This book is the operations manual for the Duckie Drone.
Contents

Part A - Media ................................................................. 3

Part B - Build ................................................................. 4

Part C - Software Architecture ........................................... 129

Part D - Flying ............................................................... 139

Part E - Troubleshooting .................................................... 151

Part F - Diagnostic .......................................................... 154

Part G - Multimeters ......................................................... 162
PART A
Media

This section contains publicity and performance videos of the DuckieDrone!

0.1. First Undergraduate Class to take the course
This video shows footage of students building and flying their own DuckieDrone’s during the first run of the course during the Fall 2017 semester. link to video

0.2. Brown University Interdisciplinary Team UTRA 2018
This video provides an overview of the platform and some of the work that was done on the drone by undergraduates in 2018. link to video

0.3. Natural Language, Mixed Reality, and Drones
This blog post demonstrates the DuckieDrone as a research platform that combines natural language processing and mixed reality.
link to blog post
PART B
Build

This section describes how to build the DuckieSky Drone. The following sections will cover the safety protocols and the required materials.

0.4. Build Diagram
The figure below is a wiring diagram of the completed build. It will be a useful reference to use while you go through these instructions.

Figure 0.1. Wiring Diagram
The goal of this section is to identify sources of danger in the drone build, and to provide safety guidelines to mitigate the chance of harm.

1.1. 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 because it may contain lead. 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 can 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
- If you are feeling light-headed after soldering for extended periods of time, take a break and get fresh air
- Wash your hands after soldering since the solder may contain lead

1.2. Battery

Lithium Polymer (LiPo) batteries are very commonly used in RC devices. However, they must be handled properly. Please read this article to learn more about the proper care of your drone’s LiPo battery.

- 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.

1.3. 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. For this reason, it is very important to fly your drone in a spacious area without people or fragile things around.

Another source of danger is a propeller braking after hitting something, and then flying off of the drone. This is why it is important to always wear your safety goggles when flying. If your propellers are on the wrong motors, they could fly off of the motor and also cause serious damage. Always make sure the correct propellers are on each motor, the propellers are tightly fastened to the motors, and the motors are tightly fastened to the drone frame.
UNIT B-2
Prerequisites

Expected Time: 1 hours

2.1. Preface
This section contains information about the most important skills used in the build. If you are new to soldering, it is recommended that you complete the soldering module before starting the build. Whether you're new too soldering or not, we recommend reading through this page to review the basic techniques used in the build.

2.2. Strip Wires
Striping is the process of removing a portion of 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.

(a) An unstripped wire  (b) A stripped wire

Figure 2.1. To Strip a Wire

2.3. 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.

1) Tinning a Wire
See this 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 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.
2.4. 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.

We recommend you complete the soldering module before starting the drone build to ensure you’ve had plenty of practice soldering practice parts before soldering your drone parts. Below, there are brief resources on soldering, as well as specific instructions for each type of soldering technique that will be used in the build.

1) Resources

For a quick overview of soldering, watch this beginner soldering tutorial YouTube video.

For a more in-depth introduction, review this article

2) Wire-to-Wire

To write: soldering instructions on wire-to-wire solders

3) Wire-to-Pad

To write: soldering instructions on wire-to-pad solder

4) Through-Hole

To write: soldering instructions on through hole soldering

5) How to Fix Solder Mistakes

To write: instructions for fixing soldering mistakes

6) Safety Tips:

- 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 or burn.
- Don’t breathe soldering fumes; use a soldering fan whenever possible.
- If you have difficulty soldering (e.g. shaky hands), please seek the advice of a TA or teacher.
2.5. **Basics of electrical circuits**

We recommend that you’ve completed the circuitry module before starting the build so that you are introduced to electrical circuits like the ones you will be creating. For an introduction or review of circuit basics and Ohm’s Law, $V = IR$, check out this (Spark-Fun article)
UNIT B-3
Included Materials

Welcome to your Duckietown Sky Drone kit! This section contains a list of parts and materials included in your kit.

**Note:** Depending on the hardware version of your drone kit, the components may look different, but the functionality will be the same.

3.1. Your drone kit

![Figure 3.1](image)

3.2. All materials included in the kit
3.3. **FPV250 Drone Frame Kit**
Figure 3.3. Drone Frame Kit

1) Frame

Figure 3.4. Drone Frame

**Units:** 1

**Description:** A 250mm plastic racing quad frame. Included in the frame box.

2) Power Distribution Board (PDB)
Units: 1
Description: An electronic component that distributes power that it receives to other components connected to it. Included in the frame box.

3) Panhead screws

Units: 3
Description: Small screws that are used to mount the Pi on top of the pi mount. Included in the frame box.

4) Landing Gears
Figure 3.7

Description: Feet that attach to the drone frame. Included in frame box

3.4. XT60 Connector

Figure 3.8

Units: 1

Description: The power connector cable that transfers power from the battery to the PDB.

3.5. 2205 2300KV Brushless Motor

Figure 3.9

Description: An actuator that spins at variable speeds. Although brushless motors can spin in either direction, the motor shafts are threaded so that as the propeller spins, it does not loosen the nut and fly off. You can identify which direction a motor should spin by noticing the direction of the arrows on top.

1) Clockwise Motor
Figure 3.10

**Units:** 2

2) **Counter-clockwise Motor**

Figure 3.11

**Units:** 2

3) **M3 Bolts**

Figure 3.12

**Units:** 16 short, 16 long

**Description:** A screw that can be screwed into a standoff or a motor. Included in the motors box.

4) **M3 Hex Key**

Figure 3.13

**Description:** Used to tighten the M3 bolts
3.6. 5040 3-Blade Propeller

Units: 2 clockwise and 2 counterclockwise (extra CW and CCW included)

Description: A device with blades that turns rotational motion into thrust. 5 refers to the diameter in inches, and 4 refers to the distance the propeller would travel if turned one rotation without slippage, e.g., in jello. Three blades gives more lift for a given diameter than two blades - at the cost of efficiency.

1) 8mm Wrench

Description: Used to tighten the motor nuts that hold down the propellers

3.7. 30A Brushless Electronic Speed Controller (ESC)
Figure 3.16

**Units:** 4

**Description:** An electronic component that sends variable amount of power to a motor, based on a specified input signal. Every motor needs one ESC.

1) *Wires for the ESCs*

Figure 3.17

**Description:** Used to wire the ESCs to the motors. Note that the colors of the wires do not affect the functionality; they just make it easier for these instructions. If you have difficulty seeing colors, do not worry about about mixing these wires up.

3.8. **2.0mm Bullet Connector**
3.9. Brass Standoffs

Units: 6
Description: A special type of screw that can also accept other screws.

3.10. Raspberry Pi 3 Model B (Pi)
**Units:** 1  
**Description:** A single board computer that can execute code loaded via an SD card.

1) **16GB Micro SD Card**

**Units:** 1  
**Description:** A memory device, especially notable because it can store code and be inserted into a Pi.

2) **Heat Sinks**

**Units:** 3 (14mm * 14mm * 7mm; 9mm * 9mm * 5mm; 12mm * 12mm * 1mm)  
**Description:** A device that helps dissipate heat. Heat sinks help to keep computer processors cool.

3) **Pi Mount**
**Included Materials**

**Figure 3.23**

**Units:** 1

**Description:** A 3D-printed mount for attaching the Raspberry Pi to the drone frame.

4) **Screwdriver**

**Figure 3.24**

**Description:** Used to tighten the Pi Mount screws

5) **Perma-Proto Raspberry Pi Hat (Pi Hat) and Pin Header**
3.11. Battery Eliminator Circuit (BEC)

Units: 1
Description: An electronic component that steps down the voltage from 12V from the battery to 5V for the Pi.

3.12. Flight Controller (FC)
Figure 3.27. Flight Controller Board and Vertical Pins

**Units:** 1 + 1  
**Description:** A flight controller is a device that contains a few sensors: an accelerometer and a gyroscope; together these sensors make an IMU. The accelerometer measures linear accelerations and a gyroscope measures angular velocities. A flight controller also sends input signals to the ESCs. There are extra wires included in the flight controller box. The drone only requires the flight controller board and the vertical pins.

1) **USB to Micro USB**

Figure 3.28  
**Units:** 1  
**Description:** Connects the flight controller to the Raspberry Pi.

2) **Foam Mounting Tape**

Figure 3.29  
**Description:** Double-sided tape used to mount the flight controller onto the drone frame.
3.13. Arducam 5MP 1080p OV5647 Camera (Pi Cam) + 15pin Flexible Flat Cable (FFC)

Units: 1 + 1
Description: A sensor that observes 2D images of the world and reports it on a FFC cable. FFC included in pi cam box.

3.14. Infrared Sensor (IR) + IR Sensor Cable

Units: 1 + 1
Description: A sensor that measures distance to an object using infrared beams, then reports it on a wire.

3.15. Analog-to-Digital Converter (ADC)
Description: A device that converts real-valued signals (i.e. analog) into discrete-valued signals (i.e. digital).

1) Pins

Figure 3.33. Pins

Description: The pins get soldered into the ADC in order to connect it to the Pi Hat

2) Wires for the ADC

Figure 3.34

Description: Used to wire the ADC to the Pi Hat. We use the colors blue, green, red, and black to make it easier to follow the instructions, but the color of the wire does not affect the functionality.

3.16. 1500mAh 3S 20C LiPo Battery

Figure 3.35

Units: 1
Description: A lithium polymer battery used to power the drone.
1) 12V 2-3S LiPo Battery Balancer/Charger

![Image of 12V 2-3S LiPo Battery Balancer/Charger]

Figure 3.36

**Units:** 1  
**Description:** Safely charges and balances the LiPo battery

2) AC/DC US Charge Adapter

![Image of AC/DC US Charge Adapter]

Figure 3.37

**Units:** 1  
**Description:** Connects the battery charger to a wall outlet

3) Battery Mount

![Image of Battery Mount]

Figure 3.38
Description: A 3D printed part that attaches to the drone frame and holds in the battery

4) Velcro Strap

![Figure 3.39]

Units: 1
Description: A velcro strip used to hold the battery to the drone.

3.17. Heat Shrink

![Figure 3.40]

Description: A heat shrink (a.k.a heatshrink or heat-shrink tubing) is a shrinkable plastic insulator tube used to insulate wires. It is commonly used as a “sleeve” over a solder joint.

3.18. Zip Ties

![Figure 3.41]

Description: A zip tie is a type of fastener for holding items together, primarily electrical cables or wires.

3.19. Extra Wires
3.20. Duckies

Description: The heart of Duckietown.
UNIT B-4

Excluded Materials

This section contains what you will need to build and fly your drone that is not included in the kit.

4.1. Base station

In order to fly the drone, you must have a laptop or personal computer that is capable of connecting to the internet. Additionally, you will need to install the following software on your base station.

1) OS requirements

At this time, it is not possible to flash your SD card on a Chromebook; therefore you will need a linux, windows, or mac device to configure SD card during the build, but you can use any OS after this the configuration.

2) Browser Requirements

Google Chrome is recommended. You will need to download a Chrome app later on, so is best to install Chrome now. Safari will work for flying. Microsoft Edge does not work.

3) Software Requirements

**Flashing the SD card:**
1. Balena Etcher. You cannot complete this step if you are using a Chromebook.

**Configuring the Flight Controller:**
1. Cleanflight. This is a Google Chrome application.
2. USB to UART driver. Download the correct driver for your OS. You will not need this driver if you are using a Chromebook; you will be able to connect to the flight controller without it.

**Viewing the screen of your drone**
1. Real VNC Viewer. This is a Google Chrome application.

4) File Downloads

1. SD Card Image
2. Flight Controller Firmware

4.2. Soldering Tools

- soldering iron
- solder
- brass wool or sponge to clean soldering iron
• solder remover– either solder sucker or desoldering wick– to fix mistakes
• wire strippers
• safety goggles
• soldering mat
• helping hands
• tweezers or pliers
• flush cutters
• fume extractor
• multimeter

4.3. Misc. Tools
• hot glue gun
UNIT B-5

Part 1: Raspberry Pi and Power Distribution Overview

5.1. 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 reintroduce 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.

5.2. Required Materials
- **Part : Quantity**
  - Battery : 1
  - Battery Charger : 1
  - Battery Charging Adapter : 1
  - Power Distribution Board : 1
  - Battery Eliminator Circuit : 1
  - Pi Hat : 1
  - XT60 Connector : 1
  - Soldering Tools : 1
  - Raspberry Pi : 1
  - Heat Sinks: 1 set
  - Micro SD Card : 1
  - Base Station : 1
  - Micro SD Card Reader : 1
  - LED : 1
  - Resistor : 1

5.3. Detailed Hardware Descriptions

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

**Capacity**

The unit milliAmp 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 a 3S battery.

**Current Output**

The “C” rating of a LiPo Battery determines how much current the battery can deliver. The maximum current 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**

Review the safety information on the battery. Take the proper precautions when using, storing, or charging is very important to keeping the battery safe. Only store the battery at room temperature and out of direct sun light. Do not discharge a battery below 10.5 V. Never leave the battery charging unattended. Unplug the battery once it is fully charged.
The Power Distribution Board is used to distribute power from the battery to electrical components of the drone. The PDB an example of 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.

3) BEC

The Battery Eliminator Circuit, or BEC, solves the issue that arises from different electrical components requiring different supply voltages. In the case of the drone, the LiPo battery used for outputs around 12V; this is the required voltage for the motors, but not for the Raspberry Pi, which requires 5V. The 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 Pi.

4) Raspberry Pi

The Raspberry Pi is a small single-board computer used for various purposes, including as a drone's control system.
The Raspberry Pi, or Pi, is a low-cost, single-board computer. The 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 Pi is used on the drone as the main computer that reads and processes all of 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 the Pi if a person had a remote controller to manually steer the drone. However, for the DuckySky Drone, the person with a remote controller is replaced by software and sensors on the Pi. The Pi, as well as the sensors you will connect later, makes autonomous flight possible.

5) Pi Hat

The 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:

![Rail highlighted on back of Pi Hat](image)

Figure 5.5. Pi Hat Front View

The rails are useful because every wire put on a rail will be electrically connected; this means that it does not matter into 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. Notice especially the long +5V and GND rails; we can use any of the holes in this 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.

6) Micro SD Card

The Micro SD card stores the operating system on the pi, as well as all of the software needed for autonomous flight.

![Micro SD Card and Adapter](image)
7) Light Emitting Diode (LED)

An LED is a device that emits light as current flows through it; the color of the light is based on the material properties of the LED. An LED has a polarity, meaning that current can only flow through the device in one direction. If electricity flows in the wrong direction, the device will be damaged; be sure to double check the LED direction before soldering. The correct direction of flow is from the *anode* (the positive terminal) to the *cathode* (the negative terminal). You will be able to identify the anode because it has a longer terminal, as shown in the image.

Another property of an LED is that it has very little resistance. Recall that Ohms states that $I = \frac{V}{R}$. The resistance of an LED is so low that we can assume it is zero. Plug zero into the equation and we get an infinite amount of current. In reality, the current is limited by the power source. Since there is such little resistance with the LED, if we created a circuit with only this element, it would burn out from too much current, and the battery may be damaged.

**Q:** *Can you think of a way that we could limit the current passing through the LED?*

**Answer:** If you thought of using a resistor, then you’re right!

8) Resistor

Resistors are passive elements that add resistance to a circuit. Each resistor has a specific resistance and a tolerance. To determine the resistance of a resistor, one must look up the color code of the colored bands on the resistor. The resistor in the image is 680 Ohms.

| Remark: | if you have poor eyesight or experience difficulty seeing colors, it is useful to ask a friend to read the colors they see and you can look up the colors on the chart. |
9) XT60 Connector

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.

5.4. Build Progress

You are going to create a circuit to provide power to the Pi. You will also create a simple circuit using an LED and resistor that will allow you to control the LED using the Raspberry Pi. Later in the course, you’ll learn how to write a script to turn the LED on and off, as well as vary its brightness!

After completing this section, your build will match the diagram below. Compare this diagram to the completed drone diagram to see how what you’re doing now fits into the final result.
The electricity flows from the battery, through the PDB to the BEC where the voltage is stepped down to 5V, then into the Pi Hat, and finally to the Pi and the LED.
UNIT B-6

Part 1: Raspberry Pi and Power Distribution Instructions

Expected Time: 3 hours

6.1. Preliminary Tasks

1) Charge the Battery

1. Start charging your drone battery so it is ready when you need it.

**Note:** Never leave the battery charging unattended. The battery takes about 2 hours to charge.

2) Download Image and Image Flashing Software

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

**Note:** 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 1 - 2 hours. In the meantime, you can move on to the next sections.

6.2. Attach the Pin Header to the Pi Hat

1) Identify the front and back

Identify the side of the 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 slot in the Pi Hat to help with matching the orientation in the following instructions.
2) Insert the Pin Header

Insert the Pin Header into the back of the Pi Hat as shown in the image.

3) Solder the Pin Header

1. Review the through-hole soldering technique
2. Use the helping hands to assist as you solder the pin header

**video tutorial**: video instructions part 1
**video tutorial**: video instructions part 2
**video tutorial**: video instructions part 3

**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. If you have a solder bridge, try the technique shown in...
this video. If this does not work, you can use desoldering wick or a solder sucker to remove the excess solder.

**Check before you continue**

Sometime too much heat when soldering the pin header can mess up the Pi Hat connections. To make sure the connections are still good, do a connectivity check on the Pi Hat; verify there is:

- A short between the hole labelled SDA on the Pi Hat and the pin at row 2 column 1 of the pin header that you soldered.

![Figure 6.3. Check for SDA connection](image)

- A short between the hole labelled SCL on the Pi Hat and the pin at row 3 column 1 of the pin header that you soldered.

![Figure 6.4. Check for SCL connection](image)

If you make a mistake while soldering, review the instructions for fixing soldering mistakes

**6.3. Attach the LED and Resistor to the Pi Hat**

Solder the 680 Ohm resistor and your LED to the Pi Hat as shown in the image.

**Remark:** The value of the resistor can be modified to adjust the brightness of the LED at full power. If you have additional resistors, you can modify this resistance by using
a resistor with a smaller resistance, or by adding resistors in parallel later on. A helpful thread for understanding why this resistance should work is found here.

**Note:** The direction of the resistor does not matter, but the direction of the LED does matter. Be sure to place the cathode (shorter end) into the GND rail.

You've just finished the LED circuit! In a future lesson, you will learn how to send electricity through GPIO pin #6 to the LED and resistor, and then back to ground.

### 6.4. Attach the BEC to the Pi Hat

1) **Prepare**

Cut the black plastic piece off the BEC OUT wires

(a) BEC wires pre-cut
Figure 6.6. BEC preparation

2) Strip and Tin

1. Strip the end of the red and black OUT wires (the wires you just cut) so that there is about 5mm of exposed wire.
2. Twist together the ends of the wires and lightly tin them. The wires need to be thin enough to fit through the holes in the Pi Hat.

3) Solder

Solder the BEC red (+) OUT wire to the +5V Rail, and solder the BEC black (-) OUT wire to the GND rail, as shown in the image.

**Remark:** Any hole on the rails will work; however, for wire organization, it is better to use a hole on the 5V rail that is shown. If you mess up soldering, it is ok to leave the mistake and use a different hole.

Figure 6.7. BEC Soldered to Pi Hat

*video tutorial:* video instructions (please ignore the extra wires and components that are already attached to the Pi Hat in the video)

**Check before you continue**

Do a connectivity check on the Pi Hat. Verify there is:

- **no short** between the +5V rail and the GND rail on the Pi Hat
6.5. Prepare the PDB

1) 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.

1. Review the pad-tinning tutorial
2. Tin every pad on the PDB, except the 5V and 12V pads.

**Note:** Be careful not to aggressively push the soldering iron tip into the PDB, as too much force will cut the pads right off!

![Tinned PDB](image)

**video tutorial:** video instructions

**Note:** For the remainder of the instructions, unless stated otherwise, red wires should be soldered to positive (+) pads and black wires should be soldered to ground (-) pads.

6.6. Solder the BEC to the PDB

1) Tin

Tin the ends of the red and black wires that are connected to the side of the BEC that is labeled IN

2) Solder

Solder the BEC red (+) IN wire to the positive (+) pad on the PDB, and solder the black (-) IN wire to the negative (-) pad on the PDB, as shown in the image.

**Note:** Any of the tinned (+) pads will work; however, using the pad shown in the image will help with wire organization later on.
Figure 6.9. BEC positive wire soldered to PDB

*video tutorial:* solder BEC to PDB (Please ignore the additional wires on the PDB, as well as which pads the BEC is soldered to - just observe the technique)

### 6.7. Solder the XT60 Battery Connector to the PDB

1) Strip and Tin

1. Strip the ends of the battery connector so that about 1cm of wire is exposed
2. Tin the exposed ends of the XT60 connector

2) Solder

Solder the XT60 **red** (+) wire to the tinned positive (+) pad on the PDB, and solder the **black** (-) wire to the tinned ground (-) pad on the PDB, as shown in the image.

*Note:* 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 and be patient.
**Note:** Do not solder the wires flat against the PDB - solder them at ~20° angle away from the board. If you solder them flat, then you will not be able to fit the PDB into the drone frame.

**Check before you continue**
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 IN side - NOT the OUT 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.

**Note:** If the battery is 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:
6.8. Put Heat Sinks on Raspberry Pi

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

(a) Heat Sinks on Top of Pi
(b) Heat Sinks on Bottom of Pi

Figure 6.11. Put Heat Sinks on Pi

6.9. Attach the Pi Hat to the Pi

1) Align the pins

Align the 2x40 GPIO pins on the Raspberry Pi with the 2x40 pin header that you soldered to the Pi Hat as shown in the image. (please ignore additional wires on the Pi Hat)

Figure 6.12. Pi Hat lined up

2) Press down

Press the pin header down onto the GPIO pins to connect the Pi Hat
6.10. Final Steps

Check before you continue
Do another connectivity check to verify that there is no short between the +5V rail and the GND rail on the Pi Hat.

1) Verify that the Pi has power
   1. Connect the battery
   2. Verify that the BEC has a solid green light.
   3. Verify that the Pi has a solid red light.

2) Insert the SD card into the Pi
Insert your (now flashed) SD card into the SD card slot on the bottom of the Pi.

Note: The SD card direction does matter - the lettering on the SD card should be facing downward.
Figure 6.14. SD Card inserted in the Pi
UNIT B-7

Part 1: Power Distribution and Raspberry Pi Checkpoint

Expected Time: 10 minutes

7.1. Connect to the Pi for the first time
You will now verify that the SD card is working properly by connecting to your Pi for the first time. The SD card is configured to serve a text editor that gives you access to all of the files on the Pi. You won’t be making any changes at this point, you will just be verifying that you can connect to the server.

1) Power up your drone build
Plug the battery into the XT60 connector

2) Connect to the drone’s wifi network
On your base station, connect to the network labelled defaultdrone. The wifi password is bigbubba

   Note: At this point you will lose internet access, because you will be connected to the Pi’s network. You can disconnect and reconnect to the network at any point without causing issues on the Pi.

3) Browse to your drone’s code server
In a web browser on your base station, browse to 192.168.42.1:8081. The first 8 numbers (192.168.42.1) are the IP address of the drone, and the numbers after the colon (8081) is the port number that the text editor is served to

4) Checkoff
If the link does not show an error, then your SD card is correctly flashed! The screen you are looking at is a text editor that allows you to edit the code on your drone. Any changes you make in this editor will be automatically saved to the drone. You can browse around the text editor and files if you’d like. You will be learning more about this interface in a later lesson.

7.2. Change your WiFi SSID
At this point, if student A is working on their drone in the same room as student B, both students would not be able to connect to their drone.

Q: Can you figure out why?
Answer: There can’t be multiple WiFi networks named “defaultdrone”. The SSID must be changed for each drone so that there aren’t multiple networks with the same name.
Now, you will change the name of your WiFi SSID. Follow the steps below along with this instruction video to change the SSID

1) Launch a terminal

1. In the code server in your browser, find the menu button in the top left corner (it looks like three horizontal lines)
2. Click the menu button, then click Terminal > New Terminal
3. A small window will open up in the bottom of your screen.

2) change the SSID

1. In the terminal, type: `sudo nano /etc/hostapd/hostapd.conf`
2. When prompted, type `bigbubba` and press enter (you will not see the password typing as a security feature)
3. Use the arrow keys on your keyboard to move the cursor to the end of the second line of this file that says: `ssid = defaultdrone`
4. Delete the phrase `defaultdrone` and type a new network name for your drone

3) Save and exit

1. While pressing down the control (ctrl) key, press the `x` key
2. Type `y` and press enter to save and exit

4) Checkoff

Disconnect and reconnect the battery to your drone. On your base station, look for the available WiFi networks. If the file was edited properly, you will now see your new network name instead of “defaultdrone”

Congrats on finishing Build Part 1!
UNIT B-8
Part 2: IR Sensor Overview

8.1. 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 infrared (IR) sensor. The IR sensor is used to measure distance. We’ll provide more details about the hardware used in this portion of the build, explain the circuit you’ll be creating, and then get into the instructions.

8.2. Required Materials

You will need Build Part 1 completed before you can begin this build part.

- **Part : Quantity**
  - Infrared Sensor and Wire : 1
  - Analog to Digital Converter and included pins : 1
  - ADC wires : 1
  - Soldering Tools

8.3. Detailed Hardware Descriptions

1) Infrared (IR) Sensor

The IR sensor is used to measure distance. On the drone, we use this sensor to measure the height of the drone above the ground. The sensor works by emitting infrared light from one side, and measuring the angle of reflection on other. The sensor outputs a voltage value that varies inversely with distance to an object. After powering up your sensor, you’ll be able to read the voltage output from the sensor using a multimeter.
2) **Analog to Digital Converter (ADC)**

The Raspberry Pi GPIO pins can only read digital signals (1’s and 0’s). However, the IR sensor outputs an analog signal (a voltage value). In order for the Pi to read the output of the IR sensor, the analog output must be converted to digital first. This is the purpose of the Analog to Digital Converter, or ADC for short.

In order to connect the IR sensor and ADC, you will be creating the following circuit:

8.4. **Build Progress**

After completing this section, your build will match the diagram below. Compare this diagram to the completed drone diagram and to the Part 1 diagram to see how what you’re building up to the final result.
The IR sensor receives power from the Pi Hat, and the sensor output signal is received by the ADC. The ADC converts the analog signal into a digital signal, and passes this to the Pi using the SCL and SDA inputs, (which is called I2C protocol).
**UNIT B-9**

**Part 2: IR Sensor Instructions**

**Expected Time**: 2 hours

### 9.1. Solder Pins to the ADC

1) **Identify the top and bottom**

   The top of the ADC has the labels near the holes, “V,” “G,” “SCL,” etc.
   
   The bottom of the ADC says “ADS1115/ADS1015”

2) **Insert pins**

   Insert the short ends of the pins into the holes in the bottom of the ADC and clamp the ADC with the helping hand as shown in the image below.

   **Note**: The ADC shown is from a previous hardware version. For your ADC, the helping hands will be on the left side and the pins will be on the right

3) **Solder**

   Solder the pins on the top of the ADC using the through-hole soldering technique
9.2. Solder the ADC to the Pi Hat

1) Detach the Pi Hat from the Pi.
Gently lift the Pi Hat near the pin header to detach.

2) Solder the ADC to the Pi Hat
Following the image below, insert the ADC into front of the Pi Hat. Then, flip the Pi Hat over so you can solder the pins on the back of the Pi Hat using the through-hole soldering technique.

**Note:** Be sure to solder the ADC to the correct location on the Pi Hat. Use the location of the slot in the Pi Hat to help you align the ADC.

3) Solder the ADC wires
Following the diagram below, insert a wire into the correct hole on the front of the Pi
Hat. Then, flip the Pi Hat over to solder the wire on the back of the Pi Hat using the through-hole soldering technique.

**Note:** Recall that the Pi Hat is a breadboard, so it has rails. The wires you solder do not need to go into exact holes; each wire just needs to go into a hole on the same rail as its corresponding ADC pin. You can look at the back of the Pi Hat and see the metal connections between holes. You can also do a connectivity check with the multimeter if you have doubts about which holes are connected.

1. Use a small piece of **red** wire to connect the V pin of the ADC to the 5V rail.
2. Use a small piece of **black** wire to connect the G pin of the ADC to the the GND rail
3. Use a small piece of **blue** wire to connect the SCL pin of the ADC to the hole labelled **SCL** on the Pi Hat
4. Use a small piece of **green** wire to connect the SDA pin of the ADC to the hole labelled **SDA** on the Pi Hat

**Remark:** For the wires, we use the colors: red, black, green, and blue. However, any color wires will work.

![Figure 9.5. ADC Wires soldered to Pi Hat](image)

**Check before you continue**

Do a connectivity check on the Pi Hat; verify:

- There is **NO** electrical connection (short) between the 5V and GND rails.
- There is **NO** electrical connection (short) between the SCL and SDA wires
- There are NO stray wire strands that are connecting adjacent pins, especially between 5V and GND (see image below)

Figure 9.6. BAD: Wire strands causing shorts

9.3. Prep the IR Sensor Wire

1) Trim

Cut the black plastic piece off the end of the IR sensor cable

Figure 9.7. IR sensor cable before and after

2) Strip and Tin

1. Strip about 5mm off of the newly-exposed ends of the IR sensor wire
2. Twist and Tin the IR sensor wires. Do not put too much solder, as the wires must fit through the holes in the Pi Hat

9.4. Solder the IR Sensor Wires

1) Solder the IR Sensor Wires to the Pi Hat

Following the diagram below, insert a wire into the correct hole in the front of the Pi Hat. Then, flip the Pi Hat over to solder the wire on the back of the Pi Hat using the through-hole soldering technique.

1. Solder the red IR Sensor Wire to the 5V rail
2. Solder the black IR Sensor Wire to the GND rail
3. Solder the yellow IR Sensor Wire to any hole in the same row as A0 on the ADC
Figure 9.8. ADC Wires soldered to Pi Hat

Check before you continue
Do a connectivity check on the Pi Hat; verify:

- There is NO electrical connection (short) between the 5V and GND rails.

9.5. **Reattach the Pi Hat to the Pi**

Reattach the Pi Hat to the Pi by aligning the GPIO pins with the pin header and pressing down. Refer to part 1 for detailed instructions on how to attach the Pi Hat.
Expected Time: 30 minutes

10.1. Overview
You will now check to see if the IR sensor on your drone is working properly. The first check will involve reading the raw IR sensor measurements using a multimeter. Then, you will use the drone software and the web interface to see the drone’s height estimate.

10.2. Reading the IR sensor using a multimeter

1) Power the Sensor
Plug the battery into your drone build

2) Setup Multimeter
Set the multimeter to read voltage up to 2V

3) Probe

Figure 10.1. Multimeter in the correct setting
1. Insert the black probe into any hole in the GND rail.
2. Insert the red probe into any hole that is on the same rail as the yellow IR wire, which is connected to A0 on the ADC

**Note:** The ADC shown in the image is from a previous hardware version. You will use the technique shown, but not necessarily the same probe placement.

![Probing the IR sensor data wire](image)

**Figure 10.2.** Probing the IR sensor data wire

4) **Test**

While holding the probes in place, point the IR sensor at an object and then move the sensor closer and further to that object within the range of 10-50cm away. Check that the value on the multimeter is getting smaller when the object is further, and larger when it is closer. If this is correct, then your IR sensor is working properly!

**Note:** if the measurement is taken closer than 10cm or further than 50cm, the sensor will not read properly. If you’re interested in why, take a look at Fig. 5 in the datasheet.

### 10.3. Reading the IR sensor on the web interface

1) **Connect to the drone**

1. Plug the battery into your drone build and connect to the drone’s wifi network. By default, the network is named, `defaultdrone`, and the password is `bigbubba`.
2. Browse to the drone’s code editor: `http://192.168.42.1:8081`
3. In a new tab, browse to the web interface: `http://192.168.42.1`. The web interface is what you will use to fly the drone. It also contains graphs that show data from the sensors.

2) **Start up the code**

1. In the code editor, click the menu bar in the top left corner, then click Terminal > New Terminal
2. Type ./start and press enter

3. Go back to the web interface tab, wait about 5 seconds, and refresh the page. Make sure that you see “Connected” at the top of the page. If you do not see this, wait a few more seconds and try refreshing again.

3) Test

1. locate the Height Readings chart on the web interface

2. Use your hand to move the IR sensor closer and further to an object and check that the Height Readings chart is changing

Congrats on finishing Build Part 2!
UNIT B-11
Part 3: Motors and ESCs Overview

11.1. Preface
In this phase of the build, you’ll be adding the essential elements of every drone– the motors, ESCs, and the flight controller.

11.2. Required Materials
- Part : Quantity
  - Flight Controller : 1
  - Counter-clockwise Motors : 2
  - Clockwise Motors : 2
  - Long M3 Bolts (included w/ motor) : 16
  - Short M3 Bolts (included w/ motor) : 4
  - Electronic Speed Controllers (ESCs) : 4
  - ESC Wires
  - Velcro Strap : 1
  - Bullet Connectors : 12 Plugs, 12 Sockets
  - Brass Standoffs : 4
  - Spare Wire
  - Soldering Tools

11.3. Detailed Hardware Descriptions

1) Motors

![Figure 11.1](image-url)
2) ESCs

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.

![ESC](image1)

Figure 11.2. Electronic Speed Controllers

3) Bullet Connectors

Bullet connectors are used to connect two wires together in a way that allows them to be disconnected easily. It is useful to use these to connect the motors to the ECSs because if a motor is spinning in the wrong direction, you can simply switch two of the motor/ESC connections and the motor direction will reverse (more on this later). Additionally, if an ESC or motor goes bad, bullet connectors make it easier to swap out those parts.

![Bullet Connectors](image2)

Figure 11.3. Bullet Connectors

4) Metal Standoffs

Standoffs are used to separate layers of electronic circuits. On the drone, the standoffs will be used to separate the PDB from the Pi.
5) Battery Monitoring Leads

The battery monitor wires leads allow the flight controller to monitor the power traversing the PDB. This is useful because the flight controller can inform the 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.

![Battery Monitor Leads](image)

Figure 11.5

11.4. Build Progress

After completing this section, your build will match the diagram below. Compare this diagram to the completed drone diagram to see how what you're doing now fits into the final result.
The Pi receives IMU sensor data from the flight controller, and sends the flight controller values for roll, pitch, yaw, and throttle. The flight controller converts these values to PWM commands for each motor, and sends the values to each ESC. The ESCs receive power from the PDB, and use the PWM signal from the flight controller to control how fast the motors are spinning.
Unit B-12
Part 3: Motors, and ESCs Instructions

**Expected Time:** 5 hours

**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.

12.1. **Solder wires onto ESC pads**

In this section, you will prepare your ESCs. Each ESC has 3 pads labeled A, B, and C. The labels are on the bottom of the ESC:

![ESC Pad Letters](image)

**Figure 12.1. Pad Letters**

1. **Cut ESC wires to length**
   1. Cut four pieces of the blue wire, 3 inch lengths
   2. Cut four pieces of the yellow wire, 3 inch lengths
   3. Cut four pieces of the red wire, 3 inch lengths

2. **Tin the ESC wires**
   1. Strip about 5mm from both ends of the each wire that you just cut (12 wires in total)
   2. Tin both ends of the wires

3. **Prepare the ESC:**
   Use your fingers (or small wire cutters) to very carefully remove excess material from each ESC pad.
4) Tin the ESC pads
Tin all three metal pads on each of the four ESCs

5) Solder the ESC wires to the ESCs

1. Review the wire-to-pad soldering technique
2. Use the helping hands to hold one ESC in place for solder
3. Solder the yellow wire to the middle pad (pad “B”)
4. Solder the red wire to pad “A”
5. Solder the blue wire to pad “B”
6. Lightly pull the wire soldered to the pad and verify it stays on.
7. Do a connectivity check between each pair of soldered wires (recommend doing this now instead of in checkoff later). The multimeter should not beep, otherwise there is a short between the pair; if the multimeter does hold a continuous beep, then you will need to fix the short by removing any excess solder forming a bridge between the problematic pair of wires.
6) Insulate the ESC wires

Put a heat shrink over the ESC. The heat shrink should cover (length-wise) the exposed soldered wires on one end and half the ESC on the other end. Apply heat to shrink the heat shrink.
12.2. ESC bullet connectors

In this step you will solder bullet connectors onto the ESC wires and the motor wires so that you can easily connect them later on.

1) Review soldering tutorial

Before beginning this section, please refer to a bullet connector soldering tutorial.

**Note:** You can use any setup to hold the bullet connector, so long as the setup is not thermally conductive. Also, if you have trouble soldering the bullet connectors, you may need to use a smaller soldering iron tip. Wait at least 15 minutes or longer for the soldering iron to completely cool down after unplugging before attempting to swap the soldering iron tip.

2) Strip and Tin the ESC power wires

1. Strip about 5mm from the ends of the red and black wires on each ESC
2. Tin the red and black wires on each ESC

3) Solder the socket bullet connectors to the ESC wires

For each ESC:

1. Solder a socket bullet connector to the end of each of the 3 ESC wires (i.e. red, yellow, blue).
2. Put a heat shrink over each solder joint. For socket connectors: the heat shrink should cover the solder joint on one end and run the entire length of the bullet connector. Apply heat to shrink the heat shrink.
12.3. Motor bullet connectors

1) Strip the motor wires
   1. Strip about 5mm from the ends of the three black wires on each motor

2) Tin the motor wires
   1. Tin the ends of the three black wires on each motor

3) Solder the plug bullet connectors to the motor wires
   For each motor:
   1. Solder a plug bullet connector to each of the motor’s wires.
   2. Put a heat shrink over each solder joint. For male connectors: the heat shrink should cover the solder joint on one end and run only the short length of the cylindrical part. Apply heat to shrink the heat shrink.
Figure 12.6. Put Heat Shrinks on Plug Bullet Connectors

Check before you continue
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.

- Visually inspect that each of the following is stripped and tinned: 4 ESCs, 4 motors
- Do a connectivity check on the XT60 connector cable; verify there is no short between the red and black wire.
- Do a connectivity check on each ESC; for each ESC, verify there are no shorts between any two wires you soldered.

12.4. Solder 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.

1) Solder each of your 4 ESCs to the PDB.
Note: Do not solder the wires flat against the PDB - solder them at ~20° angle. If you solder them flat, then you will not be able to fit the PDB into the drone frame.

12.5. Solder battery monitor leads to the PDB

1) Solder leads to PDB

Solder the 6 inch red and black wires to the PDB. Due to limited PDB pads, you will need to solder onto another pair of wires, e.g. BEC wires. The red wire should connect to a positive (+) pad and the brown wire should connect to a negative (-) pad.

Note: these wires are soldered so they go across the PDB, toward where the flight controller will be mounted. Also, please ignore that in this photo, the PDB is in the drone frame.
Figure 12.8. Battery Monitor Lead Soldered to PDB

**Note:** While trying to solder on these wires, you may accidentally unsolder the existing wires from the PDB. We recommend temporarily holding down the existing wires with long-nose pliers, tape, or helping hands.

12.6. Solder to the flight controller

1) Solder the pins to the flight controller

Solder the short edge of the straight pins to the flight controller.

**Note:** Be sure that direction you solder the pins into the board is exactly as shown in the images.
2) Solder the battery monitor leads to the flight controller

Tin and solder the battery leads to the flight controller as shown in the image.

12.7. Attach parts to drone frame

This section will cover attaching the first set of items to the drone frame. Before beginning, verify the PDB is completely soldered with all necessary parts (as covered in previous sections).

Note: the flight controller is not shown in these images; however, don’t be alarmed that your build is incorrect. For now, just move the flight controller as you are working so it is not in the way. In the next section, you will attach the flight controller to the drone frame.

For reference, here are the motor directions with respect to the frame:
1) Materials

Gather the following:
- Drone frame
- Completed PDB
- 4 motors (2 CW, 2 CCW)
- Velcro
- 4 standoffs
- 12 black screws (in motors box, not drone frame box)

2) Align the frame

Place the drone frame on a flat surface so that the back is facing you.
Figure 12.12. Orientation of Drone Frame

3) Insert the velcro strap

Feed the velcro through the center of the drone frame. Make sure rough side is facing up.

Note: The strap in the image is from a previous hardware version. You will want this to strap down around the battery on the bottom of the drone frame. Try doing so without the battery to make sure the strap is on in the correct direction.
4) Screw a short M3 motor bolt into each of the standoffs

**Note:** Although the image shows rounded top bolts, the correct ones are the short M3 bolts that are included with the motors. (The ones shown in the image came from the frame kit and they work fine, but the hex key doesn’t fit them)

5) Attach PDB to frame

Place the completed PDB into the center of the drone frame. For each of the 4 corner screw holes of the PDB, screw a standoff through the hole and into the drone frame. Note that the drone frame doesn’t have screw grooves for the standoffs - you will create
these grooves by applying downward force while screwing. Don’t screw too far, since it is easy to strip the plastic. (We realize these don’t hold into the frame that well, and are devising a better solution for the next hardware version.)

![Image](attachment:image.png)

(a) Applying Downward Force  
(b) PDB Secured in Drone Frame

**Figure 12.15.** Put PDB on Drone Frame

6) **Attach clockwise (CW) motors**

Attach CW motors to the bottom-right and top-left of the drone frame, using 2 black screws for each attachment.
7) Attach Counter-Clockwise (CCW) Motors

Attach CCW motors to the bottom-left and top-right of the drone frame, using 2 black screws for each attachment.
PART 3: MOTORS, AND ESCs INSTRUCTIONS

Figure 12.17. Attaching CCW Motors

8) Connect 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 latter phase.
Figure 12.18. Connecting Bullet Connectors
UNIT B-13
Part 3: Motors and ESCs Checkpoint

Expected Time: 20 minutes

13.1. Verify the wiring is correct

1) 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 IN side - NOT the OUT 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

2) 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

3) Do a DC voltage check on the PDB;
   **ONLY** if the connectivity check passed, 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.

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

4) ESC check
   **ONLY** if the DC voltage check passed, re-connect a battery to your drone and verify the following:
   - The ESCs emitted a quick succession of 3 beeps.
   - The bottom of the drone frame is illuminating, due to the LEDs on the bottom of the PDB.
UNIT B-14

Part 3: Flight Controller and Cleanflight Overview

14.1. Preface
In this phase of the build, you will configure and calibrate the Flight Controller and ESCs. First, you will flash the flight controller with firmware, then you will configure the firmware to the settings that work best on the drone. Firmware is a special type of software that controls the hardware on a device.

- **Part : Quantity**
  - Foam Mounting Tape : 1
  - USB to Micro USB cable : 1
  - base station

14.2. Hardware

1) Flight Controller
The flight controller (i.e. FC) 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 Pi and then sends electric signals to the ESCs which in turn change the speeds of the motors.

![Flight Controller](image1)

Figure 14.1. Flight Controller

2) USB to Micro USB cable
This cable is used for two purposes. The first use is to configure the flight controller settings in CleanFlight (introduced later); this part only needs to be done once. The second use is to send the flight commands from the Raspberry Pi to the FC. This connection allows our software on the Pi to control the motors. The Pi tells the FC what roll, pitch, yaw, and throttle values the drone should have, and then the flight controller speeds up or slows down the motors to get this values.
Unit B-15

Part 3: Flight Controller and Cleanflight Instructions

Expected Time: 1 hour

15.1. Attach the Flight Controller

1) Trim

Trim the all of the pins on the flight controller just like the four shown in the image. This helps the flight controller sit level on the drone frame.

Note: the image shows only 4 trimmed, but you will want to trim all of the pins (updated picture coming soon).

2) Apply Tape to the Bottom of the FC

1. Put double sided mounting tape on the bottom of the FC.
2. Cut off any excess tape.

3) Stick the FC to the drone frame

1. Peel off the back of the foam tape
2. Attach the FC to front of the drone. Ensure the FC is not skewed and it is pushed against the frame body. (Ignore the extra wires in the flight controller for now).
Figure 15.2. Orientation of Drone Frame

**Note:** the white plastic tabs will need to hang off the frame for the flight controller to sit level.

![Top View](image)
![Bottom View](image)

Figure 15.3. FC attached to drone frame

**Note:** Try to minimize the FC skew as much as possible. (the following photos are an example of skew. ignore the difference of flight controller).
Once the FC is attached, do a “rock test” (i.e. try to rock the FC back-and-forth by pushing the corners). If the FC rocks, then the double sided mounting tape used is too soft and is compressing under pressure. Detach the FC from the frame (e.g. carefully use a flat-head screwdriver) and replace the tape with more robust tape.

15.2. **Download the required software and files**

1) **Download the firmware**

On your base station, download the flight controller firmware

**Note:** the file extension for this file should be `.hex`. Your computer might have added a `.txt` to the end of the file, rename the file and delete `.txt` so that the file ends in `.hex`.

2) **Install the USB to UART driver**

A driver is software that allows your computer to talk to a hardware device. You will need a driver that allows the computer to talk to the flight controller.

1. On your base station, download this driver. Be sure to select the correct download for your base station’s operating system. If you are using a Chromebook, you will not need this driver.
2. Once downloaded, click on the file and go through the installation process
3. After the installation finishes, restart your base station.
3) Install Cleanflight

Cleanflight is open-source flight controller software that allows you to flash firmware to the flight controller, and configure the settings.

On your base station, install Cleanflight (you will need Google Chrome)

15.3. Flash the firmware

Please follow the written instructions below to flash firmware onto your flight controller. If you are having trouble flashing, please follow this video to flash the flight controller with the boot pins shorted.

1) Open the firmware flasher

1. Launch CleanFlight
2. Click on “Firmware Flasher” on the left sidebar.

![Figure 15.5. Firmware Flasher](image)

2) Load the firmware

Click on “Load Firmware [Local]” in the bottom right corner of the window and select the firmware file that you downloaded (the one with the .hex extension).
Figure 15.6. Loading Custom Firmware

3) Connect the FC

With the battery **disconnected**, connect the FC to the base station via a USB to micro USB cable. Click the flash firmware button at the bottom right of the screen. Flashing will be complete once the bar at the bottom of the screen says “Programming: SUCCESSFUL”.

**Note:** If the bar instead reaches halfway and then says “Verifying: FAILED”, do not worry - it has flashed successfully.
15.4. Configure the Cleanflight settings

Now that the FC has been flashed with firmware, it can be configured.

1) Disconnect and Reconnect

1. Disconnect the FC from the base station and close Cleanflight.
2. Launch Cleanflight
3. Reconnect the FC to the workstation
4. Click the “Connect” button in the top right corner of the screen (this is not needed if “auto-connect” is toggled on).
Now you will configure each of the settings that need to be changed.

2) “Ports”

1. Go to “Ports” tab on the left side of cleanflight
2. Make sure SerialRX for UART2 is disabled and click “Save and Reboot.” UART2 is a pin on the flight controller, and we want to make sure it only uses the USB.

3) “Configuration”

Go to “Configuration” tab on the left side of cleanflight
- Change the ESC/Motor protocol to “MULTISHOT”. 
Figure 15.10. Set Multishot
- Set the Minimum Throttle to 1100.

Figure 15.11. Set Minimum Throttle
- Flip the yaw by 180° (because the FC is rotated by 180° when attached to the drone frame).
Figure 15.12. Flip Yaw

- Change the receiver to “MSP RX input” (by default it is configured to receive data from an RC receiver, but we want it to take commands over MSP).

Figure 15.13. MSP RX Input

- Finally, click “Save and Reboot.”

**Note:** On the configuration page, Cleanflight might show that the direction of your motors are reversed. This is a UI bug and can be ignored. You will ensure that your motors are spinning in the correct direction in later steps.

4) “Modes”
The FC needs to be in Angle mode for its entire available range - not just the range of acrobatic mode.

Go to the “Modes” tab on the left side of cleanflight
- Under the “Angle” option, click “Add Range”.

![Figure 15.14. Angle Mode Option](image)

- Drag the sliders so that the range spans from 900 to 2100 (i.e. entire range).
Figure 15.15. Modes Configuration

- Finally, click “Save”.

5) PID Tuning

The FC PID parameters need to be changed to work better with our drone. Go to the “PID Tuning” tab. Change the “ROLL” and “PITCH” PID terms to match the image. For reference: Roll should be (Proportional: 60, Integral: 40, Derivative: 50, RC Rate: 1.00, Super Rate: 0.00, Max Vel: 200). Pitch should be (Proportional: 60, Integral: 40, Derivative: 50, RC Rate: curly bracket, Super Rate: 0.00, Max Vel: 200). Change angle limit to 50. Finally, click “Save”.

(b) Expanded Range
Check before you continue

Double check that all of the settings in cleanflight match up to the ones above. Make sure to save the settings when possible.

6) Verify

To make sure that the settings were properly saved, you will verify just one of the
changes.
1. navigate to the configuration tab
2. Scroll down to box that’s labelled “Receiver”
3. Make sure the input says “MSP RX Input”

![Configuration](image)

Figure 15.17. MSP RX Input

15.5. Connecting the ESCs to the Flight Controller

Now that the FC has been configured, it can be connected to the ESCs via the PWM wires (i.e. the white and black wires on the ESCs). Take a moment not find the numbers **1-8** on the flight controller, adjacent to the pins that your soldered in. We will be connecting the PWM wires to number **1-4** because we have 4 motors. These numbers on the flight controller indicate which PWM wire should be connected to which set of pins 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 labelled **1** on the flight controller.
Where the red arrow indicates the front direction of the drone. Recall that for your drone, the FC is on the front and the camera is on the back.

**Note:** There is a correct way to connect an ESC to the Flight Controller. Make sure the white wire of the ESC signal wire pair is facing toward the board, and the black wire is facing away.

1) Plug each PWM wire into its corresponding ESC signal wire pair.

15.6. **Test the Motors**

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

1) **Remove Propellers**

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!
2) Launch Cleanflight

1. Open up Cleanflight on your base station.

3) Connect your drone

1. Plug your drone's FC into a computer (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.

4) Navigate to Motors tab

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

5) 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:
3. the correct motor spins. If the correct motor does not spin, reconnect the ESC wires to the FC in the correct order.
4. the motor spins in the correct direction. If the motor spins in the incorrect direction, take note and you will correct it later on. Read the following Remark and Note before continuing.

**Remark:** 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.

**Note:** 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.

![Correct Motors Diagram](a.png) ![Incorrect Motors Diagram](b.png)

Figure 15.20. Motors Diagram
Repeat this process for all of the motors.

6) **Change the Motor Directions**

Power your drone off by disconnecting the power supply. For each motor that is spinning in the *incorrect* direction:

1. Disconnect any 2 of the 3 ESC pad wires from the motor, e.g. disconnect the red and yellow ESC pad wires from their corresponding motor wires. Keep track of which wires used to be connected together.
2. Swap the connections, e.g. plug the socket bullet connector of the red ESC pad wire into the plug bullet connector of the motor wire previously connected to the yellow ESC pad wire and vice-versa.

7) **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.

---

**15.7. Calibrate the ESCs**

By this point, your drone's FC should be able to spin up each of the 4 motors. This is possible because the FC 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 FC 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 FC. In this context, **calibration** means getting all the ESCs to have the same PWM-to-RPM understanding from the FC. In this section, you will calibrate your ESCs.

Note that symptoms of no calibration include: scorching hot motors, a drone that lifts to one side during flight, motors that appear to spin at different speeds.

1) **Remove propellers**

*Make sure no propellers are attached to your drone’s motors!*

2) **Disconnect Power**

Unplug the battery from your drone.

3) **Launch Cleanflight**

1. On your base station, open cleanflight
2. Connect the FC to a computer and click “Connect” in the top right of the screen
4) Navigate to the Motors tab

1. Go to the Motors tab on the left side of Cleanflight.
2. Read the safety notice and check the box that says “I understand the risks, propellers are removed - Enable motor control”.

5) Calibrate

1. 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.

1. 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.
2. 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).

Check before you continue

- Make sure the motors spin in the correct direction, i.e. according to the motors diagram in this doc.
- When you connect the drone to power, the ESCs should make a “boop boop boop” sound, followed by a “beep BEEEEEP” sound.
UNIT B-16

Part 3: Flight Controller and Cleanflight Checkpoint

Expected Time: 20 minutes

16.1. Check 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.

Figure 16.1. Motors Diagram

16.2. Check the motor attachment
Make sure the arrows on the motors match up with the diagram above. It is important that the correct motors are in the correct spot because the motor bolts are threaded so that the propeller will not loosen the nut and fly off the drone.

16.3. Check 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 arrow on the motor, which is the same direction as shown in the image above. It helps to put a small piece of tape on the side of the motor to know which way it is spinning.

16.4. Arm your drone
You will now test out arming your motors using the drone software. To arm the motors, the software sends out a sequence of PWM values that let's the flight controller know that the operator is ready to fly. After arming the motors, the software sends out a low
PWM value that leaves the motors idling at speed much slower than what would be needed for takeoff.

In this checkpoint, you go through the process of setting up your drone to fly, and then will arm the motors and make sure they are spinning. Then you will disarm the motors and make sure they stop spinning. Then you will be done with build part 3!

1) Power up your drone
   1. Make sure the propellers are off of the motors
   2. Connect the flight controller USB to the Raspberry Pi
   3. Plug in the battery

2) Connect to your drone
   1. Connect to your drone’s wifi
   2. Browse to the drone’s code editor: http://192.168.42.1:8081

3) Start up the flight code
   1. If a terminal is not already open at the bottom of the code editor, then open a terminal: Menu (top left button) > Terminal > New Terminal
   2. Start up the flight code by typing in the command ./start and press enter

4) Connect to the web interface
   1. In a new tab, browse to the web interface: http://192.168.42.1
   2. Refresh this page and ensure it says “Connected” at the top. If not, wait a few more seconds and try refreshing the page again.

5) Start the flight controller node
   1. Navigate back to the terminal in the code editor
   2. Identify the “tick” key on your keyboard. It looks like this: ` and is typically the key to the left of the ‘1’ key. It is also typically located on the same key as the tilde `~.
   3. Navigate to the flight controller node by typing “tick” and then 1: ` 1
   4. Press enter to start the node
   5. When prompted “Are you ready to fly?”, type y and press enter since we want to spin the motors

6) Get a safety failure message!
   1. Navigate back to the web interface
   2. Attempt to arm your drone by pressing the semicolon (; ) button on your keyboard. (Nothing should happen at this point, continue on)
   3. Navigate back to the code editor and read the message in the terminal. It should look like the image below:
Explanation: The drone code contains safety checks that stop the drone from flying if there is a technical problem. Based on the message in the terminal, the issue is that the drone is not receiving data from the camera. This makes sense, because we haven’t even attached the camera, yet! You will temporarily disable this safety check for the purpose of this checkoff, and then reenable it afterwards. This is useful to practice because you may want to adjust other safety thresholds later on as you fly your drone.

7) Change the safety threshold

1. On the left side of your code editor, locate the “params” folder.
2. Open the thresholds.yaml file.
3. Locate the camera parameter underneath heartbeat, and change the value from 0.25 to 60. This means the code will check for camera errors every 60 seconds, which is plenty of time for this checkoff.
4. Close the file to avoid accidentally editing it by clicking the small “x” near the file name at the top of the editor. The file change is saved automatically.

8) Arm your drone for real!

1. Restart the flight controller: click in the terminal, press the up arrow on your keyboard, type “y” and press enter.
2. Navigate back to the web interface
3. Arm your drone by pressing the semicolon (;) button on your keyboard.
4. Verify that the motors start spinning. If not, navigate back to the code editor and see if any errors were printed out in the terminal. If there are no errors and the motors are not spinning, it is recommended that you check your cleanflight settings and make sure they’re exactly like ours.
5. Observe how the flight controller automatically adjusts to disturbances by tilting up one side of the drone and noticing how the motors that are lower are spinning faster than the motors that you’ve tilted up.
6. Disarm your drone by pressing the **spacebar**.

9) **Revert the safety threshold**

1. On the left side of your code editor, locate the “params” folder.
2. Open the *thresholds.yaml* file.
3. Locate the *camera* parameter and change the value from 60 back to 0.25.
4. Close the file to avoid accidentally editing it (the file saves automatically).

Congrats on finishing build part 3!
In this section of the build, you will attach the camera, and finalize the drone assembly.

- **Part : Quantity**
  - Raspberry Pi Camera : 1
  - Pi Mount : 1
  - Pi Mount Screws : 3
  - Brass Standoffs : 2
  - CW Propellers : 2
  - CCW Propellers : 2
  - 8mm Wrench : 1
  - Battery Mount : 1
  - Hot glue gun : 1
  - Zip Ties : 10+

### 17.1. Hardware

1) **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 that. The position of the drone is found by computing the change in position between two subsequent camera frames. The drone is also able 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 at within this photo, or “map.” Finally, the drone is able the 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 its been, creating a map.

For more information about the camera, watch these lectures (you will need to create a free EdX account to view the material)
2) Propellers

The propellors are what provide lift for the drone and allow it to fly. When the propellor spins, it creates a pressure difference between the air above and below it. The pressure below the propellor 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.

Drone are specified by three numbers. The propellors on your drone are 5 x 4 x 3. The first number, 5, is the size and it indicates the length of the blades; measured from the center to the tip. The second number, 4, is the pitch and it is a theoretical measurement of how far the propellor would travel through the air in one revolution. A larger pitch means more air is moved, so the propellor travels further, while a smaller pitch would move less air and therefore move less. The last number, 3, is the number of blades.
After completing this section, your build will match the diagram below. Compare this diagram to the completed drone diagram to see how what you’re doing now fits into the final result.

Figure 17.3. Diagram for Build Part 4

The camera provides information about the position and velocity of the drone, which is used to hold the drone in one place, or move it at a specific velocity or to a specific position. After this build part, you will be ready to fly!
UNIT B-18

Part 4: Camera, Propellers, and Mounting Hardware Instructions

18.1. Glue USB Connector to Flight Controller
These little surface-mounted micro USB ports are prone to breaking upon impact; therefore, to minimize the likelihood of damaging the connect, you will thoroughly cover the port with hot glue.

1) Plug in the micro USB to USB cable
If it is not plugged in already, plug the 6” micro USB to USB cable into the micro USB port of the flight controller.

2) Glue the port
Liberally apply hot glue over the end of the usb cable and the port on the flight controller.

**Note:** You will not be able to remove the USB cable. This is intentional. Also note that the flight controller shown is from a previous hardware version; use this as an example for how to glue your FC.

![Figure 18.1. Glued USB Connector](image)

18.2. Attach the Pi Camera to the drone frame

1) Detach the Pi Hat
Gently lift near the pin header to detach the Pi hat

2) Put tape on the Pi Camera
Put foam mounting tape on the top, left, and right sides of the pi cam. Cut off any excess tape.

**Note:** The camera in the photo does not have the white flexible flat cable (FFC) attached. This is just for the photo; it is better to leave the FFC attached to your camera.

![Pi Cam with Tape](image)

**Figure 18.2. Pi Cam with Tape**

3) **Stick camera to the drone frame**

1. Peel off the back of the foam mounting tape
2. Stick the Pi Cam to the hole in the back of the drone. Attach on top so camera faces downward and the FFC goes away from the drone. Ensure that the attachment is not skewed.
18.3. Attach Battery Mount

Use zip ties to attach the battery mount to the drone frame as shown in the image.

**Note:** Ignore the propellers on the drone in the image. You will attach those in the final step.
Figure 18.4. Battery Mount

This is how the battery will be attached to the drone:

Figure 18.5. Strapping in the Battery

(a) Battery unstrapped  (b) Battery strapped in

18.4. Attach PiMount

Place the PiMount on top of the PDB and use the short black M3 bolts from the motors box to secure the mount to the standoffs.

| Note: Ensure that the triangle shape is on the side of the camera. |
18.5. Attach Raspberry Pi

1) Insert standoffs

You will now insert standoffs into the Pi that will help support the Pi Hat and keep it from bouncing during flight.

1. Screw in a pair of standoffs into bottom right hole of the Pi. It may help to first screw in a black screw into the standoffs, then use an allen wrench to screw the standoffs into the hole. Note that the hole may initially seem too small for the standoffs, but it will definitely screw in.
2) Attach Pi to Pi Mount

Leaving the Pi Hat detached (to make the next step easier), use the sliver screws included in the drone frame kit to secure the Pi to the mount. You should use 3 panhead screws as shown in picture. The Pi USB ports should face toward the front of the drone.

**Note:** The image below does not include the brass standoff from the previous step. Please ignore this discrepancy.
3) Plug in the Flight Controller USB

Plug in the Flight Controller USB to any of the USB ports on the Raspberry Pi. The USB sends the autonomous flight commands from the Pi to the FC.
18.6. Connect Pi Cam

1) Connect FFC to the Pi Camera

If not already done, attach the FFC to the Pi Cam.
2) Connect FFC to the Pi

Feed the flexible flat cable (FFC) from the Pi Cam through the slot in Pi Hat, then connect it to the Pi’s camera port, i.e. the black port close to the HDMI port.

**Note:** Do not just push the FFC into the slot. Watch this video on how to insert the camera properly.

![FFC cable through Pi Hat](image1.jpg) ![FFC cable in Pi camera port](image2.jpg)

Figure 18.11. Connecting Pi Cam

3) Re-attach Pi Hat

Put the Pi Hat back onto the Pi. Refer to part 1 for detailed instructions to attach the Pi hat.

18.7. Attach the IR Sensor

Zip tie the IR sensor at the front of the drone, facing down. It will be underneath the flight controller and the wires will go back toward the frame.
18.8. Attach BEC to Drone

Attach the BEC to the top of the USB ports using some double sided tape. Alternatively, if it does not reach, then 1) place electrical tape on the Pi Hat, 2) place double sided mounting tape on the electrical tape, and 3) place the BEC upside down on double sided mounting tape.

18.9. Ziptie ESCs and PWM Connectors to Frame Arms

Use the small zipties to secure the ESCs and motor wires to the bottom of each frame arm. To do this, thread the zip ties through the holes in each frame arm. If they do not fit, instead wrap the zip ties around the frame arm. Also, use zip ties to secure the PWM connectors to the sides of each frame arm.
18.10. 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 motors have small arrows on them in the center to indicate which type they are.

**Note:** The bolts on the motors that spin CCW tighten when turned CW, and the bolts 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.
Propeller Flat on Motor

Figure 18.15. Put on Propellers

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.

18.11. Attach Drone Feet

Attach the drone feet in the drone frame box to the four arms of the drone. This can be done with either electrical tape or zip ties. **DO NOT** use hot glue, as it will not work.
Check before you continue

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

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.
UNIT B-19
Part 4 Checkpoint

19.1. Overview
You will now make sure the camera is working properly by viewing the camera data on the web interface.

19.2. Connect to the drone

1) Connect to the WiFi
Plug the battery into your drone build and connect to the drone’s wifi network, default-drone

2) Open the web interface
Browse to the web interface: http://192.168.42.1

3) Open the code editor
In a new tab, browse to the drone’s code editor: http://192.168.42.1:8081

19.3. Start up the code

1) Open a terminal
If there is not a terminal at the bottom of your screen already, then:
In the code editor, click the menu bar in the top left corner, then click Terminal > New Terminal

2) Run the start command
Type this command and hit enter: ./start

3) Start the flight controller node
1. Identify the “tick” key on your keyboard. It looks like this: ` and is typically the key to the left of the ‘1’ key. It is also typically located on the same key as the tilde ~.
2. Navigate to the flight controller node by pressing ` followed by the number 1. Note that nothing will appear in the terminal when you type these keys.
3. Press enter to start the node
4. When prompted, “Are you ready to fly?” Type n and press enter (we do not want to fly at this point.)

4) Locate the x and y velocity graphs
1. Go back to the web interface tab, wait about 5 seconds, and refresh the page. Make sure that you see “Connected” at the top of the page
2. Find the x and y velocity graphs on the web interface

5) Test
1. Orient your drone so that the flight controller is facing away from you
2. While using your hand to move the drone to the right, verify that the x velocity graph shows positive reading
3. While using your hand to move the drone to the left, verify that the x velocity graph shows negative readings
4. While using your hand to move the drone forward, verify that the y velocity graph shows positive readings
5. While using your hand to move the drone backward, verify that the y velocity graph shows negative readings

Congrats on finishing your build!

19.4. Debugging
If you do not see anything on the velocity graphs:
1. navigate to the terminal in the code editor
2. go to the camera node by typing “tick” 4: ` 4

If you see an error printed that says something like “out of resources other than memory” the the issue is that your camera cable is not connected properly.
• make sure there are no holes or tears in your camera cable
• unplug the batter and then reinsert the camera cable into the camera and into the Pi
• repeat the steps of this checkoff again
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 multimeter, 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. This is a general tutorial for multimeters.

**NOTE:** Please turn off the multimeter by setting the dial to OFF after you finish your check.

20.1. 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 friction.

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](image1)

   ![Continuity test mode](image2)

   - 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.
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.
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. 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 (-).

### 20.2. DC Voltage Check
1. **Selecting the DC Voltage Mode**
2. Switch on your multimeter, and set the dial to DC voltage mode (indicated by a V with a straight line, or the symbol `==`).
3. **Performing the Voltage Check**

4. Place the positive (i.e. red) lead on a positive (+) terminal, and the negative (i.e. black) lead on the negative (-) terminal.

5. See the screen for a voltage measurement.

6. **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.

### 20.3. AC Voltage Check

1. **Selecting the AC Voltage Mode**

2. Switch on your multimeter, and set the dial to AC voltage mode (indicated by a V with a wavy line, or the symbol \(\tilde{V}\)).

3. **NOTE:** AC voltage does not have polarity.

4. **NOTE:** Do not let your fingers touch the lead tips. Do not allow the tips to contact one another.
5. **Performing the Voltage Check**

6. Place the positive (red) lead on the positive terminal, and the negative (black) lead on the negative terminal.

7. We can get a reading from the screen now.

8. **NOTE:** Reversing the leads won’t do any harm; it just gives us a negative reading. This contains more detailed information about AC voltage test.
Help, I accidentally cut off a chunk of my wire while stripping!
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.

Help, I cut off several strands of wire while stripping!
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.

Help, I put too much solder on my wire while tinning!
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.

Help, my wire is picking up random particles while tinning!
Clean the tip of your soldering iron with soldering wool. For future prevention, do this cleaning more frequently while soldering.

Help, the alligator clips of my helping hands are loose!
Remove the offending alligator clip from the helping hands, then use pliers to carefully pinch the end tighter. Re-insert once fit is tight.

Help, I burned some insulator onto my wire while tinning!
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.

Help, I can’t tell if I tinned my wire properly!
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.

Help, my solder keeps melting into a sphere shape instead of melting onto a wire!
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, or 3) the soldering iron station no longer works properly. Systematic debugging is required to determine the cause, preferably in the order listed. When the culprit part is determined, replace with a new one.
In this section of the textbook, we will introduce you to the software of the drone, and how it interacts with the hardware you put together in the build project. 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.

Software Architecture Diagram
INTRODUCTION

Figure 1.1

Key

Hardware Component

ROS Node

State

ROS Topic

Interface with Hardware

Figure 1.1
2.1. Overview

The Robot Operating System (ROS) is widely used robot middleware that makes communication 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.

2.2. General Components:

1) 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 is 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 PiDrone, `roscore` is called in 0 of the screen.

2) 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 PiDrone, each window of the screen is a ROS node.

Creating a ROS node in Python:

To create a ROS node in python, you first need to import rospy at the top of the file using `import rospy`. Then, initialize a ROS node using `rospy.init_node("node_name")` where "node_name" what you want to call this node. For example, if you were creating the mode_controller, you could write `rospy.init_node("mode_controller")`. After you've initialized the node, you can create any number of publishers using: `[publisher_variable_name] = rospy.Publisher("[topic]", [message_type])`. For example, if you wanted to publish the commanded mode you could write: `commanded_mode_pub = rospy.Publisher("/pidrone/command/mode", Mode)`. You can then publish messages of the type [message_type] by creating a message of that type by importing it, instantiating it, and changing the value of its fields. Using `[publisher_variable_name].publish([message]). For example, you could import our custom mode message using `from pidrone_pkg.msg import Mode` and then instantiate it with `mode_msg = Mode()`. Then you can edit the field, mode as follows: `mode_msg.mode = "ARMED"`. You could publish this message using `modepub.publish(mode_msg).`
2.3. Messages

The mode message referenced above is a custom message that we created, and to know what fields it has, you can look in the msg folder of the pidrone_pkg. To see what fields that 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 hierachal 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.

2.4. 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 a free 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 infrared sensor, you could enter rostopic echo /pidrone/infrared

2.5. Publishers

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

2.6. 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 everytime a message is published to the topic. For example, if you wanted to update the height of the drone everytime a message was published, then in a ROS node you would first create a subscriber using `rospy.Subscriber("/pidrone/infrared", Range, infrared_callback_method)`. Your callback method might look something like:

```python
infrared_callback(msg):
    drone_height = msg.range
```
UNIT C-3

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.

3.1. 0: roscore
Starts up a ROS master to allow the nodes to find each other.

3.2. 1: Mode Controller
Mode controller 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. You will need to start this node explicitly after starting up a new screen.

- Python script: mode_controller.py
- Hardware interfacing: None
- Publishers:
  - `/pidrone/commanded/mode`
- Subscribers:
  - `/pidrone/mode`
  - `/pidrone/desired/mode`
  - `/pidrone/battery`
  - `/pidrone/heartbeat/infrared`
  - `/pidrone/heartbeat/web_interface`
  - `/pidrone/heartbeat/pid_controller`
  - `/pidrone/heartbeat/flight_controller`

3.3. 2: Command Line Interface
Command Line Interface is a text-based UI that allows you to control the drone. You will still need to connect to the drone using the web interface so that the web_interface heartbeat is publishing.

- Python script: command_line_interface.py
- Hardware interfacing: None
- Publishers:
  - `/pidrone/desired/mode`
  - `/pidrone/desired/pose`
  - `/pidrone/desired/twist`
3.4. ‘3: Flight Controller
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. The Flight controller listens to the mode controller over the desired_mode topic to know what if it should listen to the fly_commands from the pid_controller which tell give it values to fly. 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 to the flight controller board.

- Python script: flight_controller_node.py
- Hardware interfacing: flight controller board
- Publishers:
  - “/pidrone/imu”
  - “/pidrone/battery”
  - “/pidrone/mode”
  - “/pidrone/heartbeat/flight_controller”
- Subscribers:
  - “/pidrone/commanded/mode”
  - “/pidrone/fly_commands”

3.5. ‘4: PID Controller
The controller node could be any controller which takes the current position of the drone and attempts to drive the drone to the desired position. We use a widely used feedback controller known as a PID. You’ll be implementing this in a project later on. In short, the PID controller uses the error between the desired and current values to compute roll, pitch, yaw, and throttle values to send to the flight controller.

- Python script: pid_controller.py
- Hardware interfacing: None
- Publishers:
  - “/pidrone/fly_commands”
  - “/pidrone/position_control”
  - “/pidrone/heartbeat/pid_controller”
- Subscribers:
  - “/pidrone/state”
3.6. `5: State Controller

State Controller subscribes to all of the sensor data and uses a filter to estimate the state of the drone. 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:
  - “/pidrone/picamera/twist”
  - “/pidrone/picamera/pose”
  - “/pidrone/infrared”
  - “/pidrone/imu”
  - “/pidrone/reset_transform”

3.7. `6: Vision

The vision node interfaces with the picamera to provide velocity and position estimates. There are several vision files used to do this. vision_flow_and_phase.py uses the python scripts analyze_flow.py and analyze_phase.py to estimate the velocity and position of the drone using optical flow vectors and rigid transformations, respectively. These values are published for use by the state_estimator. Another vision node is ‘vision_localization_offboard.py’, which uses localization to provide the position of the drone to the state estimator. Others exist as well, and you will be writing vision nodes in future projects.

- Python script: vision_flow_and_phase.py
- Hardware interfacing: Pi camera
- Publishers:
  - “/pidrone/picamera/twist”
  - “/pidrone/picamera/pose”
- Subscribers:
3.8. '7: Infrared

Infrared is used to interface with the analog to digital converter (ADC) which is connected to the infrared sensor. This node is used to convert the data from the ADC into usable range readings to publish for use by the state estimator.

- Python script: infrared_pub.py
- Hardware interfacing: Infrared sensor
- Publishers:
  - “/pidrone/infrared”
PART D
Flying

In this section, we will describe the steps necessary to get your autonomous quadrotor flying!
There are several options for your Raspberry Pi’s network structure with different tradeoffs. Here we give a summary of different options.

1.1. Drone as Access Point

The drone comes by default configured to act as an Wifi access point, and you connect your base station to the drone’s wifi as a client. It will use hostapd to give your base station an IP address via DHCP. This configuration is the default configuration out of the box. It has the advantage that your drone can work anywhere, without any existing wifi network, and without requiring you to change any settings on the drone before connecting to it with your base station. It has the disadvantage that by default, the drone only has one wifi card and no wired access when it is flying. Thus the drone does not have access to the internet in this configuration. Nor does your base station have internet access while connected to the drone, unless you have provide an alternative means. For most people who use wifi to connect to the Internet, this mode presents a problem.

You can ameliorate the internet access problem by using a wired connection to either plug your drone into the wall, or into your base station. If you plug your drone into the wall, it will use DHCP to request an IP address, and also route traffic from your base station to the Internet and back, so both the Raspberry Pi and your base station will have internet. Alternatively, you can plug the drone into your base station. To make this mode work, you must first configure your base station to perform internet connection sharing, which is possible on most operating systems. Then the drone will obtain a DHCP address from your base station and be able to access the internet when the base station is connected to ambient wifi, and you can ssh to the drone through the wired connection.

Of course, both of these solutions require an Ethernet cable to be connected to your drone, which is problematic for flight.

1.2. Drone in Managed Mode

A second option is to connect both your base station and your drone to the wifi network. Your drone will be in AP Managed mode, and obtain an IP address and internet access in the same way as your base station. You can then use this wifi network to connect between the base station and the drone. The advantage of this approach is that both machines have internet access, wirelessly. The disadvantage is that the drone must be configured to access the ambient network, with its SSID and password, and this network must allow direct connections between two clients on the network. This network must be available and working in order to fly. Furthermore you must find the drone’s IP address from your base station, unless you want to plug in a keyboard and monitor into the Pi to find out its IP address.
1) Managed Mode Configuration

These instructions will save your network's credentials (network name, password) across reboots. You only have to follow these steps the first time you wish to connect to a new network.

**On your base station:**
1. connect your base station to the default drone wifi network
2. browse to your drone's code server at the default ip address and port 8081: 192.168.42.1:8081
3. open a new terminal: Menu > Terminal > New Terminal

**In the terminal:**
1. navigate to the networking directory in the pidrone_pkg: `roscd pidrone_pkg/networking`
2. run the bash script to generate a wpa_supplicant.conf file for your network: `./generate_wpa_supplicant_conf.sh`
3. fill in the prompt for your network's ssid (your network name) and press enter
4. fill in the prompt for your network's password and press enter. Note that you will not see anything happening when you are typing the password—this is a security feature.
5. after the script finishes, move your newly created wpa_supplicant.conf file to the `/etc/wpa_supplicant` directory: `sudo mv wpa_supplicant.conf /etc/wpa_supplicant/`. Note that you will need to enter the user password when using `sudo`. By default, this is bigbubba

2) Switching to Managed Mode

These instructions need to be followed every time you wish to switch your drone to managed mode. The drone always starts up as a WiFi access point so that you will always be able to connect to your drone, even if you are not in range of the network you usually connect to.

**In the terminal:**
1. navigate to the networking directory in the pidrone_pkg: `roscd pidrone_pkg/networking`
2. run the bash script to connect your drone to the wifi network configured previously: `./connect_to_user_wifi.sh`
3. connect your base station to the same network that your drone is now connected to
4. reconnect to your drone

For this step, you can try to use your drone's hostname to connect; however this may not work. If using the hostname does not work for you, you will need to find your drone's IP address. Follow the instructions in this article to find your drone’s IP address. Note that your drone’s hostname is duckiesky-drone, not raspberrypi which is used in the article.

a) First, try connecting via hostname:
browse to your drone's code server: duckiesky-drone.local:8081

b) If connecting via hostname did not work, then connect via IP address: ip_address:8081

**Note:** If you are connecting over ssh instead of the terminal in the code server, you will need to specify the username. For example, `ssh duckiesky@duckiesky-drone.local` or `ssh duckiesky@ip_address`
UNIT D-2

Flying Your Drone

In order to fly, you will need:
- Fully assembled drone
- Charged battery
- Base station
- Safety goggles
- Highly textured planar surface, such as a poster board with scribbles. This should have considerable and distinct markings for the camera to process. Alternatively, a patterned carpet will work.

Figure 2.1. Example of a highly textured planar surface.

2.1. Environment Checks
- 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 carpet will work fine as well.

Figure 2.2. Example of a highly textured planar surface.
2.2. Hardware Checks

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

1) 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.

2) USB Connection

Make sure that the flight controller USB is plugged into the Pi. Any of the USB ports will work.

2.3. Software and Sensors Checks

1) Power

Plug the battery into your drone.

2) Connection

Connect to your drone’s wifi network. By default, the wifi is called defaultdrone and the password is bigbubba

3) Code Editor

In a new tab, browse to your drone’s code editor: 192.168.42.1:8081

4) Start the Flight Code

1. If there is not a terminal open at the bottom of the code editor, then open a new terminal: Menu (three horizontal lines in the top left) > Terminal > New Terminal
2. In the terminal, type type ros cd pidrone_pkg and press enter.
3. Calibrate the accelerometer by typing python scripts/calibrateAcc.py and press enter.
4. Start up the flight code by typing the command ./start and press enter.

5) Start the Flight Controller Node in Sensors-only Mode

1. In the terminal, navigate to the flight controller node by typing “tick 1” : ` 1`. The "tick" ` key is typically located to the left of the 1 key, and is on the same key as the tilde ~. Note that you will not see anything appear when you are typing ` 1
2. Press enter to start the python script
3. When prompted, “Are you ready to fly?” type n and press enter to run in sensors-only mode. This allows us to use the flight controller sensors without the chance of the motors spinning while we are testing the ir and camera sensor in the next steps.

6) Web Interface

1. In a new tab, browse to your drone’s web interface: http://192.168.42.1
2. Press connect on the web interface to connect to your drone. If the web interface does not say “connected”, then wait about 5 seconds and refresh the page.

7) Check the IR Sensor
While looking at the Height Chart in the web interface, move the drone up and down and make sure you see changes in the IR graph on the web interface.

8) 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.

9) Check the ROS Nodes
In the terminal of the code editor, go through each of the screens using `n, where n is a number 1-5, 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”.

1. tick 1 is the flight controller node; you’ve already seen this one
2. tick 2 is the pid controller; this sends the roll, pitch, yaw, and throttle commands to the flight controller
3. tick 3 is the state estimator; this combines sensor data to estimate the state (position, velocity, orientation) of the drone
4. tick 4 is the camera node; this gets velocity and position data from the camera
5. tick 5 is the infrared node; this gets height readings from the IR sensor

2.4. Fly

1) Familiarize yourself with the keyboard commands
Find and read the keyboard commands to control the drone at the bottom of the web interface. 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 primary keys you’ll need are the spacebar (to disarm the drone), the semicolon ; (to arm the drone), and t (to takeoff). The most important key is the spacebar.

If anything goes wrong be prepared to immediately hit 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.

2) Orient the drone
Rotate the drone so that the camera end is facing towards you and the flight controller is facing away from you. In this way the keyboard controls (i,j,k,l) will match the
drone’s orientation. I - forward (flight controller side), J - left, K - backward (camera side), L - right

3) Restart the Flight Controller Node for Flight

1. In the terminal in the code editor, navigate back to the flight controller node using tick 1.
2. Quit the flight controller node by typing control-c (hold down the ctrl key and press the c key)
3. Rerun the flight controller script by pressing the up arrow on your keyboard (this brings back the last script that was run) and then press enter
4. When prompted, “Are you ready to Fly?” type y and press enter to start the flight controller node

4) Takeoff sequence

1. First arm your drone by pressing ;. The propellers should start spinning slowly. If they spin fast, or you hear strange noises, immediately disarm the drone. If they still do not spin, check the flight_controller_node. This is where any error message will be printed out.
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!

2.5. Flying Modes

There are two modes of flight for the drone, velocity control and position control. In velocity control, the drone tries to maintain zero velocity in the x and y directions; however, the drone can drift over time using just a control loop on the velocity. In position control, the drone is able to prevent itself from drifting. Information on each mode is included below. This video shows velocity mode, where it drifts, followed by position mode, followed by velocity mode again.

1) Velocity Control

The default mode when starting is velocity mode, where the keyboard commands control planar velocity. When no keys are pressed the drone’s velocity setpoint is zero, so it tries to maintain still. It estimates its velocity using the optical flow computed from

...
the camera frames. This estimate only works over a textured surface; when flying over a non-textured surface it will cause the drone to inaccurately estimate its velocity and fly out of control. A repeating texture is fine, as long as it has texture (e.g., a carpet with a pattern). Using just the velocity, the drone will tend to drift over time. The key $v$ activates velocity mode, and the drone is always in this mode on when it starts flying.

2) Position Control

Because velocity mode can drift, we have implemented a position hold mode, where the drone computes its offset relative to automatically detected features from the downward pointing camera. This mode must be activated over a planar surface with a non-repeating texture, such as a poster board with scribbles in different colors and shapes. When position mode is activated, it takes a picture of the first frame, and then continuously estimates its offset from this frame. If it sees features in the first frame, it computes its position relative to the first image; otherwise it computes its position relative to the previous image it saw, and adds the change in position to its current position.

Type $p$ to active position hold, and $r$ to reset the first frame where offsets are computed. A well-tuned drone can hold the same position indefinitely on power, and for almost the entire battery when on battery. (At the end of the battery it will oscillate more and lose its position.)

When in position mode, the keyboard commands tell the drone to maintain a setpoint that is a defined offset from the origin, defined by the saved first frame. That is, if you fly left, it will try to hold its position a defined offset to the left of the first frame. Of course if the drone gets too far from the first frame, it can no longer compute a global estimate and will drift away. To rectify this problem, you need to compute a global map and localize in that map, as described in the next section.

**Note:** If the drone drifts too far without being able to make a position estimate, the drone automatically switches back to velocity control. This is a safety feature and the threshold can be adjusted in the `pidrone_pkg/params/thresholds.yaml` file.
The Duckiedrone includes three approaches for low-level state estimation (velocity and integrated position). The default state estimator is the Exponential Moving Average, which simply maintains a moving average across different sensor estimates. We have also implemented several variants of an Unscented Kalman Filter, which uses a control input and a measurement input to enable the estimator to smooth sensor measurements while also updating quickly when the drone changes state.

To see options for the state estimator, run `python state_estimator.py --help`. You can also read more about the Unscented Kalman Filter in the Duckiesky Learning Materials chapter on this topic.

When the UKF is working you can see a graph appear in the web interface along with the raw sensor value for the height.
UNIT D-4
Mapping and Localization

The Duckiedrone can perform localization with a map created from an image, as well as the ability to run SLAM (online or offline). Both methods use the drone’s camera to produce a pose estimate, but localization requires a map of the environment before hand, represented as images stitched together (map.jpg) whereas SLAM builds a map of the environment as it runs. SLAM does not run well online even with a fast base station. However it gives good performance when run offline to create a map, and then later using that map to localize.

These scripts replace the vision_flow_and_phase.py script that does velocity and position control. The reason is that we save time and memory by avoiding sending images to different nodes on the Pi and instead do all processing in a single node that directly connects to the camera.

The recommended workflow is to first run SLAM to create a map offline. Then after creating the map, run localization with the saved map to give the drone a global position estimate.

4.1. SLAM

The recommended mode for SLAM is to run it offline; there is not enough compute even offboard to run it during flight. It may be possible to run SLAM online with additional optimization or by rewriting the algorithm in C++; for this reason we include instructions for running online as well. Note that if you run anything offboard, ROS must be installed on the offboard machine. Running offboard is not required to run SLAM; however it may be significantly faster to create the map if you use a fast base station.

**Offline SLAM:**

**Save Flight Data:** The first step to offline SLAM is to fly the drone and collect image data to build the map. To do this, run ‘vision_localization_offboard.py –offline’ which will save the image data from a flight to ‘flight_data.txt.’ Press ‘r’ to toggle recording the data. After the second press, the script will write the data to the file and will let you know when it is done (this may take ~10 min for a longer flight).

**Build a Map:** This can be done either offline or online. Run “offline_slam.py” which will look for the “flight_data.txt” file and save a map to a file called “map.txt.”

**Localize:** Currently this only works onboard, but we will add offboard support soon. Make sure your “map.txt” file is in the scripts folder and run “vision_localization_onboard.py –read” which will perform localization over the map created by SLAM. Press “r” to restart the localization.

**Online SLAM** Online SLAM runs but not in real time, even offboard. Therefore we do not recommend it until and unless we make it fast enough to run in real time. However we are including instructions in case someone wants to try! For online slam:

**Offboard:** run vision_localization_offboard.py on the pi. On the offboard computer run offboard_slam.py. Press r to toggle SLAM.
Onboard: run `vision_localization_onboard.py` --SLAM on the pi. Press `r` to toggle SLAM.

4.2. Localization with a Stitched Map

These instructions describe how to create a stitched image of a map with your cell phone or other camera. Once you have created a map, you can use it to localize.

Take photos of the new map with a cell phone or other camera. Take them at a height of 25cm pointed downward at the workspace. Use an image stitching software to generate the map. We recommend auto-stitch or hugin. Replace map.jpg with your new map and change the following four parameters in `offboard_localization.py`, `onboard_localization.py`, and `localization_helper.py`: MAP_PIXEL_WIDTH, MAP_PIXEL_HEIGHT, MAP_REAL_WIDTH, MAP_REAL_HEIGHT. You may need to resize the image to be smaller if it is too large.

Onboard: run `vision_localization_onboard.py` on the pi. You must fly over the area captured in map.jpg. Press `r` in the web interface to toggle localization.

Offboard: run `vision_localization_offboard.py` on the pi and `offboard_localization.py` on the remote computer. You must fly over the area captured in map.jpg. Press `r` to toggle localization.
PART E
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!

0.3. Power Issues
A common issue is that your PI won't boot or the motors won't turn on. You should verify that each part of the drone is receiving power. The Pi indicates it has power with a red power LED. 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 Pi is receiving 5 volts from the BEC.

0.4. Pi Issues
Most of the next debugging steps require getting “into” your Pi. So if your PI doesn’t boot and display an access point, you are stuck. If your Pi is not powering on, verify with a multimeter that the 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.

If your 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 the Pi, so try replacing the Raspberry Pi.

If it is receiving power and turning the red light on, then something might be wrong with your SD card. Verify that your SD card has the correct image flashed on it, and that it is seated in the Pi so that it can boot. If all this doesn’t work, then find a keyboard and monitor to plug the 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.

0.5. Camera
Now verify that the camera is working. If the camera node is not starting in screen, 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 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 parts of the cable should be facing the pins in the slot.) And make sure it is seated all the way.

If none of these things make “raspistill” work, try plugging your Pi into someone else’s camera, and someone else’s Pi into your camera, and try all of the above debugging steps. You can also consider 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 Pi and cable works with someone else’s camera but not yours, try replacing the camera. If your camera and cable work with someone else’s Pi but not yours, try replacing the Pi.

0.6. Range Sensor

The range sensor passes information through the Adafruit board. Use the multimeter to verify the three wires going to the range sensor have sensible values. Check power to ground first; then check the signal wire to ground. You should see the signal voltage change in the multimeter based on the range reading (moving your hand closer and farther from the sensor.)

If that is working, then check the Adafruit board. Verify that you can read the analog range voltage reading on the appropriate pin on that board. Then verify that it is getting power.

If all that looks good, then check the connection to the Adafruit board and the Raspberry pi. Make sure the Pi GPIO pin is accurately reading the adafruit output value.

0.7. 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 Pi, make sure you can calibrate the accellerometer, and run the flight controller node. If those don’t work, go back and recheck your Cleanflight configuration. Sometimes the flight controller gets wedged into a state where you can’t talk to it in Cleanflight. You can get it unwedged by shorting two pins on the board and following the instructions in Cleanflight. Once you do this though, you’ll need to set the Cleanflight configuration again.

0.8. Flight Issues

Before each flight, physically inspect the drone. Make sure that your camera is mounted firmly, pointed downwards. Make sure the range sensor is pointed downwards and hasn’t gotten rotated. Make sure the flight controller board is level and firmly attached, or the IMU and gyroscope will return incorrect readings. Any of these issues could cause poor flight behavior. Also make sure each propellor is tightened down all the way.
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 stricking 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 (this video)[https://www.youtube.com/watch?v=csCBMF8P3qg].

After doing all these steps, you can try to tune the PID gains as described in our PID project.

A well-tuned drone can hover with velocity zero with some drifting, but not tons of drifting. It should be able to hover with position hold indefinitely.
PART F
Diagnostic

0.9. Is your battery charged?
use the multimeter to check the battery voltage by inserting the probes into the battery terminals. A full battery is 12.6 V, an empty battery is around 10.5 V. If your battery voltage is below this, especially below 9 V, the battery may be permanently damaged and needs to be recycled.

1) Yes
continue on

2) No
charge or replace the battery

0.10. Does the BEC have a solid green light?

1) Yes
continue on

2) No
• check the voltage going into the BEC from the PDB
• make sure that the BEC is wired the correct way (red wire goes to positive (+), black wire goes to negative (-))
• Make sure that the OUT side of the BEC is soldered to the Pi Hat, and the IN side is soldered to the PDB
• Make sure there are no stray wire hairs that are shorting out the 5V and the GND rails on the Pi Hat

0.11. Does the red light of the Pi stay on solid
This is the small led near the SD card

1) Yes
continue on

2) No
If the light is blinking, then the Pi is not receiving enough power. Check the voltage coming out of the BEC. It should be a constant 5 V
If the light does not turn on, then the Pi is not receiving any power.
• Make sure that the Pi Hat is attached to the Pi all the way (there is no gap between the GPIO pins and the Pi Hat pin header).
• Make sure that the OUT side of the BEC is soldered to the Pi Hat, and the IN side is soldered to the PDB
• Make sure that there is not a short between the power and ground rails on the Pi Hat.
• Make sure there are no stray wire hairs that are shorting out the 5V and the GND rails on the Pi Hat

0.12. Can you connect to your drone’s WiFi?

1) Yes
continue

2) No
• Make sure that you have flashed your SD card
• Make sure that the green LED on the Pi near the SD card is blinking. If this light is not on, then the SD card is not flashed properly. Re-flash the SD card, and if this does not work try a new SD card.

0.13. Can you access your drone’s code editor?

1) Yes
continue

2) No
If the drone is in access mode (you’re connected to the drone’s WiFi):
• make sure that the link is opening to the right address. Try manually typing 192.168.42.1:8081 and press enter in your browser
• try another browser. Google Chrome is the browser and know to work correctly
If the drone is in managed mode (the drone is connected to your network):
• make sure that you’re using the drone’s new ip address (it will no longer be 192.168.42.1). If you are trying to access the drone via hostname instead of ip, make sure that you append .local, so it would be duckie-drone.local:8081

0.14. Can you access your drone’s web interface?

Does the web interface show up when you browse to 192.168.42.1? Don’t worry for now if the interface connected or not, just whether or not the interface appears

1) Yes
same steps as the code editor, but the address should be 192.168.42.1, or duckiesky-drone.local, or new-ip-address if the drone is in managed mode.

0.15. Does the flight code start?

use roscd pidrone_pkg and then run ./start. Does the terminal look different?

1) Yes

continue

2) No

- make sure you’re running the start script in the correct directory: ~/ws/src/pidrone_pkg

0.16. Does roscore startup?

After running the start code, type 0, where is the key to the left of the 1 on your keyboard, and is on the same key as the ~. Does the last line on this screen say started core service [/rosout]?

1) Yes

continue to check the FC node

2) No

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. continue on to the next checks.

0.17. Are there multiple screens running?

In the terminal, type screen -ls. Were any sockets found?

1) Yes

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.
0.18. No
continue

0.19. Is the ROS_MASTER_URI set
In the terminal, type `echo $ROS_MASTER_URI`. Did the terminal print out `http://localhost:11311`?

1) Yes
If you’re on the drone’s network, continue on.
If you’re drone is managed mode, then type `export ROS_MASTER_URI=[ip address]:11311` where `[ip address]` is the ip address of the drone on your network.

0.20. Is the Flight controller node running?
Navigate to the flight controller node using “tick 1”: 1. Press enter to start the node, and then type `y` and press enter when prompted if you’re ready to fly. Is the last line printed out “/pidrone/battery”?

1) Yes
continue

2) no
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. (it should be hot glued in)
- 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. Was it glued down?
- 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.

0.21. Is the PID controller running?
Navigate to the PID controller node using “tick 2”: 2. Is the last line printed out PID Controller Started?

1) Yes
continue

2) No
• Try quitting the script with ctrl-c and rerunning it

### 0.22. Is the state estimator running?

Navigate to the state estimator using “tick 3”: `3. Is the last line printed out Starting filter?`

1) Yes
   
   continue

2) No
   
   • 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 re-running this script

### 0.23. Is the vision node running?

Navigate to the vision node using “tick 4”: `4. Is the last line printed out Vision started?`

1) Yes
   
   continue

2) No
   
   If the last few lines say something like: “out of resources other than memory”, then the issue is the physical connection from the camera to the 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 Pi.
   
   • On the Pi, make sure the blue side of the FFC is facing towards the USB cables
   
   • 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

### 0.24. Is the IR node running?

Navigate to the IR node using “tick 5”: `5. Is the last line printed out Publishing IR?`

1) Yes
2) No

- make sure that the IR sensor wire is firmly plugged into the IR sensor
- check the connections on the ADC:
  - make sure there are no shorts between adjacent pins
  - make sure there are no stray wire hairs causing shorts
  - make sure the yellow signal wire of the IR sensor connected to “A0” on the ADC
  - make sure there is 5V across the “V” and “G” pins
- Try rerunning the node. If it still does not work, go back through Build Part 2 to check all of the wiring.

**0.25. Is rosbridge running?**

Navigate to the rosbridge node using “tick 6”: `6. Does the last line printed out include the phrase Rosbridge websocket server started on port 9090?

1) Yes
continue

2) No

- make sure you have no other programs using the same port as the Rosbridge server (9090)
try rerunning this script.

**0.26. Does the web interface say connected?**

Open up or refresh the drone’s web interface: 192.168.42.1. Does the interface say “Connected” at the top?

1) Yes
continue

2) No

- make sure you are not using Microsoft Edge browser. Chrome is the preferred browser
wait another 10 seconds and try refreshing again.

**0.27. Is there data on the height readings graph?**

When you move the drone up and down, does the height reading graph change?

1) Yes
2) No
recheck the IR node. rerun the node if needed and then refresh the browser.

0.28. Is there data on the x and y velocity graphs?
Move the drone to the left and right over a textured surface. Do the X and Y velocity graphs change?

1) Yes
continue

2) No
recheck the vision node. rerun the node if needed and refresh the browser.

0.29. Is the data on the graphs lagging?
Is there a long delay between when you move the drone and when the graphs change?

1) Yes
If you are running the drone in access mode, then try quitting the screen with tick colon quit: `:quit`. rerun the flight code with ./start and start up the flight controller again. If this does not help, unplug and plug back in the battery to the drone and try again.

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 access mode.

2) No
continue

0.30. Does the drone arm?
Click on the web interface and then press the semicolon: ;. Do the motors on the drone start spinning?

1) Yes
continue

2) No
Plug the USB into a computer with Cleanflight and verify that the settings in cleanflight from Build Part 3 FC.
When going back through the settings, double check that:
• 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 Pi over USB) in the “Configuration” tab
• the ESC/Motor protocol is set to multishot in the “Configuration” tab.

0.31. Does the drone get off the ground?

1) Yes
continue

2) No

• 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 same direction as the arrows on the motors
• Take off the propellers from your motors and plug the battery into your drone. In cleanflight, 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
Using a multimeter effectively is an essential part of debugging any electronics project. Achieving multimeter mastery is beyond the scope of this course, but there are two modes that you should be aware of and use as part of your work.

0.32. Checking for Shorts
Most multimeters have a mode where they will beep if the two probes are connected via a circuit. You can use this mode to check that two outputs on your drone that should be connected are connected. Do this checking with the drone powered off. For example, verify that the PI GPIO pin that is reading the Adafruit input is connected to the correct output port on the Adafruit board. And verify that the 5V output from the BEC is connected to the 5V input on the Pi. And then verify that the power pad on the power distribution board is NOT connected to the ground pad on the same board.

0.33. Checking Voltage
When the drone is powered on, you can check each part of the drone for shorts. Voltage is a measure of relative electric potential between two different parts of the project. For example, verify that the power pad and ground pad on the PDF show a 12 volt difference. (It's okay if it varies between 11 and 12.5 volts, depending on the state of your battery charge.) Verify that the Pi’s power and ground pins show a 5 volt difference. Verify that the Adafruit board’s power and ground pins show a 5 volt difference as well.