In this write up I will present extra information that was not covered in the group’s final presentation. Specifically, I will give an overview on how to use the Raspberry Pi with the PixHawk to perform autonomous flight.

* One thing that I noted when researching the drone project and other robotics, was that a lot of work was done in Linux. I chose to utilize an installation of Ubuntu on my laptop for the development of this project.

Lewis Raspberry Pi Address (on the Wi-Fi labeled “myLewis”) : 10.120.18.2 (Static IP Address)

* I had to personally change the Pi’s IP to static rather than Dynamic. If it were still dynamic, then the user would have to take the Pi out of drone and connect it into a computer every time the Pi reconnected to the network, in order to find its IP Address. Having a static IP address helps testing stay consistent.

Labtop Address : 172.17.0.1 or 10.120.33.115 **<- (Make sure you change to your computer’s IP)**

* This is the address of the laptop that we will be using to communicate to the drone.
* Before utilizing autonomous functions make sure the drone can be flown in a manual mode with the RC controller to prevent possible crashing on takeoff.
* There will be many libraries you have to download to get the communication process working. If using Ubuntu, learn file manipulation and how to properly navigate the Operating System.

<https://docs.px4.io/master/en/config/> - This link will point you in the direction we took to learning all the ins and outs of the systems through the manufacturer’s documentation.

<https://mavsdk.mavlink.io/main/en/index.html> - This link will lead you to the MAVSDK frontpage. MAVSDK is the library we used with python on the raspberry Pi to communicate the actions we wanted the drone to perform.

<https://github.com/mavlink/MAVSDK-Python> - This link is for more documentation on MAVSDK and it’s setup that we followed to get our end results.

<https://github.com/mavlink/MAVSDK-Python/tree/main/examples> - This link provides example scripts written in Python for users to edit and then run on their own.

\*\*Make sure to utilize a ground control station as well, to monitor the metrics of your drone. Our group utilized QGroundControl and overall, it was very buggy and unreliable. Steer away from it and find a different ground control that is compatible with the PX4 autopilot.

**To Set Autonomous Tests w/ Scripts..** (ls / ls -a) \* In order to look into directories in Linux

#1. ssh pi@10.120.18.2 - run ssh into Pi on the same network

(\*Password for PI: raspberry)

#2. cd Desktop/Test\ Code\ for\ Drone/ - move to the directory where scripts are saved

#3. Run one of the two Commands below:

- python TakeoffAndLand.py –connect - call your python scripts name to be run

- python TakeoffAndLand.py --connect /dev/serial0 (You will need the serial configuration b/w PixHawk and Pi already done for this command)

-------------------------------------------------

\* Problems I ran into

1. Could not connect / Heartbeat not detected?
   1. There is probably a setting in your drone’s autopilot preventing communication even when the drone is properly on.
   2. Make sure your network connection is good.
   3. Sometimes the school’s Wifi simply does not allow us to connect and the reason in unknown by us.
   4. Make sure wiring is correct, wires can fall out after crashes or be destroyed so try to check on connections after every flight

---------------------------------------------------

Here I will discuss how to potentially hook up the drone to QGroundControl wirelessly, to transmit data.

* The MavProxy command shown below is essentially a way to transmit the live metrics of your drone back to an appropriate ground control program.

To get MAVProxy running...

(\*Password for PI: raspberry)

#1. ssh pi@10.120.18.2

#2. sudo -s

#3. mavproxy.py --master=/dev/serial0 --baudrate 921600 --out 172.17.0.1:14550 --aircraft MyCopter

------------------------------------------------

This will set us up in the MAVProxy environment and send out a constant signal to QGroundControl on your computer for connection over port 14550 on the same Wi-Fi. (Port used: 14550)

------------------------------------------------

To Run Camera Live Stream... (WORKING)

* In order to utilize the ArduCam with the Pi as presented below, you will have to get into the Pi’s configuration and set its core usage to 128 or more.
* This ultimately prevented the autonomous command above to stop working as the Pi became much more unreliable, most likely due to the power draw and the ability of the battery.

#1. (OPEN MEDIA PLAYER)

- Install VLC Media Player

- Open VLC Media Player before ssh into Pi

#2. ssh pi@10.120.18.2

#3. \*Run Below Command in Pi shell\*

raspivid -o - -t 0 -w 800 -h 600 -fps 12 | cvlc -vvv stream:///dev/stdin --sout '#rtp{sdp=rtsp://:8080/}' :demux=h264

#4. In VLC go to MEDIA (TOP BAR) -> Open Network Stream

#5. In network URL box (input): rtsp://raspberrypi:8080/