# Overview

This document serves as a build guide for a drone used to perform autonomous pickup and dropoff of a small payload as described in our [senior thesis](https://drive.google.com/file/d/1eyzDFhopvY5F14Urc1uSDVqPYYfMpOdw/view?usp=drive_link). The document will provide more information on the parts and assembly of the drone as well as the software setup to supplement the information presented in the thesis.

# Parts list

The following list contains all the off-the-shelf parts for the drone that we purchased:

* [Holybro PX4 X500 V2 development kit (with Pixhawk 6X and 915 MHz radio)](https://holybro.com/products/px4-development-kit-x500-v2?variant=43018371629245)
* [8 GB Raspberry Pi 4](https://vilros.com/products/raspberry-pi-4-model-b-8gb-ram?src=raspberrypi)
* [Power distribution board (used to power RPi and servo motors)](https://www.amazon.com/Distribution-FCHUB-12S-Supports-Regulators-Current/dp/B07MHKTF7F/)
* [14.8 V LiPo battery](https://www.amazon.com/HRB-5000mAh-Connector-Airplane-Helicopter/dp/B06XK8WWX1/ref=sr_1_5?crid=36WIN4IJF2ZXF&keywords=4s+5000+mah+lipo+battery+xt60&qid=1695248981&sprefix=4s+5000+mah+%2Caps%2C894&sr=8-5&ufe=app_do%3Aamzn1.fos.006c50ae-5d4c-4777-9bc0-4513d670b6bc)
* [Servo motors](https://futabausa.com/product/rs303mr/)
* [Radiomaster Controller (4in1)](https://holybro.myshopify.com/products/radiomaster-tx16s)
* [Radiomaster receiver](https://holybro.myshopify.com/products/radiomaster-r81-receiver)
* [Ark Flow optical flow sensor](https://arkelectron.com/product/ark-flow/)
* [Arducam Wide Angle Fixed-Focus Camera Module 3](https://www.arducam.com/product/12mp-imx708-usb-uvc-102-wide-angle-fixed-focus-camera-module-3/)
* [GPIO pin cables](https://www.amazon.com/dp/B01EV70C78?psc=1&ref=ppx_yo2ov_dt_b_product_details)
* [Screw terminals](https://www.amazon.com/dp/B07NSJV6NW?psc=1&ref=ppx_yo2ov_dt_b_product_details)
* [XT30 plugs](https://www.amazon.com/gp/product/B0BKK5BGCR/ref=ppx_yo_dt_b_asin_title_o01_s00?ie=UTF8&psc=1)
* [OAK D S2 RGBD Camera](https://shop.luxonis.com/products/oak-d-s2)\*
* [Intel RealSense d435i RGBD Camera](https://www.intelrealsense.com/depth-camera-d435i/)\*

\*Note: originally, we chose the RealSense camera to be our front facing camera. The RealSense has a dedicated [software suite](https://www.intelrealsense.com/sdk-2/) for reading in information; however, trying to install this suite from the link on the Raspberry Pi is not possible. Instead, we had to follow [these steps found in a GitHub thread](https://github.com/IntelRealSense/librealsense/issues/11506#issuecomment-1599168850) to build from source. The OAK D S2 does not have this issue, but we found that it is slightly slower than the RealSense. If you decide to use the RealSense, HolyBro sells a [depth camera mount](https://holybro.com/products/depth-camera-mount?variant=42724474454205) that you can use to place the camera on the drone (this mount also has space to mount an optical flow sensor underneath it). If not, you will have to make your own mount, although we provide CAD files later in this build guide for some that we made.

# Build Guide

The X500 V2 kit that formed the base of the drone ships with instructions for assembling it. For more visual aids, PX4 contains a [step-by-step tutorial with pictures](https://docs.px4.io/main/en/frames_multicopter/holybro_x500v2_pixhawk6c.html#holybro-x500-v2-pixhawk-6c-px4-dev-kit)  (note that this guide is for a kit that uses the Pixhawk 6C, but the assembly instructions themselves are still the same regardless of if you use the 6C or 6X).

Once this base was built, we modified the drone by adding the 3D printed parts described in the next section.

In addition, we had to do some soldering to connect the power distribution board for the RPi and servos connected to the power distribution board for the drone. To do this, we soldered an XT30 male connector to the 8-60 V DC IN port on the distribution board. We then soldered a screw terminal on each of the GND and 5 V pairs on the board - one of these holds the pair of wires that powers the Raspberry Pi, and the other holds the pair that powers the servo motor (see notes on the potential improvements section). Once this is done, it can be mounted onto a 3D printed mount (CAD included in next section).

Once the Raspberry Pi is mounted onto the drone (using the mount that is described in the next section), it can be connected to the Pixhawk 6X in multiple ways as described on the [PX4 documentation](https://docs.px4.io/main/en/companion_computer/pixhawk_companion.html) - ultimately, we used the [Ethernet setup](https://docs.px4.io/main/en/advanced_config/ethernet_setup.html) (see more details in ROS2 bridge section).

# CAD for 3D printed parts

There are four CAD printed parts needed for assembling this drone: the mantis claw, RGB Camera Mount, Crossbar Mount for Claw and Servo Motor, and Depth Camera and Optical Flow Mount. All 3D printed parts are made of PLA with 50% infills.

Link to Claw STL: [Files](https://drive.google.com/drive/folders/1if9A6Ysn8pVyYp9WaMxc0N_ZXBhwunZz?usp=drive_link)

Links to Mounts: [Files](https://drive.google.com/drive/folders/1IK8oUzaypBWdTEbn7lTK5G9DpVHtkovL?usp=drive_link)

1. The Claw: The mantis claw is composed of several different STL files, including claw pins, bolt cap, bolt, bolt nut, bot base, gripper claws, top bot arm, bottom bot arm, arm joints, and gripper claw joints. For assembly, you need to 3D print 40 claw pins (redundancy included in number), 1 bolt cap, 1 bolt, 1 bolt nut, 2 bot base, 10 gripper claws, 5 top bot arm, 5 bottom bot arm, 10 arm joints, and 5 gripper claw joints. A tutorial to make the mantis claw can be found here: [Tutorial](https://www.youtube.com/watch?v=yaddRVDZMao). When printing the STL files, it is important to scale each part down to 70% of its original size in order to take into account the limited space available underneath the drone body. Credit to Mantis Engineering Team for the claw idea.
2. RGB Camera Mount: The RGB camera mount can simply be downloaded from the CAD file and can be printed as is. The RGB Camera Mount will attach to the Crossbar Mount via a friction fit indent already added to the CAD files.
3. Crossbar Mount for Claw and Servo Motor: The Crossbar Mount for Claw and Servo Motor can simply be downloaded from the CAD file and can be printed as is; it is snapped onto the drone landing gears via a friction fit.
4. Depth Camera and Optical Flow Mount: The Depth Camera and Optical Flow Mount can simply be downloaded from the CAD file and can be printed as is; it is attached to the Platform board mounting rods on the drone body via a friction fit.
5. Raspberry Pi Mount: This mount can be attached to the payload board of the drone using 4 M2 screws and nuts (see [reference picture](https://docs.px4.io/main/assets/payloads_x500v2.BlKS32-f.png)).

Once these are printed, the physical components can be added to their respective location. The Servo motor and Arducam RGB Camera can be fitted into their respective mounts via a friction fit. The OAK D S2 Depth Camera and Ark Flow optical flow sensor must be attached to their respective locations via screws (M4 screw and M2 screw, respectively). The claw can be attached via adhesive to the bottom of the crossbar mount. The next step would be to connect each claw to the servo motor using nylon strings. Each nylon string was attached from the five gripper arms to a spool on a servo motor. In the servo default position, the strings are taut, but they are fastened in locations where an object would still be able to push on the bottom part of each leg, allowing it to get captured. The claws should be taut enough that when the servo is activated to rotate 90 degrees, the claw arms move upward. See Section 3 of the GRASP Senior Thesis for more specifics about the physical assembly, which can be found here: [File](https://drive.google.com/file/d/1eyzDFhopvY5F14Urc1uSDVqPYYfMpOdw/view?usp=drive_link).

# 

# Setup on QGC

QGroundControl (QGC) is a ground control station, which can be used to calibrate various drone sensors and modify parameters that impact how the drone flies. More information is available on the [PX4 documentation](https://docs.px4.io/main/en/getting_started/px4_basic_concepts.html#ground-control-stations).

To set up and calibrate the Pixhawk flight controller on QGC, we followed the steps on the PX4 documentation pages for [installing PX4](https://docs.px4.io/main/en/config/firmware.html) and on the [QGroundControl Website](https://docs.qgroundcontrol.com/master/en/qgc-user-guide/setup_view/setup_view.html).

Additionally, we had to change some parameters on QGC to allow the drone to use the Ark Flow sensor for state estimation. These steps are described [here.](https://docs.px4.io/main/en/dronecan/ark_flow.html#ark-flow)

Besides any changes in QGC parameters performed in the previous two steps we also found that changing the following parameters helped us fly the drone:

* EKF2\_RNG\_CTRL: set this to enabled so that the range sensor on the Ark Flow is used for altitude estimation as well. Before setting this parameter, altitude estimation was only done using the barometer on the Pixhawk, which gave values that did not make sense (i.e., negative values).
* EKF2\_HGT\_REF: set this to Range sensor for the same reason as above (using the Ark Flow for altitude estimation).
* COM\_OBL\_RC\_ACT: set this to Land mode so that if your drone loses connection to the offboard computer while it is flying in offboard mode it automatically lands.

# 

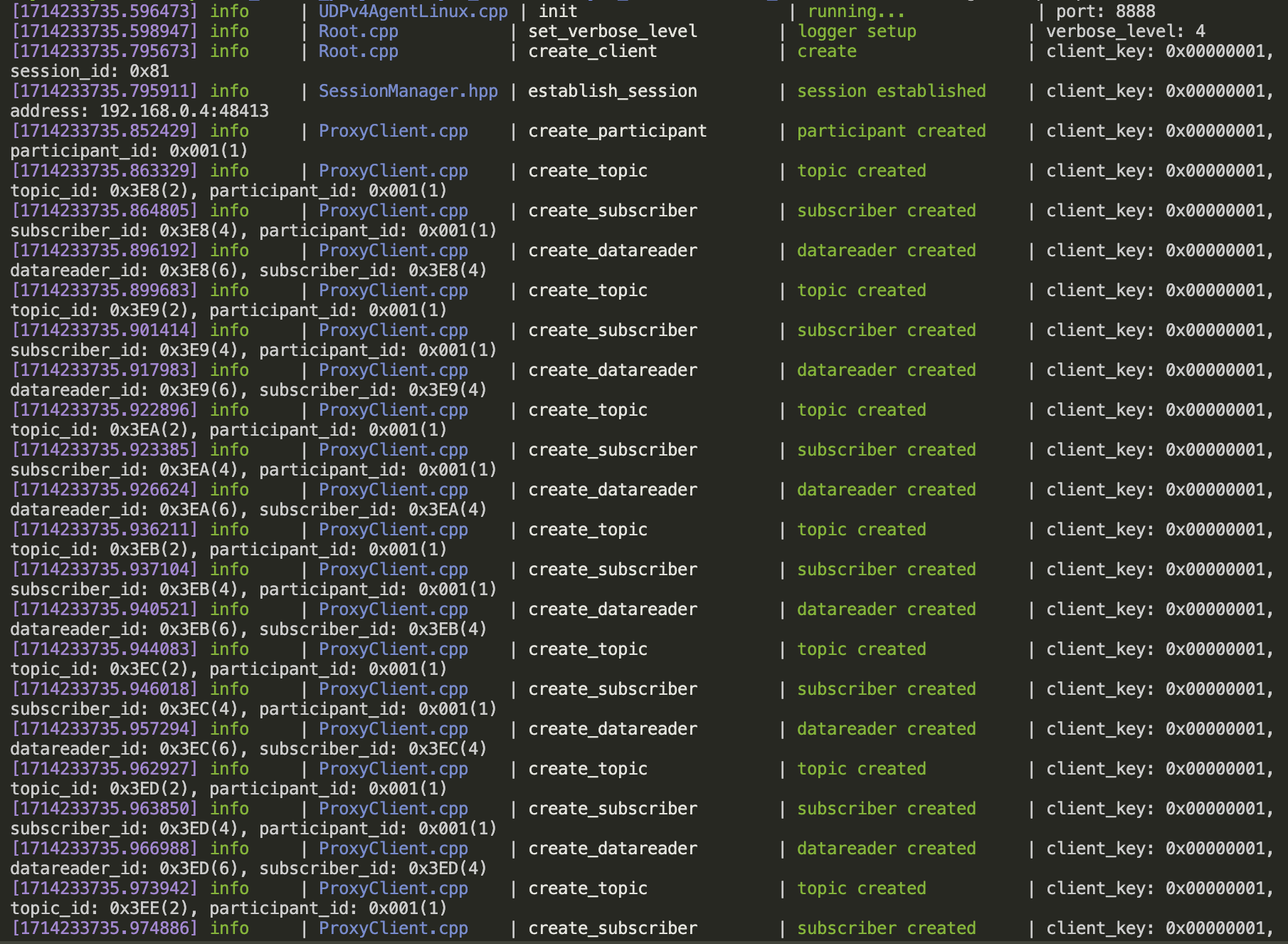
# ROS2 bridge + code

For offboard control of the drone, we chose to use the ROS2 bridge. Instructions for setting this up on the Raspberry Pi are available on the [PX4 website](https://docs.px4.io/main/en/ros/ros2_comm.html). Since the Pixhawk 6X comes with an Ethernet cable, we also followed the instructions [for the companion computer Ethernet setup.](https://docs.px4.io/main/en/advanced_config/ethernet_setup.html)

After following these instructions, you should be able to connect the Raspberry Pi with the Pixhawk 6X by running the following commands in a terminal on the RPi:

| sudo ifconfig eth0 192.168.0.1 ### this sets the IP address of the RPi Ethernet port to the needed value, need to do this every time MicroXRCE udp4 -p 8888 ### start the client that connects the two computers |
| --- |

If the connection is successful, you should see the following output in the terminal where you ran these commands:



However, this connection can be very finicky (at least for us it was), and may sometimes not work on the first try, or may even disconnect in the middle of flying and testing the drone. If this happens, we found that simply running the commands a few more times until the connection initiates solves the problem. In rare situations, the connection would start up, but whenever we tried running our code to fly the drone, nothing would happen. When this happened, we found that running the [offboard control example from PX4](https://docs.px4.io/main/en/ros/ros2_offboard_control.html) first would solve the problem.

For control scripts, the GitHub repo this build guide is in contains the exact code used to control the drone, along with standalone Python scripts and Python scripts written for execution as a ROS2 node that were used for testing camera components. The main control scripts themselves are strongly based on [PX4’s offboard control example](https://docs.px4.io/main/en/ros/ros2_offboard_control.html). In addition, we developed nodes that publish image information from the front-facing RGBD camera and bottom-facing RGB camera. Since we used [AprilTags](https://april.eecs.umich.edu/software/apriltag) for object and destination localization, the topics that these nodes carry information about an AprilTag’s distance to a camera (for the RGBD only) and location within an image (both cameras). For ease of use, we also have provided several launch files that launch different combinations of the offboard control, bottom-facing camera, and front-facing camera nodes.

Finally, to control the servo motor that was used to open/close the claw, we wrote scripts located in the [servo control folder](https://github.com/balfaroo/GRASP-Drone-Project/tree/main/offboard/offboard/servo_control) of the GitHub repo, which were developed following the instructions [here](https://www.digikey.com/en/maker/tutorials/2021/how-to-control-servo-motors-with-a-raspberry-pi). These require admin access to run, and must therefore be run with sudo <file name>.py

# Tips for use + potential improvements

In this final section we want to give some pointers for making the drone easier to use and ways to improve the goal of this project.

1. Consider the lighting conditions when you test the drone. When we were performing our initial tests, we found that the drone would sometimes drift horizontally when it was supposed to hover in place. At first, we suspected that this was because the optical flow sensor was too close to the claw, and may have detected some of the natural movement of the individual claw legs that happens during flight, potentially leading it to erroneously detect horizontal movement. This caused us to shift the position of the sensor from behind the depth camera (as in the [Holybro depth camera mount](https://holybro.com/products/depth-camera-mount?variant=42724474454205)) to directly underneath it. This caused us to mount the depth camera upside down to make room for the wire that connects to it (however, since we were not using the IMU on the camera it wouldn’t have a practical effect except needing to consider some negative signs when programming the drone). After further testing though, we found that the drift would only happen at night whenever the lights directly above the drone were on. By turning off these lights so that the testing area was only lit by lights that are adjacent to it, we were able to get rid of this drifting area.
2. You may notice that sometimes the motors of the drone do not begin spinning simultaneously when taking off. Usually this is a one-time issue that fixes itself if you stop the program running on the drone’s companion computer and restart it; however, if it persists, a likely solution is that you need to recalibrate the ESCs of the drone, which and be done on QGroundControl. Similarly, there may occasionally be a motor that “jitters” when flying, we were also able to fix this by recalibrating the ESCs.
3. For better performance of the grasping functionality for the drone, it may be worth experimenting with the following options:
   1. Consider changing the position of the bottom-facing camera to be directly inside the claw as opposed to being horizontally-offset.
   2. Similar to the above point, consider a new passive claw design that gives you more space to place the camera inside the claw, or
   3. Consider switching the design from a passive claw to an active claw.
4. Because of the need to run the servo motor control scripts with admin access, we were not able to incorporate this aspect into the ROS2 nodes, and instead executed the scripts manually based on output printed to the terminal from the offboard control nodes. This is because the nodes could not be run using the sudo command, since this would result in errors on the terminal we did not know how to work around. A potential solution to this would be to control the servo using the flight controller instead of the Raspberry Pi, since the 6X has auxiliary PWM ports that can be used for this purpose and can be set on QGroundControl. This is the initial method we wanted to use, and it seems that this would allow the servo to be controlled with ROS2 using the [ActuatorServos](https://docs.px4.io/v1.14/en/msg_docs/ActuatorServos.html#actuatorservos-uorb-message) message, but we weren’t able to find a documented use of this and were unsure how to incorporate it. However, this would be a more elegant solution and is worth looking into.

# 