**Set up a Headless Raspberry Pi**

The drone control computer no needs GUI, we will set up a **headless** Raspberry Pi. It can save the disk space and also the CPU load.

**Download the Raspberry Pi OS Lite**

If your RPi didn’t install the OS or you want to reinstall it, you can download the [system image](https://www.raspberrypi.org/downloads/raspberry-pi-os/) here.

**Writing image to SD card**

You can download and use the [etcher](https://www.balena.io/etcher/) to write the image to SD card.

**Writing image to SD card (using Mac Terminal)**

**If you use**[**etcher**](https://www.balena.io/etcher/)**, you can skip this step.** If you are using Mac and want to use Terminal to write the image, it is for you.

1. insert the SD card to Mac (suggested minimum 8GB disk)
2. look up the disk id:

diskutil list

1. the screen-print will be something like below. The SD card id is **disk2**. (**Your disk id may be difference, please find it carefully**)

/dev/disk2 (external, physical):

#: TYPE NAME SIZE IDENTIFIER

0: FDisk\_partition\_scheme \*31.9 GB disk2

1: Windows\_FAT\_32 boot 268.4 MB disk2s1

2: Linux 7.8 GB disk2s2

3: Linux 23.9 GB disk2s3

1. replace the "diskx" to your disk id then unmount the disk

diskutil unmountDisk /dev/diskx

1. format the disk

sudo newfs\_msdos -F 16 /dev/diskx

1. replace the "/path/of/image/xxx.dmg" to the image file path then write the image to disk

sudo dd if=/path/of/image/xxx.dmg of=/dev/diskx bs=1m

**Login default username and password**

You can use the default username and password log in to the RPi.

Default user: pi

Default password: raspberry

**Expanding disk space**

Even if you are using a larger card, but the system still using 4GB disk space. This is because writing the disk image to your microSD card creates a partition. The result is that the rest of the disk is unusable unless you expand the file system.

You can run the following command to expand it:

sudo raspi-config --expand-rootfs

**Config the WiFi**

[Setting up a wireless LAN via the command line](https://www.raspberrypi.org/documentation/configuration/wireless/wireless-cli.md)

You can use ifconfig to check the wifi status. If it prints the wwan0 information, means the wifi connection fine.

wwan0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST> mtu 1500

inet 169.254.xxx.xxx netmask 255.255.0.0 broadcast 169.254.255.255

You can try to ping a domain to test the connection to Internet.

ping google.com

64 bytes from 172.217.163.238: icmp\_seq=1 ttl=58 time=13.067 ms

***Note: In case you need to modify or check the WiFi config file directly, find here:***

/etc/wpa\_supplicant/wpa\_supplicant.conf

**Security Setup**

The document below provides detailed information to protect your RPi. You can choose to execute items that suitable for you, but at least change the default username and password.

[Securing your Raspberry Pi](https://www.raspberrypi.org/documentation/configuration/security.md)

**Login SSH**

If the everything fine, you can login to your RPi through SSH. You can find the RPi IP using ifconfig in wwan0

ssh my-username@my-rpi-ip

**Enabling Serial Console**

The serial console lets Raspberry Pi communicates with the px4. You can check [here](https://learn.adafruit.com/adafruits-raspberry-pi-lesson-5-using-a-console-cable/enabling-serial-console) for how to enable it.

**Installing 4G USB modem to Raspberry Pi**

**HUAWEI E3372 modem**

If you are using HUAWEI E3372, it can plug & play on Raspberry Pi, no need to install any program.

**HUAWEI E3370 modem**

I have an old HUAWEI E3370 modem that needs config manually. The method also can help me find the problem when the modem didn’t work.

[](https://i0.wp.com/bellergy.com/wp-content/uploads/2020/10/IMG_6905-scaled.jpg?ssl=1)

**Did the USB modem loaded?**

Run the command below in you RPi, to check did the USB modem loaded.

$ lsusb | grep Huawei

12d1:1506 Huawei Technologies Co., Ltd. Modem/Networkcard

**Did the E3370 loaded?**

The 12d1 is Huawei and the 1506 is E3370 modem.

$ dmesg | grep 1506

idVendor=12d1, idProduct=1506, bcdDevice= 1.02

**Install wvdial**

If the modem has been loaded but it didn’t dial up. You can install the wvdial.

$ sudo apt-get install wvdial

The default config of wvdial is fine for me. If you need to edit it, the config file is here:

$ sudo pico /etc/wvdial.conf

It is the default config file:

[Dialer Defaults]

Init1 = ATZ

Init2 = ATQ0 V1 E1 S0=0

Modem Type = Analog Modem

Baud = 9600

New PPPD = yes

Modem = /dev/ttyUSB0

ISDN = 0

Phone = \*99#

Password = internet

Username = internet

Stupid Mode = on

**Testing the dial up:**

$ sudo wvdial

It is the success dial up screen-print:

--> WvDial: Internet dialer version 1.61

--> Initializing modem.

--> Sending: ATZ

OK

--> Sending: ATQ0 V1 E1 S0=0

OK

--> Modem initialized.

--> Sending: ATDT\*99#

--> Waiting for carrier.

ATDT\*99#

CONNECT 150000000

--> Carrier detected. Starting PPP immediately.

--> Starting pppd at Sun Oct 18 14:45:08 2020

--> Pid of pppd: 1329

--> Using interface ppp0

--> pppd: X/?[01]X/?[01]

--> local IP address xxx.xxx.xxx.xxx

--> pppd: X/?[01]X/?[01]

--> remote IP address xxx.xxx.xxx.xxx

--> pppd: X/?[01]X/?[01]

--> primary DNS address xxx.xxx.xxx.xxx

--> pppd: X/?[01]X/?[01]

--> secondary DNS address xxx.xxx.xxx.xxx

--> pppd: X/?[01]X/?[01]

**Create the service**

I need the USB modem can auto dial-up, I created a wvdial.service file.

$ sudo pico /etc/systemd/system/wvdial.service

[Unit]

Description=wvdial

[Service]

ExecStart=/usr/bin/wvdial

Restart=on-failure

RestartSec=5

Add the code to the 99-com.rules file. It will call the wvdial.service when the USB modem loaded.

$ sudo pico /etc/udev/rules.d/99-com.rules

SUBSYSTEM=="tty", KERNEL=="ttyUSB0", TAG+="systemd", ENV{SYSTEMD\_WANTS}+="wvdial.service"

**Testing the connection**

Run the command:

ifconfig

If the modem runs correctly, you can find the ppp0 output:

ppp0: flags=4305<UP,POINTOPOINT,RUNNING,NOARP,MULTICAST> mtu 1500

inet xxx.xxx.xxx.xxx netmask 255.255.255.255 destination 10.64.64.64

ppp txqueuelen 3 (Point-to-Point Protocol)

RX packets 11065 bytes 1348146 (1.2 MiB)

RX errors 0 dropped 0 overruns 0 frame 0

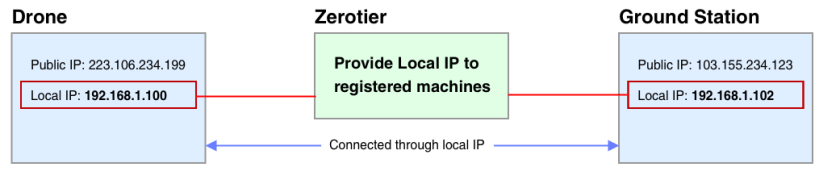
TX packets 12168 bytes 1585065 (1.5 MiB)

TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0

**Install Zerotier**

**Why ZeroTier?**

ZeroTier provides a convenient way for us to easily connect the drone and the ground station through the 4G network without having to build a VPN by ourselves.

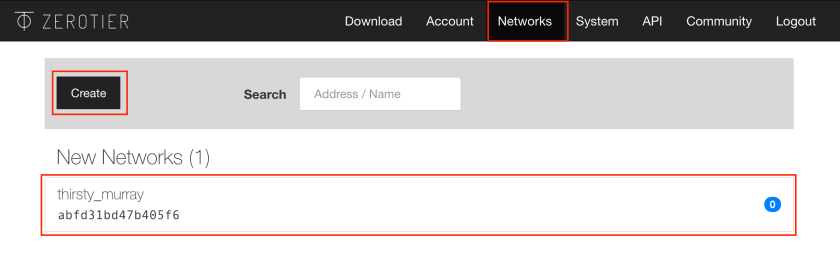
[](https://i0.wp.com/bellergy.com/wp-content/uploads/2020/10/zerotier-diagram.png?ssl=1)

**The setting process**

1. Register and set up a ZeroTier account
2. Install zerotier-one client on Raspberry Pi
3. Install the ZeroTier One client on the ground station PC
4. Approve the connection of Raspberry Pi and ground station in ZeroTier account

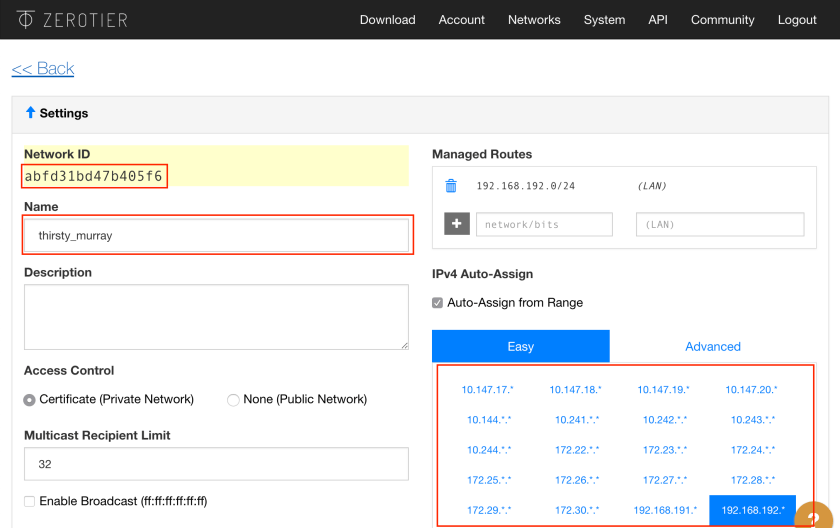
**1. Register and set up a ZeroTier account**

* Register a new account on [zerotier.com](https://www.zerotier.com/)
* Login and create a new Network

[](https://i0.wp.com/bellergy.com/wp-content/uploads/2019/02/Screenshot-2019-02-25-at-6.27.57-PM-1.png?ssl=1)

In the Network setting page:

* **Write down the Network ID**, you will need it to config the zerotier later
* Change the Network Name you prefer
* Choose a IP range you prefer

[](https://i0.wp.com/bellergy.com/wp-content/uploads/2019/02/Screenshot-2019-02-25-at-6.30.05-PM.png?ssl=1)

**2. Install zerotier-one client on Raspberry Pi**

Through Zerotier, we can create a VLAN that put the Raspberry Pi drone and Ground Station in the same local network. Let us connect the Raspberry Pi to the ground station more easily.

Execute the following shell command in Raspberry Pi to download the Zerotier:

curl -s https://install.zerotier.com | sudo bash

*for more information of installation, please*[*read here*](https://www.zerotier.com/download/)

**Testing the installation**

sudo zerotier-cli status

If you can see 200 info [ID] [version] ONLINE then successed

**Make the Zerotier auto-start on system boot**

sudo systemctl enable zerotier-one

**Join the zerotier to your network**

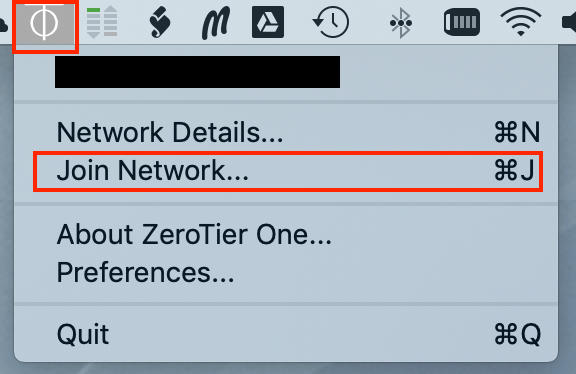
sudo zerotier-cli join [Network ID]

*You can find the Network ID in your Zerotier account*

**3. Install the Zerotier One client on the ground station PC**

[Download](https://www.zerotier.com/download/) and install the Zerotier to your PC

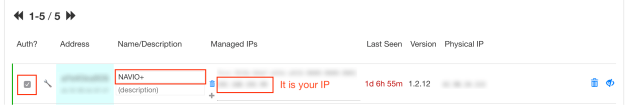
Then select Join Network and input the Network ID

[](https://i0.wp.com/bellergy.com/wp-content/uploads/2019/02/Screenshot-2019-02-27-at-1.26.24-PM-1.png?ssl=1)

**4. Approve the connection of Raspberry Pi and ground station in your Zerotier account**

Go to your ZeroTier network setting page, you will find the records of Raspberry Pi and the Ground Station PC.

* Checked the Auth? checkbox
* Change the Name you prefer
* Write down the Managed IPs of Raspberry Pi and ground station

Approve the machine in zerotier

**Wiring and connecting Pixhawk to Raspberry Pi**

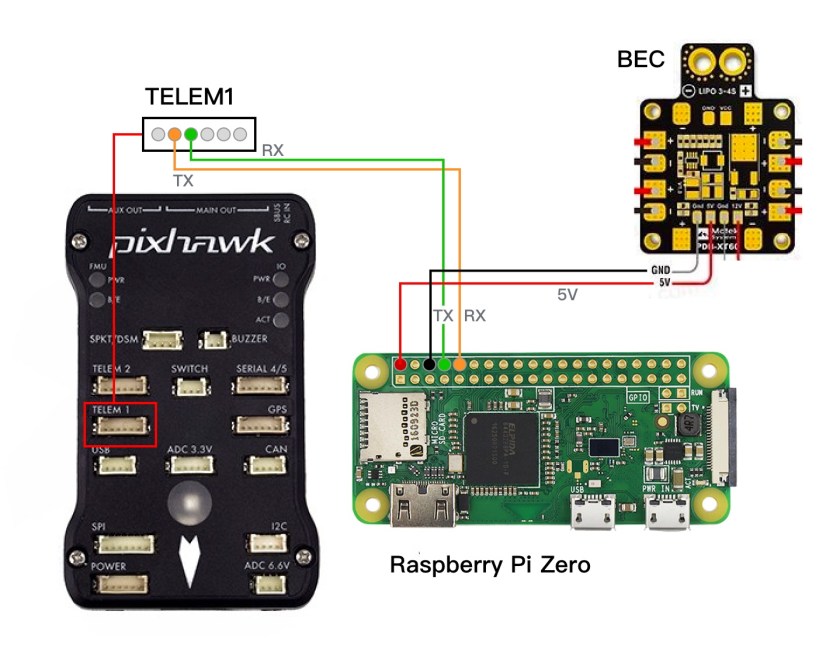
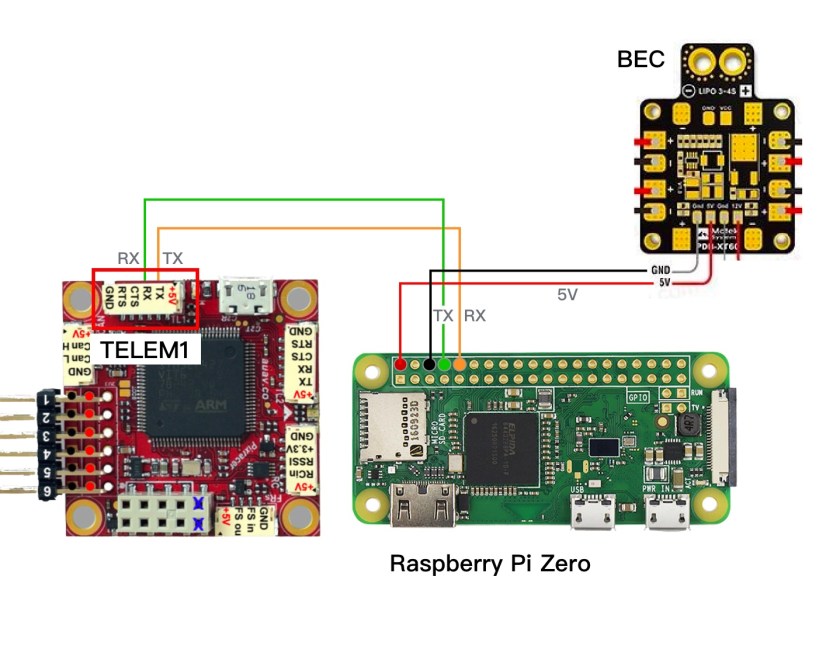
**How to send data between the flight controller to companion computer?**

In our case, we need to send data between Pixhawk to Raspberry Pi.

1. Hardware level, wiring the cable
2. Software level, set up the Mavlink connection

**Wiring**

Connect Pixhawk to Raspberry Pi as the diagram below. I suggest connecting 5V power source from a BEC board, not from Pixhawk TELEM port. When you connect the 4G USB modem and camera to the Raspberry Pi, the Pixhawk can’t provide enough current to them. The Raspberry Pi will be force restart.

[](https://i0.wp.com/bellergy.com/wp-content/uploads/2020/10/pixhawk-rpi-wiring-2.jpg?ssl=1)**Pixhawk to Raspberry Pi wiring**[](https://i0.wp.com/bellergy.com/wp-content/uploads/2020/10/pixracer-rpi-wiring-1.jpg?ssl=1)Pixracer to **Raspberry Pi** wiring

**Enabling Serial Console in the Raspberry Pi**

The serial console lets Raspberry Pi communicates with the px4. You can check [here](https://learn.adafruit.com/adafruits-raspberry-pi-lesson-5-using-a-console-cable/enabling-serial-console) for how to enable it.

**Setup the MAVlink Port in the QGroundControl**

We need to tell the flight controller to send the MAVlink data to the connected port. I use [QGroundControl to change the parameters](https://docs.px4.io/master/en/advanced_config/parameters.html" \l "findingupdating-parameters).

If you followed the guide above that connected the wire to TELEM 1, the [default setting](https://docs.px4.io/master/en/peripherals/mavlink_peripherals.html#default-mavlink-ports) is fine. I changed the SER\_TEL1\_BAUD to 38400 or below to reduce the bandwidth usage on 4G.

# PX4 parameters:

MAV\_0\_CONFIG = TELEM 1

MAV\_0\_MODE = Normal

MAV\_0\_RATE= 1200 Bytes/s

MAV\_0\_FORWARD = True

SER\_TEL1\_BAUD = 38400 baud

**Install and setup mavlink-router (connecting drone to ground station)**

**What is mavlink-router?**

Whether the flight controller communicates with the Raspberry Pi or the Raspberry Pi communicates with the ground station, they are all through mavlink.

When the Raspberry Pi receives the mavlink data from the flight controller via the serial port, the mavlink-router will distribute the mavlink data to other devices. (e.g. ground station, simulator or your computer visual program, etc.)

**Install mavlink-router**

Login to your raspberry pi, then:

*If you didn’t install git, you may install it*

sudo apt update

sudo apt install git

**Download the source**

git clone https://github.com/intel/mavlink-router.git

cd mavlink-router

git submodule update --init --recursive

sudo apt install git meson ninja-build pkg-config gcc g++ systemd

**Build and install**

*If you didn’t install pip3, you can install it by sudo apt install python3-pip*

sudo pip3 install meson

meson setup build .

ninja -C build

For the detail of the installation, please read the document [here](https://github.com/intel/mavlink-router.git).

**Create a config file**

sudo mkdir /etc/mavlink-router

cd /etc/mavlink-router

sudo pico main.conf

My config is:

[General]

TcpServerPort=5760

ReportStats=false

MavlinkDialect=common

[UartEndpoint serial0]

Device=/dev/serial0

Baud=38400

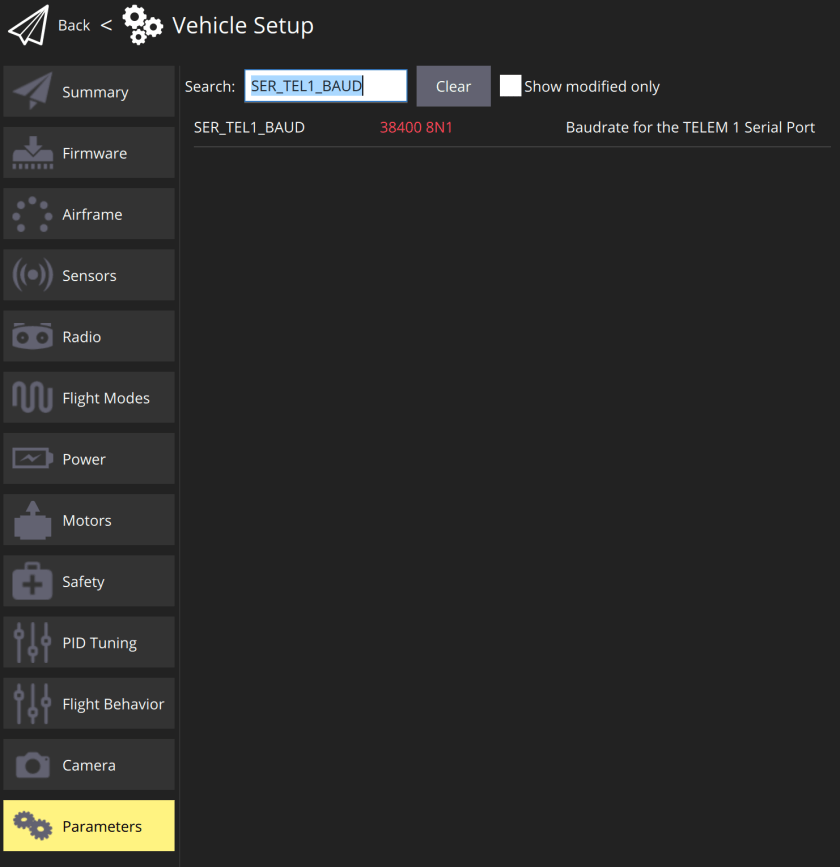
[UdpEndpoint local]

Mode=normal

Address=127.0.0.1

Port=14550

It tells mavlink-router where is the serial port of the mavlink input. In my case, it is /dev/serial0. And, it configed Baud=38400, make sure it is same setting in the QGroundControl Parameters SER\_TEL1\_BAUD



Please find the detail settings of the [config file here](https://github.com/intel/mavlink-router/blob/master/examples/config.sample).

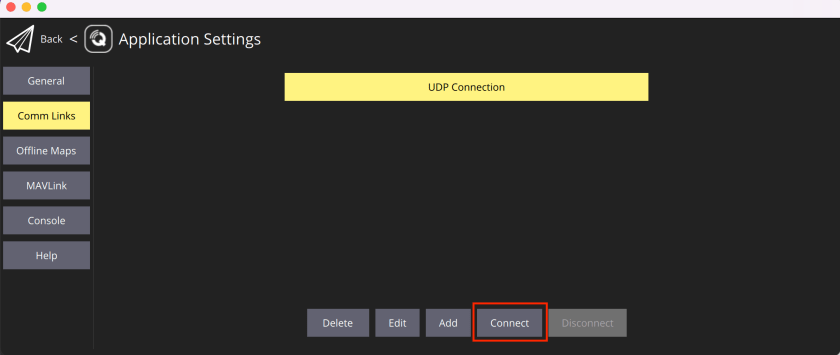
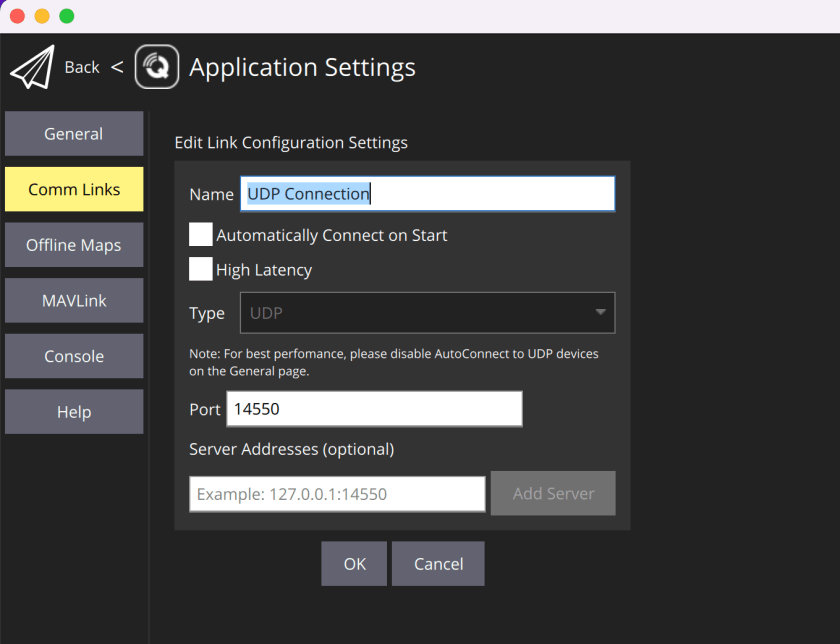
**Run**

The command below will read the mavlink data then forward it to the 192.168.1.100 port 14550 (assume your ground station is in 192.168.1.100).

mavlink-routerd -e 192.168.1.100:14550

**Connect the Ground Station**

In the QGroundControl open the Application Settings > Comm inks and add a UDP Connection Port 14550 then click "Connect" button.



**It is a sample video that how it works:**

**Auto start mavlink-router when system boot**

**Create the file below to /etc/systemd/system/mavlink-router.service**

[Unit]

Description=MAVLink Router

Wants=zerotier-one.service

After=network-online.target zerotier-one.service

[Service]

Type=simple

ExecStart=/usr/bin/mavlink-routerd -e {groundstation}:14550

Restart=on-failure

RestartSec=3

[Install]

WantedBy=multi-user.target

*Replace the {groundstation} to the IP of your ground station*

*Please remember, both the drone and the groundstation should use the zerotier IPs. If you didn’t setup the Zerotier, plaese read*[*here*](https://bellergy.com/4-install-zerotier/)*.*

**Start the service**

sudo systemctl start mavlink-router.service

**View the log**

If there is error, you can view the log:

sudo journalctl -u mavlink-router.service

**Enable the service**

If the start runs fine, we can enable the service.

sudo systemctl enable mavlink-router.service

**Check the status**

If you find the mavlink-router didn’t start up, you can check the status:

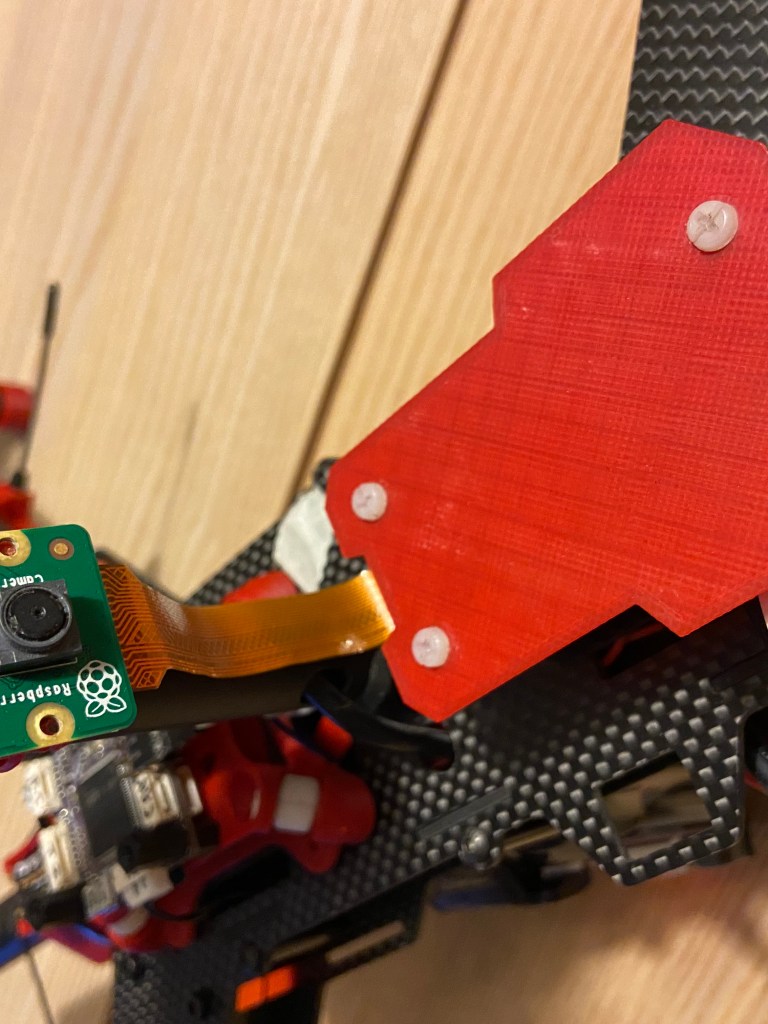
systemctl status mavlink-router.service

**Set up video streaming**

I need an FPV from the drone to the ground station. It is what this setup can do:

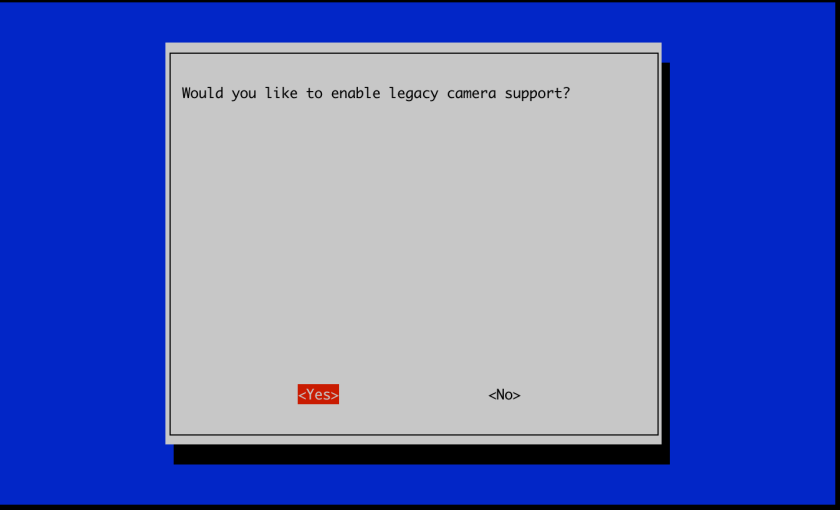
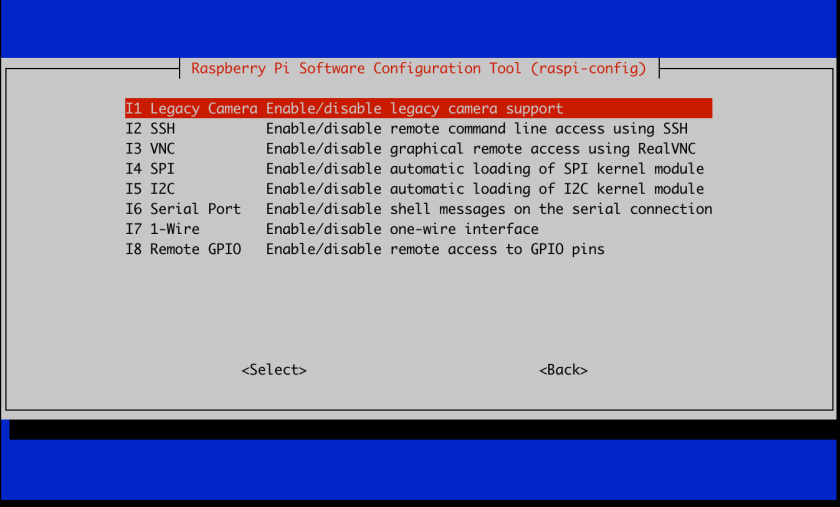
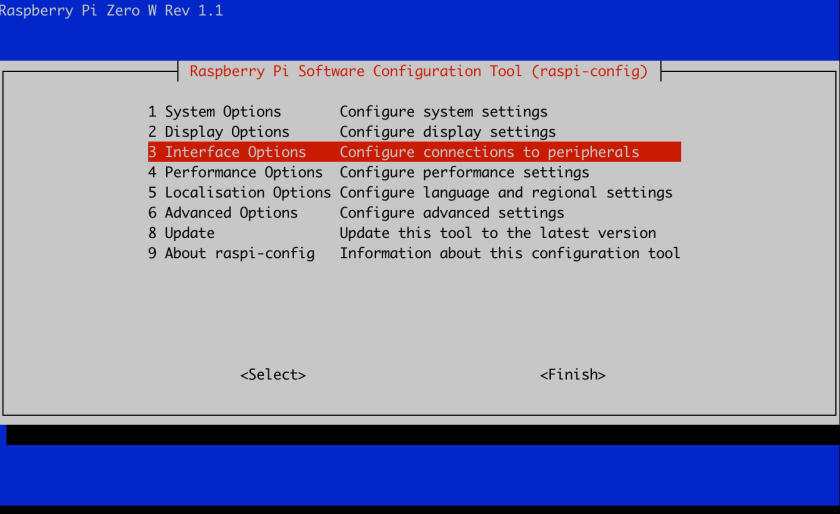
* Using Raspberry Pi Camera Module captures the live image;
* Using GStreamer to set up the video streaming;
* Sending the video streaming to the ground station through UDP;
* Using mavsdk to monitoring the drone’s status and start the video streaming when the drone is armed;
* Stop the video streaming when the drone is disarmed
* Save the video to file;

**My hardware**– Raspberry Pi Zero 2  
– Raspberry Pi Camera Module 2



**Connect the Raspberry Pi Camera Module**

Connecting the Camera Module to Raspberry Pi is straightforward, the only thing that should note is using raspi-config to enable the camera.



**Install gstreamer1.0**

I use gstreamer to encode the video and send it to the grand station through UDP.

sudo apt-get update

sudo apt-get install gstreamer1.0-tools gstreamer1.0-plugins-good gstreamer1.0-plugins-bad

**Start video streaming on the Raspbarry Pi**

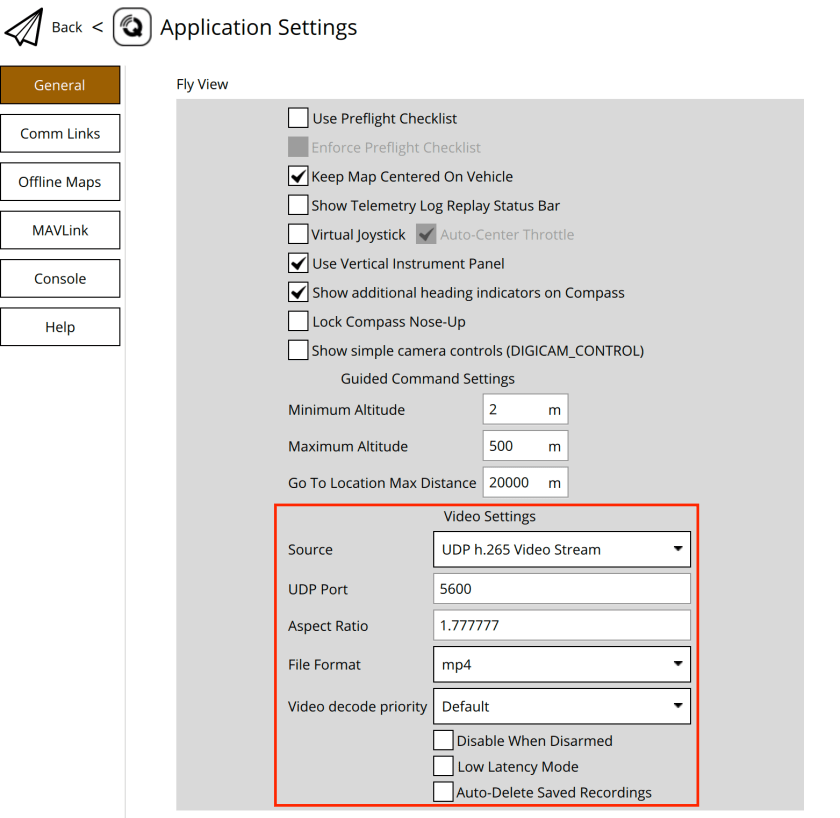
The the gstreamer install success, you can test the video streaming by the command below. It will send the video to the ground station through the UDP on 5600 port.

raspivid -n -w 640 -h 360 -b 1000000 -fps 15 --flush --timeout 0 -o - | gst-launch-1.0 -v fdsrc ! h264parse ! rtph264pay config-interval=10 pt=96 ! udpsink host={ground-station\_ip} port=5600

**Replace the {ground-station\_ip}** to the IP of your ground station.

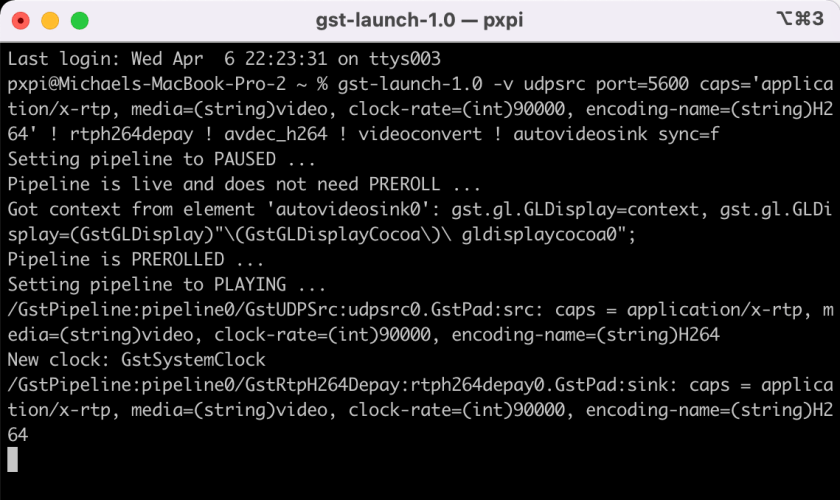
**Receive the video**

The setup above is the default setting of the QGroundControl. If you are using Windows and do the setup below, the video can be shown automatically.

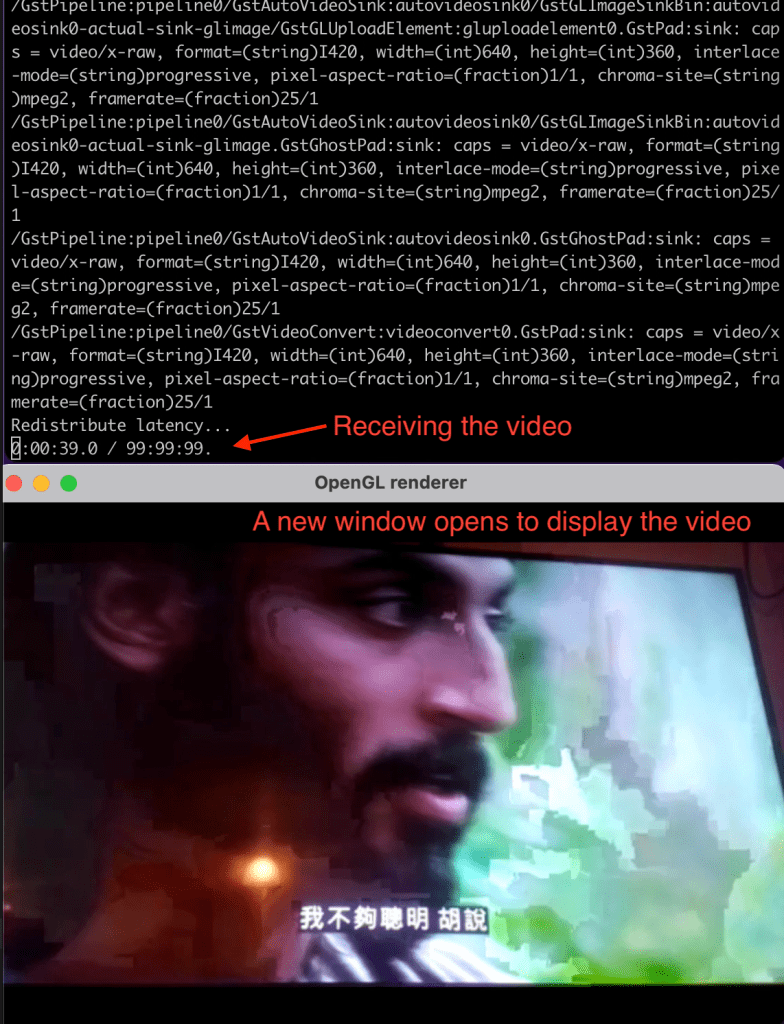


Unfortunately, I can’t make it work on Mac with QGroundControl. I don’t know the reason, the workaround is to use the gst-launch to receive the video. It is the command that I used on Mac:

gst-launch-1.0 -v udpsrc port=5600 caps='application/x-rtp, media=(string)video, clock-rate=(int)90000, encoding-name=(string)H264' ! rtph264depay ! avdec\_h264 ! videoconvert ! autovideosink sync=f

The command is running and waiting for the video stream.

When it receive the video, it will open a new window to display it.



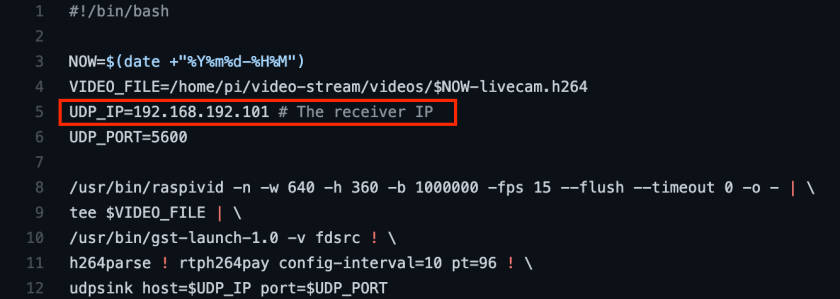
**Auto start and stop the video streaming**

I want the video starts only when the drone is armed and it will stop the stream when disarmed. I used the [mavsdk](https://github.com/mavlink/MAVSDK-Python) to listen the drone’s status.

**INSTALL THE SCRIPTS**

I write a python script to achieve it. Please read my [github](https://github.com/rc-bellergy/pxpi/tree/master/home/pi/video-stream).  
You can copy two files to your raspi:

* [stream.sh](https://github.com/rc-bellergy/pxpi/blob/master/home/pi/video-stream/stream.sh): It is the shell script to start the video streaming.
* [controller.py](https://github.com/rc-bellergy/pxpi/blob/master/home/pi/video-stream/controller.py): is the python script that starts/stops the stream when the drone is armed/disarmed.

Remember to change the UDP\_IP to your receiver IP.

I assume you save the files to /home/pi/video-stream/

Change the files to executable:

cd /home/pi/video-stream/

chmod u+x stream.sh

chmod u+x controller.py

Create a video folder to save videos. The video files will be saved to /home/pi/video-stream/videos/\*

mkdir videos

Install required python modules:

sudo pip install mavsdk psutil

**RUN THE SCRIPT**

sudo python controller.py

A demo of using video-stream/controller.py and receiving the video on Mac.

**START THE CONTROLLER.PY WHEN SYSTEM BOOT**

I need the controller.py auto start when the system boot. I added this line to the /ect/rc.local. It is my [rc.local](https://github.com/rc-bellergy/pxpi/blob/master/etc/rc.local) file.

# Strat video-control

/usr/bin/python /home/pi/video-stream/controller.py