How to set up a plane using Pixhawk

By Christian Hales

(REQUIRES QGROUNDCONTROL)[1]

Introduction

MAV flying and autopilot are fun and rewarding projects. PX4 Software and its associated hardware, Pixhawk, are powerful and well-supported means of accomplishing MAV autopiloting. However, it can be difficult for new (or experienced!) members of the MAGICC Lab who have little or no experience in PX4 to get started.

That is where this document comes in. After a summer of deep learning in PX4 and its related hardware and software in the MAGICC Lab, I found that I was able to lead the 2024 MAGICC Lab Hackathon's PX4 team in setting up all necessary hardware and software. Since I was the one who knew how to do all the steps, it was natural to assign me, amongst other team members, to write this paper.

I hope this introduction to PX4 and plane assembly saves you weeks of failed experiments, rabbit holes, painful realizations that the webpage of instructions you and your coworkers wanted 2 weeks ago just required a slightly different web search, and the otherwise general banging of ones' own head against the desk that can accompany hardware, software, and middleware development.[2]

To Start

If setting up everything, one can find a helpful start at:
https://docs.px4.io/main/en/dev_setup/config_initial.html or by searching "PX4 setup" on the web.

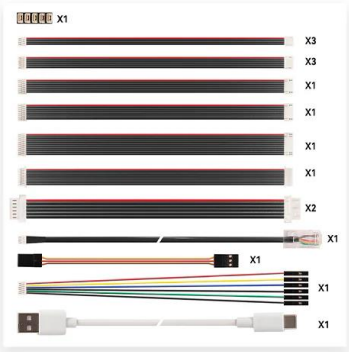
If just working in simulation, go down to SITL development below.

The first decision to make is to choose which Pixhawk you will use. Each Pixhaw has slightly different documentation on how to attach all the sensors and such properly to it. Included in this document are instructions on how to set up the Pixhawk 6c mini and Pixhawk 6x, and notes on what the online documentation, as of August 30, 2024, does *not* tell you.

How to set up Pixhawk 6X:

[1] QGroundControl is always needed, except for offboard work sometimes on SITL.

[2] DISCLAIMER: PX4 Documentation changes often enough, and enough files are buried elsewhere, that the links (and some steps) can become obsolete. The goal of this documentation is to show principles and concepts in such a way that they resist changes in the PX4 Flight Stack and can allow anyone to know just what exactly a Pixhawk and a PX4 are, and what to do about them.



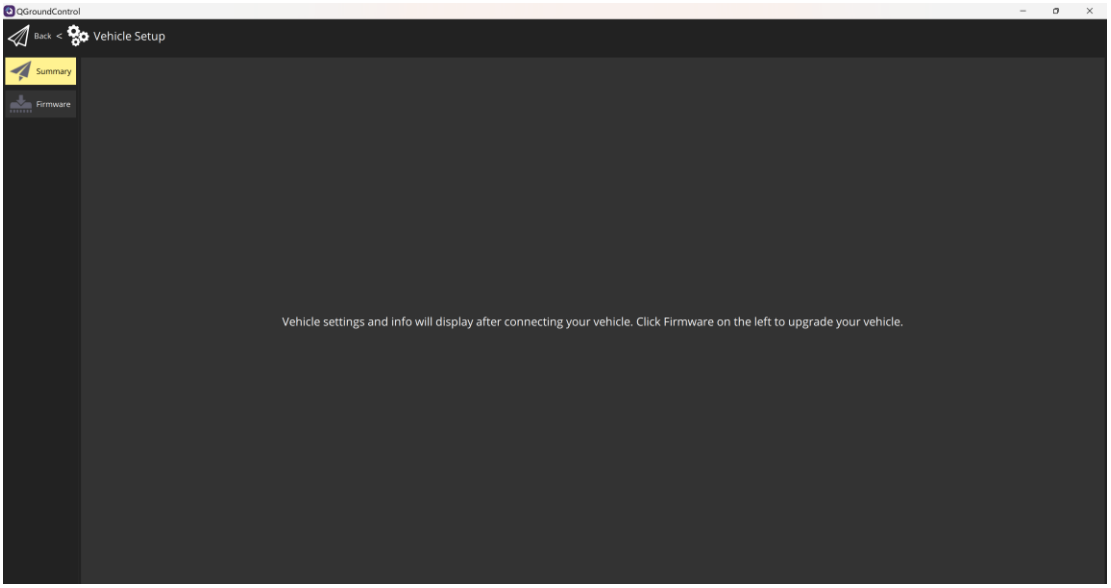
Top left: [The Pixhawk 6X.]
Bottom Left: The GPS.

Top Right: The Power Module.
Bottom Right: Some included cables, including USB-USBC, JST-JST, JST-LargePowerOutput, etc.

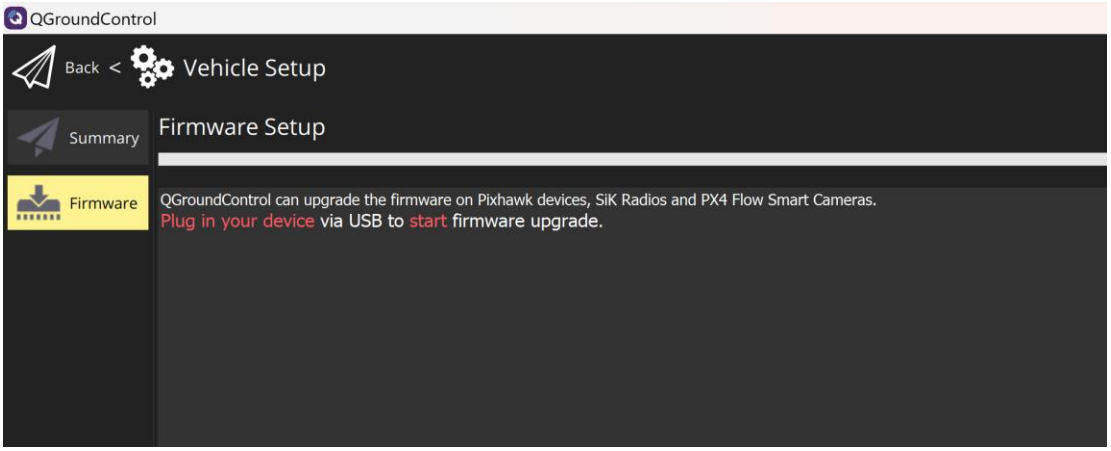
1. Install QGroundControl. You will need it for your laptop eventually as a "ground station" when you fly your vehicle, so it's a good idea to start on the laptop you'll be using instead of some lab computer.[3] Instructions are as follows:
    a. Start up QGroundControl.

[3] If only working in simulation, you will likely work on an Ubuntu machine. QGroundControl is still necessary for simulation debugging and for manual simulated flight.

b. Navigate as such:  (QGroundControl_icon) > Vehicle Setup.



c. Click on "Firmware." This is the page where QGroundControl will flash firmware updates to Telemetry Radios and Pixhawks.



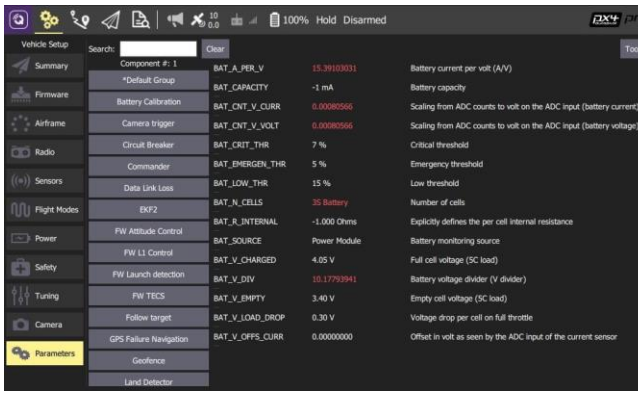
d. Now you will flash a firmware to the Pixhawk.[4] Options are either to flash the latest version or to flash a custom firmware. Custom firmware will be preferred if you are altering the PX4 code in any way, shape or form.[5]

2. Plug in the GPS (the one that came with the Pixhawk) into the Pixhawk.[i]
   a. About the GPS:
      i. Yes, you could plug in the Pixhawk directly first. I like having the GPS plugged in first because, as you will learn, the GPS makes some beeps and shows colors on its LED that are very helpful for debugging.
      ii. See the UI and LED section of this page for an analysis of those colors. In case the link changes later, you can get there by Googling "LED Meanings Pixhawk series PX4" or "GPS Pixhawk colors".

[4] See the main documentation. You can also search the web for "flash the firmware px4"
[5] See this link or search the web for "make command pixhawk px4" for more instructions.

iii. The device will likely be flashing red or holding blue till most everything is set up. I imagine you will be doing most of the setup inside, under several floors of a building. That means the global position of GPS will not work well. That will matter later, when you discover that switching to Offboard mode will be rejected (i.e. a request from a Raspberry Pi to take control will be denied).

3. Plug in the Pixhawk directly to the computer with the USB-to-USB-C cable.
    a. While you could start with telemetry radios, I find it easier to start by plugging the Pixhawk directly into your computer (the one you installed QGroundControl on). You will need to have it plugged in directly when you calibrate the ESCs (Electronic Speed Controllers) used on the motors.
4. Choose your airframe. Then hit “apply and restart”.
    a. Most changes in QGroundControl require you to reboot (i.e. restart) the vehicle. In the “parameters” tab of QGroundControl, as seen below, you can reboot even while using telemetry.



<-- “Tools” Button

Click “Reboot Vehicle” in the “Tools” tab that pulls up

5. Set up the Radio. [6] In QGroundControl, run the Calibration.
6. Set up Flight Modes.
    a. Make sure the top left control stick can control Arming or Disarming.
    b. The Flight Mode will stay red until one of the channels (preferably the same as switching modes<>) is assigned to <>
    c. Set one of the channels (preferably channel 7 for quick access and its halfway-to-on feature) to switching to offboard mode
7. Battery/Power section
    a. This needs proper battery setup, and requires restarting when done.
    b. IMPORTANT: ESC Calibration must be done with the USB-C cable plugged in, and not from the telemetry Radio.[7]
    c. **DANGER:** DO NOT EVER have a propeller attached while calibrating ESCs, unless you want a fully powered motor(s) ramming the vehicle at full speed in any given direction. This will possibly send individuals to the hospital to get stitches for a propeller hitting their leg. This will DEFINITELY damage the fuselage and related plane components.

[6] See “Setting up the FrSky RC” below.

[7] When the Pixhawk is already nested good and tight and double-taped into the plane to avoid sensor errors from jostling, it can be difficult to reach a USB-C cable. You will find it very helpful to attach a USB-C extender cable, whose female end can easily be reached from the fuselage opening. In some cases, it is a good idea for that extender cable to have a right-angle male USB-C end.

d. Wear sunglasses or protective goggles when setting up ESCs or debugging a plane while at the RC park.

8. Actuator/Motor setup
    a. The documentation online and the process shown in QGroundControl's GUI are self-explanatory. The main thing that might be confusing is that the PWM Main on QGroundControl is I/O Board Output, and PWM AUX is FMU Board Output. Hopefully, adding all these links can increase the reader's confidence that all the documentation for how to do all this is (mostly) there, if not hard to understand for new readers.
    b. Remember that not only does the FMU/IO board need to be powered by a battery separately from the main board of the Pixhawk, but each ESC needs to be powered by a battery too.
        i. That does NOT mean that one big battery can't power everything. It just means that both sides of the Pixhawk need to get power, and so does each ESC.
        ii. The FMU/IO pinout gets power from a battery plugged into an ESC via a 3-wire S/+/- connection.

How to set up Pixhawk 6C Mini:

Very similar to 6X, but with a few differences:

- The wires needed to connect sensors to the Pixhawk are different from the Pixhawk 6X because sensors need to be connected differently on the 6C Mini. For example, the Frsky connecter will be connected by a JST instead of by a 3-wire.
- The idea generalizes to other Pixhawks. As they make newer and better ones, they will continue to change ports and setup slightly.

Setting up the FrSky RC

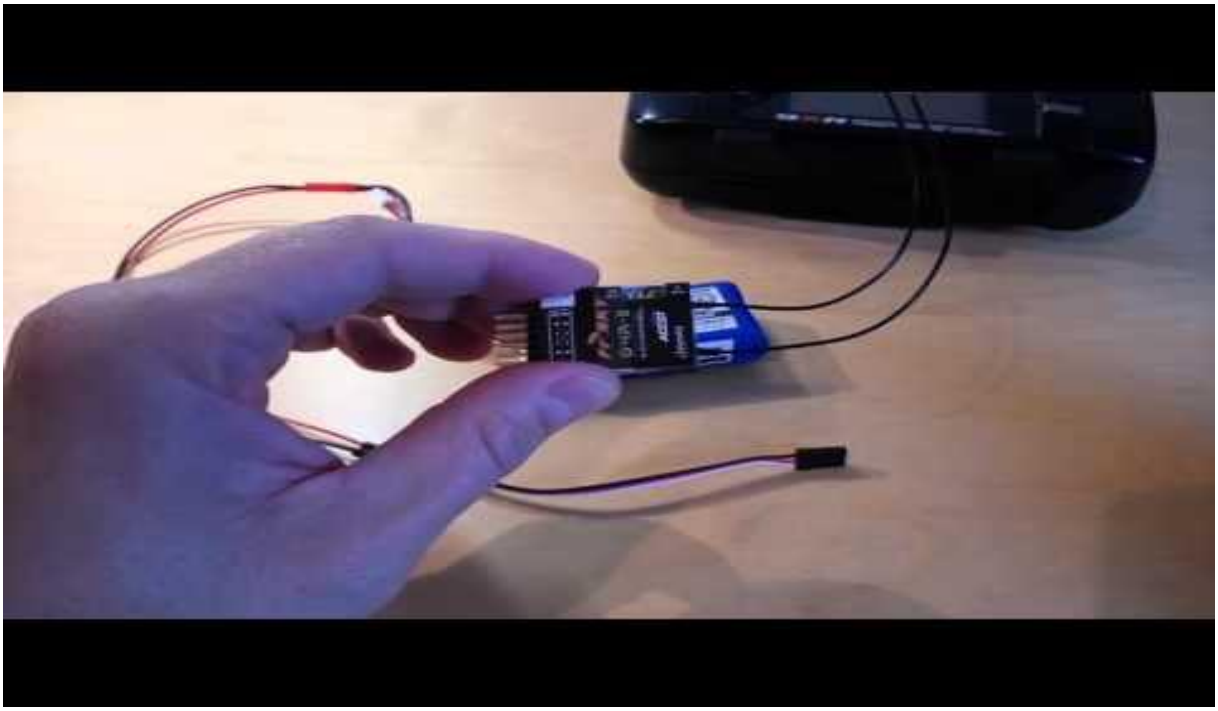
The terminology of Transmitter and Receiver can get a little confusing, so I added pictures with labels.

FrSky Transmitter	**FrSky Receiver**
(a.k.a. the RC Controller)	We use the D4r-II FrSky Receiver most. (read the manual! A good policy for all pieces of tech: https://www.frsky-rc.com/wp-content/uploads/2017/07/Manual/D4R-II.pdf)

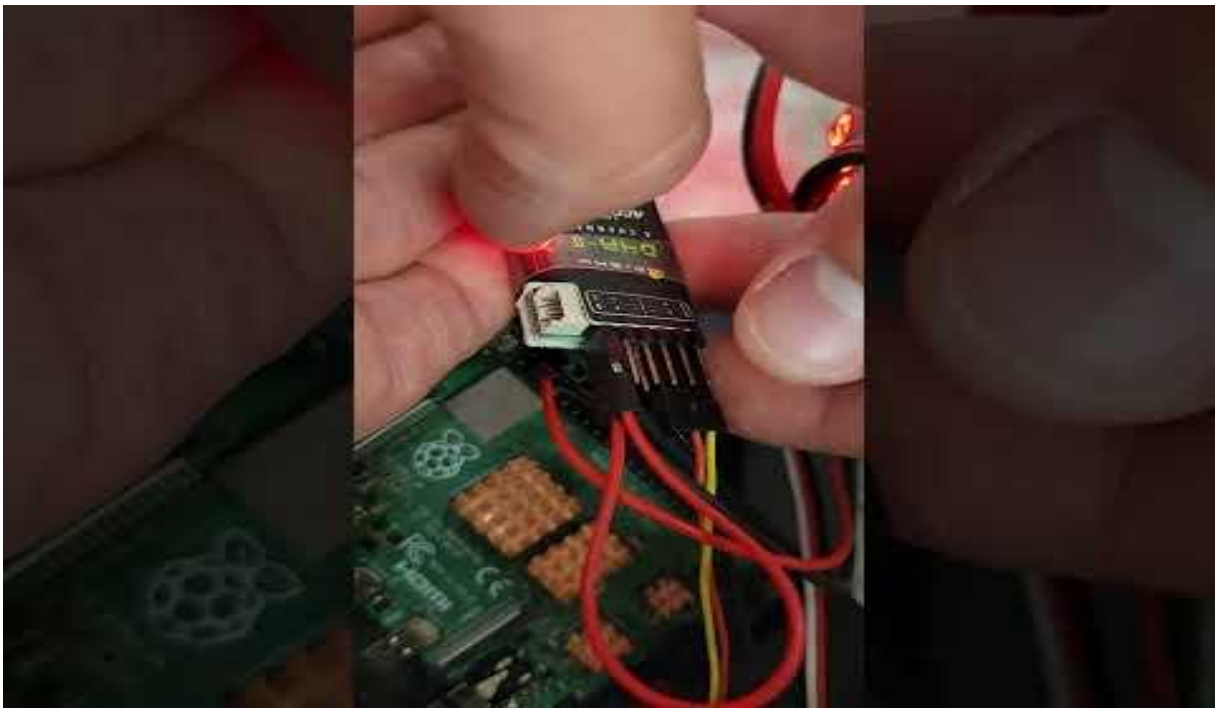
	

The manual for the receiver contains most of the important information about how to bind the receiver to the transmitter. The Pixhawk comes with the appropriate wire to connect the receiver to the Pixhawk. For example, the Pixhawk 6X connects via an RSSI cable (Signal/+/-), while the Pixhawk 6C Mini connects to it from an RSSI_to_JST-like cable.

If you are *only* looking for instructions on binding the receiver to a general transmitter, the documentation above or this video is helpful: https://www.youtube.com/watch?v=D3Jxs89I7XU



For setting up and connecting the RC radio receiver and transmitter *using MAGICC Lab equipment and specifically how we use them in our planes*, see this video : https://youtu.be/xDy_Sb82Gsw



IMPORTANT: As mentioned in the self-made video and in the manual, you will see that the PPM connection required for connection WILL NOT HAPPPEN unless signals 3 and 4 are connected via a F-F wire. That will save you lots of time.

Once a receiver is bound to a certain transmitter, it will always be connected every time you plug it in.

Setting up SiK Radio (Holybro)

This lets you connect to QGroundControl wirelessly, which is super important:

- When all the wiring and everything is done for the plane
- When you are testing motors with propellers or something somewhat dangerous
- When you are flying you always need QGroundControl no matter what, so this is how you get that to work.

Don't use the old 3dp ones; they just don't seem to connect anymore.

Here is the Quick Start Guide. You can also just look it up: https://docs.holybro.com/radio/sik-telemetry-radio-v3

It's a good idea to flash the firmware for both in QGroundControl, then plug both in to make sure they work. Plug one of them into TELEM1 (via the JST cable that comes with the SiK Radio) and the other via USB-MicroUSB to the computer. [8]

IMPORTANT: If the QGroundControl is flashing the firmware but not recognizing the connected SiK Radio pair as though it were a direct USB connection (which it SHOULD recognize!), then you need to uninstall QGroundControl and re-install it. That's what worked for me (Summer 2024).

The Holybro pairs are bought together; once you have tested a pair and know that they work, make sure to use the label maker to make good labels that clearly ascribe both as a pair. For example, one pair in the lab is denotes as "TE-A1" and "TE-A2".

Running SITL (PX4 Simulation)

You may have noticed that, up to this point in instruction, you have not downloaded any code. There is a PX4 GitHub, but I have not put in any documentation for it.

Up until Version 1.13, there was no need for a GitHub; one simply had to flash the firmware to the Pixhawk to do anything. The benefit of Version 1.14 is NOT to alter the PX4 code; you do not want to TOUCH that code directly[9]. The benefit of Version 1.14 and on is to allow for connection to Gazbeo simulations and other such non-Pixhawk items.

Setup Gazebo

But, the real question is, how to set it up? Until you know where to look in the documentation, it can be confusing. Use this link or search the web for "PX4 Toolchain Installation".[10]

Alternatively, you can use the Docker Image I made: https://github.com/FredJones4/dockerized_ros2_control.

With the whole environment setup as described in either of those links, Gazebo can be properly used.

What is Gazebo?

Gazebo is the officially supported simulation environment of PX4. You can search the web for the documentation on PX4 Gazebo simulation or use this link. MAKE SURE YOU USE THE MAIN VERSION, not

---

[8] TELEM 1 is designed to just naturally connect to whatever is set up.

As is explained in the section on Raspberry Pi connection to Pixhawk, you must go to Vehicle Setup > Parameters > MAVLINK and enable "Mav_1_config" and setting it to TELEM 2. This is what allows you to follow the instructions to set up the Pi. It's not that the documentation isn't there; it's just somewhat hidden. See https://docs.px4.io/main/en/companion_computer/pixhawk_rpi.html#:~:text=For%20information%20about%20how%20serial%20ports%20and%20MAVLink%20configuration%20work%20see%20Serial%20Port%20Configuration%20and%20MAVLink%20Peripherals and its links for more details.

[9] As of August 28, 2024, there has arisen a case where you DO want to touch that code: if altering the uORB topics or dds middleware. See my PX4-Autopilot repository for an example, as well as a pdf example from BYU's CS department on how to properly manage such a system while keeping the latest updates from PX4-Autopilot.

[10] I prefer using Ubuntu 22.04 (as of Summer 2024) for simulation, which requires ROS Humble. It seems that most often, 2 years of development makes for best support. Perhaps in 2026, the Ubuntu 24.04 will work best.

version 1.12. Some links (like this one) use old versions and old instructions. Don't do the classic; just use the main one.

You will see instructions on how to simulate get any VTOL, fixed-wing, or quadrotor. The BIG benefit here, aside from being able to practice flying with a plane[11], is to test offboard control commands[12].

IMPORTANT: As of August 28, 2024, documentation exists on a supported method to include more topics in both SITL and physical setups. Please read to practice more effectively and compare real life to simulation.

Practicing Flying in SITL

We have in the lab, a RC Radio that can be plugged in by USB.

Use this transmitter for manual control in simulation.

Use the SITL explanation above to practice flying in SITL. Plug in the RC Controller to the computer.

Turn on QGroundControl and set up the vehicle and RC Controller to your preferences. This will be especially important for practicing quadplane-to-fixed wing transition and other unusual components of RC flying.

In the Simulation, type **commander takeoff** to take off. Type **commander –help** (two dashes) for more commands,

Running SITL with Offboard (MAVSDK-Python, ROS 2, *with-or-without Raspberry Pi*)

After setting up practice-flying in SITL, you can use your own code to control the simulation.

On MAVSDK-Python, this happens over a UDP:14540 connection.

[11] See "Practicing Flying in SITL"

[12] See Running SITL with Offboard. You will notice that not all commands are accepted upon startup by the Gazebo simulation with PX4. Unlike ROSFlight, the PX4 software differs in simulation from physical testing.

On ROS 2, this happens over a “micro-ROS” UDP4:8888 connection.[13]

There exists example code for MAVSDK-Python for running offboard mode, which I have altered to include data gathering. Unfortunately, I learned from a discussion I had with a PX4 member[14] that actuator controls are not supported on v1.14 (the simulator version) and older, though attitude and altitude are still controllable.

While there exists some similar example code for the supported medium (ROS 2) in C++, Python is much easier to work with. So, I wrote code in Python[15] to do the same thing.[16]

If all you care about is setting waypoints, attitude and altitude, then MAVSDK is just fine. [17]

If you care about data collecting, actuator control, or anything else beyond setting waypoints (which you can also just do with QGroundControl, by the way), then use ROS 2. As mentioned in the forum link above, ROS 2 is the only currently supported method for actuator and motor control. Further, some empirical tests I have run prove that data gathering in MAVSDK happens (at best) at 15 Hz, while ROS 2 data gathering occurs at well over 100 Hz.

For a basic test in ROS 2, use the offboard_test I wrote with instructions in README.md:

For a basic test in MAVSDK, click here.

Some thoughts on ROS 2

You will need to know:

- How fast topics are being published (run **ros2 topic hz <insert topic name>** after sourcing the terminal)

---

[13] See https://design.ros2.org/articles/ros_on_dds.html for more information on the history and the “why” of all this. “Micro-ROS” is a fancy term for the XRCE-DDS agent that ROS is designed to work with. See the ROS 2 PX4 User Guide for how to set up ROS 2 libraries and connections and how to turn on the micro-ROS connection (it’s run with a CLI command “MicroXRCEAgent udp4 -p 8888”).

[14] Very good note: use PX4 Forum to reach out and ask questions to the developers. They are very good about responding quickly.

[15] See the OffboardTesting.py file in the ros2_px4_interface package found here: HYPERLINK "https://github.com/FredJones4/vtol_ctrl_ros2/tree/main/src/ros2_px4_interface"https://github.com/FredJones4/vtol_ctrl_ros2/tree/main/src/ros2_px4_interface

[16] Go to the ROS 2 tutorials found online to learn how it all works. For Ubuntu 22.04, which my version of everything was using, use Humble: https://docs.ros.org/en/humble/Tutorials.html. Learning up to Learner: CLI tools and Learner: Client Libraries is enough to understand the tutorial (and just how cool ROS 2 is: you can encapsulate all these separate pieces of code and make nodes like candy).

[17] As of August 30, 2024 MAVSIM has not been wrapped with MAVSDK; such a method would require actuator control, which MAVSDK cannot do. Such an application is mainly for drones. While a fixed-wing setup with waypoints using a Dubins path (which naturally occurs in waypoint setting in QGroundControl) could possibly have been achieved, a lack of time and a previous conversation I had with PX4 developers led me to believe that our time would be better spent developing with ROS 2. *It might be of some interest to future MAGICC Lab Researchers to set waypoints for fixed-wing aircraft in SITL or in real life.*

- Which topics from PX4 are being accepted (in hardware, see dds.topics.yaml. In SITL, see the list that is spouted from the terminal when running **make px4_sitl <whichever gz simulation>.** *They can be different.* See also the list from **ros2 topics list** after sourcing a new terminal while an active terminal is running.). IMPORTANT: As of August 28, 2024, the PX4 team has released support for including other uORB topics that were not included in dds.topics.yaml beforehand.

MAVSIM: Bridge to PX4 via ROS 2

Many MAGICC Lab students will be familiar with Dr. Randy Beard's UAV class, where students write a powerful flight simulator called MAVSIM from scratch. Since it is a class project, there will likely never be a public repository for MAVSIM. However, the ROS 2 PX4 Interface package that I made includes a wrapper that allows for MAVISM to control PX4 via ROS 2.

Setting up Raspberry Pi

The documentation for setting up the Raspberry Pi as a companion computer is very good, especially for setting up MAVSDK, but it is not very good at explaining one important detail:

You must go to Vehicle Setup > Parameters > MAVLINK and enable "Mav_1_config" and setting it to TELEM 2. This is what allows you to follow the instructions to set up the Pi. Otherwise, the TEL2_Baud will NOT show up. It's not that the documentation isn't there; it's just not super obvious. See this link and its links for more details.

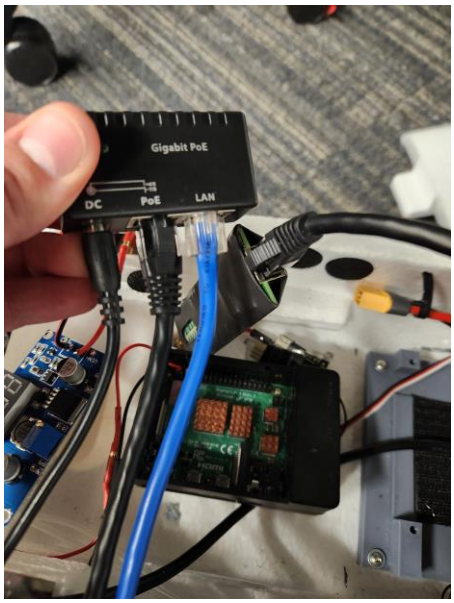
Technically, altering the settings for ROS 2 sets the TELEM settings back to standard; but unless you realize that MAVLink was turned back off, and you want to connect to TELEM 2 again, you will not see TEL_2_BAUD until you reset the board (i.e. reboot the vehicle).

Ubiquiti Bullet Connection

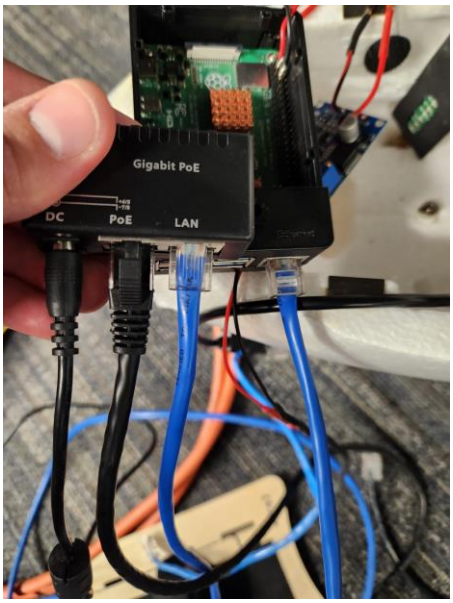
The lab documentation and online documentation is a little out of date.

The only "hardware" setup that needs to happen is done by the MAGICC Lab Comms master. All the everyday MAGICC Lab user should need to do is:

1. Plug in an Ethernet POE's DC port to get power (in a clean way!) to the Raspberry Pi (that will be the cleanest way; other computers will be fine).
2. Plug in the Ethernet POE's LAN spot to the Raspberry PI (or other computer).
3. Plug in the Ethernet POE's PoE port to the Ubiquiti Bullet.



The Black PoE is connected to the Bullet

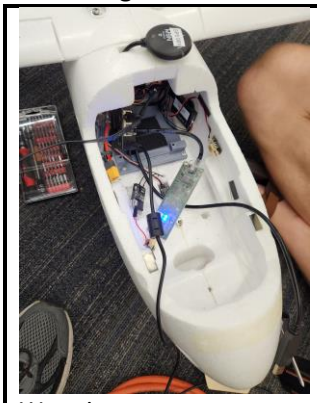
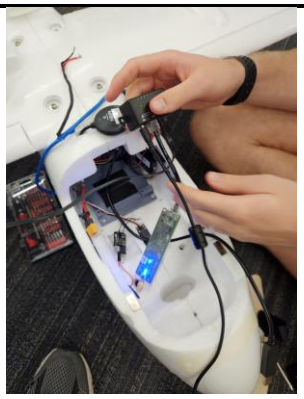


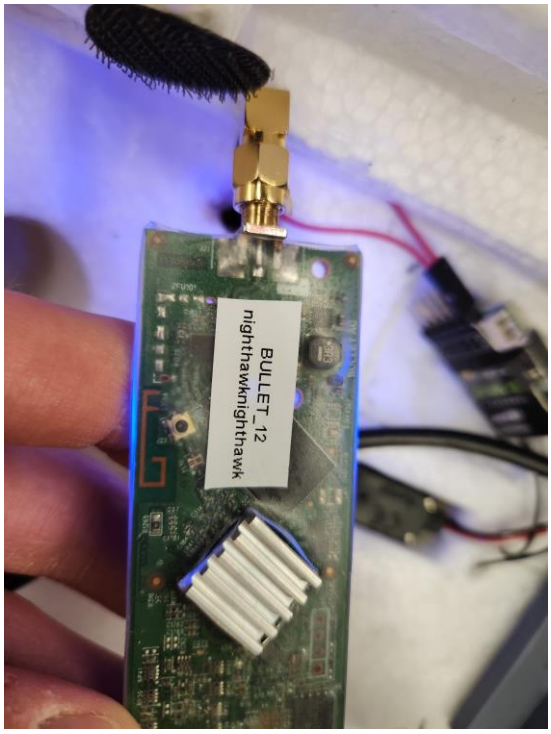
The Blue LAN is connected to the Raspberry Pi

Ubiquiti Bullet - Instructions Specifically for Raspberry Pi

1. On the Pi, select the ethernet wired connection provided by the PoE. At this point, there should already be two blue lights on the Bullet. These lights coming on can take about 16 seconds with good ethernet cables. Make sure you have good ethernet cables.

2. On the device's internet explorer, enter https://192.168.1.20. You should get a sign-in for the Ubiquiti Bullet, requiring your username and password.
3. Sign in. Find the rocket. You should not need to do much else.

		
Warming up	More Lights are turning on	Fully Connected



WARNING: Do not have the heat sink of the bullet facing upwards, or touching anything but the frame. It will get hot enough to burn wood or melt plastics.

NOTE: the username in this case was actually lower case. If the standard default (ubnt, magicc123) doesn't work, try lower case username with the password listed.

Summary

I hope this document is helpful for setting up anything and everything related to PX4.

Appendix:

A sample of what the tutorials (on PX4, ROS 2, MAGICC Lab) do not tell you:

- (PX4) ESC's must be calibrated, and battery installed to BOTH the ESCs and the pinout (plus however the other side of the board is powered), before motors and actuators can be tested in their assignments.
- (PX4) Not all motors and actuators work. Motors and actuators must be tested.[18]
- (MAGICC) Ubiquiti needs some resets and uses new software, Hostifi. (See new instructions above.)
- (PX4) How to reset Holybros telemetry (don't use3dp telemetry. Flash on QGroundControl. Redownload QGroundControl if needed).
- (PX4, MAGICC) What equipment is needed both for offboard and manual.
- (MAGICC) Link on the MAGICC Lab hardware guide for Pixhawk is old https://wiki.magiccvs.byu.edu/#!computers/pixhawk.md
- (MAGICC) "We however only recommend using PX4 for multirotor-type aircraft as their fixed-wing development is not well documented." https://wiki.magiccvs.byu.edu/#!computers/pixhawk.md Have the documentation say, 'You

[18] See the tutorial below about testing actuators and motors.

should google ROS 2 Raspberry pi integration'. That way, the links don't need to be updated so often.

- (PX4) Receiver and transmitter setup (and which is which)
- Don't alter the PX4-Autopilot code. Any "changes" should happen by git pulls from the main branch.

---

[i] It's a very good idea to label the GPS and Pixhawk so you remember which Pixhawk the GPS came with. It's less about which one fits into the slot and more about compatibility.