 param1 21 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 开始执行降落失败
3 执行降落失败(etc.地面有障碍物)
4 停止降落失败

python测试脚本

```
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()
```

强制降落命令

MAV_CMD_DO_LAND_START 189

返回command = MAV_CMD_REQUEST_MESSAGE
返回值 param1 189 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 开始执行降落失败
3 执行降落失败

python测试脚本

```
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_DO_LAND_START,
        0,
        0, 0, 0, 0, 0, 0, 0
    )

land()
while True:
    msg = master.recv_match()
    if not msg:
        continue
    if msg.get_type() == 'COMMAND_LONG':
        print("\n\n*****Got message: %s*****" % msg.get_type())
```

```
        print("Message: %s" % msg)
        print("\nAs dictionary: %s" % msg.to_dict())
```

返航命令以及取消返航命令

MAV_CMD_NAV_RETURN_TO_LAUNCH 20

param1 0为返航 1为取消返航
返回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

python测试脚本

```
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)
while True:
    msg = master.recv_match()
    if not msg:
```

```
            continue
        if msg.get_type() == 'COMMAND_LONG':
            print("\n\n*****Got message: %s*****" % msg.get_type())
            print("Message: %s" % msg)
            print("\nAs dictionary: %s" % msg.to_dict())
```

经纬度位置控制

GLOBAL_POSITION_INT 33

lat	int32_t	degE7	经
lon	int32_t	degE7	纬
alt	int32_t	mm	高

```
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=388647525, lon=1214500745, alt=7000
    )
```

返回command = MAV_CMD_REQUEST_MESSAGE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

经纬度飞行停止

COMMAND_CANCEL 80

Field Name	Type	Values	De
command	uint16_t	MAV_CMD	Co co

```
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()
#command_cancel
def cancel():
    master.mav.command_cancel_send(
        int(1e3 * (time.time() - boot_time)),  # ms since boot
        command=33
    )

cancel()
```

返回command = MAV_CMD_REQUEST_MESSAGE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

飞行控制

根据命令设置不同飞行状态以及目标

SET_POSITION_TARGET_LOCAL_NED 84

time_boot_ms	uint32_t	ms		Timestan boot).
target_system	uint8_t			System I
target_component	uint8_t			Compone
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(NE
y	float	m		Y坐标(NE
z	float	m		Z坐标(NE
vx	float	m/s		X速度(NE
vy	float	m/s		Y速度(NE
vz	float	m/s		Z速度(NE
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

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

python测试脚本

```
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()
```

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

设置云台角度

GIMBAL_MANAGER_SET_PITCHYAW 287

Field Name	Type	Units	Values
target_system	uint8_t		
target_component	uint8_t		
flags	uint32_t		
gimbal_device_id	uint8_t		
pitch	float	deg	

返回command = MAV_CMD_REQUEST_MESSAGE
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 无法初始化云台控制
2 无法设置云台模式
3 云台到达限位
4 云台无法执行命令

python测试脚本

```
import time
# Import mavutil
from pymavlink import mavutil

# Create the connection
master = mavutil.mavlink_connection('udpin:0.0.0.0:14550')
```

```
# Wait a heartbeat before sending commands
master.wait_heartbeat()

def look_at(pitch):
    master.mav.gimbal_manager_set_pitchyaw_send(
        master.target_system,
        master.target_component,
        mavutil.mavlink.GIMBAL_MANAGER_FLAGS_RC_MIXED,
        1,
        pitch,
        0,
        0,0)

# make camera up and down
look_at(10)
time.sleep(5)
look_at(-10)
```

gimbal_device_id 为1时为固定角度旋转gimbal_device_id 为2时为速度旋转 角度以10hz频率发送gimbal_device_id 为3时为云台回中gimbal_device_id 为4时为云台绝对角度控制范围为 -90 到 45 单位deg

下视推流以及停止下视推流

MAV_CMD_VIDEO_START_STREAMING 2502

param1 0 开启下视推流
param1 1关闭下视推流

python测试脚本

```
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 pushDown():
    master.mav.command_long_send(
        master.target_system,
        master.target_component,
        mavutil.mavlink.MAV_CMD_VIDEO_START_STREAMING,
        0,
        0, 0, 0, 0, 0, 0, 0
    )

pushDown()
while True:
    msg = master.recv_match()
    if not msg:
        continue
    if msg.get_type() == 'COMMAND_LONG':
        print("\n\n*****Got message: %s*****" % msg.get_type())
        print("Message: %s" % msg)
        print("\nAs dictionary: %s" % msg.to_dict())
```

视频流获取

运行mavlink_demo

IP地址为192.168.28.221:5600
视频格式为h264视频流

视频流接收脚本示例

环境需要安装opencv和gstreamer

Ubuntu20.04安装gstreamer

```
sudo apt install libgstreamer1.0-dev libgstreamer-plugins-base1.0-dev
sudo apt-get install libgtk-3-dev
pkg-config --modversion gtk+-3.0
```

Ubuntu20.04安装opencv

```
sudo apt install libopencv-dev
```

主视频流接收并转换为opencv的cv mat

```
#include <gst/gst.h>
#include <gst/video/video.h>
#include <opencv2/opencv.hpp>

static GstFlowReturn on_new_sample_from_sink(GstElement *sink, gpointer data) {
    GstSample *sample = nullptr;
    GstBuffer *buffer = nullptr;
    GstMapInfo map;
    cv::Mat frame;

    g_signal_emit_by_name(sink, "pull-sample", &sample);
    if (sample) {
        buffer = gst_sample_get_buffer(sample);
        gst_buffer_map(buffer, &map, GST_MAP_READ);

        GstCaps *caps = gst_sample_get_caps(sample);
        if (caps) {
            GstVideoInfo info;
            if (gst_video_info_from_caps(&info, caps)) {
                gint width = info.width;
                gint height = info.height;
```

```
                gint stride = GST_VIDEO_INFO_PLANE_STRIDE(&info,
0);

                // 创建Y分量
                cv::Mat yPlane(height, width, CV_8UC1, map.data, st
ride);

                // 获取U和V分量的起始地址
                gpointer uPlane = (guchar*)map.data + stride * heig
ht;
                gpointer vPlane = uPlane + (stride * height / 4);

                // 创建U和V分量
                cv::Mat uPlaneMat(height / 2, width / 2, CV_8UC1,
(void*)uPlane, stride / 2);
                cv::Mat vPlaneMat(height / 2, width / 2, CV_8UC1,
(void*)vPlane, stride / 2);

                // 将YUV420P转换为BGR
                cv::Mat bgrFrame(height, width, CV_8UC3);
                cv::cvtColor(cv::Mat(height * 3 / 2, width, CV_8UC
1, map.data, stride), bgrFrame, cv::COLOR_YUV2BGR_I420);

                // 现在bgrFrame包含了转换后的BGR图像
                // 你可以保存或显示bgrFrame
                // 例如保存为图片
                static int frame_count = 0;
                std::string filename = "frame_" + std::to_string(fr
ame_count) + ".png";
                cv::imwrite(filename, bgrFrame);
                frame_count++; // 增加帧计数，以便为下一帧生成不同的文件名
            }
        }

        gst_buffer_unmap(buffer, &map);
        gst_sample_unref(sample);
    }

    return GST_FLOW_OK;
}

static gboolean bus_call(GstBus *bus, GstMessage *msg, gpointer dat
a)
{
    GMainLoop *loop = (GMainLoop *)data;
```

```
    switch (GST_MESSAGE_TYPE(msg))
    {
    case GST_MESSAGE_EOS:
        g_print("End of stream\n");
        g_main_loop_quit(loop);
        break;
    case GST_MESSAGE_ERROR:
    {
        gchar *debug;
        GError *error;

        gst_message_parse_error(msg, &error, &debug);
        g_printerr("Error: %s\n", error->message);
        g_error_free(error);
        g_free(debug);

        g_main_loop_quit(loop);
        break;
    }
    default:
        break;
    }

    return TRUE;
}

int main(int argc, char *argv[])
{
    gst_init(&argc, &argv);

    GstElement *pipeline = gst_parse_launch("udpsrc uri=udp://192.168.17.51:5600 ! application/x-rtp, media=video, clock-rate=90000, encoding-name=H264 ! rtph264depay ! avdec_h264 ! videoconvert ! videoscale ! appsink name=sink emit-signals=true", NULL);
    GstElement *sink = gst_bin_get_by_name(GST_BIN(pipeline), "sink");

    g_signal_connect(sink, "new-sample", G_CALLBACK(on_new_sample_from_sink), NULL);

    GstBus *bus = gst_element_get_bus(pipeline);
    GMainLoop *loop = g_main_loop_new(NULL, FALSE);

    gst_bus_add_watch(bus, bus_call, loop);
```

```
    gst_element_set_state(pipeline, GST_STATE_PLAYING);
    g_main_loop_run(loop);

    gst_element_set_state(pipeline, GST_STATE_NULL);
    gst_object_unref(bus);
    gst_object_unref(pipeline);
    g_main_loop_unref(loop);

    return 0;
}
```

运行命令g++ -o video_stream video_stream.cpp `pkg-config --cflags --libs gstreamer-1.0` `pkg-config --cflags --libs gstreamer-video-1.0` `pkg-config --cflags --libs opencv4`

下视视频流获取并转换为opencv的cv mat

```
#include <gst/gst.h>
#include <gst/video/video.h>
#include <opencv2/opencv.hpp>

static GstFlowReturn on_new_sample_from_sink(GstElement *sink, gpoi
nter data) {
    GstSample *sample = nullptr;
    GstBuffer *buffer = nullptr;
    GstMapInfo map;
    cv::Mat frame;

    g_signal_emit_by_name(sink, "pull-sample", &sample);
    if (sample) {
        buffer = gst_sample_get_buffer(sample);
        gst_buffer_map(buffer, &map, GST_MAP_READ);

        GstCaps *caps = gst_sample_get_caps(sample);
        if (caps) {
            GstVideoInfo info;
            if (gst_video_info_from_caps(&info, caps)) {
                gint width = info.width;
                gint height = info.height;
                gint stride = GST_VIDEO_INFO_PLANE_STRIDE(&info,
0);

                // 创建灰度图
                cv::Mat grayFrame(height, width, CV_8UC1, map.data,
```

```
stride);

                // 将灰度图转换为BGR
                cv::Mat bgrFrame(height, width, CV_8UC3);
                cv::cvtColor(grayFrame, bgrFrame, cv::COLOR_GRAY2BG
R);

                // 现在bgrFrame包含了转换后的BGR图像
                // 你可以保存或显示bgrFrame
                // 例如保存为图片
                static int frame_count = 0;
                std::string filename = "frame_" + std::to_string(fr
ame_count) + ".png";
                cv::imwrite(filename, bgrFrame);
                frame_count++; // 增加帧计数，以便为下一帧生成不同的文件名
            }
        }

        gst_buffer_unmap(buffer, &map);
        gst_sample_unref(sample);
    }

    return GST_FLOW_OK;
}

static gboolean bus_call(GstBus *bus, GstMessage *msg, gpointer dat
a)
{
    GMainLoop *loop = (GMainLoop *)data;

    switch (GST_MESSAGE_TYPE(msg))
    {
    case GST_MESSAGE_EOS:
        g_print("End of stream\n");
        g_main_loop_quit(loop);
        break;
    case GST_MESSAGE_ERROR:
    {
        gchar *debug;
        GError *error;

        gst_message_parse_error(msg, &error, &debug);
        g_printerr("Error: %s\n", error->message);
        g_error_free(error);
        g_free(debug);
```

```
        g_main_loop_quit(loop);
        break;
    }
    default:
        break;
    }

    return TRUE;
}

int main(int argc, char *argv[])
{
    gst_init(&argc, &argv);

    GstElement *pipeline = gst_parse_launch("udpsrc uri=udp://192.168.17.51:5700 ! application/x-rtp, media=video, clock-rate=90000, encoding-name=H264 ! rtph264depay ! avdec_h264 ! videoconvert ! videoscale ! appsink name=sink emit-signals=true", NULL);
    GstElement *sink = gst_bin_get_by_name(GST_BIN(pipeline), "sink");

    g_signal_connect(sink, "new-sample", G_CALLBACK(on_new_sample_from_sink), NULL);

    GstBus *bus = gst_element_get_bus(pipeline);
    GMainLoop *loop = g_main_loop_new(NULL, FALSE);

    gst_bus_add_watch(bus, bus_call, loop);

    gst_element_set_state(pipeline, GST_STATE_PLAYING);
    g_main_loop_run(loop);

    gst_element_set_state(pipeline, GST_STATE_NULL);
    gst_object_unref(bus);
    gst_object_unref(pipeline);
    g_main_loop_unref(loop);

    return 0;
}
```

运行命令g++ -o down_stream down_stream.cpp `pkg-config --cflags --libs gstreamer-1.0` `pkg-config --cflags --libs gstreamer-video-1.0` `pkg-config --cflags --libs opencv4`

QGC遥控

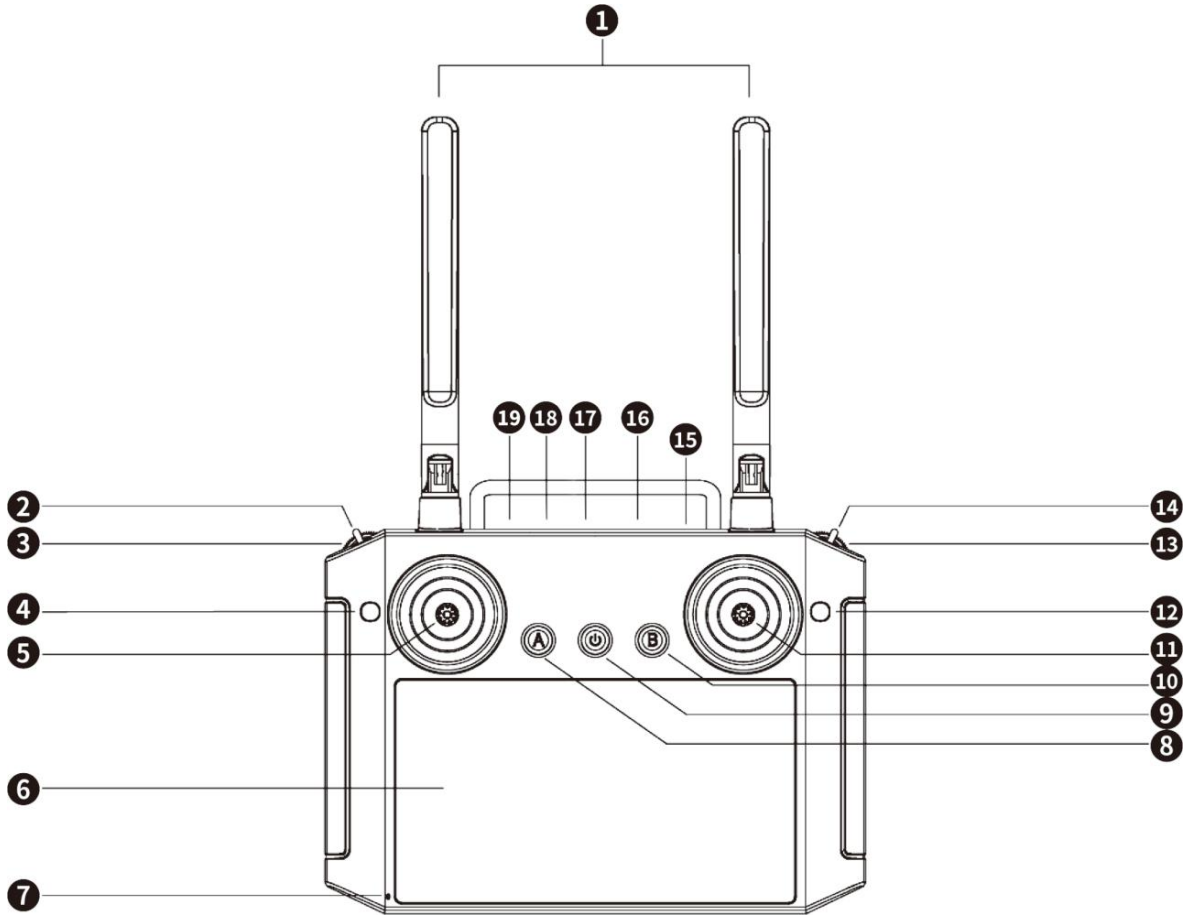
三方遥控

第三方遥控


C
A
B
D

遥控器参数			
型号	H12PRO	通道数	12
工作电压	4.2V	射频功率	17.5dBm
频段	2.400-2.483GHz	跳频	OFDM跳频
升级	APP在线升级	重量	660克
尺寸	190*152*94毫米	电池	10000mA/H
续航时间	6-20小时	充电接口	TYPE-C
调制方式	CCK,DQPSK,DBPSK	传输速率	6.5-144Mbps
发射功率	2.4G: 16.0±1.5dBm&<-15dB@11b 11Mbps;14.0±1.5dBm&<-25dB@11g 54Mbps13.0±1.5dBm&<-28dB@11n-HT20-2.4G-MCS7;13.0±1.5dB-m&<-28dB@11n-HT40-2.4G-MCS7		
接收机灵敏度	2.4G:1Mbps:-94dBm@PER<8%;11Mbps:-85dBm@PER<8% 54Mbps:-70dBm@PER<10%;HT40-MCS7-2.4G:65dBm@PER<10%		
传输距离	空旷地对空：6-10KM　空旷地面：1.2-1.5KM　室内3~4堵墙：30m		

遥控器介绍

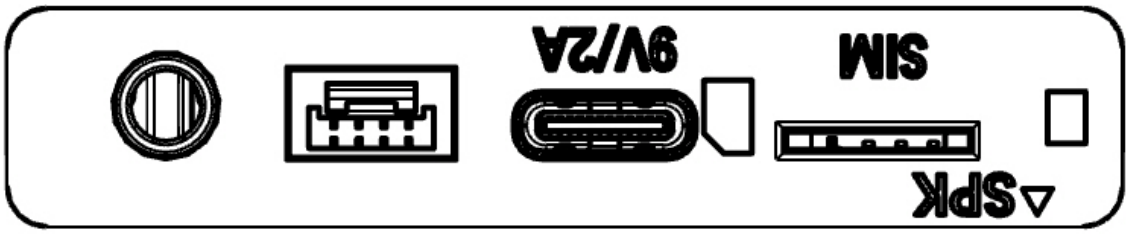


序号	注解	序号	注解
1	2.4G 4dBm天线	11	右摇杆X2、Y2
2	拨动三段开关/飞行模式	12	按键D（暂无功能）
3	波轮G（暂无功能）	13	波轮H（暂无功能）
4	按键C（暂无功能）	14	拨动三段开关（暂无功能）
5	左摇杆X1、Y1	15	防尘塞
6	显示屏	16	SIM卡接插口
7	喊话MIC	17	Type-C充电口

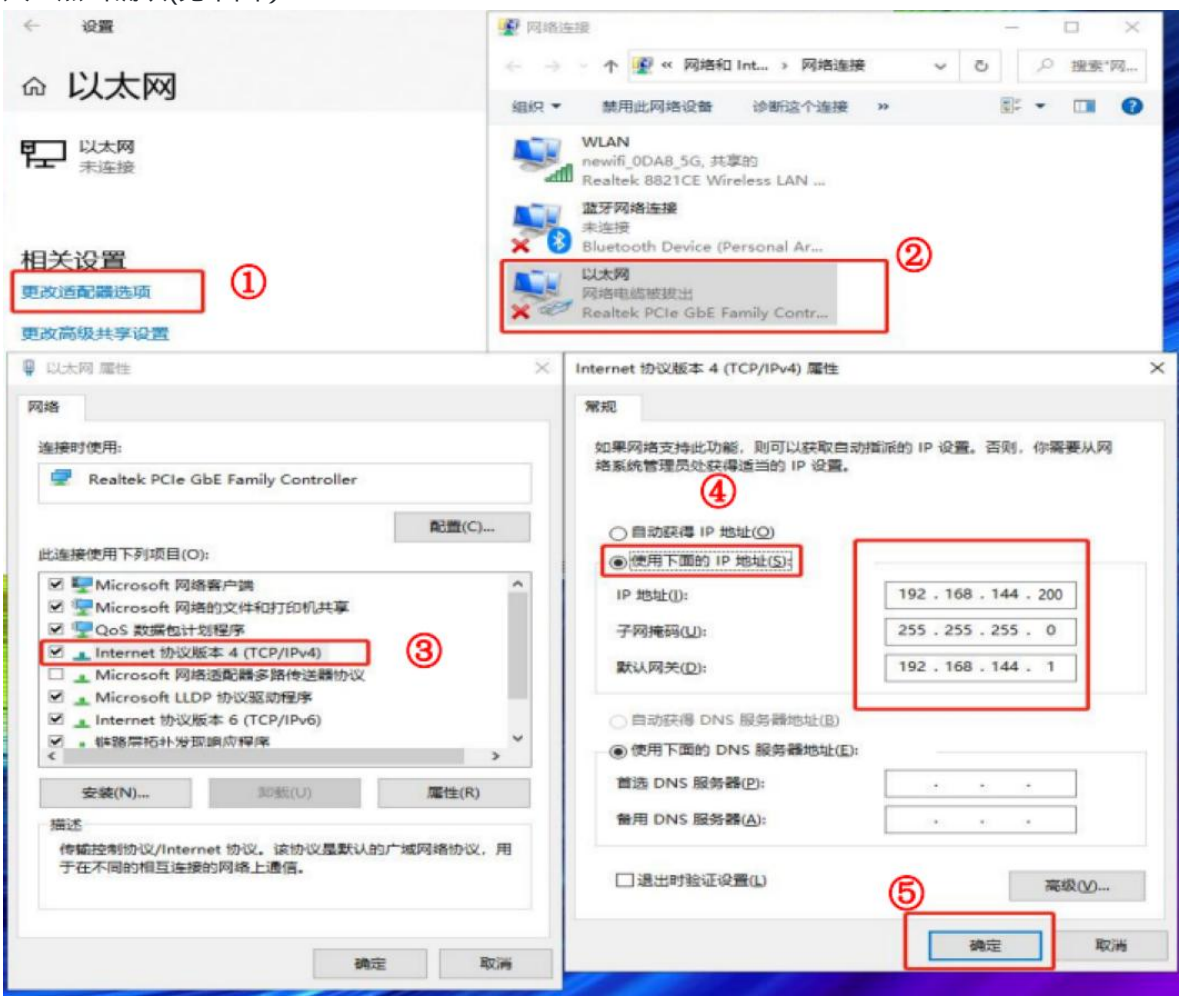
8	按键A（暂无功能）	18	4pin网口
9	开关机按键	19	暂无功能
10	按键B（暂无功能）		

遥控器网口外接

第三方地面站软件网口（如上表“18”）配置



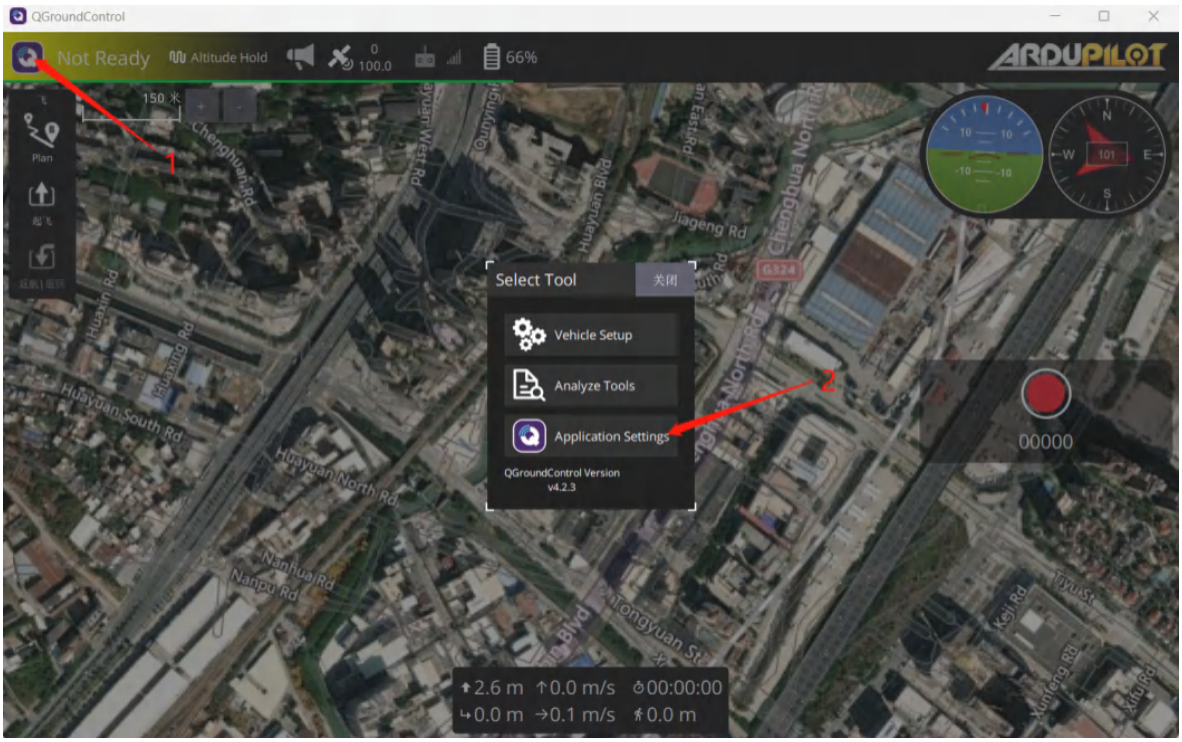
打开电脑设置->点击适配器->选择以太网->选择IPV4协议->选择自定义IP地址->按图中的方式输入->点击确认(见下图）



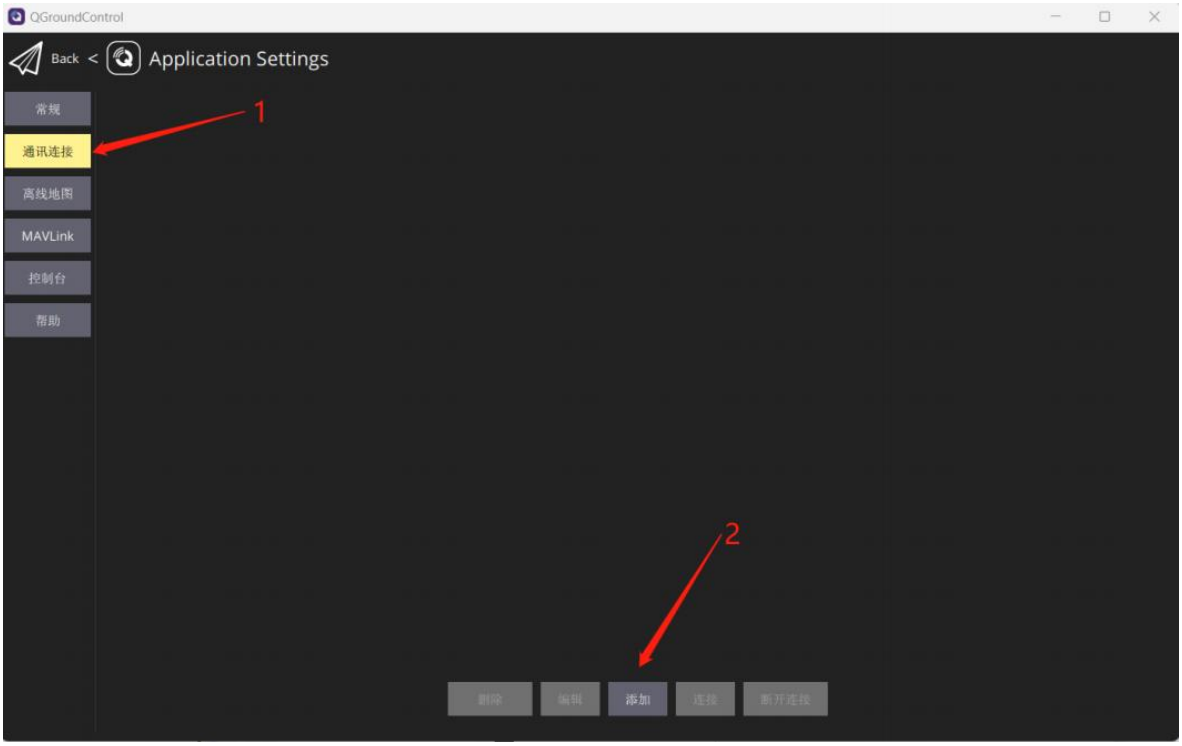
QGC地面站

QGC地面站

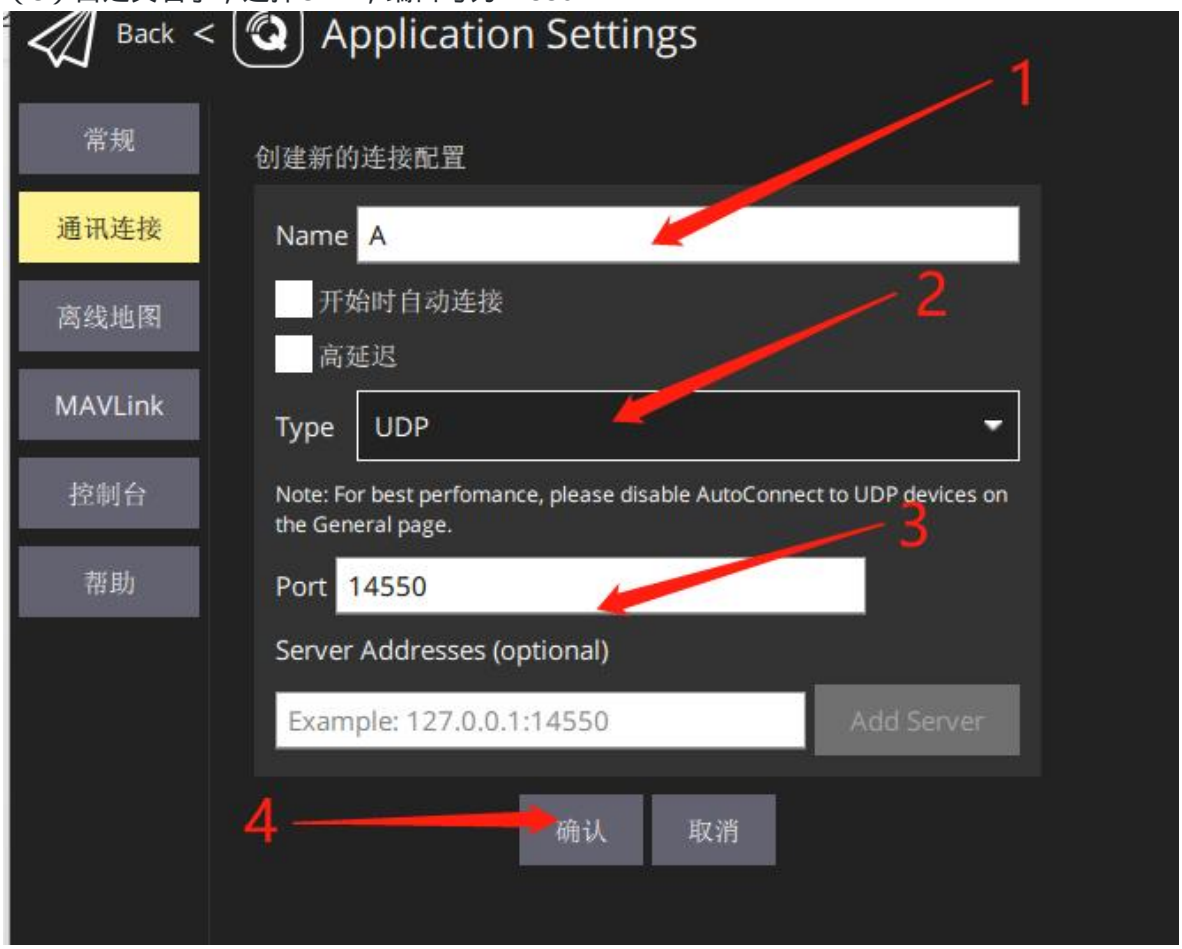
（1）打开电脑 QGC，选择Application Settings



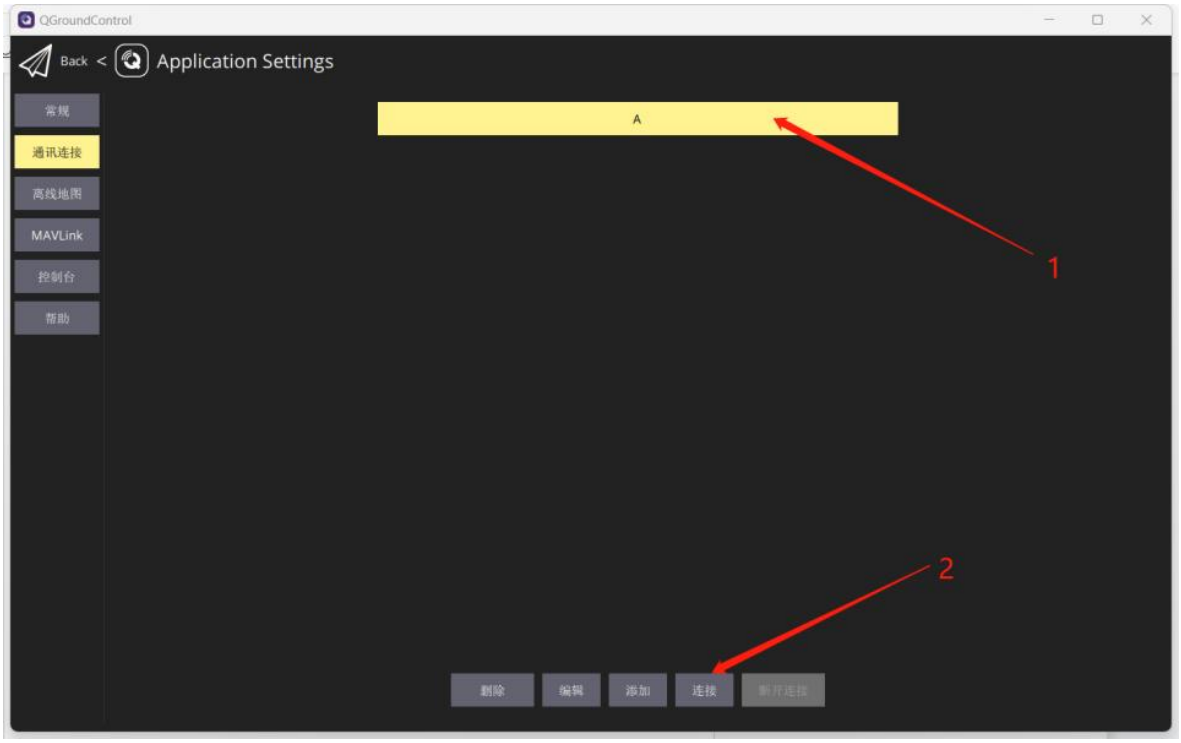
（2）选择通讯连接，添加连接方式



（3）自定义名字，选择UDP，端口号为14550

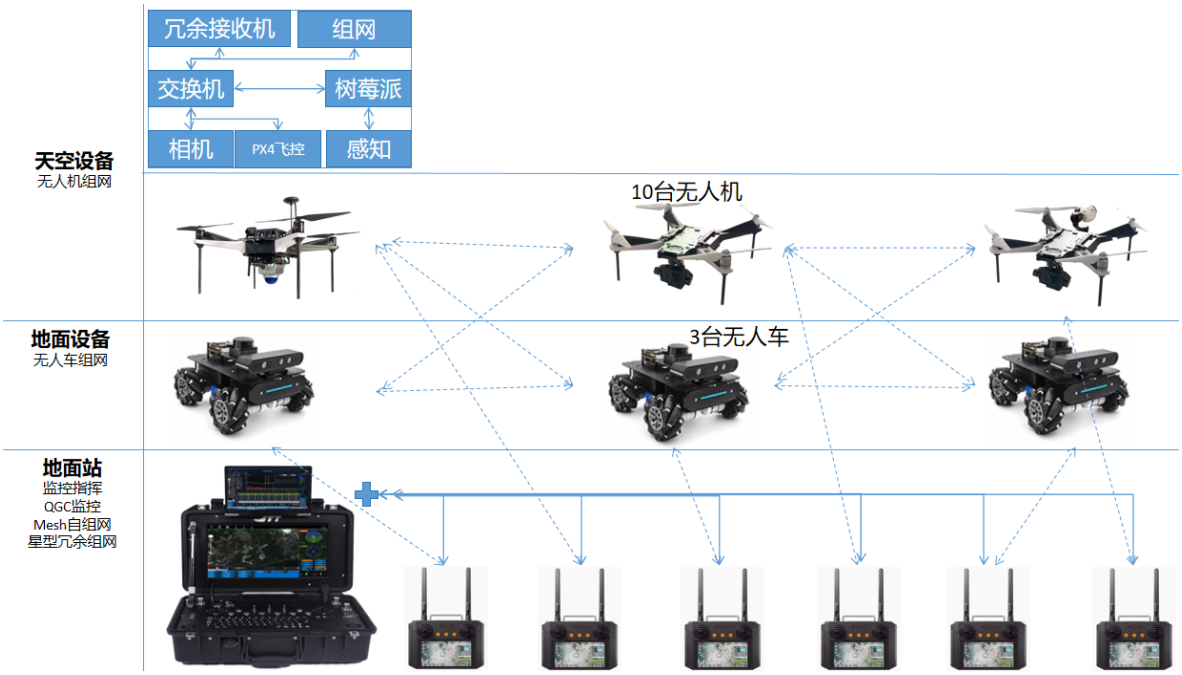


（4）选择建立的连接方式，点击连接



（5）星型组网方案

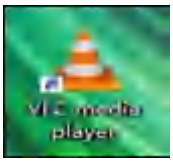
多机协同MESH及星型冗余方案——6台机器人（3台无人机，3台无人车）



共享图传

共享图传

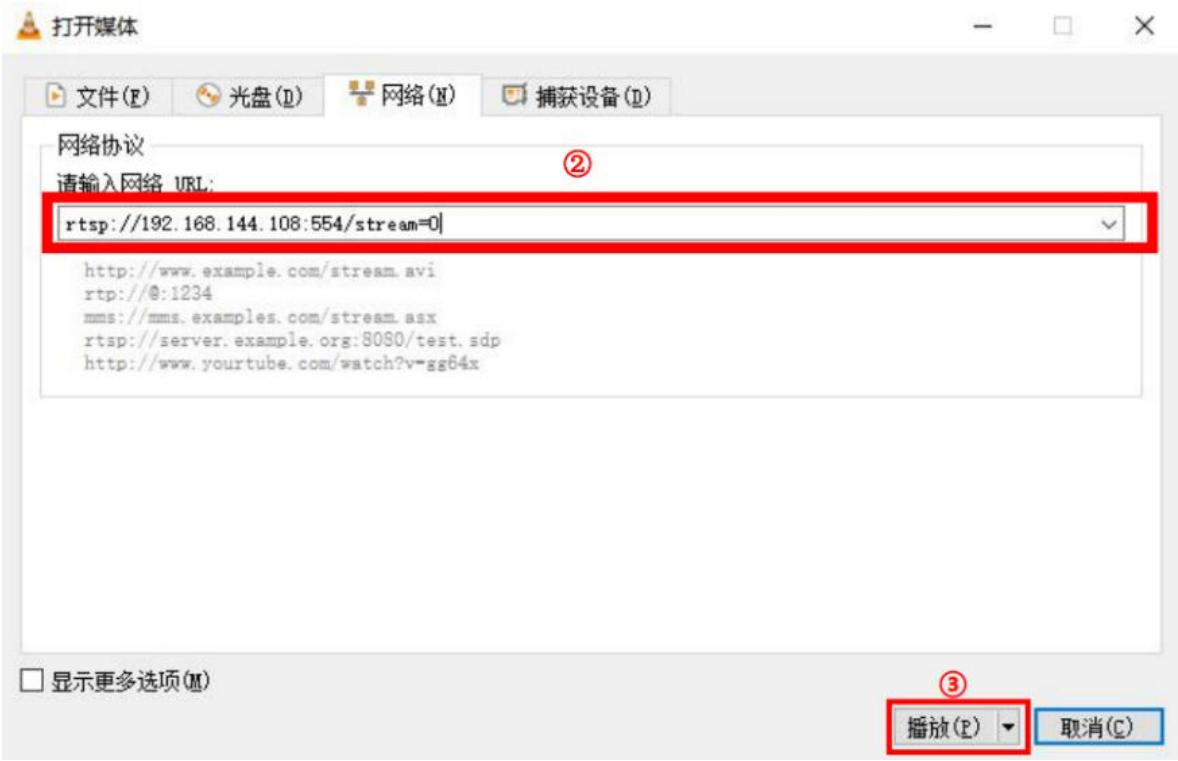
按照上述连接遥控器与电脑后，下载软件VLC来显示图像



（1）打开电脑VLC软件，在媒体中选择网络串流



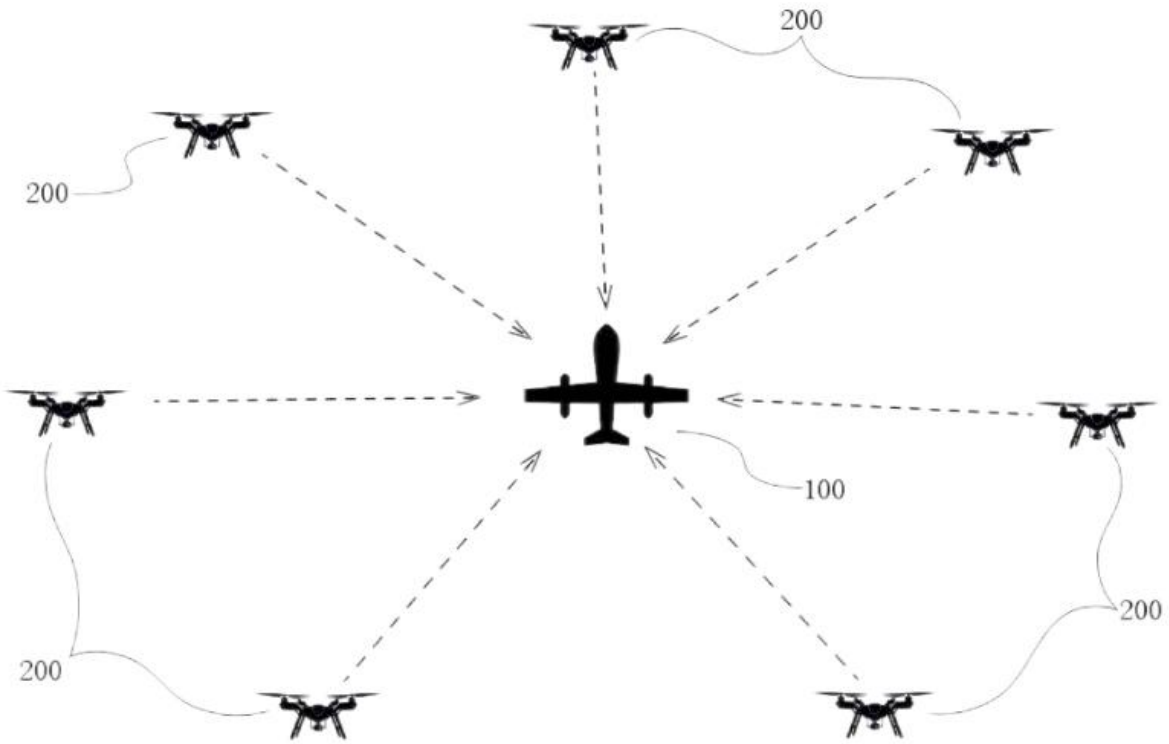
（2）在网络 URL 的输入栏中，输入 rtsp://192.168.144.108:554/stream=0 后点击播放



通信技术

星型组网（有中心组网）

星型组网示意图



星型自组网技术特点

- 传输距离通视条件下，最远17km。
- 视频接口：网口。
- 调制方式:OFDM。
- 数据接口：UART（TTL）。
- 工作频率：800MHz/1.4GHz/2.4GHz，用户可设置。
- 工作模式：自主配置为：点对点模式；中继模式（点对多点），最大节点数16个。
- 跳频/定频：定频模式（用户可固定频点）；跳频模式，自动避开干扰频段。
- 工作带宽：3/5/10/20MHz。
- 传输速率：30Mbps。
- 最大发射功率：25dBm。
- 工作温度：-20℃~+75℃

- 工作温度：-20℃～+75℃。

星型组网技术参数

项目	功能	描述
射频	频率	806-826MHz；1427.9-1447.9MHz；2401.5-2481.5MHz
	发射功率	2.4G/1.4G/800M20-25dBm±2dBm
	灵敏度	2.4G:10Mbps-102dBm；5MHz-104dBm；3MH 106dBm
		1.4G：10Mbps-103dBm；5MHz-106dBm；3MHz-108dBm
		800M:10Mbps-103dBm；5MHz-106dBm；3MHz-108dBm
接口	以太网口	RJ45*3
	UART	UART(Data)*1；UART(Config)*1
	USB	USB*1
电源	电源输入	-20℃～+75℃
整机功率	最大峰值功率	小于10W（发射功率分为1W,2W,5W）
传输模式	天线	双天线：主天线收发模式，辅助天线接收模式（1.5dbi\5dbi\7dbi；全向，方向可选）
信道带宽	800M/1.4GHz/2.4GHz	1.4MHz/3MHz/5MHz/10MHz/20MHz
性能	速率	速率可配置/支持高达30Mbps
	距离	800m-17km， 无人机实测最远传输17km/1080P图像
时延	数据传输时延	空口延迟=15ms
	开机时延	<15s上电-建链完成
命令接口	WEB配置	支持web页面配置管理

温度范围	存储温度	-40℃ ~ +85℃
	工作温度	-20℃ ~ +75℃
湿度	存储湿度	5% ~ 95%

星型组网实物-EC-OPi-ALpro系列-M3



星型组网实物-EC-OPi-ALpro-M30

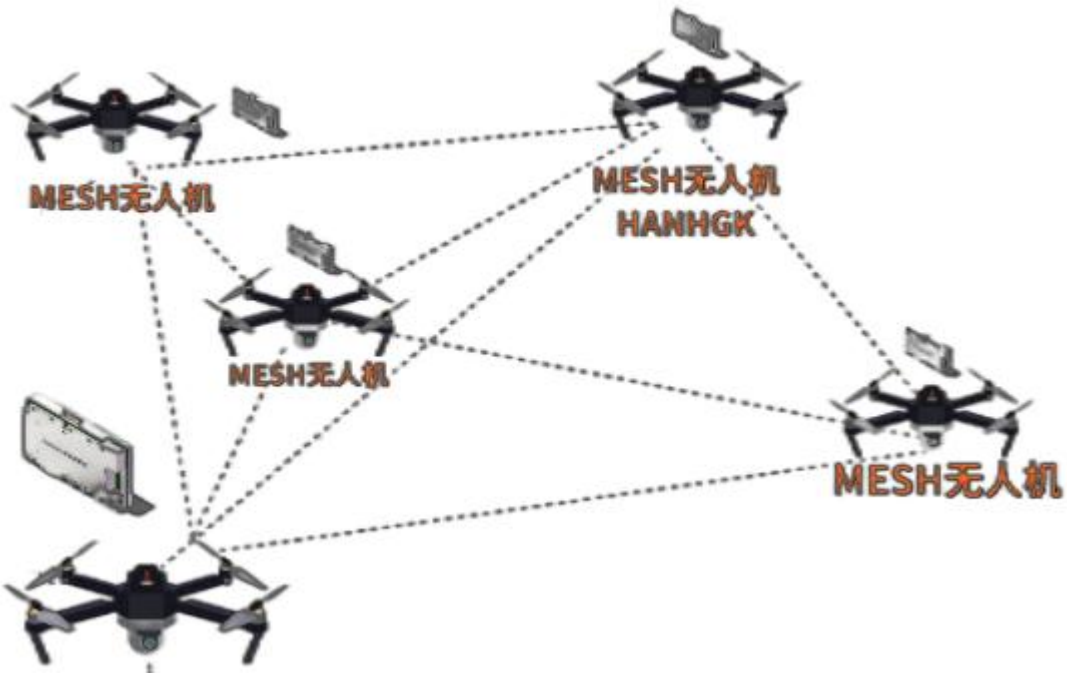


星型组网实物-EC-OPi-ALpro-M300/M350



MESH组网（无中心组网）

MESH组网示意图



MESH自组网技术特点

- 无需架设中心站及复杂系统配置现场开机上电后自动组网，“秒级”即可展开通联；
- 任意拓扑、多跳接力、中继转发；
- 可灵活配置各节点的逻辑属性；
- 目前同频组网可支持≥32个节点互连；
- 信道采用AES加密；
- 系统数据带宽峰值为90Mbps；
- 抗干扰能力强、频谱效率高、传输距离远、抗衰落能力、绕射能力强。

MESH组网技术参数

功能	描述
通信频率	1407~1497MHz，1MHz步进可调
网络规模	单频支持40个节点
发射功率	2*30dBm（2*1W），1dBm步进可调
带宽模式	5/10/20MHz
调制方式	多载波TDD-COFDM

功能	描述
载波调制	BPSK/QPSK/16QAM/64QAM（自适应或固定式）
接收灵敏度	-98dBm@10MHz
通信距离	30km（地空/空空视距），8km（地地视距）
通信速率	峰值90Mbps（自适应）
传输时延	单跳约2ms
多跳能力	视频可达4跳
启动时间	≤25S
入网时间	小于1秒
路由切换	小于1秒
数据接口	网口x1，RS232x2（选配）
整机功耗	6~12W，待机6W，接收8W，满负载发送12W
工作温度	-40~+80℃

应用案例

自组网编队

4台无人机自组网编队，使用QGC做无人机实时动态显示。

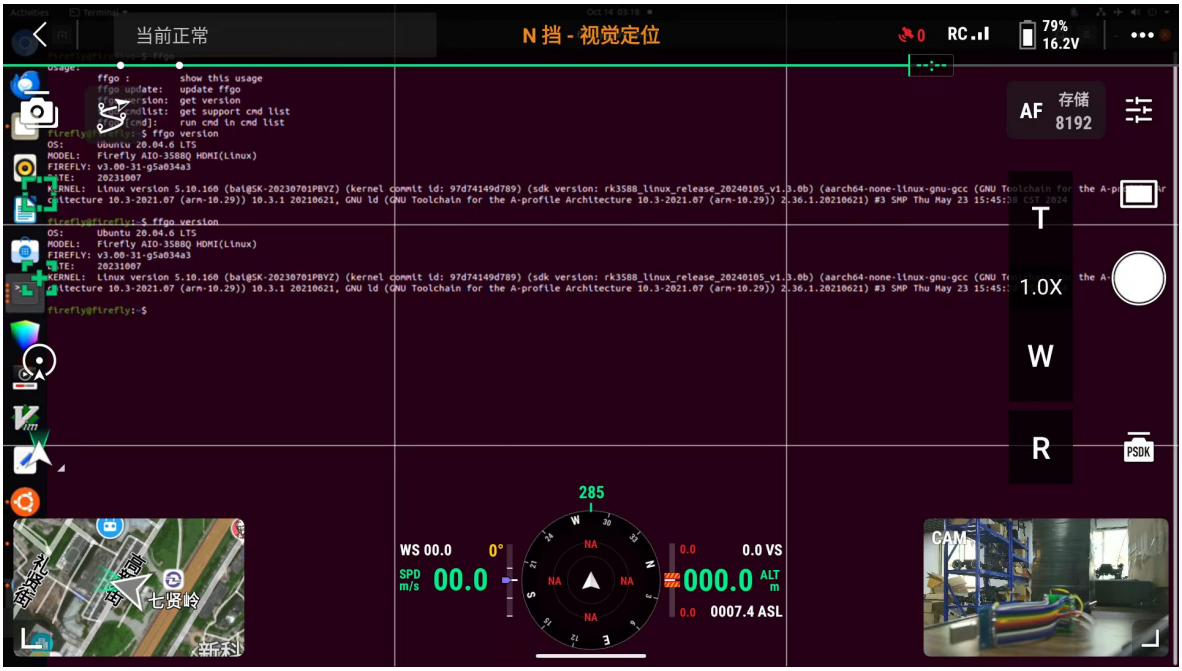








Ubuntu桌面推流到遥控器


当前正常
N 挡 - 视觉定位
RC
79%
16.2V
AF
存储
8192
T
1.0X
W
R
PSDK
285
WS 00.0
0°
SPD
m/s
00.0
0.0
0.0 VS
000.0
ALT
m
0.0
0007.4 ASL
CAM





参考文献

- [1]大疆PSDK环境文档,https://developer.dji.com/doc/payload-sdk-tutorial/cn/quick-start/config-develop-environment.html