Vertical Flight Society

2020-2021 Design Build Vertical Flight Competition

Appendix A - Achieving Autonomy: An Overview

Preface

“Autonomy” as it is referred to in this document pertains to the unmanned aerial vehicle (UAV) flying a programmed mission without manual interference from the pilot in command (PIC). The software that runs on the aircraft is referred to as the “firmware” or “flight stack”, and the hardware that it runs on is referred to as the “flight controller” or “autopilot”. The autopilot is an integrated part of the aircraft, so it is something that should be considered from the beginning of the aircraft’s design. Ardupilot and PX4 are the main two firmware stacks used in the hobby, both of which are open source, free to download, have a wide support community, and allow for broad aircraft control and customization. They feature a flight stack to be flashed to a compatible flight controller (Ardupilot, PX4) and a downloadable ground control station (GCS) software. Once flashed, the user can handle all initial setup and custom tuning within that GCS. Both firmware stacks have instructional documentation that can be found looking through the websites, and the purpose of this document is to offer some clarity about the system as a whole and to offer the necessary links in a single location.

Once the aircraft is functionally flying, it is relatively simple to operate waypoint missions with the necessary hardware mentioned below and some knowledge of the aircraft’s stall speed from testing or aerodynamic analysis to input the proper parameters from the GCS. Both Mission Planner and QGroundControl offer “point and click” waypoint creation, allowing the user to pre-plan waypoint missions. Once the aircraft is in the take-off position, the user can instruct the aircraft to begin the waypoint mission from the GCS, and the aircraft will take off to a designated altitude, follow the waypoint mission as instructed, and land the aircraft at a designated landing location. A functional waypoint mission can be completed with the hardware below (along with the other basic hardware necessary for any RC plane or multicopter), but both autopilots offer many options for adding additional sensors and peripherals to increase the aircraft’s awareness and precision.

Hardware required for simple waypoint mission:

- Compatible flight controller (PX4, Ardupilot)
- GPS
- Telemetry
- Airspeed sensor

Example hardware cost:

- Compatible flight controller + GPS + PDB combo:
    - Holybro Pixhawk 4 Autopilot + Neo-M8N GPS + PM07 Combo
        - $220
- Telemetry
    - Holybro 500mW Transceiver Telemetry Radio Set V3 (915Mhz)
        - $45
- Airspeed sensor
    - Holybro Air Speed Sensor
        - $55
- Total for simple autonomy: $320

Airframe Configuration

A fundamental part of the autopilot is setting up the aircraft configuration, and both firmware stacks mentioned here (PX4 & Ardupilot) have a library of stock airframes that can be used. If an airframe that is not available is chosen, there are ways to "hack" the setup to make it work, or the user can look into the developer side of the firmware stacks to implement custom code for the aircraft. By exploring the user guides and other parts of the websites linked, you can find lots of general and specific information to learn about the complete architecture of the firmware and the aircraft. Although successful flight is never guaranteed to be straightforward, there is more risk associated with the two custom options.

If going with a stock airframe (Ardupilot & PX4 references), there are direct set-up guides that can be found online here for PX4 and here for Ardupilot. If the aircraft being designed is similar to a stock airframe, that airframe's firmware could still be used by "hacking" the hardware. For example, say the desired aircraft is a "Lift + Cruise" aircraft with eight VTOL motors but none are stacked. Referencing the PX4 airframes reference from above, the Standard VTOL airframe could be used but rather than having an output from the autopilot for all eight motors, the user could use four output signals that are y-split such that the motors in each corner are a pair receiving the same signal. As far as the autopilot knows, there are only four motors but the control strategy is similar enough that this approach can work with proper motor rotation directions and PID tuning. This limits the capabilities of the octocopter since each motor isn't being controlled individually, but it could achieve the functionality the user is trying to obtain. This "hacking" approach can work, but it should be considered a last resort. Extra caution must be taken in testing to ensure that everything is operating safely and risk mitigation should be a priority at all times.

If attempting to create a custom airframe, the user is taking on a much larger workload for firmware development. Additionally, custom firmware must be compiled in order to be flashed to the autopilot, so it is recommended to have a Linux operating system to reduce compatibility issues. Implementing custom firmware certainly can be done, but this is typically avoided except for long-term projects due to the amount of time it takes to overcome the learning curve for becoming familiar with the flight stack as a developer and for troubleshooting/testing the custom firmware to work out any bugs in the code.

Ground Control Station (GCS)

The GCS is the laptop, tablet, or other device that is running the software that shows flight data, allows the user to create waypoint missions, and can give the aircraft in-flight commands when operating autonomously. In this document, the GCS will refer to that software. The common GCS used for Ardupilot is called Mission Planner, and the common GCS used for PX4 is called QGroundControl. On initial setup, the GCS will walk the user through selecting an airframe, calibrating the sensors, and from there the user has options for customizing the setup for their specific needs. This typically includes setting up channel mapping for the radio controller, setting motor number and spin directions, setting up fail-safes, tuning PID values during testing, setting servo ranges, viewing in-flight data, planning waypoint missions (Figures 1, 2), and viewing flight logs.

Figure 1: Mission Planner waypoint view

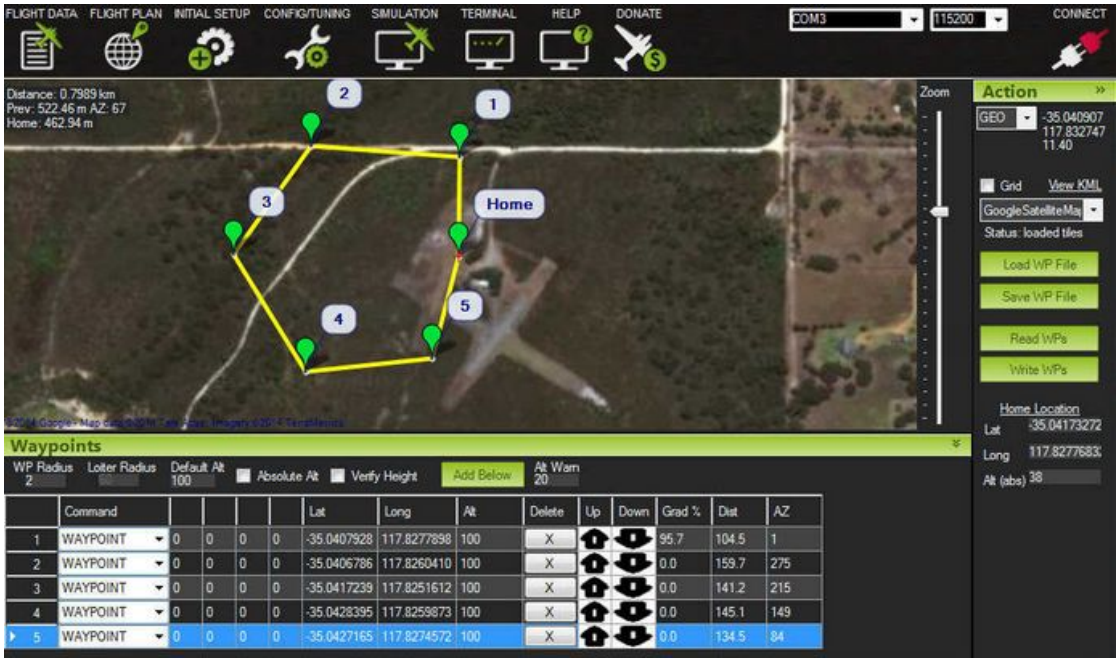
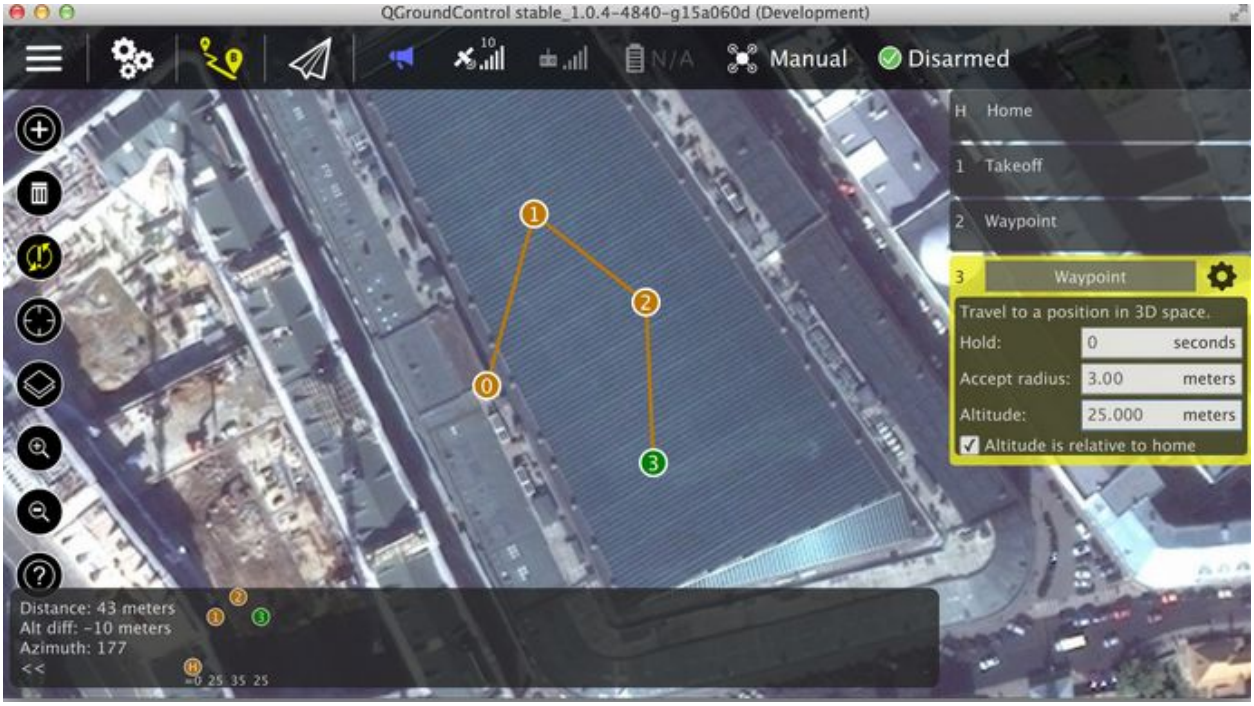


Figure 2: QGroundControl waypoint view



Basic Setup

Once the GCS is installed, the user can follow the setup for whichever firmware they are going with. Here are the first-time setup guides for PX4/QGroundControl and Ardupilot/Mission Planner. The general steps are to upload the firmware to the autopilot, select the vehicle type/airframe (Figures 3, 4), calibrate the sensors in the autopilot, wire the autopilot, and set up the channels for the radio transmitter. Within PX4 Basic Assembly and Ardupilot Wiring reference, the user can find guides on how to wire the autopilot and its various inputs and outputs.

For the radio transmitter setup, the user will calibrate the joysticks, assign any additional channels to the desired switches, and also set up the fail-safes. Fail-safes are put in place to determine what conditions you can safely fly under, and trigger an action if those conditions are not met. The two most common fail-safes that are put in place are in the case of RC loss and low battery. There are several options for the action taken if a fail-safe is triggered, which vary depending on which firmware is being used. Some common fail safe actions include a warning, aircraft loiter/position hold, or "return to home". For firmware specific guidelines, check here for PX4 and here for Ardupilot.

Once everything is wired and calibrated, the user can begin bench testing the aircraft with propellers removed to ensure functionality, and use these tests to begin tuning the aircraft. This can include, but is not limited to, checking motor spin direction, ensuring servo alignment for control surfaces, limiting servo range, adjusting the throttle curve, checking radio transmitter switch functionality, and carefully moving the aircraft while in the desired flight modes to ensure proper autopilot reactions for stabilization.

After all systems are working properly, the next recommended step is to begin tethered testing with proper test procedures in a safe environment to begin tuning VTOL PID gains. Once stable hover is achieved, the user can then move on to other tests such as fixed-wing flight, waypoint missions in VTOL or fixed-wing flight, manual transition, and waypoint missions operating through the entire flight envelope (includes transition).

Figure 3: Ardupilot/Mission Planner airframe selection

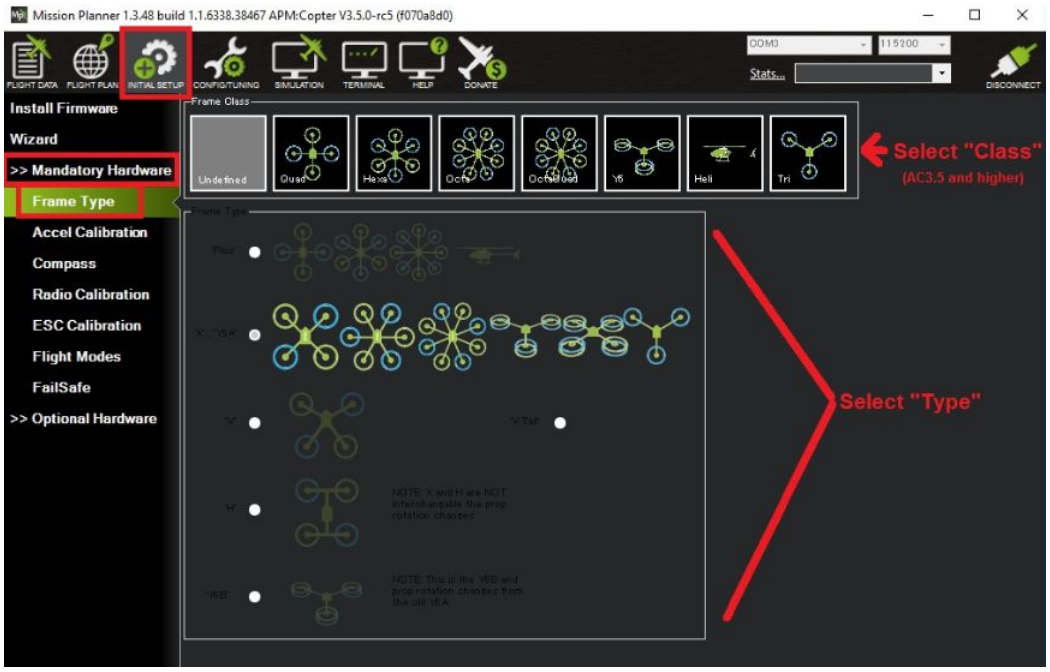
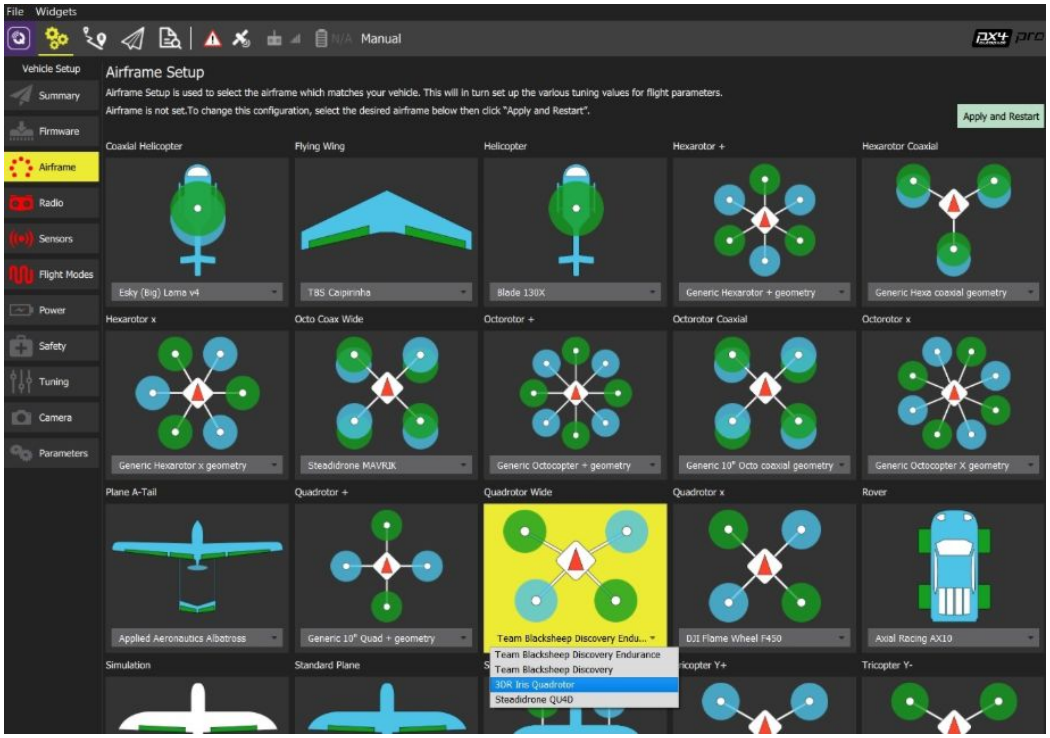


Figure 4: PX4/QGroundControl airframe selection



Additional Peripherals

To continue increasing aircraft functionality and the autopilot's awareness of the environment, additional peripherals can be added to the system. Here are links to the Ardupilot and PX4 listed peripheral hardware. Common uses for the peripherals include distance sensors for object avoidance and various proximity functions, battery monitors to monitor aircraft energy storage, and cameras for pilot first-person-view or image processing.