The Duckiedrone Operation Manual

Contents

Preliminaries

Hardware build DD21

Drone handling

Using the drone

Learning Experiences

Software Architecture

Welcome to the Operation Manual for your shiny new Duckiedrone.

In this manual you will learn how to build your Duckiedrone, test it and have an introduction to its inner workings. Are your duckies ready to fly?

Safety

> ⚠ **Read carefully**
>
> Building your Duckiedrone can be lots of fun, however you'll be handling some tools which can be dangerous if used improperly. **Read carefully** this page to know how to handle them correctly to minimize risk!

Soldering

A majority of the drone build involves soldering. Soldering is fun and safe when done with proper care; however, the soldering iron is very hot, and is a significant burn hazard. By-products of soldering are also dangerous: the fumes produced from the flux evaporating, as well as handling the solder itself (it may contain lead, a heavy metal).

> ⚠ **Danger**
>
> Be sure to follow the following safety guidelines whenever soldering:
>
> - Solder in a **well ventilated** space and use a fume extractor to suck the fumes away from you while working
> - Always **wear safety goggles** when soldering to prevent solder from splashing into your eyes, and to help keep the solder fumes away from your eyes
> - Make sure the soldering iron stand is sturdy (the soldering iron has to sit in the stand without falling over) before plugging in the soldering iron
> - **Never** touch the metal part of the soldering iron after it has been plugged in (you'll get burned)
> - If you are feeling light-headed after soldering for extended periods of time, take a break and get some fresh air
> - Wash your hands thoroughly with soap after soldering as the solder may contain lead

Battery

Lithium Polymer (LiPo) batteries are very commonly used in RC devices (as well as many cell phones!) However, they must be handled properly. Because they store energy, they can start a fire or explode if handled incorrectly. Please [read this article](#) to learn more about the proper care of your drone's LiPo battery.

> ⚠ **Danger**
>
> - **Never** leave a LiPo battery charging unattended.
> - If the battery or charger starts smoking when you plug in the battery to charge, unplug it immediately and get a new battery and charger (this happened once as a result of a faulty charger, but the battery was fine).
> - Never overcharge the battery
> - Always have a fire extinguisher nearby in case of fire. - If one is not readily available, drenching the battery and surrounding area with water will work.
> - Always inspect your battery for signs of damage such as punctures or if it is puffy. If there are any signs of damage to your battery, properly [dispose of it following this article](#).
> - It is recommended that you add the battery in the drone before screwing on the second level of the frame.

Flying

The main sources of danger when flying are the spinning propellers. The propellers can cause serious injury if they come into contact with a person, and can cause serious damage if they come into contact with things. Another source of danger is a propeller breaking after hitting something, and then flying off of the drone. Also, if your propellers are on the wrong motors, they could fly off of the motor and also cause serious damage. For these reason, it is important to follow these guidelines:

Fig. 1 RED: **counter clockwise** propellers

BLACK: **clockwise** propellers/motors

Prerequisites

This section contains information about the most important skills used in the build.

⚠️ **Warning**

Whether you're new to soldering or not, we recommend reading through this page to review the basic techniques used in the build.

Strip Wires

Stripping is the process of removing a portion of an insulator from a wire in order to expose its strands. It is done by using a wire stripper. The exposed wire is then able to be tinned and soldered.

Tin

Tinning is the process of applying solder to exposed wire or metal pad. It is done by using a soldering iron to heat up the metal, and then solder melts into the wire or onto the pad. The purpose of Tinning is to make the soldering process easier.

Tinning a Wire

Watch the following tutorial to learn how to tin and join two wires.

> **ⓘ Note**
>
> Sometimes parts will have wires already tinned out-of-the-box by the manufacturer (i.e. pre-tinned). You can identify this by:
>
> 1. The "shininess" of the tip of a wire
> 2. The inability to fray the wire strands of the tip of a wire.
>
> However, such tinning is often ineffective. Cut off any pre-tinned tips, then strip and tin the part yourself.

Soldering

Soldering is the process of joining two metal components by melting an alloy; namely, solder. Since solder is conductive, the resulting joint acts as a bridge for electricity traveling between the two metal components.

Below, there are brief resources on soldering, please review them before starting the build.

> **⚠ Danger**
>
> - Be careful holding wires and components with your bare hands while soldering, as **they will get very hot very quickly**. We recommend using long-nose pliers or helping hands whenever possible.
> - Don't touch the soldering iron tip (or any other metal piece) while the soldering iron is on, since doing so can cause burns. If you get burned, rinse the affected area with cold water immediately.
> - Likewise, don't use the soldering iron on anything you don't intend to solder. The high heat will cause things to melt/burn.
> - Don't breathe soldering fumes; use a soldering fan whenever possible.

Resources

For a quick overview of soldering, watch this beginner soldering tutorial YouTube video. For a more in-depth introduction, review this article.

Troubleshooting

Troubleshooting	
SYMPTOM	I accidentally cut off too much wire while stripping.
RESOLUTION	Grab another wire of the same color, then strip and tin it on one end. Solder this new wire to your original wire. Cover the solder joint with either a heat shrink or electrical tape.

Troubleshooting

SYMPTOM I cut off several strands of wire while stripping.

RESOLUTION If you cut off just a couple of strands, then the wire is probably still safe to use. If you cut off a large percentage of strands, then you will need to get a new wire.

Troubleshooting

SYMPTOM I put too much solder on my wire while tinning.

RESOLUTION The easiest way to remove excess solder is to use a solder sucker or copper wick. Alternatively, excess solder can be removed by carefully picking up the excess with a soldering iron, then cleaning the soldering iron with soldering wool. Repeat as needed

Troubleshooting

SYMPTOM My wire is picking up random particles while tinning

RESOLUTION Clean the tip of your soldering iron with soldering wool. For future prevention, do this cleaning more frequently while soldering

Troubleshooting

SYMPTOM The alligator clips of my helping hands are loose.

RESOLUTION Remove the offending alligator clip from the helping hands, then use pliers to carefully pinch the end tighter. Re-insert once fit is tight

Troubleshooting

SYMPTOM I burned some insulator onto my wire while tinning.

RESOLUTION A little bit of insulator burn is probably fine. If a lot has been burned, use a wire cutter to carefully cut off the burned parts. If that fails, you will need another wire

Troubleshooting

SYMPTOM I can't tell if I tinned my wire properly.

RESOLUTION Use a wire cutter to cut off the tip of the tinned wire. Visually inspect the core. If solder is not in the core or if the wire strands can be spread with your fingers, then the wire is not tinned properly

Troubleshooting

SYMPTOM My solder keeps melting into a sphere shape instead of melting onto a wire.

RESOLUTION This happens because the solder is not getting hot enough to fully melt. This could be a consequence of:

1. the soldering iron tip has reached the end of its life
2. the solder has expired
3. the soldering iron station no longer works properly.
4. you are not using rosin core solder, try painting some soldering flux on the wires or try different solder.

How to Use Multimeters

During this step you will learn about how to use multimeters to do a continuity check and a voltage check.

A multimeter or a multitester, also known as a *VOM* (volt-ohm-milliammeter), is an electronic measuring instrument that combines several measurement functions in one unit. A typical multimeter can measure voltage, current, and resistance.

> **→ See also**
>
> This is a general tutorial for multimeters.

> **❗ Attention**
>
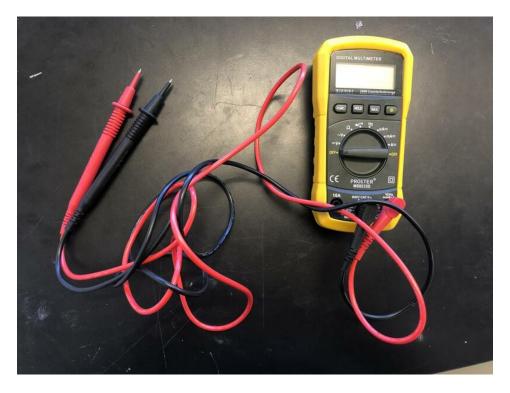> Please turn off the multimeter by setting the dial to `OFF` after you finish your check.

Fig. 2 Multimeter

Continuity Check

In electronics, a continuity check is a test of the resistance between any two points of a circuit (that it is in fact a complete circuit). If there is zero resistance between two points, then there is a **short** between the points. Shorts are potentially dangerous because they may cause far too much current to flow throughout the circuit - thus resulting in the circuit frying due to heat generated by Joule effect.
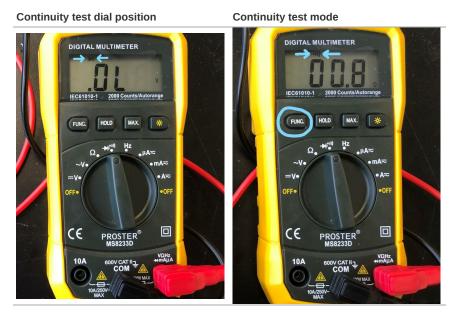
> **ⓘ Note**
>
> There might be slight differences with your multimeter model, but the general instructions should be very similar.

Performing a continuity check is a safe way to debug if a circuit has an undesired short because the check does not require a power source to be connected to the circuit.

1. Select the Continuity Function
   - Turn the multimeter dial to the continuity test position. Then press the `"FUNC."` button to switch to the continuity test mode (indicated by an icon that looks like a sound wave).

Continuity test dial position	Continuity test mode

   - Test the continuity test mode by touching and holding the multimeter leads together. A continuous beep will be audible for as long as the leads are held together.
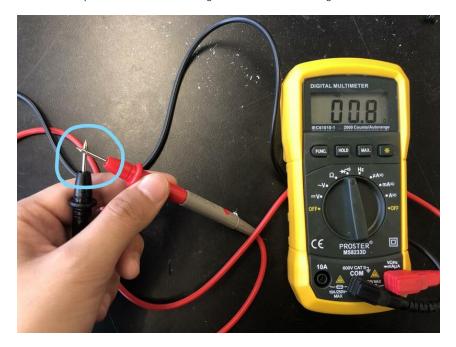

Fig. 3 Leads held together (a continuous beep is audible)

2. Perform the Continuity Check
   - Place each lead at a point of the circuit or component you want to test.
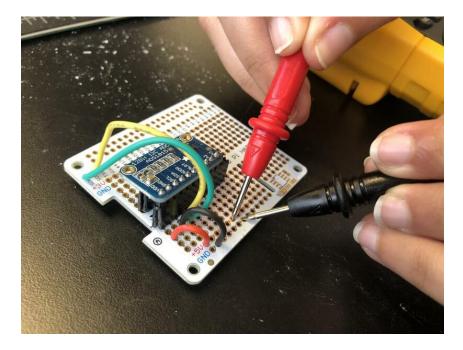
Fig. 4 Multimeter leads on breadboard

- If the path between the two points is continuous (i.e. is a short), then the screen will display a value of zero (or near zero) and the multimeter will emit a continuous beep for as long as the leads are held in place.

> **ℹ Note**
>
> If you hear a short beep followed by silence while the leads are held in place, then you can safely ignore the short beep.

> **💡 General Continuity Check Strategy**
>
> - Check every two positive (+) terminals to make sure every pair of these terminals is continuous.
> - Check every two negative (-) terminals to make sure every pair of these terminals is continuous.
> - Check every positive terminal (+) to make sure it is **not continuous** with any negative terminal ('-').

> **↪ See also**
>
>
>
> What is Continuity and How to Test for it With a Multimeter

DC Voltage Check

1. **Selecting the DC Voltage Mode**

   Switch on your multimeter, and set the dial to DC voltage mode (indicated by a v with a straight line -v, or the symbol ⎓).

Fig. 5 Switch the Dial to the DC Voltage Test Function

2. **Performing the Voltage Check**
   - Place the positive (i.e. red) lead on a positive (+) terminal, and the negative (i.e. black) lead on the negative (-) terminal.
   - See the screen for a voltage measurement.

> ℹ **Note**
>
> Reversing the leads (i.e. red on - and black on +) won't do any harm; it will simply give a negative reading of the same magnitude.
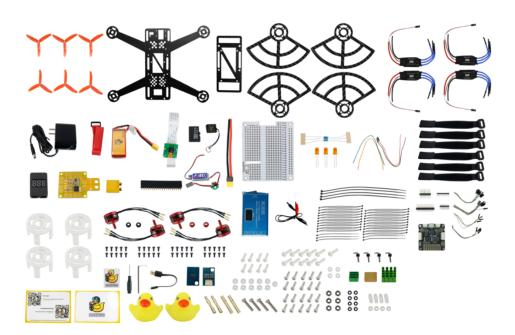

How to Use a Multimeter for Beginners - How to Measure V...

> ↪ **See also**
>
> For an introduction or review of circuit basics and Ohm's Law, $V = IR$, check out this [SparkFun article](#).

Included materials

Fig. 7 What's inside the box

In the picture there are all materials included in the Duckietown kit. A more detailed overview of each item is given in the *Overview* chapter of each assembly section.

Other Required Materials

Base Station (Laptop)

This section contains what you will need to build and fly your drone that is not included in the kit. In order to fly the drone, you must have a laptop or personal computer that is capable of connecting to the internet.

OS Requirements

At this time, it is not possible to flash your SD card on a Chromebook. You will need a Linux, Windows, or macOS device to configure the SD card during the build, but you can use any OS after this the configuration.

Software Requirements

Additionally, you will need to install the following software on your base station.

1. *Cleanflight Configurator*. This is an application that will allow you to set up your Flight Controller.

> **ⓘ Note**
>
> You will install this later in the manual.

2. Google Chrome is recommended
3. USB to UART driver. (**macOS and Windows only**) Download the correct driver for your OS. You will not need this driver if you are using a Chromebook or Linux, you will be able to connect to the flight controller without it.

Soldering Tools

Essential:

Fig. 8 Soldering iron with base, sponge and solder

- soldering iron
- solder
- brass wool or sponge to clean soldering iron

12

10

8

AWG
SOLID
6-32

AWG
STRANDED
45-6

IDEAL INDUSTRIES INC
MADE IN U.S.A.

8-32

14

12

10

DO NOT USE ON LIVE CIRCUITS

REFLEX T·-Stripper

Fig. 9 Soldering iron with base, sponge and solder

- wire strippers
- safety goggles
- fume extractor
- multimeter
- helping hands

Nice to have:

- solder remover (either solder sucker or de-soldering wick) to fix mistakes
- soldering mat
- tweezers or pliers
- flush cutters

Misc. Tools

- Double-sided tape

Software initialization

> ℹ **What you will need**
>
> - A computer (a.k.a. "base station") with an internet connection
> - Balena Etcher or similar program
> - A micro SD card (32GB, U3, Class 10), e.g., that from your Duckiebox
> - A micro SD card reader, e.g., that from your Duckiebox

> ℹ **What you will get**
>
> - A DD21 initialized and customized micro SD card

Flashing the SD card

In this section you will install the Duckiedrone software on the microSD card.

1. If you have not already, on a base station, download the image flashing tool [Etcher](#).
2. If you have not already, on a base station, download the latest drone image:

   [DD21 system image]

3. Connect the micro SD card to the base station. Use the micro SD to USB card reader if the base station does not have a micro SD port.

Fig. 10 Micro SD Card adapter

4. Open Etcher and select the downloaded drone image. Then select the micro SD card as the drive to flash. Finally, click the `"Flash"` button.

Watch this video to see how the process looks like.

We couldn't verify the security of your connection.

Access to this content has been restricted. Contact your internet service provider for help.

We couldn't verify the security of your connection.

Access to this content has been restricted. Contact your internet service provider for help.

> ⚠ **Warning**
>
> **Double check** that the "drive" is your micro SD card.
>
> You may be prompted to enter the base station password to proceed. This is normal: flashing an SD card deletes everything that is on it, so Etcher is making sure this process is OK with you.

> ⓘ **Note**
>
> Flashing will take 10 - 15 min.

Customizing the Duckiedrone `hostname` and client network settings

Changing the Duckiedrone `hostname` (also known as the "robot name") is needed to prevent conflicts when multiple Duckiedrones are operating in the same environment. If you intend to operate the drone in isolation from other Duckiedrones (e.g., at home), you can maintain the default settings.

1. To change your robot's `hostname` navigate to the newly flashed SD card.
   - You will have to unplug it from your base station first and plug it back in as Balena Etcher dismounts the drive after finishing the flashing process.
   - If the flashing is successful, you will see that it has one partition named `boot`, with many files inside and `overlays` folder. **Do not** manually alter these files.

Fig. 11 `boot` partition, **do not** alter!

2. There will be a second partition called `config`. Open this partition. You will find two files inside:
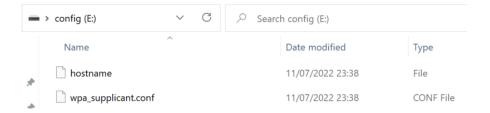   - `hostname`
   - `wpa_supplicant.conf`

Fig. 12 `config` partition content

3. Open the `hostname` file with any text editor program (e.g., Notepad on Windows) and replace the default robot name, `amelia`, with one of your choosing.

> ⚠️ **Warning**
>
> Make sure to follow the naming guidelines in the `attention` box above.

A wrong `hostname` will mean having to reflash the SD card and start from step 1.

> ❗ **Attention**
>
> **Save** the file before closing it.

4. Open the `wpa_supplicant.conf` with a text editor of your choice and either edit the default connection settings or duplicate them to add a new network. These parameters will be used only when booting the Duckiedrone in client-network mode, i.e. connecting to an existing Wi-Fi access point.

```
ctrl_interface=DIR=/var/run/wpa_supplicant GROUP=netdev
update_config=1
country=US

# NOTE: the following block is a template, use it to define connection to
custom wifi networks
network={
  id_str="network_1"
  ssid="duckietown"
  psk="quackquack"
  key_mgmt=WPA-PSK
}
```

- `country`: change it if you are not in the US (e.g., `CH` for Switzerland, `CA` for Canada; [full list](#) of country codes)
- `id_str`: an identifier for the network; change it if adding a new one;
- `SSID`: name of the Wi-Fi you want the Duckiedrone to connect to;
- `psk`: password for the above Wi-Fi;

You can add as many Wi-Fi settings as you want, e.g., for home, school, office, etc., by copying and pasting the first block.
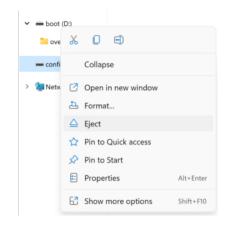
> ℹ️ **Note**
>
> This file can be edited after the first boot as well if you want to add other networks.

```
ctrl_interface=DIR=/var/run/wpa_supplicant GROUP=netdev
update_config=1
country=US

# NOTE: the following block is a template, use it to define connection to
custom wifi networks
network={
  id_str="network_1"
  ssid="duckietown"
  psk="quackquack"
  key_mgmt=WPA-PSK
}

network={
  id_str="network_2"
  ssid="example-second-network"
  psk="quackquack2"
  key_mgmt=WPA-PSK
}

network={
  id_str="network_3"
  ssid="example-third-network"
  psk="quackquack3"
  key_mgmt=WPA-PSK
}
```
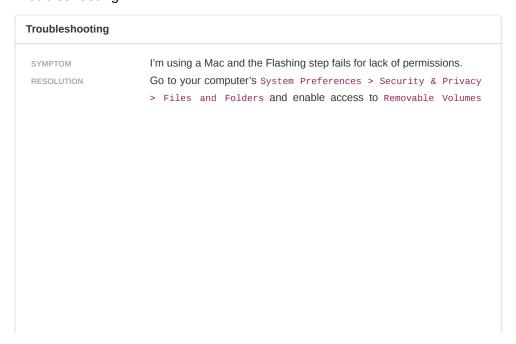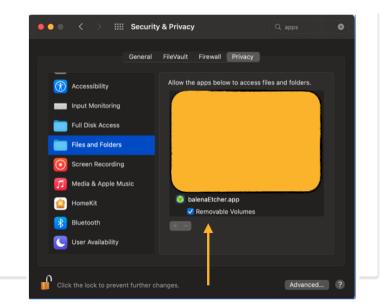
⚠ **Eject your SD card safely.**

Do not just unplug the SD crad from the base station

You are now ready for the first boot.

Troubleshooting

Troubleshooting	
SYMPTOM	I'm using a Mac and the Flashing step fails for lack of permissions.
RESOLUTION	Go to your computer's `System Preferences > Security & Privacy > Files and Folders` and enable access to `Removable Volumes`

Initializing the Flight Controller

> ℹ **What you will need**
>
> - A base station computer
> - Flight Controller
> - USB to micro USB cable

> ℹ **What you will get**
>
> - An up-to-date, initialized Flight Controller

The Flight Controller (FC) implements several low-level behaviors, e.g., stabilizing the Duckiedrone around roll, pitch, and yaw through three different PID controllers. Correctly configuring the Flight Controller is critical for flying safely.

Installing Cleanflight Configurator (CFC)

Cleanflight Configurator is an app that allows the base station to connect directly to the Flight Controller and access its configuration interface.

> ℹ **Note**
>
> Cleanflight Configurator used to be a Chrome App, however Chrome Apps' support has been dropped from Google so you will use the native app for your OS.

Steps:

1. Download the correct version of Cleanflight Configurator v2.4.0 for your OS from **this link**.
2. Install Cleanflight Configurator on your system
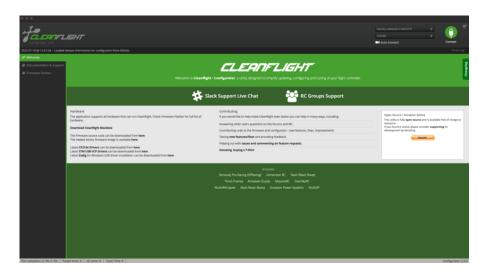3. Start Cleanflight Configurator on your base station. You should see the below interface.

Fig. 13 Cleanflight Configurator (CFC) `welcome screen`

Flashing the correct firmware

Updating the Flight Controller firmware (Flashing the Flight Controller)

Flight Controllers might have different versions of firmware (i.e., the software that runs on the Flight Controller microcontroller) out of the factory. Normally, this only needs to be done once initially. Follow this procedure to update your firmware.

> ⚠ **Attention**
>
> During flashing, do not press the `"Connect"` button in Cleanflight Configurator.

Our current target firmware is:

- **BTFL v3.3.3** (Download the `.hex` file below to your base station) [Download BTFL v3.3.3 here](#)

Prepare for flashing the Flight Controller firmware

> ⚠ **Attention**
>
> Regardless of the firmware version, if it is the first time setting up the Flight Controller, we recommend performing the below flashing procedure once anyways in order to start from a clean state.

Perform the following 2 steps with the Flight Controller **disconnected**.

- In the default Welcome page of Cleanflight Configurator, in the left sidebar, please click on the `Firmware Flasher` tab
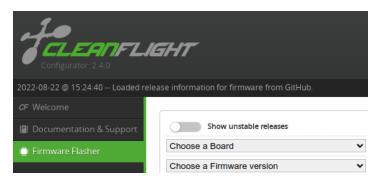

Fig. 14 `Firmware Flasher` tab in Cleanflight Configurator

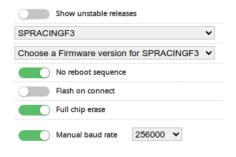- Configure the options in the tab as below:

Fig. 15 `Firmware Flasher` parameters to set

Different Flight Controller versions

There are currently 2 types of Flight Controller hardware. Use the steps corresponding to your hardware. Videos for both hardware versions are provided later in the document.

> ⚠ **Attention**
>
> If you have the `OSD` version of the Flight Controller you need to solder the micro USB adapter before being able to connect to the base station.
>
> **Choose in the following tabs which Flight Controller version you have**

> ↪ **See also**
>
> Identify your flight controller in **this chapter**.

OSD   ACRO

You will need to solder the pins to the microUSB board and connect it to the Flight Controller.

> ⓘ **what you'll need**
>
> - soldering tools
> - Flight Controller microUSB board
> - 90° header pins
> - 6 pins cable connector (in the Flight Controller bag)

> ⓘ **what you'll get**
>
> - `OSD` Flight Controller connected to the base station

1. Solder the 90° pins to the microUSB board as shown

Fig. 16 Pins soldered to the microUSB board

2. Connect the 6 pins cable that came with the Flight Controller to the microUSB board according to the following table (there's one unused `GND` pin on the board).

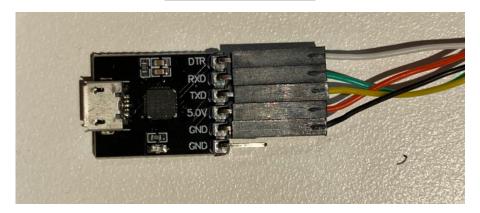| USB board pin	Cable color
GND	Black
5.0V	Red
TXD	Yellow
RXD	Green
DTR	White

Fig. 17 USB board pinout connection

> ⚠️ **Attention**
>
> **Once you verify the connection** between the base station and the Flight Controller, it is recommended to hot glue the pins in place.
>
>

3. Connect the white connector of the 6 pins cable to the Flight Controller port shown here.
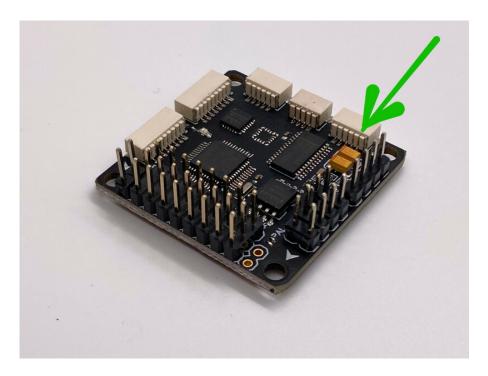
Fig. 18 Connect the white end of the 6 pin cable to this port on the Flight

Controller

Flashing the Flight Controller

The videos to flash the **ACRO** version and the **OSD** version of Flight Controllers are below in this section, in their respective tab.

A summary is provided here.

Most operations are the same with both versions, and any version-wise operations are written in the respective tab.

> ❗ **Attention**
>
> Normally, when connected to the base station, the Flight Controller connects in `Normal mode`.
>
> This can be identified by:
>
> - Having a red blinking LED on the Flight Controller board
>
> If this is the case during this section, unplug from base station and try to reactivate `bootloader mode` as detailed below.

To flash the firmware, we need the Flight Controller to be in `bootloader mode`.

Start by having the Flight Controller disconnected

OSD   ACRO

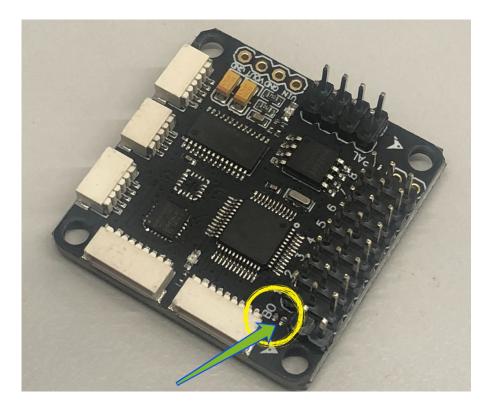1. Identify the `"BO"` pins on the board

Fig. 19 Location of the boot pins `"BO"` on the `OSD` Flight Controller

2. Use some conductive metal tool to short the `"BO"` while connecting the USB cable
3. Once connected, you can take the pin-shorting material/tool away

> ↪ **Check**
>
> there is no red blinking LED. There should be no LED on now on the Flight Controller (there is a red LED on the USB board).

4. The Flight Controller is now in `bootloader mode`

Now that the Flight Controller is in `bootloader mode` you can flash the correct firmware:

1. In the Cleanflight Configurator Firmware Flasher tab, click the `Load Firmware [Local]` button (bottom right), and select the `.hex` file downloaded at the beginning of this section.

> ↪ **Check**
>
> The progress bar should look like "`Loaded Local Firmware: (… bytes)`"

2. Click the Flash Firmware button (bottom right) and check the progress bar.

> ↪ **Check**
>
> The progress bar shows: "`Flashing…`" => "`Verifying…`" => "`Programming SUCCESSFUL`"

> 💡 **Tip**
>
> In case the progress bar turns red, see the Troubleshooting section below

3. If successful, without needing to reconnect the cable, the Flight Controller should go back to the `Normal mode`.

> ➤ **Check**
>
> > 1. Verify the red blinking LED is back on
> > 2. Click `"Connect"` and verify the firmware version is correct as detailed below

> ❗ **Attention**
>
> To see the whole process for your version of the Flight Controller choose the correct tab below:

OSD   **ACRO**

We couldn't verify the security of your connection.

Access to this content has been restricted. Contact your internet service provider for help.

Connecting to the Flight Controller

1. Unplug the battery from your drone

   > ❗ **Attention**
   >
   > Double-check the battery is unplugged

2. Connect the micro USB cable from your Flight Controller to your base station
3. Identify the `"Connect"` button in the top right corner (right now it will be green as the Flight Controller is not connected yet)

Fig. 21 `"Connect"` button in the top-right corner of Cleanflight Configurator

4. Select the correct connection port

5. Leave the baud rate (number) at the default value of `115200`
6. Press the `"Connect"` button and a `"Setup"` page should greet you with a rendering of your drone (see figure below).

Fig. 22 Cleanflight Configurator `"Setup"` page

Checking the firmware version

On the top left of the Cleanflight Configurator interface, one could check for the Firmware version. For example, in the figure below, the firmware version of the Flight Controller is `BTFL 3.3.3`.

Fig. 23 Top left of Cleanflight Configurator, check Cleanflight Configurator version and

Flight Controller firmware version here

Restoring the correct settings

We will restore the correct settings for the Flight Controller that reflect the setup we have on Duckiedrones (i.e. Flight Controller upside down, ESCs communication protocol, etc.)

The settings for the Flight Controller can be saved and restored through the CLI interface of Cleanflight, which is akin to a shell used to interact with the firmware.

We have created a file with the required setup for you, so you will only need to restore it without having to tweak parameters through the Cleanflight Configurator.

To do this:

1. Download this `.txt` configuration file.
2. Open it in the notepad app of your base station
3. Copy all content of the file from the notepad by simply clicking on the text and using `CTRL` + `A` or `CMD` + `A`
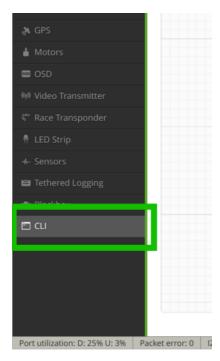4. Go in the `CLI` tab of Cleanflight Configurator

Fig. 24 CLI tab

5. Paste all the text you previously copied using `CTRL` + `V` or `CMD` + `V` in the text field at the bottom (the one with the text `Write your command here. Press Tab for AutoComplete`).
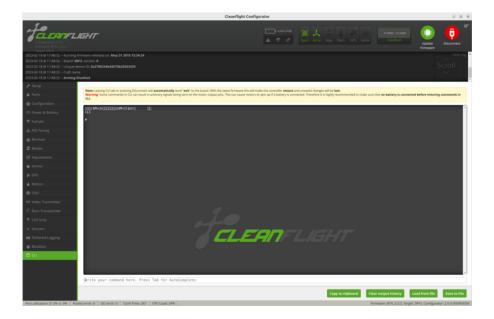
Fig. 25 CLI interface

6. Press `Enter` on your keyboard to execute the commands and wait for the shell to finish (it should take about 30 seconds)
7. Save the configuration by executing the command `save` in the CLI.

The Flight Controller will now reboot and reconnect.

> ↪ **Check**
>
> In the `setup` tab check that the drone now results facing up when the side of the Flight Controller with the chips faces downward.

Troubleshooting

Troubleshooting	
SYMPTOM	The `OSD` version of the Flight Controller does not enter `bootloader mode`
RESOLUTION	It is tricky to jump the exposed Boot pins. Try applying a bit of force when pressing on the pins. If possible, also try verifying they have actually been jumped with a multi-meter.

Troubleshooting	
SYMPTOM	The progress bar turns red and shows `"No response from the bootloader, programming:FAILED"`
RESOLUTION	It might happen when, on the Firmware Flasher tab, the Flash Firmware button is clicked. This is likely due to the Flight Controller not being in Bootloader mode. Please double check the indicators of that mode in the section above.

Troubleshooting	
SYMPTOM	`"Flashing..."` started, but progress bar turns red with a `"Timeout"` error
RESOLUTION	It might happen during "Flashing…" or "Verifying…". Click the `Flash`

`Firmware` button and try again

Raspberry Pi & Power Board

Overview

Preface

In this first part of the build, you will create a circuit to provide power to the Raspberry Pi from the battery. You will make the circuit connections using the soldering skills that you've gained from the Soldering Module.

First, we will introduce you to the components that you will be working with. Then, you will do some preliminary tasks that will run in the background. As those tasks are running, you will go through the first portion of the build.

Required Materials

What you will need	- Battery - Battery Charger - Battery Charging Adapter - Power Distribution Board (PDB) - Universal Battery Eliminator Circuit (UBEC) - Raspberry Pi Hat - XT60 Connector - Soldering Tools - Raspberry Pi - Heat Sinks - Micro SD Card - Base Station
What you will get	Assembled Raspberry Pi Hat and PDB

Detailed Hardware Descriptions

Battery

Fig. 26 LiPo Battery

The Battery on your drone is a 1500mAh 3S 20C LiPo Battery. Let's go over what all that means.

Capacity

The unit milliampere-hour, or mAh, is a measure of how much charge a battery can hold. The higher this number is, the more charge the battery can hold; therefore, the battery will "last" longer and your drone will fly for a longer time without needing to be recharged.

Structure

A Lithium-ion polymer (LiPo) battery is made up of one or more LiPo cells. Each cell has a voltage of 3.7 V when it is discharged, and 4.2 V when it is charged. The cells in your battery are connected in series, so that the voltages add together to get a total of 11.1 V when discharged, and **12.6 V when charged**. There are three cells connected in series in your battery, so it is called a 3S battery.

Current Output

The "**C**" rating of a LiPo Battery determines how much current the battery can deliver. The maximum current draw is equal to the battery's C rating multiplied by the battery's capacity. For the drone's 1500mAh 20C batteries, the maximum current draw is 1500mAh x 20C = 30A.

Precautions

> ⚠ **Warning**
>
> Review the safety information on the battery.

Taking the proper precautions when using, storing, or charging is very important to keeping the battery safe.

> ⚠ **Danger**
>
> 1. Only store the battery at room temperature and out of direct sunlight.
> 2. Do not discharge a battery below 10.5 V.
> 3. Never leave the battery charging unattended.
> 4. Unplug the battery once it is fully charged.

Battery charger and charging adapter

The battery charger and charging adapter allow you to draw power from an outlet to recharge the LiPo battery of your Duckiedrone. The blue charger has a display that cycles through showing the voltage on each cell and the overall voltage of the battery while it is charging.

Fig. 27 Battery charger

Fig. 28 Charging adapter

> 💡 **Tip**
>
> During development/testing you might need to keep your drone on for more time than the battery allows.
>
> A good solution for this scenario is to connect the Raspberry Pi to an external power supply, disconnecting the battery from the PDB.
>
> In this way you can activate all the nodes (including Flight Controller) but the motors will not get power to spin.

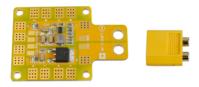## Power Distribution Board (PDB)

Fig. 29 Power Distribution Board (PDB)

The Power Distribution Board is used to distribute power from the battery to electrical components of the drone. The PDB is an example of a Printed Circuit Board (PCB), which is a circuit board that has connections within its structure. For the PDB, the internal wiring connects all of the positive (+) pads together, and all of the negative (-) pads together; this allows for the battery to be connected to one set of positive and negative pads, and all of the other pads receive power.

UBEC (Universal Battery Eliminator Circuit)

The UBEC, solves the issue that arises from different electrical components requiring different supply voltages. In the case of the drone, the LiPo battery is used for outputs around 12V: this is the required voltage for the motors, but not for the Raspberry Pi, which requires 5V. The Universal Battery Eliminator Circuit eliminates the need to carry multiple batteries of different voltages by converting the 12V supply from the battery down to a 5V supply for the Raspberry Pi.

Raspberry Pi

Fig. 31 Raspberry Pi 3 Model B

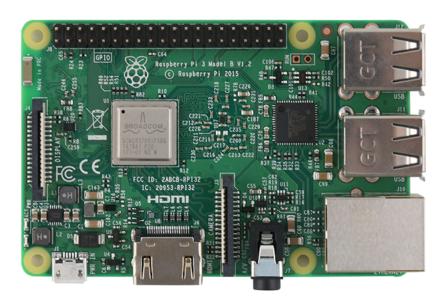The Raspberry Pi is a single-board computer. The Raspberry Pi is capable of running a full desktop operating system: you can use it as a computer similar to the one you're using now. The Raspberry Pi is used on the drone as the main computer that reads and processes all the sensor data and user inputs, and then sends commands to the flight controller to control the drone motors. The drone would

be able to fly without it if a person had a remote controller to manually steer the drone. However, for the Duckiedrone, the person with a remote controller is replaced by software and sensors on the Raspberry Pi. This, as well as the sensors you will connect later, makes autonomous flight possible.

Raspberry Pi Hat

The Raspberry Pi Hat is a special type of breadboard. One useful property of a breadboard is that it has rails. A rail is a sequence of holes that share an electrical connection:
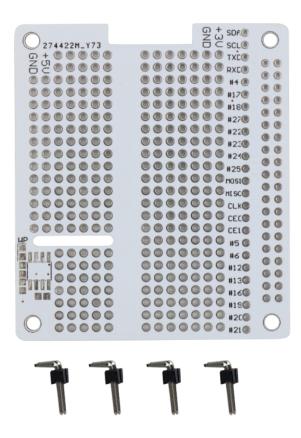

Fig. 32 Raspberry Pi Hat shield

Fig. 33 Raspberry Pi Hat header

The rails are useful because every wire put on a rail will be electrically connected; this means that it does not matter which hole along a rail a wire is inserted. This is useful for wire organization, and if a soldering mistake is made in one hole, an adjacent hole in the same rail can be used instead.
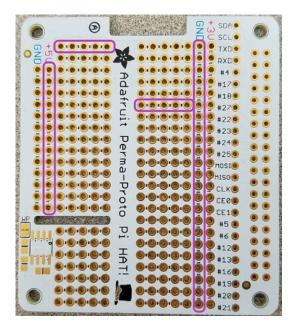
Fig. 34 PiHat with rails highlighted in purple

Notice especially the long +5V and GND rails; we can use any of the holes in these rails to provide power to components. Also notice that there is a 3.3V rail, be sure not to mix this up with the 5V rail because electrical components will only work at the correct voltages.

Micro SD Card

Fig. 35 microSD card

The micro SD card stores the operating system on the Raspberry Pi, as well as all the software needed for autonomous flight.

XT60 Connector

Fig. 36 XT60 connector cable

An XT60 connector cable is a component which provides power when a power source (e.g. battery) is connected to it. By soldering it to the PDB, the PDB will get power to distribute to other components.

Heatsinks

The heatsinks help regulate the temperature of your Raspberry Pi.

Instructions

Preliminary Tasks

Charge the Battery

Start charging your drone battery so it is ready when you need it. To charge the battery connect it to the `3 Cells` plug of the charger and connect it to the charging adapter.

Fig. 37 Battery connected to the charger (the display will show different codes)

The battery takes about 2 hours to charge. When charging, the battery flashes between the voltage on each cell and `"ALL"`, the voltage on all cells together.

⚠ **Warning**

The battery is fully charged when `"ALL"` is 12.6 volts.

The charger will also start emitting a `BEEP` sound when charging is complete.

Disconnect your battery.

You can also tell by measuring the voltage across the battery's power and ground with a multimeter. When the battery is plugged into a charger and the charger is not plugged into a wall, it uses power from the battery to display the voltage on the battery. Over time this will drain the battery. If a battery's voltage gets too low, the battery can then no longer be charged.

Attach the pin Header to the Raspberry Pi Hat

Identify the front and back

Identify the side of the Raspberry Pi Hat that has writing on it - this side is the front, and the side without writing is the back.

> **💡 Tip**
>
> Take note of where the slot in the Raspberry Pi Hat is to help with matching the orientation in the following instructions.
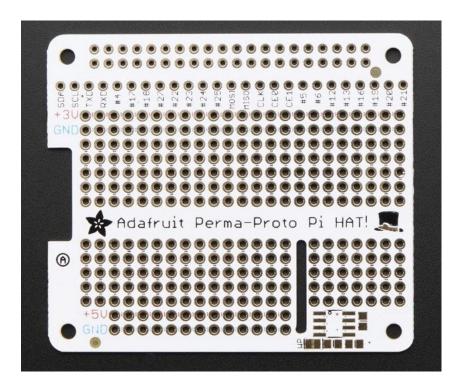
Fig. 38 Raspberry Pi Hat front
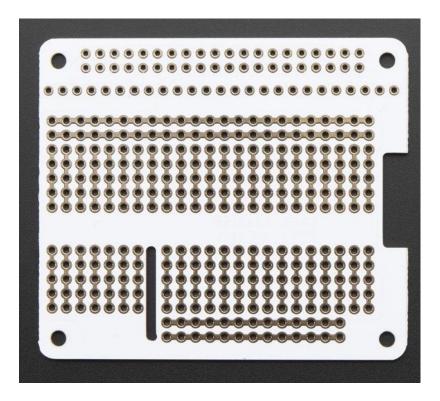
Fig. 39 Raspberry Pi Hat back

1 - Insert the Pin Header

Insert the Pin Header into the back of the Raspberry Pi Hat as shown in the image (the black plastic side will be on the side without any writing).
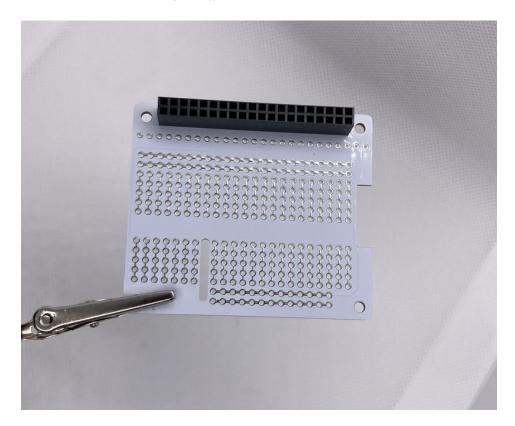
Fig. 40 Pin header inserted into the back of Raspberry Pi Hat

2 - Solder the Pin Header

> **💡 Tip**
>
> **video tutorials**
>
> - [part 1](#)
> - [part 2](#)
> - [part 3](#)
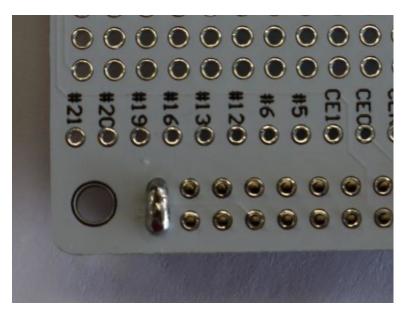
> **ℹ Note**
>
> It's very easy to add too much solder and create a solder bridge between adjacent pins. Visually inspect your soldering to make sure that adjacent pins are not connected by globs of solder.
>
> **Good soldering**:
>
>

> **⚠ Attention**
>
> **Bad soldering**:
>
>
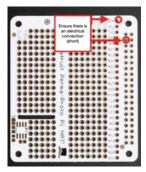
If you have a solder bridge, try the technique shown in [this video](#). If this does not work, you can use a desoldering wick or a solder sucker to remove the excess solder.
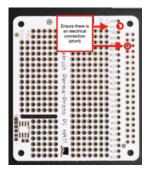
> ↪ **Check**
>
> Sometime too much heat when soldering the pin header can mess up the Raspberry Pi Hat connections. To make sure the connections are still good, do a **connectivity check** on the Raspberry Pi Hat. Verify there is:
>
> - A short between the hole labelled `SDA` on the Raspberry Pi Hat and the pin at row 2 column 1 of the pin header that you soldered.
>
>
>
> - A short between the hole labelled `SCL` on the Raspberry Pi Hat and the pin at row 3 column 1 of the pin header that you soldered.
>
>
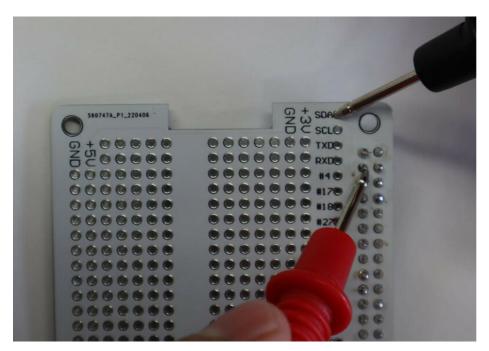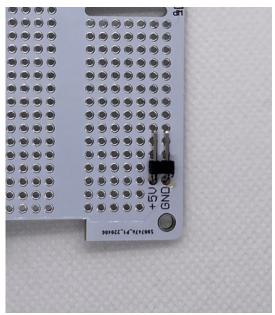
> 💡 **Tip**
>
> Use the multimeter to check for connections:
>
>

Attach the UBEC to the Raspberry Pi Hat

The UBEC is a battery eliminator circuit that lowers the voltage from the battery from 12V (used by the motors) to 5V (used by the Pi). We will power the Pi by connecting it to the battery via the UBEC. Within your kit you are provided with a few 90 degrees header pins. You will need to solder one of them (with two pins) into the +5V and GND rails as shown.
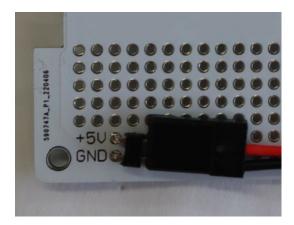
The UBEC comes with a black connector which you can plug into your newly soldered pins.

While you could directly solder the UBEC to the Raspberry Pi Hat, using the pins provides a plug and play feature when needing to disassemble the drone. Also allowing the user to test motors while not having the Raspberry Pi drain power from them.

> **! Attention**
>
> Make sure to plug in the conector correctly with red wire on +5V and black wire respectively on GND. Leaving the last pin hole on the UBEC connector free.
>
>

> **! Attention**
>
> Do a **connectivity check** on the Raspberry Pi Hat.
>
> Verify there is no short between the +5V rail and the GND rail on the Raspberry Pi Hat

Attaching Wi-Fi mode pins

The Raspberry Pi has a unique way to understand how to switch between its two different Wi-Fi modes.

You will be soldering another set of 90 degree pins in port #5 and port #6 of the Raspberry Pi Hat as shown in the following image.
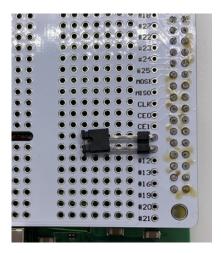
Fig. 41 WiFi mode pins with jumper (not inserted in the pic)

> **❶ Attention**
>
> The other piece of hardware shown in the image, which is a black cap-like device next to the soldered pins, is a jumper to short the pins. This piece is very small, be careful not to lose this.
>
> It is advised to insert it into the pins for safe keeping.

> **⚠ Warning**
>
> Remember that it does matter whether the jumper is plugged into both pins or just one when booting your Raspberry Pi up:
>
> - Jumper plugged in both pins → **access point mode**
> - Jumper plugged in only one pin → **client mode**

Preparing the PDB

The PDB (Power distribution board) takes power from the battery and distributes it to the motors (at 12V) and to the Raspberry Pi (via the BEC which converts it to 5V). Note that the PDB itself includes a separate BEC, but in our testing we found it did not provide enough amps to power the Pi, so we provide an external UBEC.

Tin the PDB

Similar to exposed wires, the metal pads on a PDB need to be tinned. This will allow tinned wires to be joined to the pads - and therefore the PDB.

> ⚠️ **Warning**
>
> It is **very important** to use a chisel tip soldering iron rather than the pointed tip you used for soldering the header to the Raspberry Pi Hat.
>
> Make sure to unplug your soldering iron and let it cool down **completely** before switching the tip.
>
> This tip applies more heat quickly to the pad, so that the solder flows onto the pad when you tin it.

Tin every pad on the PDB, including the GND 12V pads.

> ⚠️ **Warning**
>
> Be careful not to aggressively push the soldering iron tip into the PDB, as too much force will cut the pads right off.

Here's a video of the process.

We couldn't verify the security of your connection.

Access to this content has been restricted. Contact your internet service provider for help.

Solder the XT60 Battery Connector to the PDB

1. **Cut** the metal-covered ends off of the battery connector wires
2. **Strip** the ends of the battery connector wires so that about 1 cm of wire is exposed
3. **Tin** the exposed ends of the XT60 connector (The original tinning is harder to connect to than freshly-tinned ends)
4. **Solder** the XT60 **red (+) wire** to the tinned positive (+) opening on the PDB, and solder the **black (-) wire** to the tinned ground (-) opening on the PDB.

💡 Tip

This wire is very thick and it will take a while for the solder to melt.

Make sure your soldering iron is turned all the way up, you're using a chisel tip and be patient.

It also helps to keep the cable at an angle to have it touch the copper hole while soldering.

Fig. 42 Tinned PDB with XT60 soldered

Solder the UBEC to the PDB
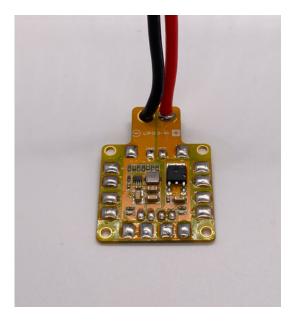
Tin the ends of the red and black wires that are connected to the side of the UBEC labeled `INPUT` (they're pre-tinned, you should cut the cable and tin it again).

Solder the BEC red (+) `INPUT` wire to the positive (+) pad on the PDB `12V` pad, and solder the black (-) `INPUT` wire to the `GND` pad on the PDB, as shown in the image.
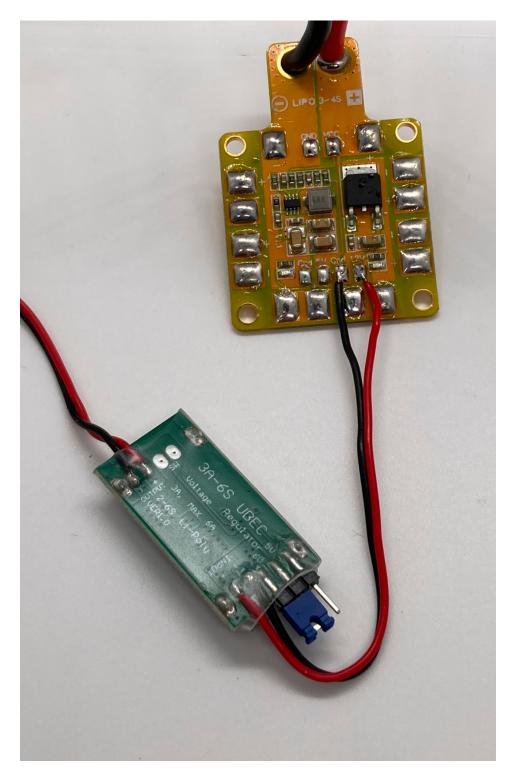
Fig. 43 UBEC soldered to PDB

Visually inspect the drone to **verify the following**:

- All red wires connected to the PDB are connected to positive (+) pads
- All black wires connected to the PDB are connected to negative (-) pads
- The wires on the `INPUT` side - NOT the `OUTPUT` side - of the BEC are soldered to the PDB.

Do a connectivity check on the PDB.

Verify there is:

- a short between any positive (+) pad and any other positive (+) pad
- a short between any negative (-) pad and any other negative (-) pad
- **no short** between any positive (+) pad and any negative (-) pad

only if the connectivity check passed, do a *DC voltage* check on the PDB.

Plug in a 12V battery and verify there is:

- ~0V between any positive (+) pad and any other positive (+) pad
- ~0V between any negative (-) pad and any other negative (-) pad
- ~12V between any positive (+) pad and any negative (-) pad.

ℹ **Note**

If the battery is charged to X volts instead of 12 volts (e.g. 10), then the multimeter will show X volts instead of 12 volts.

only if the DC voltage check passed, re-connect a battery to your drone and verify the following:

- red LEDs on the PDB light up.
- red LED on Raspberry Pi lights up.

Put Heat Sinks on Raspberry Pi

Attach the heat sinks to the Raspberry Pi as shown in the pictures.

Fig. 44 Top heatsinks

Fig. 45 Bottom heatsink

Attach the Raspberry Pi Hat to the Raspberry Pi

Align the 2x40 GPIO pins on the Raspberry Pi with the 2x40 pin header that you soldered to the Raspberry Pi Hat as shown in the image.

Fig. 46 Raspberry Pi Hat lined up with Raspberry Pi pins

Press the pin header down onto the GPIO pins to connect the Raspberry Pi Hat.

Fig. 47 Raspberry Pi Hat inserted into Raspberry Pi pins

Final Steps

> ➡ **Check**
>
> On the Raspberry Pi Hat, do another connectivity check to verify that there is **no short** between the +5V rail and the GND rail.
>
> **Verify that the Raspberry Pi has power:**
>
> 1. Connect the battery
> 2. Verify that the BEC has a solid red light.
> 3. Verify that the Raspberry Pi has a solid red light.

Insert the SD card into the Pi

> ⚠ **Disconnect the UBEC from the Raspberry Pi Hat**
>
> You will boot the Raspberry Pi once the build is complete, for now on until the first boot make sure the Raspberry Pi isn't powered.

Insert your (previously flashed) micro SD card into the micro SD card slot on the bottom of the Pi.

> ❗ **Attention**
>
> The micro SD card **direction does matter** - the lettering on the SD card should be facing away from the Pi.

Only do this step if SD-card is already flashed!

Fig. 48 Micro SD Card being inserted into the Raspberry Pi

TOF Sensor

Overview

Preface

At a high level, a **sensor** is a device that *observes* something about the world and reports its observations on an electrical wire. For example, a camera can be a sensor.

In contrast, an **actuator** is a device that *does* something when provided power via an electrical wire. For example, a motor can be an actuator.

The simplest possible robot is one that has only actuators. However, a robot with any amount of autonomy would also require sensors. This is because such a robot would need observations about its world in order to decide what to do with its actuators.

In this part of the build, you will be adding your first sensor to the drone: the time of flight (ToF) sensor. The ToF sensor is used to measure distance. We'll provide more details about the hardware used in this portion of the build, and then get into the instructions.

> ⚠ **Attention**
>
> You will need to have completed the Raspberry Pi and PDB section before you can begin this build section.

Required Materials

What you will need	• ToF Sensor • ToF Sensor wire • Soldering tools
What you will get	• Connection to ToF Sensor

ToF Sensor

The ToF (**Li**ght **d**etection **a**nd **r**anging - **Lidar**) sensor is used to measure distance. On the drone, we use this sensor to measure the height of the drone above the ground. Lidar enables autonomous vehicles to "see" by generating and measuring millions of data points in real time, creating a precise map of its ever-changing surroundings for safe navigation.

Fig. 49 ToF sensor

ToF Sensor cable

This cable allows you to connect the ToF sensor to the Raspberry Pi Hat. One end has a 4-pin connector that is attached to the sensor, while the other has 4 wires that will be soldered to the breadboard.

Fig. 50 ToF sensor cable

Build progress

After completing this section, you will have the Raspberry Pi Hat connected to the Raspberry Pi, and the former will have soldered on:

- A 2-pins connector to connect the UBEC, soldered to the +5V and GND rails.
- A 2-pins connector to select the WiFi mode of the Raspberry
- The ToF cable

See the following figure:
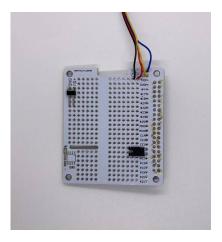
Fig. 51 Raspberry Pi Hat progress

The ToF sensor receives power from the Raspberry Pi Hat, and the sensor output signal is received by the Raspberry Pi, using the SCL and SDA inputs (the sensor uses a protocol called I2C).

Instructions

1. Strip about 5 mm off of the newly-exposed ends of the ToF sensor wire
2. Twist and Tin the ToF sensor wires.

   > **⚠ Attention**
   >
   > Do not put too much solder, as the wires must fit through the holes in the Raspberry Pi Hat

3. Solder the ToF sensor wires to the Raspberry Pi Hat following the figure below:
   - insert a wire into the correct hole in the front of the Raspberry Pi Hat.
   - Solder the red wire to the +3V rail
   - Solder the black wire to the GND rail
   - Solder the yellow wire to SCL hole
   - Solder the blue wire to SDA hole

4. Trim the excess cable on the back side of the Pi Hat (where you just soldered it)

Fig. 52 Soldered TOF sensor cable

Checkpoint

> ↪ **Check**
>
> Do a **connectivity check** on the Raspberry Pi Hat. Verify that:
>
> - There is NO electrical connection (short) between the +3V and GND rails.
> - There are no wire strands touching other pads

Here is a picture of how the Pi Hat should look like.
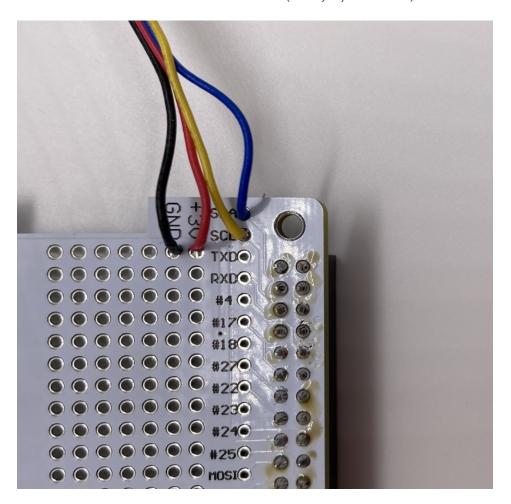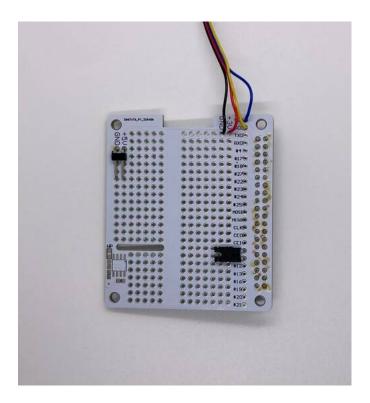
Fig. 53 Raspberry Pi Hat progress

Reattach the Raspberry Pi Hat

Reattach the Raspberry Pi Hat to the Raspberry Pi by aligning the GPIO pins with the pin header and pressing down. Refer to [previous chapter's instructions](#) for detailed instructions on how to attach the Raspberry Pi Hat.

Motors and ESCs

Overview

Preface

In this phase of the build, you'll be adding the essential elements of every drone– the motors, ESCs, and the Flight Controller.

Required Materials

What you will need	- Counter-clockwise Motors - Clockwise Motors - Long gray metal M3 Bolts - Nylon M3 Bolts and nuts - Electronic Speed Controllers (ESCs) - Propeller guards - Plastic standoffs - Battery leads Wire - Soldering Tools - Zip ties
What you will get	Motors connected to power

Detailed Hardware Descriptions

Motors

The motors used are Brushless Direct Current (BLDC) motors. They are a particular type of electric motor commonly used in drones for their high reliability and low wear during usage.

Fig. 54 BLDC motors

black nut : *counter*-clockwise motors

red nut : clockwise motor

Electronic Speed Controller (ESC)

An Electronic Speed Controller (ESC) is used to regulate the speed of a motor according to a signal from the Flight Controller. A brushless motor would not be able to spin without an ESC, as they are responsible for changing the magnetic fields that generate a moment to make the motor spin.

Fig. 55 Electronic Speed Controller

Battery Monitoring Leads

The battery monitor wires allow the Flight Controller to monitor the power traversing the PDB. This is useful because the Flight Controller can inform the Raspberry Pi of the battery voltage. The benefit of this is that the software will prevent the battery from draining too low and permanently damaging it.

You will be using the extra red and black wire that came with the kit to make the battery monitor leads.

Fig. 56 Battery monitoring leads

Long M3 screws

These long gray screws are used to attach the motors to the frame together with the propellers guards (prop guards) and the plastic standoffs used as landing legs.

Fig. 57 **left:** right screws to use

right: wrong screws, do not use

Nylon M3 bolts and nuts

These plastic bolts are used to attach the PDB to the frame of the Duckiedrone.

Fig. 58 Nylon M3 bolts (8) and nuts (11)

Zip Ties

Zip Ties will help you tie up your Duckiedrone together and manage its cabling (Duckiecaptain wants its ship to be tidy!).

Fig. 59 Zip Ties (Large on top, small on bottom)

Propeller Guards

The propeller guards protect both the propellers and the environment where the drone flies.

Fig. 60 Propeller guard

Standoffs feet

These white plastic standoffs are mounted beneath the motors' attachment point on the frame and provide stable ground contact points for the drone.

Fig. 61 Drone feet standoffs

Instructions

> ℹ️ **Note**
>
> Sometimes parts will have wires already tinned out-of-the-box by the manufacturer (i.e. pre-tinned).
>
> You can identify this by:
>
> 1. the "shininess" of the tip of a wire and
> 2. the inability to fray the wire strands of the tip of a wire.
>
> However, such tinning is often ineffective.
>
> Cut off any pre-tinned tips, then strip and tin the part yourself.

> ⚠️ **Warning**
>
> Visually inspect each ESC and verify that the heat shrinks are on properly; there should be no exposed wires and each heat shrink should be a tight fit.

Do a connectivity check on the XT60 connector cable; verify there is no short between the red and black wire.

Soldering ESCs to the PDB

An ESC (i.e. Electronic Speed Control) is a component which requires power. It takes this power and provides a variable amount of it to a motor; since a motor's RPM depends on how much power it gets, an ESC can control how fast a motor spins by controlling how much power it supplies the motor.

Solder each of your 4 ESCs to the PDB, one by one:

1. Strip about 5 mm of wire from (+) red and (-) black cables of the ESC.
2. Tin the wire
3. Solder on the pad, the (+) red cable solders to the (+) on the PDB and the (-) black cable to the (-) pad.
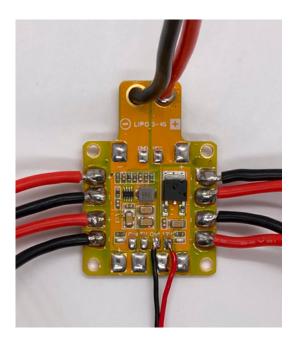
Fig. 62 ESCs cables soldered to the PDB on the side pads.

Solder battery monitor leads to the PDB

Solder the 6-inch red and black wires to the PDB to the `VCC` and `GND` pads respectively.
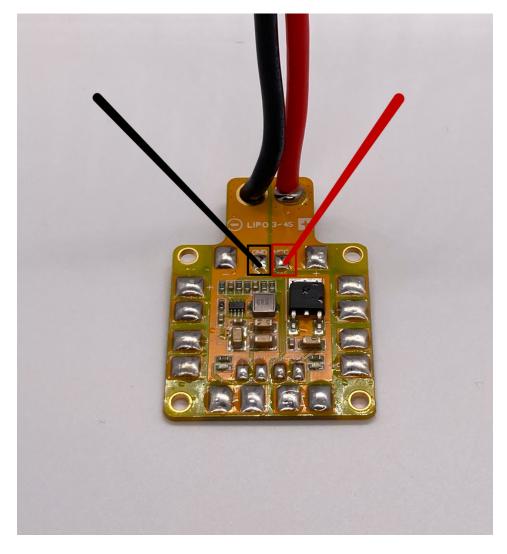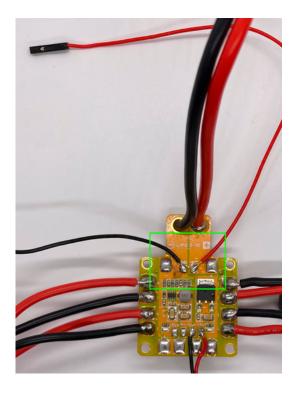
Fig. 63 Pads for soldering the red and black monitor battery leads

These wires are soldered as to go out of the PDB.

Fig. 64 Battery leads soldered to the PDB

> ⚠ **Warning**
>
> It's important not to strip the wires too much at the PDB end to prevent the positive and negative leads from touching.
>
> Pull **very** slightly the two wires to make sure they stay connected.

Attaching parts to drone frame

This section will cover attaching the first set of items to the drone frame.

> ❗ **Attention**
>
> Before beginning, verify the PDB is completely soldered with all necessary parts (as covered in previous sections).

For reference, here are the motor directions with respect to the frame:
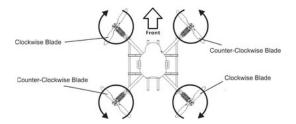
Fig. 65 Your motors MUST spin in these directions

Fig. 66 Notice the colors of the motors nuts in this pic

(**red**->*counter*-clockwise, **black**->clockwise)

Align the frame

Place the drone frame on a flat surface so that the back is facing you.

Attach PDB to frame

Place the completed PDB towards the bottom of the drone frame, with plastic M3 screws in their respective holes. For each of the 4 corner screw holes of the PDB, add a plastic nut on the other end to secure it.

Fig. 67 PDB Secured in Drone Frame (front side up)

Attach motors

There are two clockwise and two counter-clockwise motors. Unfortunately the motors themselves are not labeled, so only the color of the nut will let you tell the difference. The clockwise motors have a nut threaded so that when the propellers spin clockwise, the nut will tend to tighten, and the counter-clockwise motors are the opposite.

> **⊘ Attention**
>
> - The counter-clockwise motors have a red nut.
> - The clockwise motors have a black nut.

Take the motors out of their bags.

> **⊘ Attention**
>
> Immediately screw the nut that came in the bag onto the main bolt sticking out of the motor. Screw the nuts on gently by hand - **do not** force it.

This will prevent you from mixing up the motors.

> **💡 Tip**
>
> Motors, frame, prop guards and standoffs all have 4 holes each for the screws. These holes are slightly asymmetrical, so that they can only be attached in one direction.

Get the standoffs, insert two screws through the holes and insert them in the holes on the frame (on the back side of the frame).

Get the prop guards, and put a prop guard on the frame. The prop guards will slot underneath each motor, between the motor's casing and the prop guard.
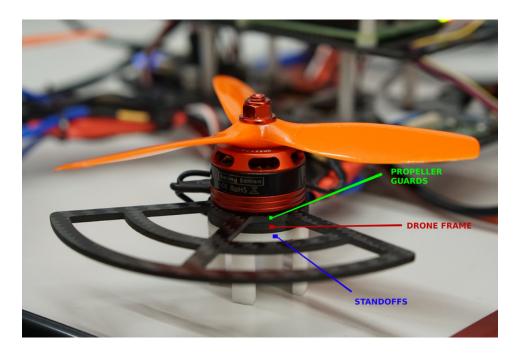
Fig. 68 **top** : propeller guards

middle : drone frame

bottom : standoffs

Attach **clockwise** motors to the bottom-right and top-left of the drone frame, using 4 long silver M3 screws for each attachment.

> ⚠ **Danger**
>
> Do not use the black screws that come with the motor
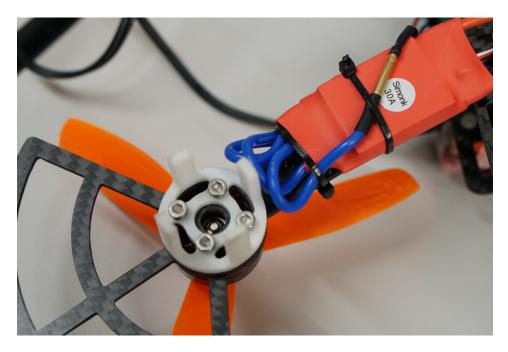
Fig. 69 Motor screwed on the frame with standoffs and prop guards

notice: the gray M3 screws are used

Attach **counter-clockwise** motors to the bottom-left and top-right of the drone frame, using 4 long M3 gray screws for each attachment.

Connecting the Motors to the ESCs

For each motor, connect its plug bullet connectors to the socket bullet connectors of the ESC in the motor's corner (e.g. top-left motor connects to top-left ESC). Any connection order will suffice for now, as you will be able to change them in a later phase.
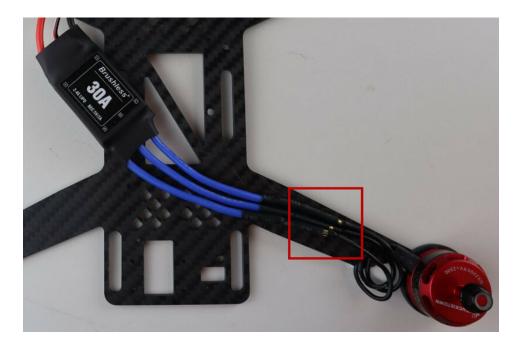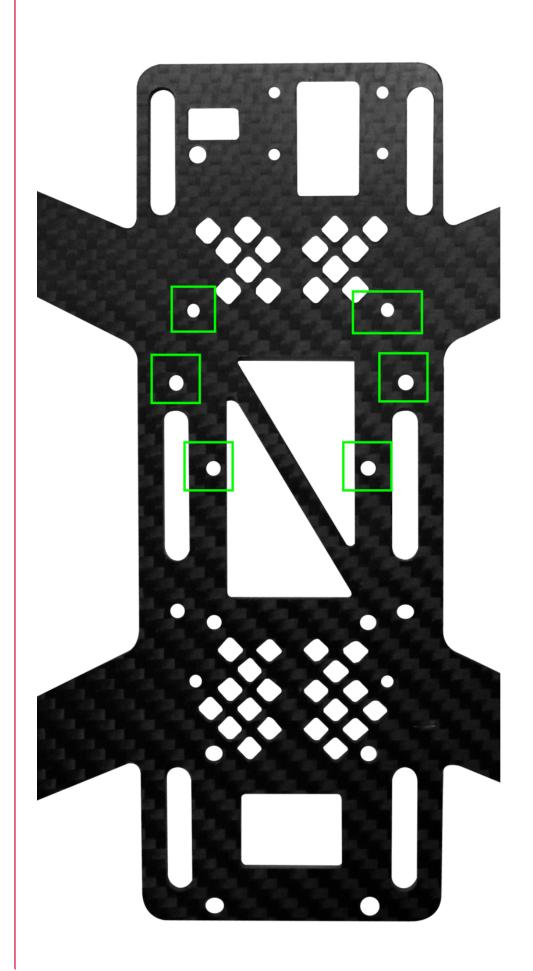
Fig. 70 ESCs plugs connected to motors

Attaching the ESCs to the frame

Each ESC has to be fixed to the frame. Due to the length of the wires powering the ESCs, you will have to fix the bottom two to the arms and the top two to the arms and the body of the drone, as shown below.

> ⚠️ **Warning**
>
> When attaching the top two ESCs to the frame make sure not to cover any of the round holes on the frame of the drone.
>
> They will be used to screw components on later.

Use two thin zip ties provided to fix each ESC, as shown.
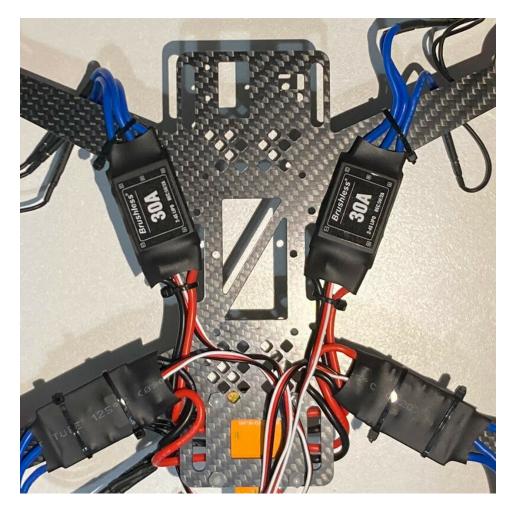
Fig. 71 ESCs attached to the frame

> ⚠ **Attention**
>
> Make sure to still be able to connect both the 3-pin wires and the banana-plug wires.

> 💡 **Tip**
>
> Fixing first the top two ESCs will allow you to route their cables more easily.

Cut the excess zip ties off to tidy up.

Checkpoint

Verify the wiring is correct

Visually inspect the drone

Verify the following:

- All red wires connected to the PDB are connected to positive (+) pads
- All black wires connected to the PDB are connected to negative (-) pads
- The wires on the `INPUT` side - **NOT** the `OUTPUT` side - of the BEC are soldered to the PDB
- For the battery monitor lead: the red wire is connected to a positive (+) pad while the brown wire is connected to a negative (-) pad

Do a *DC voltage check* on the PDB.

Plug in a 12V battery and verify there is:

- ~0V between any positive (+) pad and any other positive (+) pad
- ~0V between any negative (-) pad and any other negative (-) pad
- ~12V between any positive (+) pad and any negative (-) pad.

ESC check

Re-connect the battery to your drone.

Verify the following:

- The PDB board lights up.
- Each ESC emits a noise (it does so by performing a slight rotation, so you should also notice some subtle movements of each motor)

Flight Controller & Cleanflight

Overview

In this phase of the build, you will attach the Flight Controller to your drone and to the ESCs.

You have already configured the Flight Controller software, now you will use it to test and calibrate ESCs.

Flight Controller

What you will need
- Flight Controller
- USB to Micro USB cable
- Nylon M3 white bolts and nuts
- M3 Rubber spacers
- Base station

What you will get
- Working Flight Controller stack.

Flight Controller

The Flight Controller (i.e. Flight Controller) contains multiple sensors: an Inertial Measurement Unit (IMU) and a gyroscope. The IMU measures linear accelerations and the gyroscope measures angular velocities. The Flight Controller also receives commands from the Raspberry Pi and then sends electrical signals to the ESCs which in turn change the speeds of the motors.

⊗ **Attention**

You should already have installed the correct version of Cleanflight on your Flight Controller.

↪ **See also**

If you haven't or you're not sure you can reflash the Flight Controller following the Flight Controller initialization instructions here.

The Flight Controller bag will also contain header pins to be soldered and connectors. These differ between the two Flight Controller versions, use the tabs below to select yours.

OSD   **ACRO**

In the Flight Controller bag you'll also find:

- 1 set of 8x3 straight pins
- 1 set of 2x4 straight pins
- 1 strip of straight pins
- 1 strip of 90 degrees pins
- Cabling

Fig. 72 OSD Flight Controller
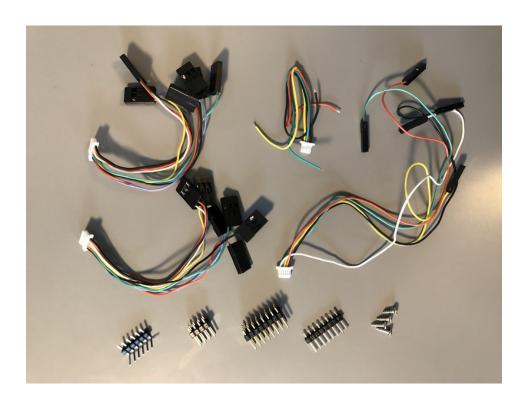
Fig. 73 Cabling included with OSD Flight Controller

> **ℹ Note**
>
> Due to packaging variations there might be a slightly different number of straight pins or you might find only 8x3 straight pins rather than 90° bent pins.
>
> This is okay, you should anyways get the needed pins to have the functionality required by your Duckiedrone.

USB to Micro USB cable

This cable is used for two purposes:

1. To configure the Flight Controller settings in CleanFlight

   > **ⓘ Note**
   >
   > This part only needs to be done once.

2. To send the flight commands from the Raspberry Pi to the Flight Controller. This connection allows our software on the Raspberry Pi to control the motors. The Raspberry Pi tells the Flight Controller what roll, pitch, yaw, and throttle values the drone should have, and then the Flight Controller speeds up or slows down the motors to achieve these values.
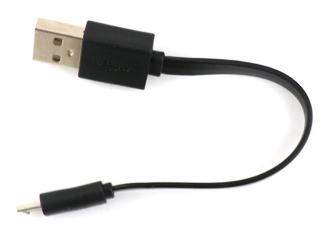
Fig. 75 Micro USB cable

Rubber spacers

These spacers are used to isolate the Flight Controller and dampen the vibrations from the ESCs, to improve accuracy in the accelerometer readings.

Fig. 76 M3 Rubber spacers

Nylon M3 bolts and nuts

These plastic bolts are used to attach the PDB to the frame of the Duckiedrone.

> ⚠ **Warning**
>
> You will find both M3 and M2 bolts and nuts in your Duckiebox, pay attention to use the M3 in this section.
>
> You can distinguish them by:
>
> 1. comparing the two; the M3 bolts will be slightly thicker
> 2. the M2 bolts only have 4 corresponding nuts
> 3. the M3 bolts have 11 corresponding nuts
>
> The M3 bolts will fit firmly in the PDB mounting holes, whereas the M2 bolts would wobble and be loose.

Fig. 77 Nylon M3 bolts (8) and nuts (11)

Identifying your Flight Controller board

> **Attention**
>
> There are currently 2 types of Flight Controller hardware.
>
> Identify here which type of Flight Controller you have and use the steps corresponding to your hardware.
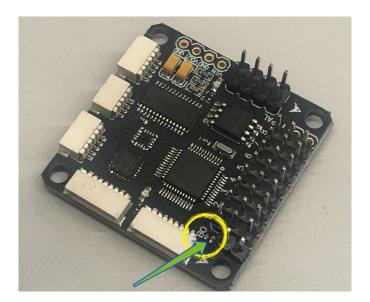
OSD   **ACRO**

Fig. 78 OSD version Flight Controller

- There is no micro USB port on the Flight Controller (it is on a separate smaller board)
- As highlighted below, the Boot pins are exposed near the `B0` marking.

Instructions

Flight Controller soldering

Here you will solder the Flight Controller.

Take the Flight Controller out of its plastic casing, exposing both sides of the PCB.

Soldering the ESCs pins

Here you will need to solder the pins to connect your ESCs to your Flight Controller.

There are subtle variations between the two Flight Controller versions, **choose in the tabs here your Flight Controller type**.

> ⚠ **Attention**
>
> Be sure that the direction you solder the pins into the board is exactly as shown in the images

OSD   **ACRO**

If you only have a 3x8 straight pins connector you'll need to solder those rather than the 90° pins.
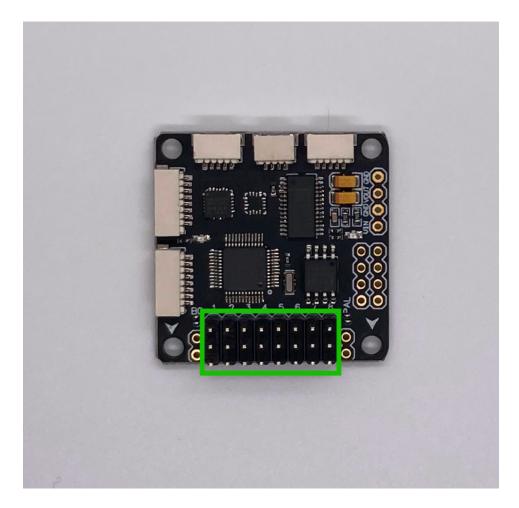
Fig. 80 3x8 straight ESCs pins soldered

Solder the battery monitor pins to the flight controller

Pick the the 6 pins header that came in the Flight Controller wiring and use pliers to break off a 2 pins connect.

Solder the 2 pins connector to the flight controller, as shown in the image below (select your Flight Controller version in the tab).

> **❗ Attention**
>
> The battery leads bins have to be soldered on the same side you soldered the ESCs pins on

OSD   **ACRO**

The battery lead pins `VBAT` are in between the ESCs pins and the bottom-left mounting hole of the Flight controller.
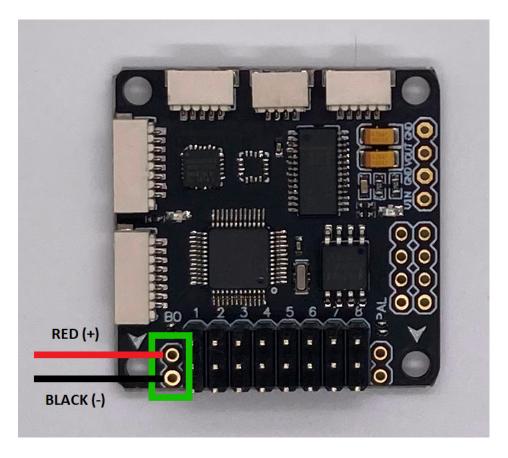
Fig. 82 Solder the battery leads pins to these two pads (they must stick out on the

same side as the other pins)

Attaching the Flight Controller

Now you will fix the Flight Controller to the Duckiedrone frame. The Flight Controller is located on the bottom side of the Duckiedrone, in the center (his "belly").

Fig. 84 Identify the front and back of your Duckiedrone
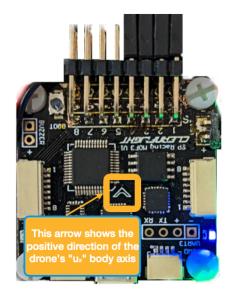
The axes of the Flight Controller, according to the measurements of linear and angular accelerations provided by the IMU, are as follows:
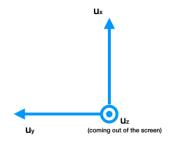
- ux is positive in the direction identified by the arrow serigraph on the PCB;

- uz is orthogonal to the plane identified by the Flight Controller's PCB, and positive "coming out of the screen";
- uy is such that (ux, uy, uz) is a right-handed reference system.

> **❶ Attention**
>
> Make sure to orient the arrow on the Flight Controller towards the front side of the Duckiedrone.
>
>

> **⚠ Warning**
>
> This is a crucial setting. A mistake here will cause your drone to confuse left/right, up/down and most probably crash immediately.

To fix the flight controller to the frame use the 4 white M3 Nylon screws:

1. Insert the bolts **from the front** side of the frame.

2. Insert the rubber spacers on the bolt from the bottom side of the frame.
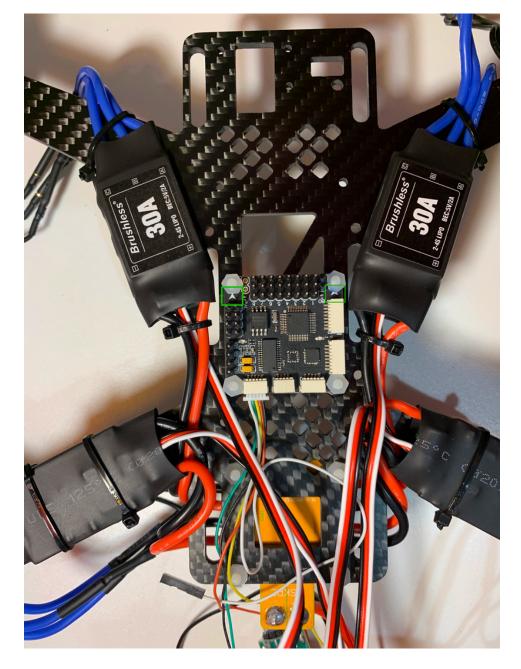
3. Screw the nuts on the bolts from the bottom side.

> **❗ Attention**
>
> The `OSD` version of the Flight Controller has the soldered battery leads header pins partially occluding the space for the nut to be screwed.
>
> The suggested solution is to simply file the nut about 1 mm to have enough clearance once screwed on the bolt.

In the end your Duckiedrone bottom should look like this:

Fig. 85 Flight controller fixed to the bottom of the frame in the correct orientation

(notice the arrows on the Flight Controller pointing towards the front)

Connecting the ESCs to the Flight Controller

Now that the Flight Controller has been attached to the frame, it can be connected to the ESCs via the PWM wires (i.e. the thin white, red and black wires coming out of the ESCs).

Take a moment to find the numbers 1-8 next to the flight controller 3x8 header pins (the big row of pins that you soldered in).

We will be connecting the **PWM wires to numbers 1-4** because we have 4 motors.

These numbers on the Flight Controller indicate which PWM wire coming out of the motor should be connected to which set of pins on the flight controller.
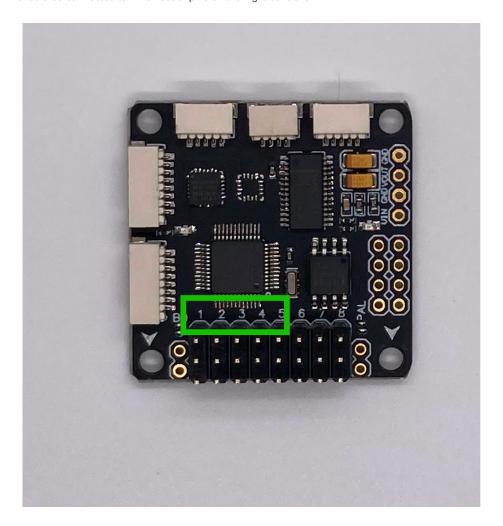
Fig. 86 Motors numbering on the Flight Controller

For example, in the image below, **motor 1 is in the bottom right**; therefore you will take the PWM wire from the ESC connected to the motor in the bottom right of your drone, and connect this to the pins labeled 1 on the flight controller.
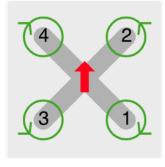
Fig. 87 Motors numbering

> **⚠ Attention**
>
> There is a correct way to connect an ESC cable to the Flight Controller.
>
> See below on how to connect your Flight Controller version to the ESCs.

OSD   **ACRO**

Make sure the white wire of the ESC signal wire pair is facing in from the board, and the black wire is facing out.

Testing the Motors

With the ESCs connected to the Flight Controller, your drone's motors can be tested. In this section, you will verify that the motors are spinning correctly.

> **⚠ Danger**
>
> **Make sure no propellers are attached to your drone's motors!**

You will be spinning the motors and you don't want your drone to fly off your desk!

Launch Cleanflight Configurator

1. Open up Cleanflight Configurator on your base station.

Connect your drone

1. Plug your drone's Flight Controller into your base station (via the USB to micro USB cable)
2. Press `"Connect"` in the top right corner of Cleanflight. (You won't need to do this if `"autoconnect"` was selected)
3. Plug the battery into your drone.

Navigate to Motors tab

1. Go to the `Motors` tab in Cleanflight.
2. Read the safety notice and check the box that says `"I understand the risks, propellers are removed - Enable motor control"`.

Test each motor
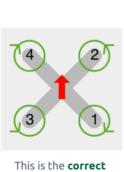
1. Slowly spin up the first motor by slowly dragging the **1** slider up
2. Use the motors diagram to verify that:

- the correct motor spins. If the correct motor does not spin, <u>reconnect the ESC wires to the Flight Controller</u> in the correct order.
- the motor spins in the correct direction. If the motor spins in the incorrect direction, take note and you will correct it later on.

> 💡 **Tip**
>
> One way to find out which direction the motor is spinning is to put a piece of tape on the motor to create a flap. Then, use a pencil or other object and touch it to the tape while the motor is spinning to see which direction it is pushed.
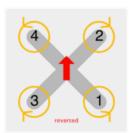
> ⚠️ **Attention**
>
> **DO NOT** follow the incorrect motors diagram. If Cleanflight shows the incorrect motors diagram, then ignore it - the diagram is a UI bug and does not affect the spin directions of the motors.
>
>
>
> This is the **correct** configuration.
>
>
>
> This is the **incorrect** configuration.

Repeat this process for all the motors.

Change the Motor Directions

For each motor that is spinning in the **incorrect** direction:

1. Disconnect any 2 of the 3 ESC wires from the motor. Keep track of which wires used to be connected together.
2. Swap the connections of the two wires.

Verify

Re-connect a power supply to your drone and check that the motors are now spinning in the correct direction. If not, repeat the swapping process.

Calibrate the ESCs

By this point, your drone's Flight Controller should be able to spin up each of the 4 motors. This is possible because the Flight Controller is sending *PWM signals* to each of the 4 ESCs, which in turn sends electrical signals to each of the 4 motors.

A **PWM signal** is a signal that communicates at how much RPM an ESC should spin a motor. For example, the PWM signal "1000" might correspond to 2300 RPM.

However, note that your drone has not 1, but 4 ESCs - which may not all have the same PWM-to-RPM understanding. For example, ESC 1 might think the PWM signal "1100" from the Flight Controller means 2300 RPM while ESC 2 might think the PWM signal "1000" means 2300 RPM.

The solution to this problem is to *calibrate* the ESCs with the Flight Controller. In this context, **calibration** means getting all the ESCs to have the same PWM-to-RPM understanding from the Flight Controller. In this section, you will calibrate your ESCs.

> ⚠ **Symptoms of no calibration**
>
> - scorching hot motors
> - a drone that lifts to one side during flight
> - motors that appear to spin at different speeds.
> - Duckiecaptain might become endangered

Disconnect Power

1. Unplug the battery from your drone.

> ⚠ **Danger**
>
> **Make sure no propellers are attached to your drone's motors!**

Launch Cleanflight

1. On your base station, open cleanflight
2. Connect the Flight Controller to a computer and click "Connect" in the top right of the screen

Navigate to Motors tab

1. Go to the **Motors** tab in Cleanflight.
2. Read the safety notice and check the box that says `"I understand the risks, propellers are removed - Enable motor control"`.

Calibrate

1. **With the battery disconnected**, drag the master slider up to `full`. All 4 motor sliders should automatically move up to full accordingly (e.g. `2000`).

2. Plug the battery into your drone.

   The ESCs will make an interesting set of sounds, kind of like music. If they do not, stop and try the previous steps again.
3. After the music stops, drag the master slider to the bottom of the bar. Correspondingly, all 4 motor sliders should automatically be at the bottoms of their bars (e.g. `1000`). The motors will make another set of sounds.
4. After the sounds stop, spin up each motor and verify it is spinning in the correct direction (i.e. according to the motors diagram in this doc).

We couldn't verify the security of your connection.

Access to this content has been restricted. Contact your internet service provider for help.

Checkpoint

Checking the ESC communication with Flight Controller

- When you connect the drone to power, the ESCs should make a "`boop boop boop`" sound, followed by a "`beep BEEEEEP`" sound.

> **ⓘ Joke**
>
> We are working on enabling a "`quack quack`" sound.

Checking the ESC ordering

Ensure the PWM wires are connected to the correct pins on the flight controller. Do this by following the PWM wires from the flight controller back to which motor the ESC is connected to. Make sure that this matches up with the image below.
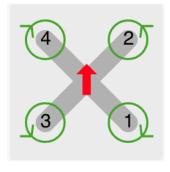
Fig. 88 Motors numbering

Checking the motor direction

Make sure the motors are spinning in the correct direction. To do this, connect your Flight Controller to Cleanflight with the **propellers off** and the battery plugged in, and spin up each motor to make sure it is spinning in the same direction as the direction shown in the image below.

💡 **Tip**

It helps to put a small piece of tape on the side of the motor to know which way it is spinning.
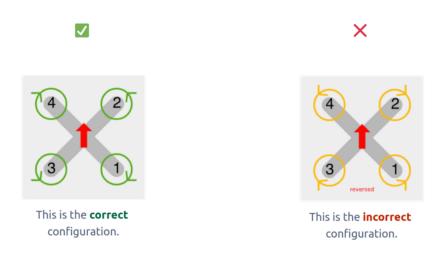
Fig. 89 Motors correct spin direction

Camera, propellers & Mounting HW

Overview

In this section of the build, you will attach the camera, and finalize the drone assembly.

What you will need	
	• Raspberry Pi Camera
	• Frame nylon standoffs
	• CW Propellers
	• CCW Propellers
	• Nylon M2.5 bolts
	• Nylon M3 bolts and nuts
	• Steel M2 bolts
	• Nylon M2 bolts and nuts
	• 8mm Wrench
	• Zip Ties
	• Double sided tape
	• Velcro strap
What you will get	• Fully assembled Duckiedrone

Hardware overview

Camera

The camera is used to measure the planar position and velocity of the drone; by planar, we mean that the camera provides position and velocities in the x and y-axis but not z (left and right, forward and back, but not up and down).

The planar velocity of the drone is found using optical flow, which is a technique that computes the change in position between two "features" over time. A feature is a distinct group of pixels in an image. The position of the drone is found by computing the change in position between two subsequent camera frames.

The drone is also able to localize the drone in a known map. This means that if you upload to the drone a photo of the surface that the drone will be flying over, the drone will be able to know where it is within this photo, or "map".

Finally, the drone is able to Simultaneously Localize And Map (SLAM) which means that it can build its own map of what it is flying over, and then localize within the map that it creates. SLAM is used on all robot vacuums that clean your house for you. Without knowing what your house looks like, the robot bumps around and remembers where it's been, creating a map.

> **→ See also**
>
> For more information about the camera, watch [these lectures](#) (you will need to create a free EdX account to view the material)
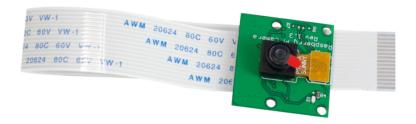
Fig. 90 Raspberry Pi Camera

> **⚠ Attention**
>
> Before attaching the camera, make sure that:
>
> - The plastic film on the lens is removed
> - The black lens assembly is attached to the PCB with the double sided tape on the back
> - The orange camera optics flex cable is connected to the connector on the PCB

Propellers

The propellers are what provide lift for the drone and allow it to fly. When the propeller spins, it creates a pressure difference between the air above and below it. The pressure below the propeller is greater than the pressure above, generating lift. Another way to think about how a drone prop works is that it

pushes air down and the opposite reaction is the thrust lifting the drone up.

The drone propeller is specified by three numbers. The propellers on your drone are 5 x 5 x 3.

1. The first number, 5, is the size, and it indicates the length of the blades; measured from the center to the tip.
2. The second number, 5, is the pitch, and it is a theoretical measurement of how far the propeller would travel through the air in one revolution. A larger pitch means more air is moved, so the propeller travels further, while a smaller pitch would move less air and therefore move less.
3. The last number, 3, is the number of blades.
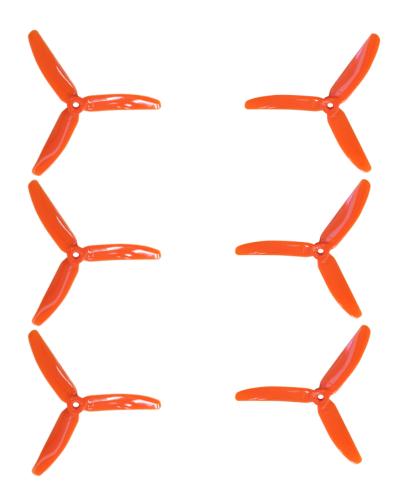
Fig. 91 Propellers

Frame nylon standoffs
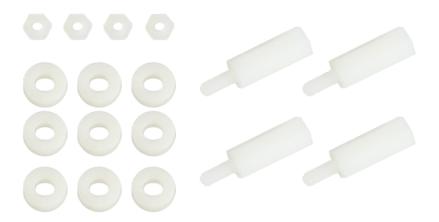
These long (30 mm) nylon standoffs are used to mount the top part of the Duckiedrone frame (the smaller rectangular piece of the frame) to the bottom part.

Fig. 92 Frame M2 nylon standoffs
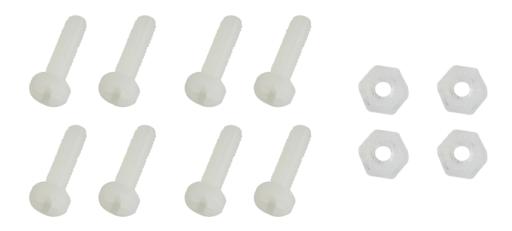
Short M2.5 standoffs and spacers

Used to mount the Raspberry Pi Hat on the Raspberry Pi.
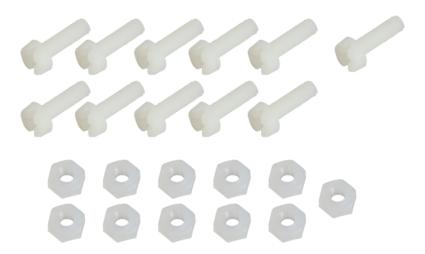
Nylon M2.5 bolts and nuts

Used to mount the frame and the standoffs.

Nylon M3 bolts and nuts

You will use one nylon bolt and nut to attach the ToF sensor to the frame.

Nylon M2 bolts and nuts

These will secure the camera to the frame.

> **ℹ Note**
>
> Old hardware revisions of the Duckiedrone only had metal M2 bolts. In this case screw the M2 nylon nuts on the metal bolts.
>
> Sometimes if screwed too tight they can short the camera and make the Raspberry Pi misbehave; if this is the case mount the camera with tape.

Wrench

This little 8-10 mm wrench will be helpful in tightening the propellers on the motor's shaft.

Fig. 93 8mm wrench

Velcro strap

Useful to secure the battery to the frame of the drone.

Battery Monitor

The Battery Monitor is a safety device that connects to the battery and monitors its voltage, emitting a very loud sound when it goes below a certain threshold voltage.

This warns you to disconnect the LiPo battery from the PDB to avoid draining it too low, thus damaging its cells.

It also displays the voltage of each cell and the cumulative voltage of the battery on the 7-segments displays on the front, just like the battery charger does.

Fig. 94 Battery Monitor

Instructions

Attach the standoffs to the bottom frame

Attach the 4 long nylon white standoffs to the bottom of the frame by screwing them with M2 nylon screws on the back of the bottom frame, as shown in the image.
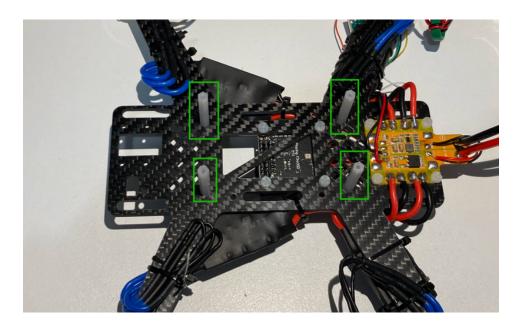
Fig. 95 Nylon standoffs mounted on bottom frame

Secure the battery to the frame

Place the battery in between the nylon standoffs and secure it to the frame by using one zip tie and a Velcro strap.

> 💡 **Tip**
>
> To allow the battery to be hot-swapped easily after a flight it is suggested to attach the velcro strap such that you can open it from the bottom side of the drone.

> ❗ **Attention**
>
> Make sure to secure the battery tightly to the frame to avoid it moving during flight.

> 💡 **Tip**
>
> You can use the holes in the rear part of the frame (towards the PDB) to attach the battery cables using a zip ties.
>
> Keep this zip tie loose to allow swapping out the battery quickly.
>
> This minimizes the sideways wobbling of the battery.
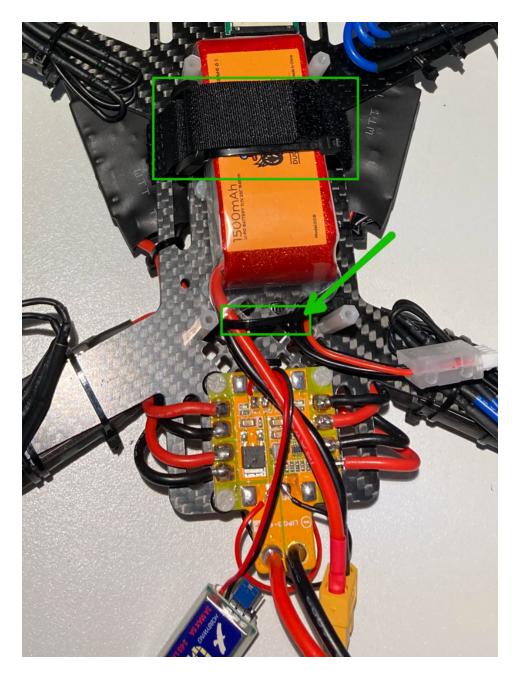
Fig. 96 Battery secured to the frame using zip ties and velcro strap

Attach the ToF sensor

> **! Attention**
>
> There's a tiny piece of yellow tape on the black tab, make sure to remove it before attaching it to the drone

1. Place the ToF sensor on the smaller left hole, making sure to align it so that the small black chip is looking down **through** the hole.
2. Using one M3 nylon bolt and nut, screw the ToF sensor to the frame.

Attach the Raspberry Pi Camera to the drone frame

The Raspberry Pi Camera has to be secured to the frame.

The Raspberry Pi Camera has tape on the back of the black lens assembly that has to be fixed to the PCB **before** mounting it on the frame

1. Peel off the tape on the back side of the lens assembly of the Raspberry Pi Camera and attach it to the PCB
2. Make sure the flex connector (called *FFC*) from the lens assembly is attached to its connector.
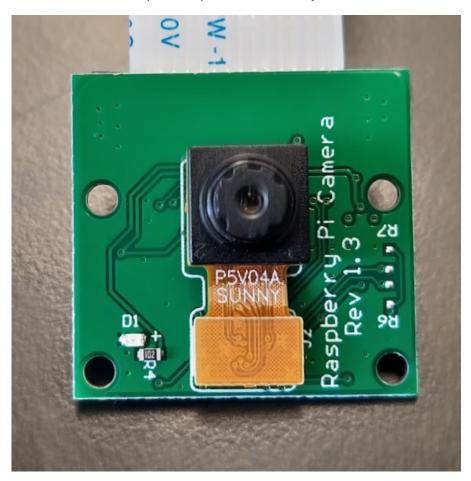
Fig. 97 Camera with lens assembly attached and connected

3. Screw the Raspberry Pi Camera to the right hole in the front part of the drone using the small nylon bolts and nuts. Attach on top, so that the camera faces downward and the FFC goes towards the drone's battery.

Fig. 98 Camera (and ToF sensor) attached to the front of the frame

> ⚠ **Warning**
>
> Do not use the metal M2 screws as they might short the PCB.
>
> Only use them if you have an old hardware revision, see Troubleshooting if you have power issues.

Mount the top frame

1. Identify the smaller black rectangular laser cut piece of the frame.
2. Attach the large nylon spacers through the small screw holes using four M2 nylon screws.

> ❗ **Attention**
>
> Make sure the upper frame is oriented in the correct direction, as shown in the figure. There is a larger rectangular opening in it that shall be oriented towards the back of the drone.
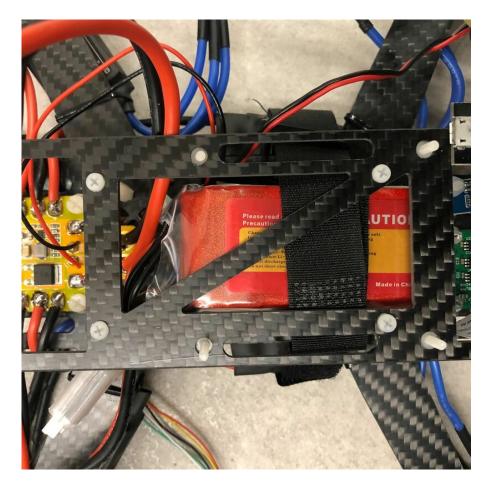
Fig. 99 Top frame mounted on the standoffs (ignore the bolts protruding)
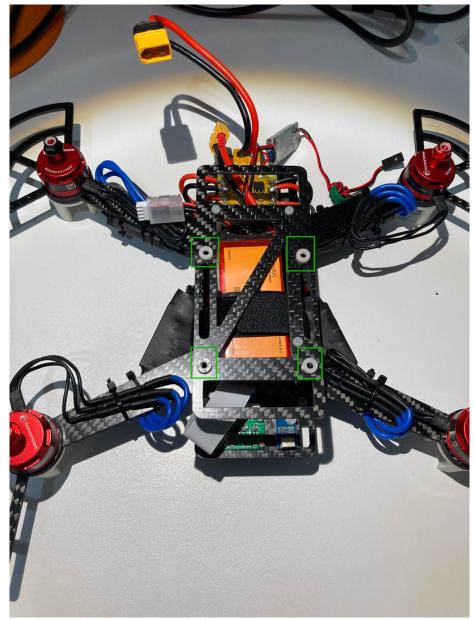
> 💡 **Tip**
>
> You might need to slightly squeeze the nylon standoffs together due to the battery being in between them.
> This is okay, and it has also the benefit of further securing the battery to the frame

Mount the Raspberry Pi

Now you are going to attach the Raspberry Pi to the top of the frame using zip ties.

1. Identify the 4 holes on the top frame and place nylon spacers on each

2. Place the Raspberry Pi on top, taking care not to move the spacers
3. Mount the Raspberry Pi on top using the 4 M2.5 nylon screws: insert them from the bottom and screw them in the short M2.5 spacers.
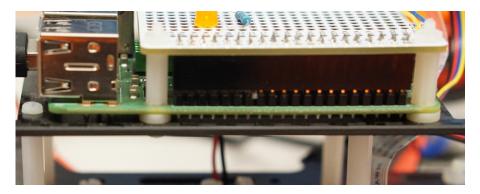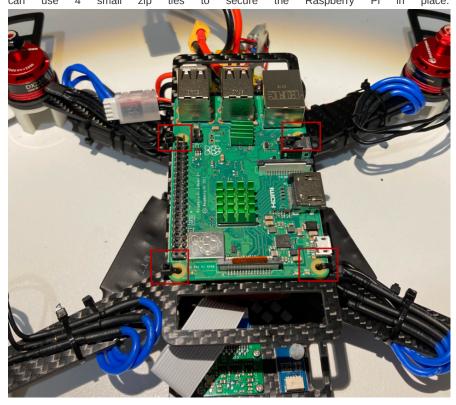
Fig. 100 Raspberry Pi mounted on the upper frame (with Raspberry Pi Hat)

Connect the camera and attach the Raspberry Pi Hat

You will now connect the camera to the Raspberry Pi and attach the Raspberry Pi Hat.
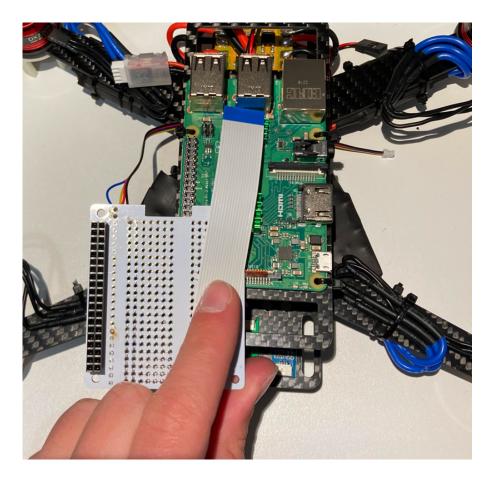
> **⚠ Attention**
>
> The camera connector cable (called *FFC*) has to go through the opening of the Raspberry Pi Hat *before* this is attached to the Raspberry Pi in order to be able to connect it to the Raspberry Pi.

1. Feed the *FFC* cable through the opening of the Raspberry Pi Hat as shown in the picture.

   > **ⓘ Note**
   >
   > Make sure that the Raspberry Pi Hat is oriented with the pins facing up and on the same side as the pins of the Raspberry Pi, and that the blue tape is facing up as in the figure.

Fig. 101 Raspberry Pi Camera flex cable going through the Raspberry Pi Hat

2.
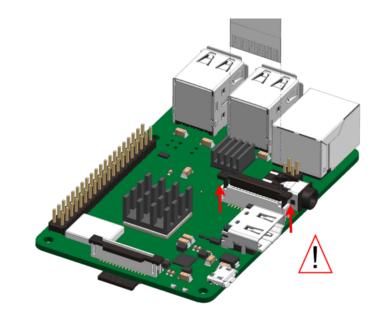
Fig. 102 Lift the camera connector plastic flap.

3.
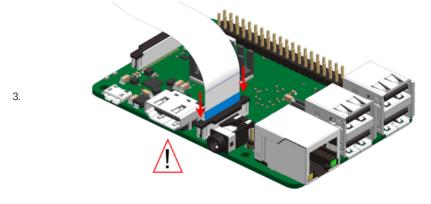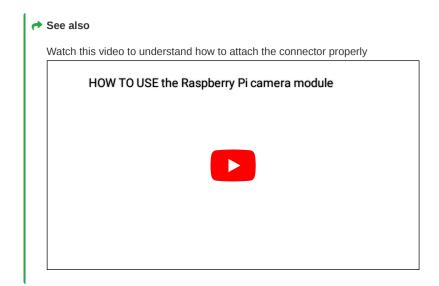
Fig. 103 Insert the camera cable in the connector with the blue side facing towards
the Ethernet port and close the plastic flap.

> **See also**
>
> Watch this video to understand how to attach the connector properly
>
>
>
> HOW TO USE the Raspberry Pi camera module

4. Attach the Raspberry Pi Hat to the GPIO pins of the Raspberry Pi being careful when handling
   the flexible camera cable.
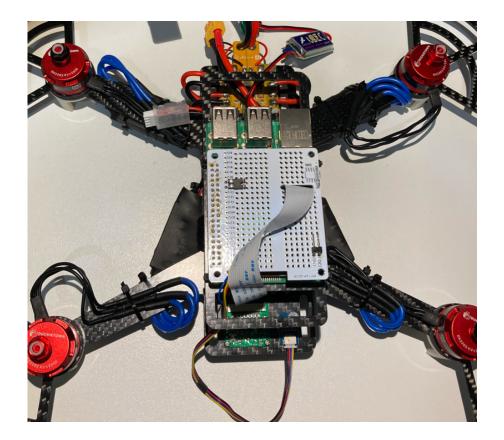
Fig. 104 Raspberry Pi Hat connected to the Raspberry Raspberry Pi with the

camera cable routed through its opening

Attach the UBEC

Use a piece of foam double-sided tape to attach the UBEC to the top of the USB ports of the Raspberry Pi as shown.
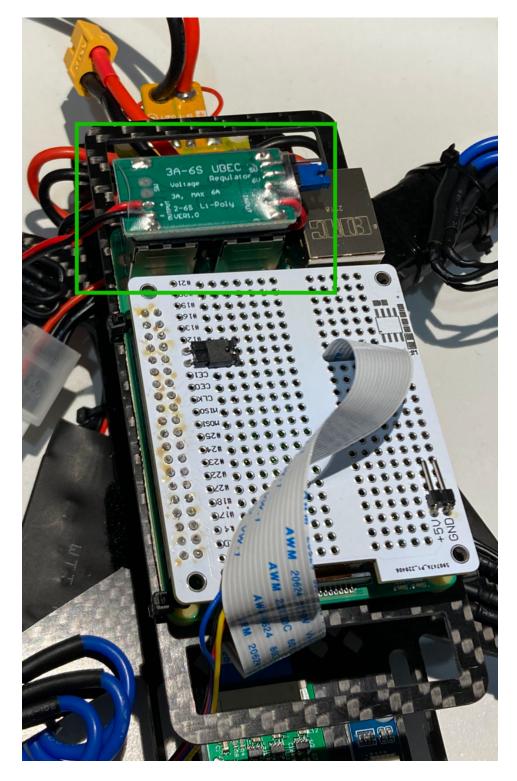
Fig. 105 UBEC attached to the Raspberry Pi USB ports

Connect sensors and UBEC

1. Connect the UBEC to the pins soldered on the `5V` and `GND` rails

> ⚠ **Danger**
>
> Be careful to not connect the UBEC to the WiFi mode selection pins as doing so will damage your Raspberry Pi

2. Attach the soldered ToF cable from the Raspberry Pi Hat to the ToF sensor.

3. Connect the Flight Controller USB cable to one of the USB ports on the Raspberry Pi

> **ⓘ Note**
>
> If you have the OSD version of the Flight Controller, you can use a piece of double sided foam tape to attach the USB connector board to the frame, right beneath the PDB board.
>
>
>
> *Fig. 106* USB board attached to the frame (OSD Flight Controller)
>
> Angle it so that the USB connector doesn't protrude much, otherwise you might encounter clearance issues with the propellers.
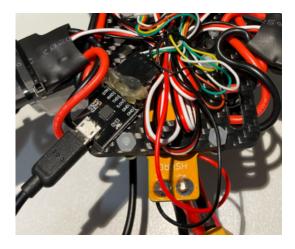
Fix cables to the frame

Make sure to attach all dangling cables using zip ties to the frame, such that no wires can hit the propellers.
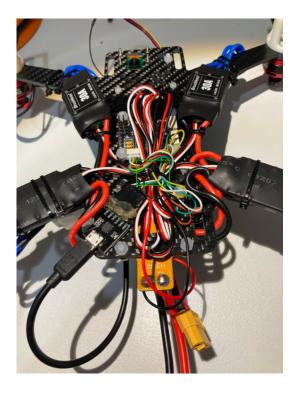
Fig. 107 PWM wires fixed to the bottom of the frame (zip tie in the center)

⚠ Danger

Make sure ESC-motor wires are ziptied down properly. If not, you risk having a short.

Fig. 108 Properly tied ESCs cables

Attach propellers

Attach the propellers to the drone so that it may fly: attach CW propellers to the CW motors, and CCW propellers to the CCW motors.

The propellers have small arrows on them in the center to indicate which type they are.

Fig. 109 Arrows on the propellers indicating spin direction

The nuts for the motors that spin CCW tighten when turned CW, and the ones on the motors that spin CW tighten when turned CCW.

Use the 8 mm wrench to tighten the bolts down so that the bottom of the propeller is flat on the top of the motor.

Screw bolts down tightly, but not so tight that you could not remove the propellers if you had to.

Fig. 110 Propeller attached to the motors

Battery Monitor

In your Duckiebox there is one item that we haven't used so far: the Battery Monitor.

> **ℹ Note**
>
> This is optional and not required for flight.

To use this safety device you are going to have to connect it to the 4-pins connector of the LiPo battery, when it is not charging:

1. Identify the row of pins at the bottom of the Battery Monitor.
2. Connect the 4-pins connector to the left-most pins, having the first pin connected to the black (-) wires of the connector.
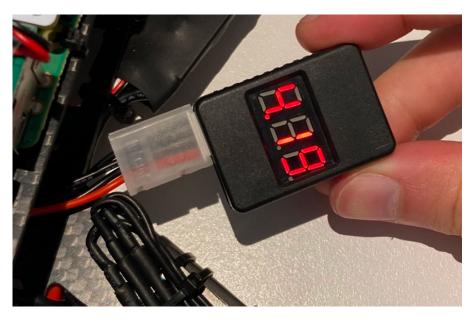
Fig. 111 Battery connected to the Battery Monitor

3. Secure the battery monitor to the chassis of the Duckiedrone.

> ℹ️ **Note**
>
> Due to the length of the battery cable and the limited space on the frame there is not a great mounting spot.
> You should pick one making sure that:
> - The Battery Monitor and battery wire do not interfere with the propellers.
> - The Battery Monitor will not move during flight.
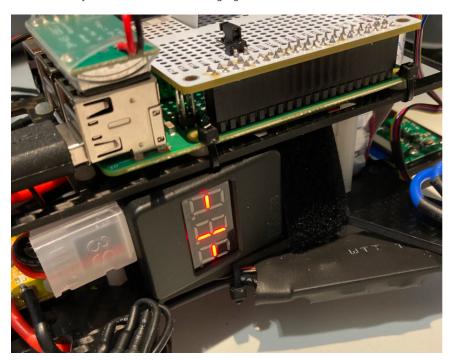
Fig. 112 A possible location to attach the Battery Monitor

> 💡 **Tip**
>
> You can use some double-sided tape to fix the battery monitor.

Checkpoint

Cabling

Check that the cabling doesn't get in the way of the drone's propellers.

> ⚠️ **Danger**
>
> - Make sure the ESCs cables are attached to the drone's arms
> - Make sure the battery connector doesn't get in the way of the propellers.
> - Make sure the USB to microUSB cable connecting the Flight Controller to the Raspberry Pi doesn't obstruct the propellers

> ⚠️ **Danger**
>
> Make sure ESC-motor wires are ziptied down properly. If not, you risk having a short.

> 💡 **Tip**
>
> Spin the propellers manually with your finger. Ensure no wires are hit by the propellers.

Make sure no wires or parts are dangling from the drone frame.

Propellers

> ℹ️ **Check before you continue**
>
> **Propeller Direction**:
>
> 1. The arrows on the propellers should be on visible from the top of the drone
> 2. The arrows should be going in the same direction as the arrows on the motors.
>
> **Propeller Attachment**:
>
> 1. The propellers must be flat on the base of the motor
> 2. Make sure there is no gap between the propeller, the motor, and the motor nut.
> 3. Holding the motor still, try to spin the prop and make sure the props cannot spin around the motor shaft; the motors and the props should spin together.

First boot

There is only one first time you can connect to your Duckiedrone. Savor the experience.

What you will need	• An assembled `DD21` • A `DD21` initialized SD card, see [Software initialization](#)
What you will get	• A live `DD21`

Before getting started

The first time a newly flashed SD card is inserted in the Duckiedrone a special "first boot" procedure is executed.

> ℹ️ **Note**
>
> The first boot procedure will take roughly 10-15 minutes, during which your Raspberry Pi might look unresponsive. It is crunching as fast as it can, do not worry.

During this process the Duckiedrone will require a stable power source.

> ⚠️ **Attention**
>
> Make sure you have a wall outlet power adapter, e.g., a phone charger (5V, 2-3A) or a fully charged Duckiebattery before starting the process.

Do not power the Raspberry Pi just yet.

Getting started

1. Make sure the Raspberry Pi is **not powered.**
2. Make sure you have a wall adapter or fully charged battery available.
3. Networks are typically one of the biggest headaches in robotics. We offer different network configurations to minimize these headaches. If you are not sure which choice to make, the right answer typically is: if you are in a university go for AP mode. If you are at home go for CL mode. In both cases, you need to place the jumper accordingly on your 5 & 6 pins before getting started.

Access point (AP) mode   Client (CL) mode

If you want to have the drone emit its own Wi-Fi network that your base station can connect to. This is the go-to choice if you do not have a network (WLAN) or admin access to the existing network where you are operating.

1. Pros:
   - You can connect to your drone without the need for a pre-existing existing network infrastructure
2. Cons:
   - You might not have access to the internet (which is not required, but useful during development), unless your base station has a secondary network adapter (e.g., an Ethernet port) and you bridge the connection.
3. How to:
   - **short** pins 5 & 6 on the breadboard by using the provided jumper.
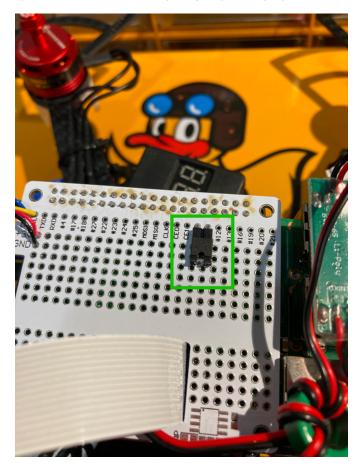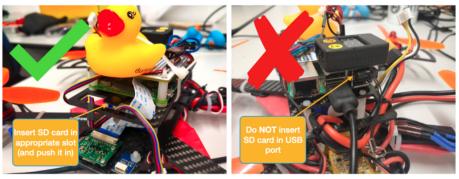
Fig. 113 Pins 5 & 6 shorted.

4. If you haven't already, insert the initialized micro SD card inside the micro SD card slot of the Raspberry Pi, as shown [here](#).

> ⚠️ **Attention**
>
> **Do not** connect the SD card inside the adapter to a USB-A port of the Raspberry Pi.

5. Power the Raspberry Pi
   - You will see a red and green light turning on the Raspberry Pi. The green light shows computation usage. You should expect it to become solid green for several minutes.

   > ↪ **See also**
   >
   > Watch a short video of a busy Raspberry Pi booting up for the first time: [Raspberry Pi first boot](#)

6. Wait until the first boot sequence is complete:

Access point (AP) mode   Client (CL) mode

Scan available networks through the base station: once the booting procedure is complete you will find a network called `duckietown-<hostname>-ap`, where `<hostname>` is the name of the robot, as determined during the initialization procedure. The default name is `amelia`.

Congratulations, you are now ready to connect to your Duckiedrone for the first time!

Troubleshooting	
SYMPTOM	I disconnected my `5` & `6` pins but cannot see my robot on the network.
RESOLUTION	Sometimes things go awry during the first boot. It is possible that the Wi-Fi detection container times out. Search for a `duckietown-hostname-ap` network instead. Reboot the Duckiedrone (with disconnected pins) to have it join the configured existing network.

First connection

You are now ready to connect to your Duckiedrone through the Duckietown Dashboard.

What you will need	- A live `DD21` ([First Boot](#)) - A base station with wireless connectivity, or a pre-existing network
What you will get	A fully operational `DD21`

Connecting to the Duckiedrone

Make sure you are on the same network as your Duckiedrone:

Access Point (AP) mode   Client (CL) mode

Connect to `duckietown-<hostname>-ap` if the drone is in AP mode, where `<hostname>` is the robot name chosen during the initialization procedure.

If you forgot to change it, the default hostname is `amelia`.

Accessing the Duckiedrone functionalities

> ℹ **Cheatsheet**
>
> Default robot name: `amelia`
>
> Default ssh user name: `duckie`
>
> Default ssh user password: `quackquack`
>
> Ssh always possible: `ssh duckie@amelia.local`
>
> **Default** access point (**AP**) network configuration:
>
> - SSID: `duckietown-amelia-ap`
> - Password: `quackquack`
>
> **Default** client (**CL**) network configuration:
>
> - SSID: `duckietown`
> - Password: `quackquack`

Troubleshooting

Troubleshooting	
SYMPTOM	I cannot connect to my Duckiedrone in AP mode.
RESOLUTION	Try using client mode and shut down the docker container `dt-access-point` through the Portainer interface (accessible through your browser from your base station at `<hostname>.local:9000`)

Troubleshooting

It is very very common for something to go wrong during your build. Count on it. The goal is to systematically figure out what is wrong and fix it. Mastering this process is essential to any robot project, because things will always go wrong.

The high-level bit when troubleshooting is to try to isolate the problem systematically. Rather than simply redoing part of the build or replacing a part, try to systematically verify what parts are working and what parts are not working. Your drone will not fly until everything works!

Power issues

Troubleshooting

SYMPTOM	My Raspberry Pi does not boot.
RESOLUTION	You should verify that each part of the drone is receiving power. The Raspberry Pi indicates it has power with a *red* power LED.
	If your Raspberry Pi is not powering on, verify with a multimeter that the Raspberry Pi pins are receiving the right voltage on input. You can find a mapping of the GPIO pins [here](#). Verify that each power pin is receiving 5 Volts compared to each ground pin with the multimeter.
	Make sure you did not use metal screws to mount the camera to the frame as they can cause a short.
	If your Raspberry Pi is receiving 5 volts on its power/ground pins, but no red light turns on, then it might have gotten fried. This can happen if you wire or short the power/ground pins on it, so try replacing the Raspberry Pi.

Troubleshooting

SYMPTOM	The motors do not turn on.
RESOLUTION	The motors indicate they are receiving power by beeping once. You can also check each part with the multimeter. Verify that there is a 12 Volt connection between power and ground on the power distribution board. And verify that the Raspberry Pi is receiving 5 volts from the UBEC.

Raspberry Pi Issues

Most of the next debugging steps require getting "into" your Raspberry Pi using ssh, check the [First connection](#) chapter if you are not connected.

Troubleshooting

SYMPTOM	My Raspberry Pi is receiving power and turning the red light on, but it doesn't boot.
RESOLUTION	Something might be wrong with your SD card.

- Verify that your SD card has the correct image flashed on it.
- Check that the SD card is inserted in the Raspberry Pi so that it can boot.

If all this does not work, find a keyboard and monitor to plug the Raspberry Pi into during boot, to see what is going on during the boot process.

There may be an error message being printed on the screen that will give more information.

Camera

Now verify that the camera is working.

Troubleshooting

SYMPTOM	I'm in the Screen session, but the camera node is not starting.
RESOLUTION	Use `raspistill` to verify that it is plugged in.
	You can try `raspi-config` and make sure it is enabled. Also it is very common for the camera cable to be plugged in backwards, or plugged into the wrong slot on the Raspberry Pi. (There are two possible slots that fit the cable.) Make sure it is plugged into the slot marked `"camera"`, and that the cable is facing the right way. (The metal pins of the cable should be facing the pins in the slot.) Make sure it is seated all the way.
	If none of these things make `raspistill` work, try plugging into your Raspberry Pi someone else's camera, and someone else's Raspberry Pi into your camera, and try all of the above debugging steps. You can also check if the cable is bad. For example if you bend the cable too much, it will fatigue and then break the wires; or if a prop strikes the cable, it might cause the cable to break.
	If your Raspberry Pi and cable work with someone else's camera but not yours, try replacing the camera. If your camera and cable work with someone else's Raspberry Pi but not yours, try replacing the Raspberry Pi.

Flight Controller

Finally check the Flight Controller. When the Flight Controller connects to the motors, it will make a "low beep, high beep" sound. So verify you hear the "do do do" from the motors, indicating they have power, and then the "low, high" indicating the Flight Controller can talk to them. If that doesn't work, check the connection between the Flight Controller, ESCs, and motors.

Inside the Raspberry Pi, make sure you can calibrate the accelerometer, and run the Flight Controller node. If those don't work, go back and recheck your Cleanflight configuration.

Flight Issues

Before each flight, physically inspect the drone.

Make sure that:

- your camera is mounted firmly, pointed downwards.
- the range sensor is pointed downwards and hasn't gotten rotated.
- the Flight Controller board is level and firmly attached, or the IMU and gyroscope will return incorrect readings.
- each propeller is tightened down all the way.

Any of these issues could cause poor flight behavior.

If your drone flips the first time you try to take off, the motors are spinning the wrong way, or the props are on upside-down. If your drone makes funny noises when arming, either the props are not tightened all the way, or they are striking a wire. Tape everything down as much as possible.

If the drone is not stable during flight, you should make sure that the props are all tightened down. Make sure the ESCs have been calibrated following as in [Calibrate the ESCs](#).

A well-tuned drone can hover with velocity zero with some drifting, but not too much. It should be able to hover with position hold indefinintely.

Datasheets

Flight Controller

- [SP Racing F3 OSD](#) (USB port on a separate PCB)
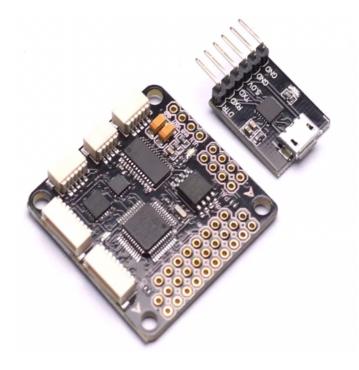
Fig. 115 The "OSD" version has the USB port on a separate PCB.

- [SP Racing F3 Acro](#) (USB port on the same PCB)

Fig. 116 The "Acro" version has a mini USB port on the board.

Camera

- [Arducam 5mp 1080p OV5647](#)

Time-of-Flight (ToF)

- [VL53L0X] (https://learn.adafruit.com/adafruit-vl53l0x-micro-lidar-distance-sensor-breakout)

Fig. 117 ToF Sensor

Raspberry Pi 3B+

- [Raspberry Pi 3B+](#)

Power Distribution Board (PDB)

- PDB-XT60

Electronic Speed Controller

- XXD 30A 2-4S ESC Brushless Motor Speed Controller

UBEC

- Hobbywing 5V 3A UBEC

Environment setup

Development container setup

To make it possible for you to develop your own code on the drone, you need to set up a docker workspace from the source code of the repository `pidrone_pkg`.

1. Run `ssh duckie@[yourdrone].local` from your base station to ssh into your drone. The password is `quackquack`.
2. Clone the repository to your Duckiedrone's SD card and switch to the branch `ente`:

   ```
   mkdir -p catkin_ws/src && cd catkin_ws/src
   git clone https://github.com/h2r/pidrone_pkg && cd pidrone_pkg
   ```

   > **ⓘ Note**
   >
   > The next step will take a long time because it has to download all the dependencies to build the image. Make sure your Raspberry Pi is plugged into an external USB power supply.

3. Finally build the Docker image needed to run the software (which is back on an older version of ROS, ROS Kinetic).

   ```
   rake build && rake create
   ```

4. You also need to download the `raspicam_node` package to be installed later:

   ```
   cd ~/catkin_ws/src/
   git clone https://github.com/UbiquityRobotics/raspicam_node
   ```

5. You can start the container and go inside it by running, from the `pidrone_pkg` directory:

```
cd ~/catkin_ws/src/pidrone_pkg
rake start
```

6. Once in the container, you need to install the `pidrone_pkg` and `raspicam_node` ROS packages. To do this execute:

```
cd ~/catkin_ws
catkin_make && catkin_make install
```

7. Now that all the packages are installed, to access the workspace you will use to control the drone, run:

```
screen -c pi.screenrc
```

This will start a screen session with each of the ROS nodes needed to run the drone and make it fly.

> **ⓘ Note**
>
> Anytime you'll want to interact with the drone software you will first need to start the container by running `rake start` from the `~/catkin_ws/src/pidrone_pkg` directory and then, inside the container, running `screen -c pi.screenrc`.

Flying Your Drone

What you will need	- Fully assembled drone - Charged battery - Base station - Safety goggles - Highly textured planar surface
What you will get	- Flying your Duckiedrone - Duckiecaptain will be delighted

Environment Checks

> **⚠ Warning**
>
> Flying your Duckiedrone is safe **only** when done in an appropriate environment. Make sure:
>
> - You are in an open space that is free of obstructions
> - You've alerted those around you that you are going to fly and have told them to clear the area
> - You are wearing safety goggles
> - The surface you are flying over is not reflective, and is not uniform in details. Ideally, you've created a highly textured planar surface, which is a poster board with a bunch of scribbles and shapes. A patterned carped will work fine as well.
>
>
>
> **Fig. 118** Example of a highly textured planar surface
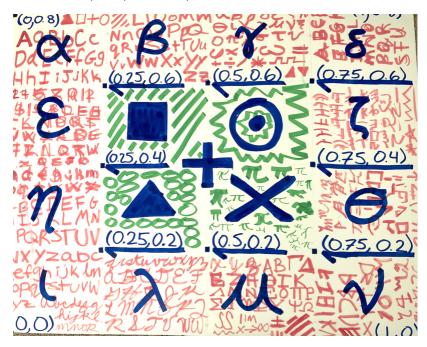
Hardware Checks

If not already, disconnect the battery before performing the following safety checks.

Wire Management

Spin the props with your finger and make sure there are no wires in the way. If wires do get close, use a zip tie to hold the wires to the frame, away from the props.

USB Connection

Make sure that the Flight Controller USB cable is plugged into the Raspberry Pi. Any of the USB ports will work.

Software and Sensors Checks

Power

Plug the battery into your drone.

Screen Environment

Connect to your drone using ssh by running `ssh duckie@<hostname>`.

Navigate to the `pidrone_pkg` directory and run `screen -c pi.screenrc` to start the Screen session. Screen is a program that allows you to run multiple terminals in one ssh session. The screen will persist even if the ssh session is disconnected and you log in again. However, it will disappear if you

reboot the drone.

Fig. 119 Screen terminal interface

Make sure to only have one screen session running at a time.

> 💡 **Tip**
>
> When in doubt, run `screen -x` to see what screen sessions are running and reconnect to a session that you have disconnected from.

The screen session will start the ROS nodes needed to run your drone and make it fly. However it will not start the flight controller node: we ask you to start this node manually because you should only run it if you are prepared for your propellers to spin.

The screen 'escape' key is back-tick. The "tick" ` key is typically located to the left of the `1` key, and is on the same key as the tilde `~`.

> ℹ️ **Note**
>
> You will not see anything appear when you are typing `` ` ``1

To navigate the screen tabs, use the following commands:

- `` ` ``1, `` ` ``2, etc to navigate to a particular window.
- `` ` ``" : Open tab menu
- `` ` ``n : Go to next tab
- `` ` ``p : Go to previous tab
- `` ` ``' : Ability to enter full numbers (such as `10` and `11`)

Start the Flight Code

1. Go to a free screen (`` ` ``11)
2. Calibrate the accelerometer by typing `python scripts/calibrateAcc.py` and pressing enter.
3. Start up the Flight Controller node by going to the `1*$FC` screen and pressing enter.

Web Interface

The Web Interface allows you to control the drone from your base station. To open it up follow these steps:

1. On your base station, clone the git repository and switch to the correct branch:

```
git clone https://github.com/h2r/pidrone_pkg
cd pidrone_pkg
git checkout ente
```

2. In your file manager browse to the directory containing the repository you just cloned and open the file `index.html` in `/pidrone_pkg/web/`.
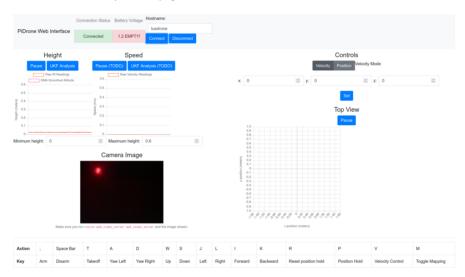
Fig. 120 Duckiedrone web interface

3. At the top of the web interface type the `hostname` you assigned to your Duckiedrone in the `Hostname:` field and press the `Connect` button to connect to your drone.

> **ⓘ Note**
>
> If the web interface does not say `"connected"`, then wait about 5 seconds and refresh the page.

Check the ToF Sensor

While looking at the `Height` graph in the web interface, move the drone up and down and make sure you see changes in the height graph on the web interface

Check the Camera

While looking at the X Velocity Chart in the web interface, move the drone to the left and right and verify that the graph changes. While looking at the Y Velocity Chart in the web interface, move the drone forward and back and make sure the graph changes.

Check the ROS Nodes

In the terminal of the code editor, go through each of the screens using `n, where n is a number between 1-8, and make sure there are no errors printed out. It is normal that there may be an error at the top of each screen that says something about not connecting to ROS master, but that is OK because it takes a bit for ROS to startup. Each screen should say "started" or "publishing".

The screen terminals contain the following nodes:

1. `1 is the flight controller node; you've already seen this one.
2. `2 is the PID controller; this sends the roll, pitch, yaw, and throttle commands to the flight controller
3. `3 is the state estimator; this combines sensor data to estimate the state (position, velocity, orientation) of the drone
4. `4 is the camera node; this gets velocity and position data from the camera
5. `5 is the Optical Flow node, getting the info from the Raspicam.
6. `6 the Rigid Transform node computes the pose of the DuckieDrone
7. `7 is the ToF node; this gets height readings from the ToF sensor
8. `8 is the rosbridge node, exposing ROS topics and parameters to the web interface

Controlling the flight

Familiarize yourself with the keyboard commands

Find and read the keyboard commands to control the drone at the bottom of the web interface.

> **ⓘ Note**
>
> The keyboard focus must be on the web page for these commands to work: you may need to click on the screen before typing the first keyboard command.

The main keys you'll need are the `spacebar` (to disarm the drone), the semicolon `;` (to arm the drone), and `t` (to takeoff).

> **⚠ Danger**
>
> If anything goes wrong **be prepared** to immediately hit the `spacebar` to disarm the drone.

Other useful keys are `i j k` and `l` which allow you to fly the drone around. The drone will attempt to hover in one place, but if it moves too much to one side, you can steer it back to the center using these keys.

Orient the drone

Rotate the drone so that the camera end is facing away you and the PDB is facing away from you. In this way the keyboard controls (`i`,`j`,`k`,`l`) will match the drone's orientation.

They have the following map:

- `i` - forward (Camera side)
- `j` - left
- `k` - backward (PDB side)
- `l` - right

Takeoff sequence

1. First arm your drone by pressing `;`. The propellers should start spinning slowly.

> **⚠ Danger**
>
> If they spin fast, or you hear strange noises, **immediately** disarm the drone.

2. Disarm your drone (`spacebar`) and ensure that the propellers slow down almost immediately and then stop spinning. If there is a delay, then there is likely network latency and this could cause flight issues. If you are connected to the drones network and there is delay, try restarting the Pi. If you are connected to your home network and there is delay, restart the Pi and use the drone's network to fly instead.
3. Try arming (`;`) and disarming (`spacebar`) again to ensure that the drone is responsive.
4. If all goes well, arm the drone again, then press `t` to takeoff. Be prepared to disarm the drone if anything goes wrong.
5. Move in the plane using `i`, `j`, `k`, `l` on the keyboard. When not moving the drone will try to maintain zero planar velocity but may drift.

Congrats on flying!

Troubleshooting	
SYMPTOM	After arming the drone the propellers do not spin at all.
RESOLUTION	Check the Flight Controller node screen. This is where any error message will be printed out.

Flight Controller PID Tuning

What you will need	• A fully operational `DD21`
What you will get	• A `DD21` that flies stably and responsively in manual mode

For a consumer level drone, when you launch it for the first time (e.g. with virtual joysticks on phones, or with remote controls), you would expect it to take off and fly more or less stably, instead of shaking and going out of control. This is indeed what a Duckiedrone is capable of doing, too. In this section, we will have a look at the parameters on the flight controller that make it happen. And you will be able to play with them!

Expectation with good parameter settings

The Flight Controller (FC) runs high-frequency control loops to stabilize the drone. It utilizes the sensors (e.g. IMU & Gyro) to estimate the state of the drone. In the **Angle Mode** that we configure the FC to, the following is expected:

- On the roll / pitch axis:
  - when commands are issued, the FC tries to make the drone reach the commanded roll/pitch as soon as possible, and minimize the fluctuation and stay at the designated angle.
  - When no commands are given, the drone would try to return to a neutral roll and pitch.
- On the yaw axis:
  - When commands are issued, the FC tries to keep the yaw rate to the command.
  - When no commands given, the drone should not rotate along the vertical axis.

PID terms in drones and tuning

Primarily, a **proportional–integral–derivative controller** (PID controller) is used on the FC to achieve these. Here are some materials helping explain the influence of `P`, `I` and `D` terms, in the context of drones, and how should they be tuned based on the behaviors of the drone.

- FPV Drone PID Explained by Oscar Liang
- CleanFlight PID Tuning on OpenTXU by John Case
- A video on recognition of PID problems. It is not necessary to use the string method to tune a Duckiedrone, but the behaviors are also clearly presented in that part, when certain parameters are far from good values.

Setting your PID values

How to update in CleanFlight

Fig. 121 GUI tab in CleanFlight to set the PID values (with recommended starter

values)

As shown in Fig. 121, here is how to set the PID values:

1. Connect the FC to the workstation.
2. Go to the "PID Tuning" tab.
3. Change the values and click "Save" on the bottom right of the tab

Recommended starting values

Axis	P	I	D
Roll	80	60	75
Pitch	80	60	75
Yaw	70	45	

In addition, change the `Angle Limit` to 55.

Finally, click `Save`.

Your tuning loop

> ℹ **Note**
>
> In your own PID tuning process, you should only change the `P`, `I` and `D` on ROLL, PITCH and `P` and `I` on YAW. Only change the other values if you are sure what they mean and what you are doing.

After apply the recommended starter values, with the knowledge from the previous subsection, you can keep trying the following for improving the basic manual performance of your Duckiedrone:

- Flying the drone manually and look out for erroneous behaviors
- Connect the FC to Betaflight, and adapt the PID values

Common issues

Troubleshooting	
SYMPTOM	The red light on the Raspberry Pi : • is blinking • does not turn on
RESOLUTION	The Raspberry Pi is not receiving enough power. • Check that the voltage coming out of the UBEC is a constant 5V

- Make sure that the Raspberry Pi Hat is attached to the Raspberry Pi all the way (there is no gap between the GPIO pins and the Raspberry Pi Hat pin header).
- Make sure that the `OUTPUT` side of the UBEC is attached to the Raspberry Pi Hat, and the `INPUT` side is soldered to the PDB
- Make sure that there is not a short between the power and ground rails on the Raspberry Pi Hat.
- Make sure there are no stray wire hairs that are shorting out the `5V` and the `GND` rails on the Raspberry Pi Hat

Troubleshooting

SYMPTOM

The flight code does not start

RESOLUTION

Make sure you are running the start script in the correct directory: `~/ws/src/pidrone_pkg`, **inside** the container.

Troubleshooting

SYMPTOM

The `roscore` screen does not startup.

RESOLUTION

Quit the screen by typing the tick (`` ` ``) followed by colon (`:`) and the type the word `quit` and press enter. You will not see the tick and colon typing, but you will see "quit" as you type at the bottom of the screen.

Check if there re multiple screens running by typing `screen -ls` in the terminal.

You will need to quit each socket found so that only one screen session is running. To do so:

For each socket found, there is a name that looks like `2503.pts-0.duckiesky-drone`. The four numbers at the beginning, `2503` are the session id. Run this command for each session id:

`screen -S [session id] -X quit`

For example, for this session it would be `screen -S 2503 -X quit`
Be sure that the 'S' and 'X' are capitalized.

Troubleshooting

SYMPTOM

The flight controller node does not start correctly.

RESOLUTION

The last line printed out should be `/pidrone/battery`.

If the last line printed out says that the USB is not plugged in:

- make sure that the USB is plugged into any one of the four USB ports on the Pi.
- make sure that the micro USB is plugged into the flight controller.
- rerun `python flight_controller_node.py`.

  If you get the error again:

- make sure that the flight controller is lighting up in some way. If it is not, the micro USB port on the flight controller may be broken.
- try wiggling the micro USB end or using a different USB port on the Pi. If the flight controller never lights up, it may need to be replaced.

Troubleshooting

SYMPTOM	The PID controller node is not running.
RESOLUTION	The last line printed out should state `PID Controller Started`. If this is not the case:

- Try quitting the script with `ctrl-c` and rerunning it

Troubleshooting

SYMPTOM	The state estimator is not running.
RESOLUTION	In the state estimator screen the last line printed out should state `Starting filter`.

- make sure the flight controller node is running, as data is needed from the imu to start the filter.
- continue with the checks to make sure the other sensors are working, then try rerunning this script

Troubleshooting

SYMPTOM	The vision node is not running.
RESOLUTION	The last line printed out in the vision node screen should state `Vision started`.

If the last few lines print: `out of resources other than memory`, then the issue is the physical connection from the camera to the Raspberry Pi.

- make sure that the sunny flap is shut (push on the small silver rectangle on the front of the camera and make sure it's attached firmly)
- make sure that the camera cable, or FFC (flexible flat cable) is fully inserted into the camera and the Raspberry Pi.
- On the Raspberry Pi, make sure the blue side of the FFC is facing towards the USB ports
- On the camera, make sure the blue side of the FFC is facing up
- Make sure that there are no holes or rips in the FFC. This is a common issue: a crash could have caused a tear, or a hole could have been made when soldering. If this is the case, you will need a new FFC
- rerun the vision script

Troubleshooting

SYMPTOM	The rosbridge node is not running-
RESOLUTION	The last line printed out should include the phrase `Rosbridge websocket server started on port 9090`

If not:

- make sure you have no other programs using the same port as the Rosbridge server (9090)

try rerunning the script.

Troubleshooting

SYMPTOM   The web interface does not say `connected` at the top.

RESOLUTION
- try using Google Chrome to open the web interface

  wait another 10 seconds and try refreshing again.

Troubleshooting

SYMPTOM   When I move the drone up and down the height reading graph does not change.

RESOLUTION   Recheck the ToF node. Rerun the node if needed and then refresh the browser.

Troubleshooting

SYMPTOM   There is no data on the x and y velocity graphs or they do not change when moving the drone

RESOLUTION   Check that the vision node in Screen is running. Rerun the node if needed and refresh the browser.

Troubleshooting

SYMPTOM   There is a long delay between when you move the drone and when the graphs change.

RESOLUTION   If you are running the drone in managed mode, this is probably due to latency in your home network. Take some of the devices offline, or restart the drone and fly in AP mode.

Troubleshooting

SYMPTOM   The motors on the drone do not spin when armed on the web interface

RESOLUTION   Plug the USB into a computer with Cleanflight Configurator and verify the following settings in Cleanflight:

- the yaw is flipped 180 degrees in the `"Configuration"` tab
- the receiver is set to `MSP RX Input` (so that the FC can receive commands from the Raspberry Pi over USB) in the `"Configuration"` tab
- the ESC/Motor protocol is set to PWM in the `"Configuration"` tab.

Troubleshooting

SYMPTOM   The drone does not get off the ground when commanded to takeoff.

RESOLUTION
- make sure that the arrows inscribed on the propellers are visible from the top of the drone
- make sure that the arrows on the props are in the correct direction
- Take off the propellers from your motors and plug the battery into your drone. In Cleanflight Configurator, navigate to the motors tab, click `"I agree to the risks"`, and try to spin up each motor. Make sure that each motor spins in the correct direction.

- Make sure that when you spin up motor 1, the correct motor spins (the bottom right). Do this for all of the motors.

Introduction to Learning Experiences

This section lists all Duckietown learning experiences (LXs):

- **Supported**: these are polished LXs that are actively maintained. You should be able to follow the instructions and obtain the expected outcomes.
- **Experimental**: these LXs are either under development or worked at some point in the past. This content is not actively supported at this point in time but might work with some particular attention. We leave this non-polished content here to provide ideas on projects or classes. We welcome contributions to these materials.

Supported Learning Experiences

What you will need	An internet connection; About 10 minutes; A computer with the Duckietown Shell command [installed and correctly setup](#); Duckietown token correctly set up;
What you will get	Knowledge

Learning experiences for the Duckiedrone are hosted on GitHub. Instructions for engaging in them are available at the following links:

- [Linux and Networking](#)
- [Introduction to ROS](#)
- [Duckiedrone's Sensors - IMU](#)
- [Duckiedrone's Sensors - Time Of Flight Sensor](#)
- [Duckiedrone's Sensors - Camera](#)
- [Duckiedrone height PID tuning](#)
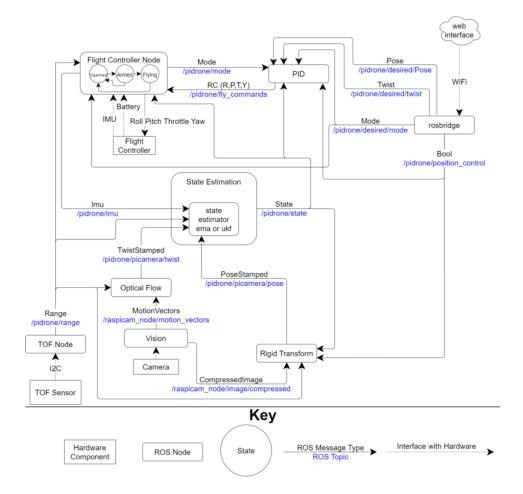- [Duckiedrone Kalman Filtering](#)

Introduction

In this section, we will introduce you to the software of the drone, and how it interacts with the hardware you assembled. First, we will offer a brief explanation of the [Robot Operating System (ROS)](#), and the ways its tools are specifically implemented on the DuckieDrone to create the programs that allow the drone to fly autonomously. Next, we will look at a diagram which provides a visual overview of how all of the components needed to fly the drone fit together. Finally, we will describe each ROS node that is running while the drone flies to convert the data from the sensors into controls to the actuators (the four motors). In doing so, we will look closely at each component to understand its purpose, where it exists in the code, what ROS topics it interacts with, and what hardware it interfaces with.

The software architecture is designed to be modular and so that the code is extensible, or it is easy to replace each component when a better or different one when desired. The sensors are interfaced into ROS nodes which publish the data for the state estimator to combine into and estimate of the state of the drone. Based on this estimate, the controller (ours is a PID) is used to move the drone to a desired velocity or position based on the inputs from the web interface or a higher level behavioral engine. The mode controller ensures that the desired mode changes (such as 'Armed' to 'Flying') are legal, and will stop the flight if any safety checks failed (for example, the web interface heartbeat stops).

1.1. Diagram of the Software Architecture

This is the general layout of the software architecture. Take a look at the key to better understand its meaning.

ROS

Overview

The Robot Operating System (ROS) is widely used robot middleware that makes communication between processes, known as *nodes*, extremely easy through the use of ROS *topics* which can be *published* and *subscribed* to. Each topic has a certain *message* type that tells the publisher or subscriber what kind of data can be sent and received from over a topic. The components of ROS are described in more detail below

General Components:

ROS Master

In order for ROS nodes to communicate to each other, there must be a master node running which all other nodes register to. A better and more detailed description can be found on the ROS wiki site [here](). A ROS master node is created by running the command `roscore` in the terminal of a computer that has ROS installed. On the DuckieDrone, `roscore` is called in `0 of the screen.

ROS Nodes

ROS nodes are programs that communicate with other programs via publishing and/or subscribing to ROS topics. A better and more detailed description of nodes is found on the ROS wiki site [here]() and [this link]() includes a shorter description along with brief descriptions of other key ROS components. On the DuckieDrone, each window of the screen is a ROS node.

Messages

To see what fields standard ROS messages have, you can google them and look at their parameters. For example, google "ROS pose message" and click on the first link. You'll encounter the documentation and see that there are two parameters, `position` and `orientation`. If you click on

position, you'll be taken to another message description that says `float64 x`, and the same for y and z. ROS messages are built up from their primitive types. In this case, the position message contains three parameters, `x`, `y`, and `z`, that are of type float64, which is a float that takes up 64 bits of storage. The `pose` message contains two parameters, `position` and `orientation` which are their own types of ROS messages built on primitives. To create a pose message, you can import the message using `from geometry_msgs import Pose` and then instantiate it: `pose_msg = Pose()`. Then, you can modify its values in hierarchal order, for example, if you wanted to change the x position to `3`, you could write: `pose_msg.position.x = 3`. If you ever have any questions as to how to access a value, just google the ROS message as we did above; if the message is a custom message (from pidrone_pkg), just look in the msg folder.

Topics

Topics are what ROS messages are published and subscribed to. From the [ROS wiki](#), "Topics are named buses over which nodes exchange messages." Topics have message types which must be followed. Creating a ROS topic involves creating a publisher that publishes to the topic. Then, any node on the same ROS Master can subscribe to this topic to get the data from the messages being published to it. You can print out all of the topics running by entering `rostopic list` into an empty window after running 'screen -c pi.screenrc' on your drone. We followed a specific naming convention when writing the ROS topics used for the drone. All of the topics start with `/pidrone`. Then, there may be a sub category, such as topics coming from the camera: `/pidrone/picamera`. This keeps things orderly and makes it easy to identify where the data is coming from. You can also have messages that are being published to a topic printed out by navigating to an empty window in the screen and entering `rostopic echo [topic_name]`. For example, if you wanted to see the data coming from the distance sensor, you could enter `rostopic echo /pidrone/range`

Publishers

Publisher are used to publish specific message types to specific topics. Publishers are useful for sharing data across nodes. For example, the tof_node which interfaces with the time of flight distance sensor publishes its data to `/pidrone/range`, and this data can be used by other nodes by subscribing to that topic. On the DuckieDrone, the state_estimator (you'll be writing this later) will subscribe to this data to as a measurement for the height of the drone.

Subscribers

Subscribers are used to read the messages being published to a ROS topic. When creating a subscriber, you must identify the topic, message type, and a callback method which takes in the message as an argument, and will called every time a message is published to the topic. For example, if you wanted to update the height of the drone every time a message was published, then in a ROS node you would first create a subscriber using `rospy.Subscriber("/pidrone/range", Range, range_callback_method)`. Your callback method might look something like:

```
range_callback_method(msg):
  drone_height = msg.range
```

Nodes

This section elaborates on all of the ROS nodes that run on your drone to make it fly autonomously. These are described in the order in which they appear in `pi.screenrc`: the pattern `# in the headers serves to remind that in screen you can move between the various nodes by typing `# where # is the node number.

`0: roscore

Starts up a ROS master to allow the nodes to find each other.

- Runs: `roscore`

`1: FC

The flight controller node controls what mode the drone should be in based on the user input and on safety checks. For example, if any of the heartbeats stop publishing, the mode controller disarms the drone. If the mode is "ARMED" or "DISARMED", the flight controller node sends static command values, but if the mode is "FLYING", then the node sends the fly_commands topic to the flight controller board.

Flight Controller interfaces with the flight controller board to extract the IMU and battery data, and to publish the roll, pitch, yaw, and throttle commands which are used to control the attitude of the drone.

You will need to start this node explicitly (press enter on screen 1) as a safety measure to prevent the drone from flying until you are really ready.

- Python script:
  - `flight_controller_node.py`
- Hardware interfacing:
  - flight controller board
- Publishers:
  - `/pidrone/imu`
  - `/pidrone/battery`
  - `/pidrone/mode`
- Subscribers:
  - `/pidrone/fly_commands`
  - `/pidrone/desired/mode`
  - `/pidrone/heartbeat/range`
  - `/pidrone/heartbeat/web_interface`
  - `/pidrone/heartbeat/pid_controller`
  - `/pidrone/state`

`2: PID

The PID controller node controls the flight of the drone by running a PID controller on the error calculated by the desired and current velocity and position of the drone.

Most of the PID loop tuning values, and the implementation of a PID control loop is actually in `scripts/pid_class.py`.

- The 5 columns of numbers that are printed on this screen actually come from pid_class.py, and represent the following:
  - throttle command for flight controller (1200=least lift)
  - target height minus actual height (millimeters)
  - component of throttle command caused by Proportional component of PID
  - component of throttle command caused by Integral component of PID
  - component of throttle command caused by Derivative component of PID
- Python script:
  - `pid_controller.py`
- Hardware interfacing:
  - None
- Publishers:
  - `/pidrone/fly_commands`
  - `/pidrone/position_control`
  - `/pidrone/heartbeat/pid_controller`
- Subscribers:
  - `/pidrone/state`
  - `/pidrone/desired/pose`
  - `/pidrone/desired/twist`
  - `/pidrone/mode`
  - `/pidrone/desired/mode`

- /pidrone/position_control
- /pidrone/reset_transform
- /pidrone/picamera/lost

`3: SE

The State Estimator node unifies the different state estimators so that the user only has to call this script to interact with state estimators. This node publishes to `/pidrone/state`, which offers the best state estimate based on whichever state estimators are to be used, depending on the command-line arguments that the user passes into this script.

The different state estimators (in `scripts/StateEstimators/`) are:

- ema: uses an exponential moving average
- ukf2d: UKF with a 2D state vector
- ukf7d: UKF with a 7D state vector
- ukf12d: UKF with a 12D state vector

The state typically consists of the x,y,z positions and velocities, and the yaw of the drone. We've implemented several state estimators that vary in complexity. You will be implementing a state estimator that uses an Unscented Kalman Filter (UKF) in a future project.

- Python script:
  - `state_estimator.py`
- Hardware interfacing:
  - None
- Publishers:
  - /pidrone/state
- Subscribers (one of):
  - /pidrone/state/ema
  - /pidrone/state/ukf_2d
  - /pidrone/state/ukf_7d
  - /pidrone/state/ukf_12d

`4: Vision

- Runs:
  - raspicam_node (from github.com/UbiquityRobotics/raspicam_node)
- Hardware interfacing:
  - Raspberry Pi camera
- Publishers:
  - /raspicam_node/motion_vectors

`5: Optical Flow

The Optical Flow node subscribes to the optical flow motion vectors published by the vision node and publishes linear velocity calculated from that.

- Python script:
  - `optical_flow_node.py`
- Hardware interfacing:
  - None
- Publishers:
  - /pidrone/picamera/twist
- Subscribers:
  - /raspicam_node/motion_vectors
  - /pidrone/range

`6: Rigid Transform

This node uses OpenCV to calculate the change in position of the drone using the camera.

- Python script:
  - `rigid_transform_node.py`
- Hardware interfacing:
  - None
- Publishers:
  - `/pidrone/picamera/pose`
  - `/pidrone/picamera/lost`
- Subscribers:
  - `/pidrone/reset_transform`
  - `/pidrone/position_control`
  - `/pidrone/state`
  - `/raspicam_node/image/compressed`
  - `/pidrone/range`

`7: TOF

The TOF node communicates with the VL53L0X Time Of Flight distance sensor using I2C and publishes the sensor's range readings.

- Python script:
  - `tof_node.py`
- Hardware interfacing:
  - Time Of Flight sensor
- Publishers:
  - `/pidrone/range`

`8: rosbridge

This node allows the web dashboard to communicate with ROS nodes on the drone.

- Publishers:
  - `/pidrone/desired/mode`
  - `/pidrone/heartbeat/web_interface`
  - `/pidrone/position_control`
  - `/pidrone/desired/twist`
  - `/pidrone/desired/pose`
  - `/pidrone/reset_transform`
  - `/pidrone/map`

`9: web_vid_serv

This node allows the web dashboard to request a camera stream of the drone. This node doesn't run by default but if you want to see what the drone's camera is seeing and if you have the processing power to spare on the drone you can press enter on this screen.

`10 free1

This screen is an unused console that you can use to run any additional commands you need to. One way to get to this screen is to type `9 then `n

`11 free2

This screen is an another unused console that you can use to run any additional commands you need to. One way to get to this screen is to type `9 `n `n

---