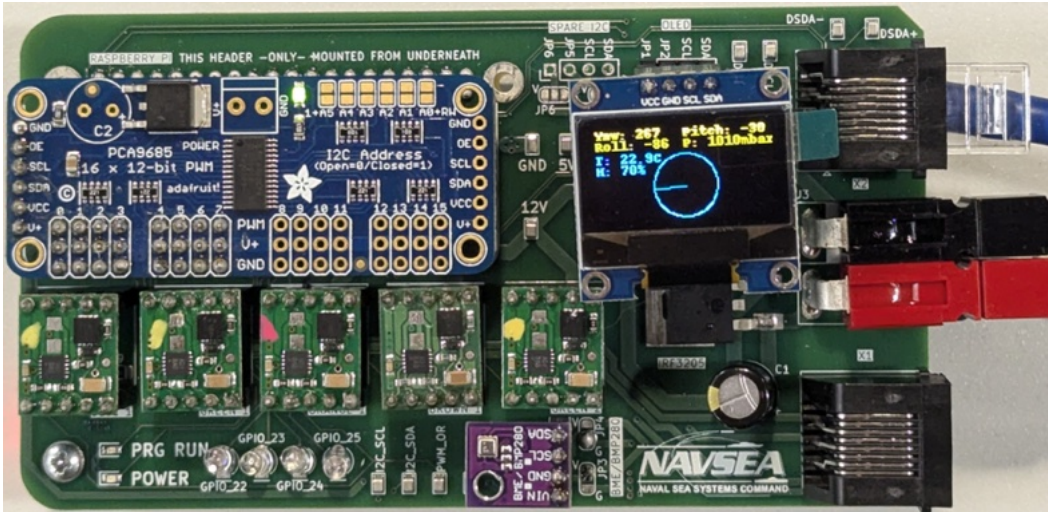
Raspberry Pi Single-Board-Computer Control of a USB Gamepad for Use With SeaPerch

Undersea Technology Apprentice Program
Educational Outreach
Naval Undersea Warfare Center
Newport, RI

john.dicecco.civ@us.navy.mil
diceccoj@ele.uri.edu

July 9, 2024



Abstract

This document describes the use of the Raspberry Pi Rev. 4 single-board-computer (SBC) to interface with a Universal Serial Bus (USB) gamepad for directional and variable speed control of the underwater vehicle SeaPerch. This forms the basis of the Undersea Technology Apprentice Program (UTAP), held at the Naval Undersea Warfare Center, Division Newport, RI. The hardware and software components as well as the basic circuit requirements will be discussed. Due to the complexity of the circuit, a printed circuit board (PCB) will be provided, to which the student will add all the necessary electrical components. A basic working Python (a software language) program will be provided that will enable the Raspberry Pi to interface with the USB gamepad, establishing variable speed thruster control to the SeaPerch. Teams of students will modify the baseline SeaPerch mechanical and propulsion design to complete a series of challenges simulating real-life mission scenarios. The students will modify the stock Python code to accommodate the new vehicle configuration and propulsion layout. Fully completed hardware and software designs will compete against other designs to determine the best design and execution.

****NOTE: This document assumes you have already built a standard SeaPerch kit!****

1 Introduction

The Raspberry Pi (RPi) model 4 is a Broadcom BCM2711, Quad core Cortex-A72 (ARM v8) 64-bit System on a Chip (SoC) running at 1.5GHz single board computer capable of hosting a number of Linux distributions (operating systems). This means, for all intents and purposes, the Raspberry Pi is computer in the modern sense of the word. It can establish communications on the Internet for web access, it runs stand-alone applications like LibreOffice (an Open Source alternative to Microsoft Office products), Mathematica (a software equations solver), as well as a host of games (yes, even Minecraft.) The board itself doesn't appear to be much of anything resembling a computer - just a collection of integrated circuits and USB ports, with an Ethernet jack and HDMI (High Definition Multimedia Interface) ports. But don't let appearances fool you, this board can do virtually anything your Smartphone can do (with the right add-ons), and quite a few things that your Smartphone can't.

We will be using an extremely tiny portion of the Raspberry Pi's capabilities for this project. However, what it provides in the process is a fantastic platform to learn about rapid prototyping, computer communications, Linux (Unix computing), Python programming language, and the Internet of Things (IoT).

At the completion of the build, the control module will be used to control a modified version of the underwater vehicle SeaPerch [2]. Each team will compete against each other in timed trials to complete

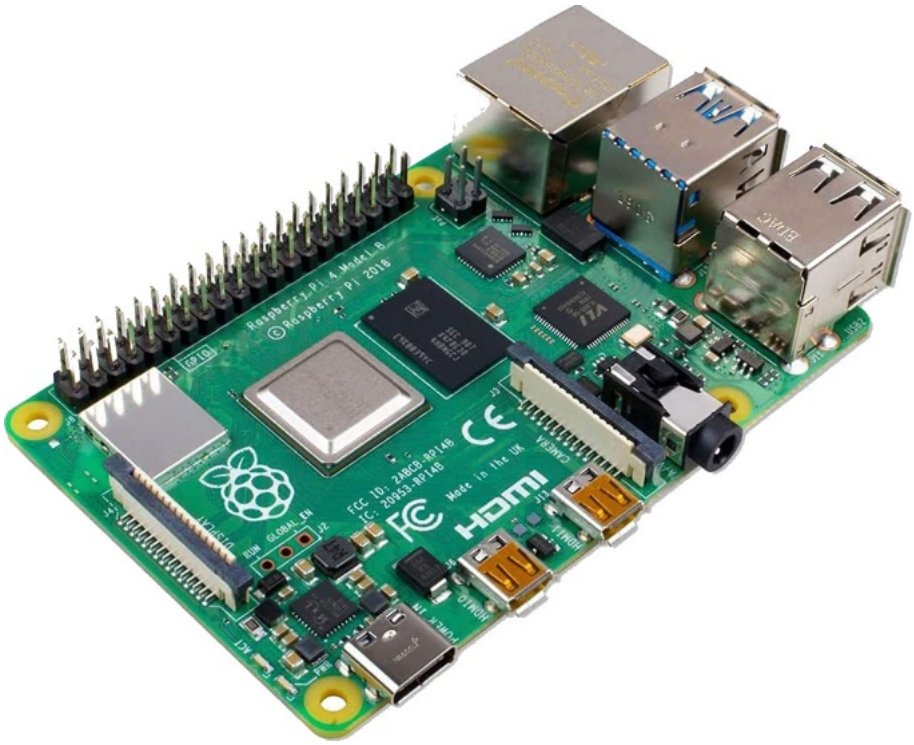


Figure 1: Raspberry Pi, rev. 4.

a series of mission scenarios. The team with the best score will win that portion of the competition. However, you will also be judged on your ability to keep an engineering notebook and write a final comprehensive report. The team with the overall highest score from all three phases will win the competition.

2 The Raspberry Pi

The RPi is a computer - connect it to a monitor and add a mouse and keyboard and it will be indistinguishable from any other computer you have ever used. It has a desktop and folders, drop down menus, and indicators to tell you it's connected to wireless protocols (Wi-Fi, Bluetooth). To reduce the number of peripherals (monitors, keyboards, mice, etc.) needed to host multiple RPis during this apprenticeship, each RPi has been assigned a static IP (Internet Protocol) address so you can communicate with it using your group's assigned laptop. (You can even communicate with it using your Smartphone, but let's hold off on that for now.) We will be using what is known as a Secure Shell (SSH) interface, which is a protocol that one computer uses to talk to another computer. Google's Chrome Browser has an extension (integrated application) cryptically called Secure Shell Beta which we will use to interface with the RPi.

Using the shell, you have only a command line to navigate all the files and folders on your RPi. It will likely take a little time for you to become comfortable navigating a computer using nothing more than a keyboard, but eventually, you'll be surprised how efficient it really is. Remember, this is how ALL computing was done in the pre-windows (both small 'w' and big 'W') days. However, the RPi does have a desktop GUI and for the savvy student, a window can be created that will host the RPi's graphic desktop on a remotely connected computer (i.e. Virtual Desktop).

The Raspberry Pi Foundation, the "company" responsible for the RPi, started producing these single-board computers in 2012. They initially thought they'd be lucky to sell 50 thousand. Ten short years later, they have sold over 38 Million RPis. Check out `https://www.raspberrypi.org` for more information.

2.1 Connecting to the Pi

As mentioned, each RPi has been assigned its own individual IP address. This implies that your RPi has already been provided a microSD card with a pre-loaded operating system - which it has. Until recently, the preferred OS distribution for RPi was Raspbian Buster (Raspbian is a portmanteau of Raspberry and Debian, a popular Linux distribution. The "flavor" Buster is named after Buster from Toy Story . Prior to the Buster distro (short for distribution), there was Stretch, and before that there was Jessie, and before that there was Wheezy, and before that was Squeeze, and before that was Lenny, and before that was Etch ... you get the point.) The entire operating system is stored on a microSD card. If you remove the microSD card, the RPi will stop functioning and it will need to be rebooted with the card reinserted.

To coincide with the release of the newest Raspberry Pi hardware (Raspberry Pi 4), the operating system is now simply referred to as Raspberry Pi OS. The Toy Story lineage is dead - long live the Toy Story lineage!!

Like all modern computers, the RPi has a Graphical User Interface (GUI) desktop, just like the one you're using now. However, because the RPi runs a Linux operating system, it can be run perfectly well from the command line. What is a command line, you ask? Before the mouse and GUI environments (aka window (small 'w') environments) there was the command line - a directory based access point to the files and programs on a computer. Windows (big 'W') has a command line interface, accessible from the "Search" window as "cmd" (figure 2). In Linux, the command line isn't just a feature, it's a way of life[1]. Because your RPis have been supplied with an operating system (it's on the removable microSD card), and because each RPi has it's own unique IP address, you can connect to and control it using the previously mentioned SSH. Insert the microSD card in the slot under the RPi (figure 3).

[1]Without going into much detail, Linux as an operating system, is a clone of the Unix operating system, developed in the 1970s at the former AT&T Bell Labs. Linux was originally crafted by software engineer Linus Torvalds - the resulting name Linux is a mash of Linus and Unix.

Make sure your RPi has power by inserting the USB C side of the cable into the RPi and the other end into your laptop - just like charging your phone.

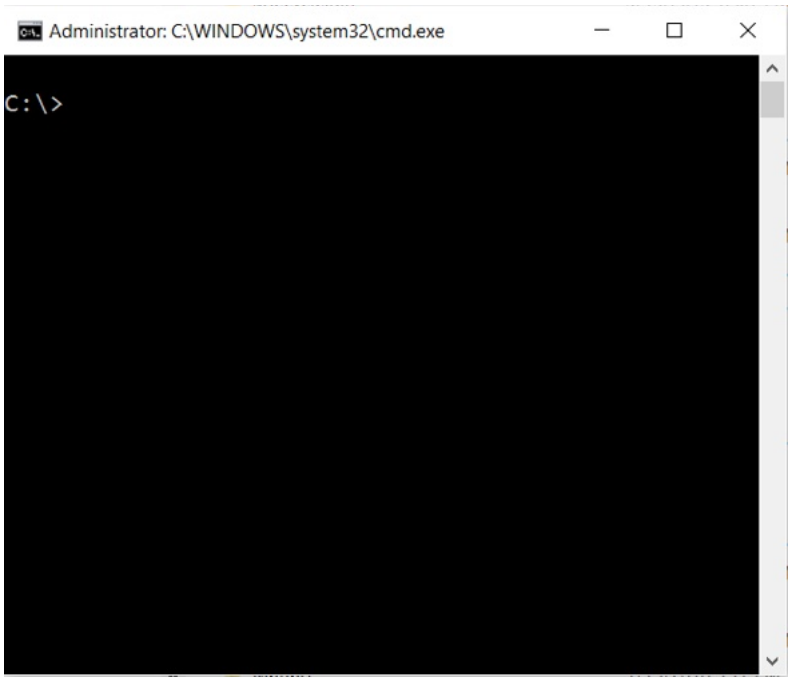


Figure 2: A command window in Windows

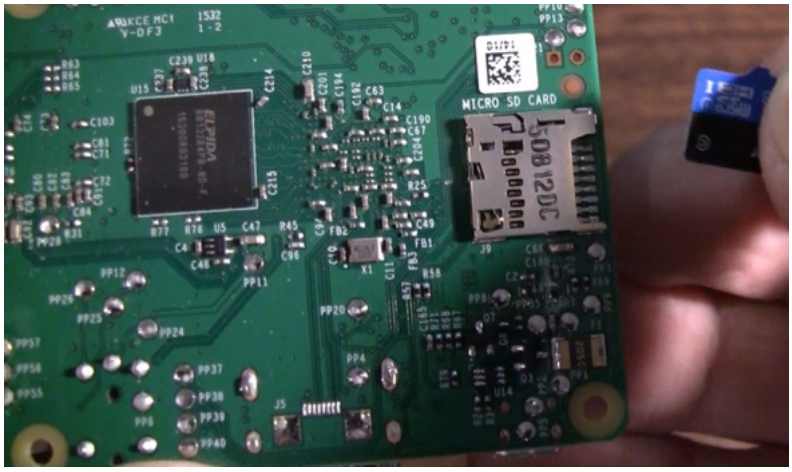


Figure 3: Insert the microSD into the RPi (from `https://arachnoid.com/raspberry_pi/`)

On your laptop, open the Google Chrome browser, click the Extensions icon (figure 4, left) to the right of the address bar and launch the Secure Shell application (figure 4, right). If all goes well, it should

look like figure 5. Where you see the left red arrow, you will enter `pi@192.168.1.XXX`, where `XXX` will be your particular address. For instance, figure 5 shows `pi@192.168.1.151`. For the right red arrow (port), enter the number 22. Port 22 is the standard SSH TCP (Transmission Control Port) port. Other well known ports are 80 (http), 443 (https) and 5900 (Virtual Network Connection), to name but a few.

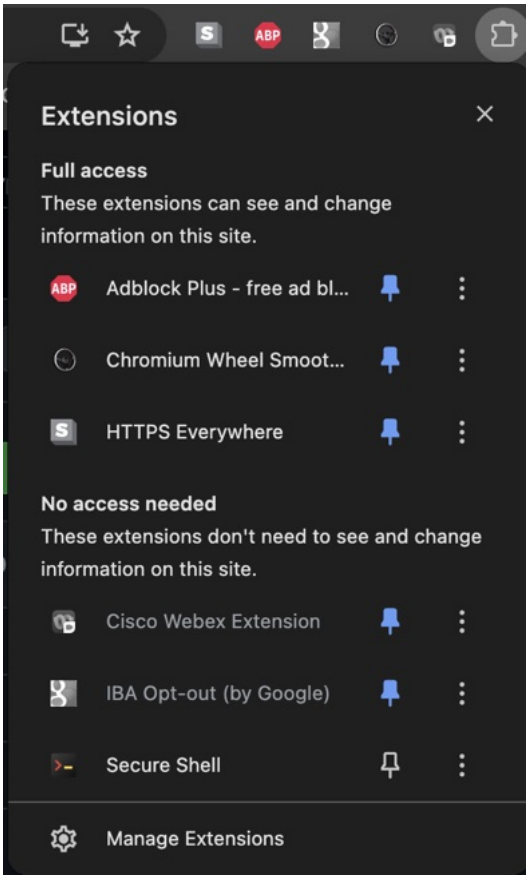


Figure 4: Access the Extensions link and select Secure Shell.

The example in figure 6 has been populated with the address `pi@192.168.1.211`. YOURS WILL BE DIFFERENT! When the address and port have been entered, simply hit ENTER at the bottom of the screen and you should see the screen in figure 7. Allow the connection by typing out `yes`. You'll be prompted for the password, which we changed from the default `raspberry` to `cranberry` - case sensitive. The connection will be authenticated and you will see the screen depicted in figure 9.

A word of warning, i.e. pay attention to this! The default username for all Raspberry Pis is `pi` and the default password is `raspberry`. We are using the local area net-

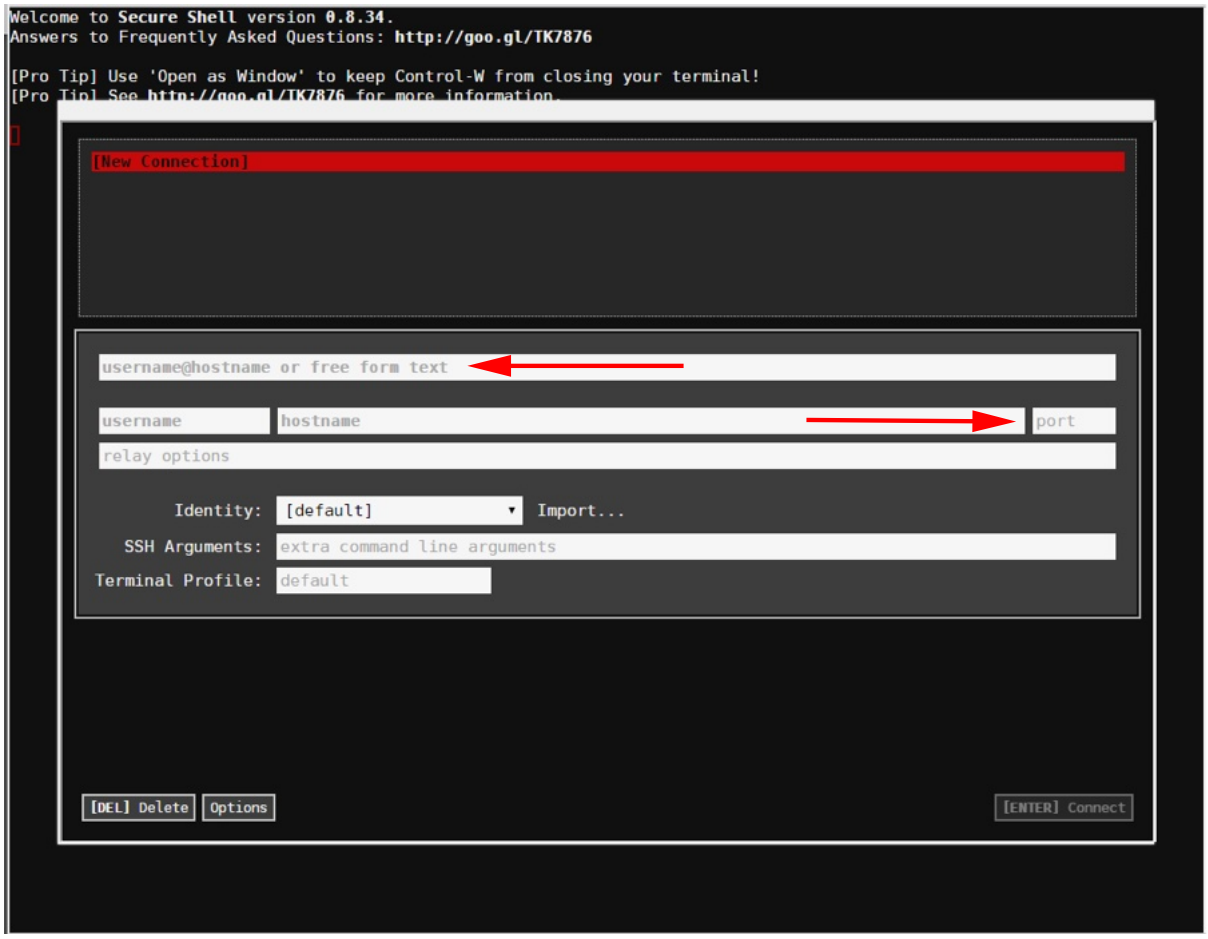


Figure 5: This screen will launch when you select the Secure Shell App.
ask a

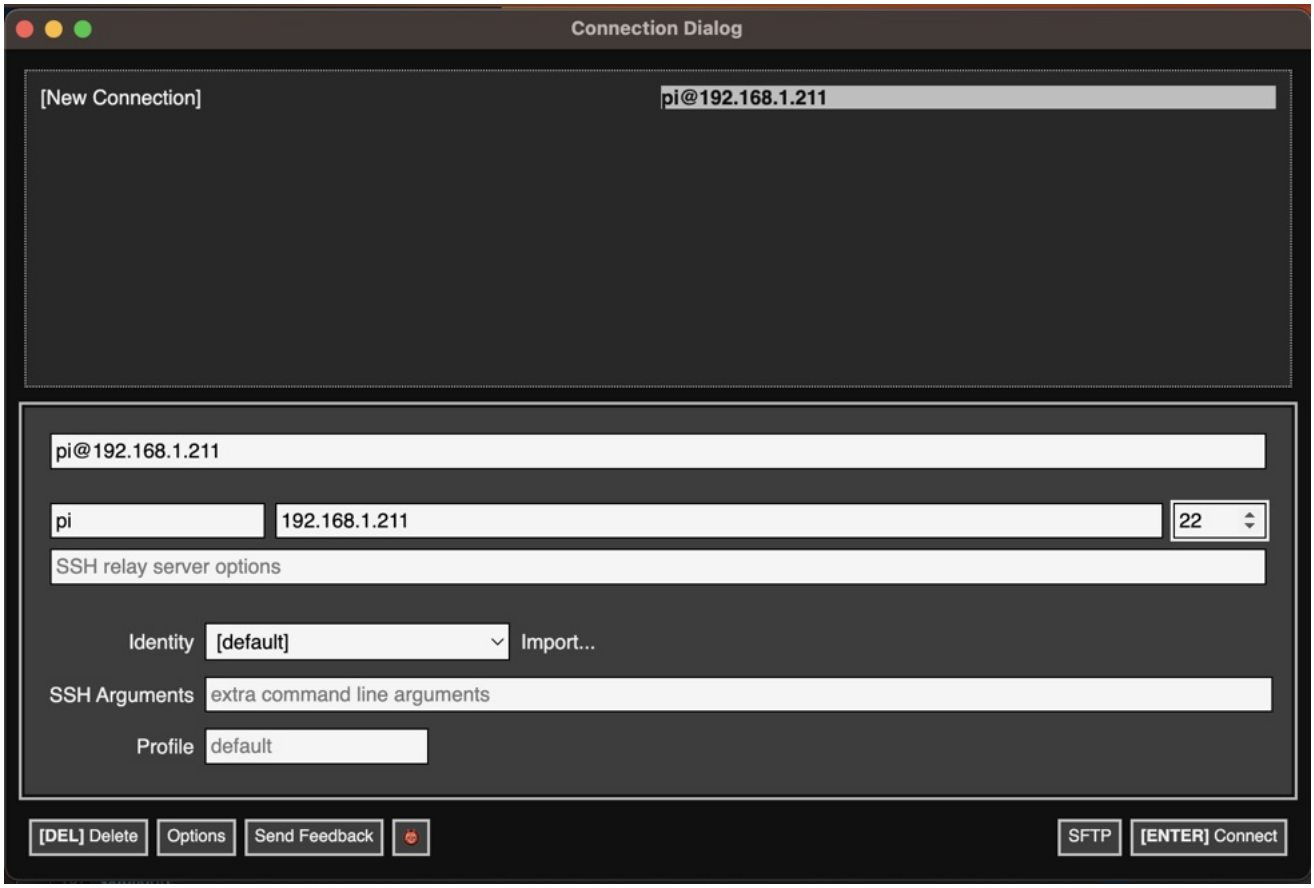


Figure 6: Populated SSH screen. YOURS WILL BE DIFFERENT!

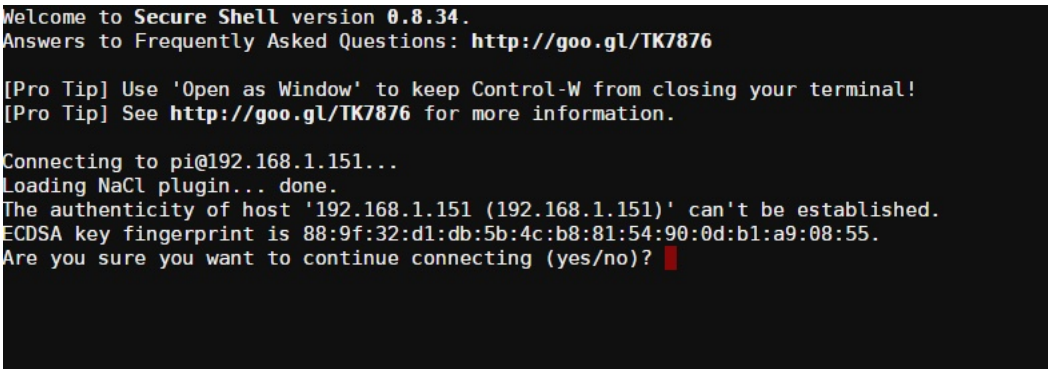


Figure 7: This screen will display if you have entered the information for your RPi correctly - do not be alarmed.

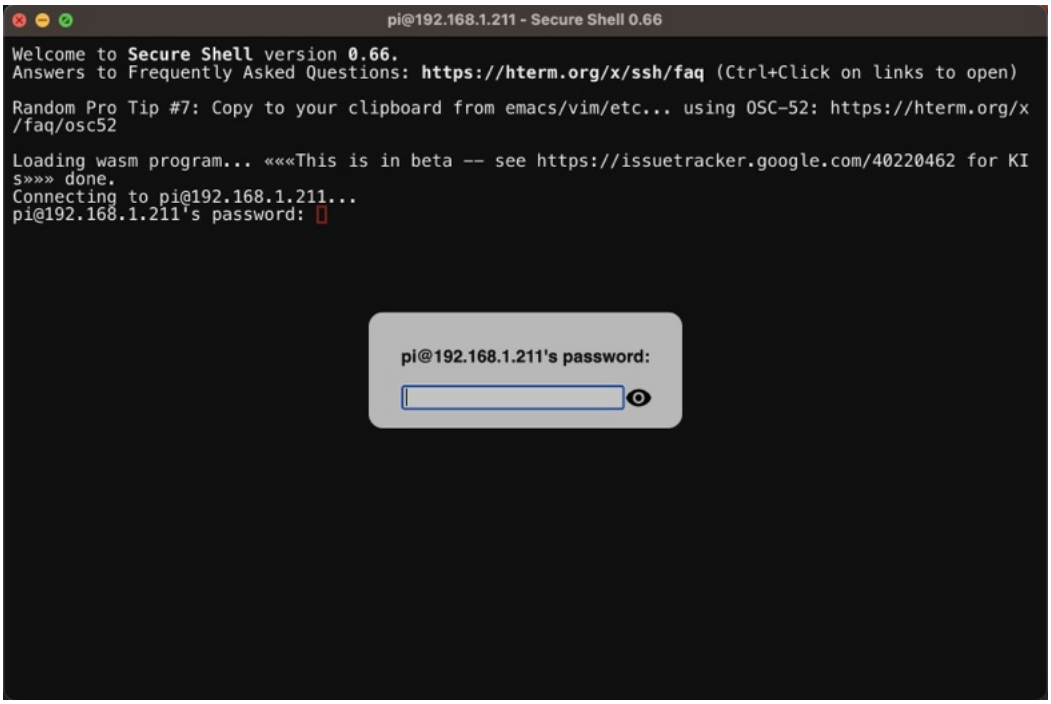


Figure 8: Enter your password

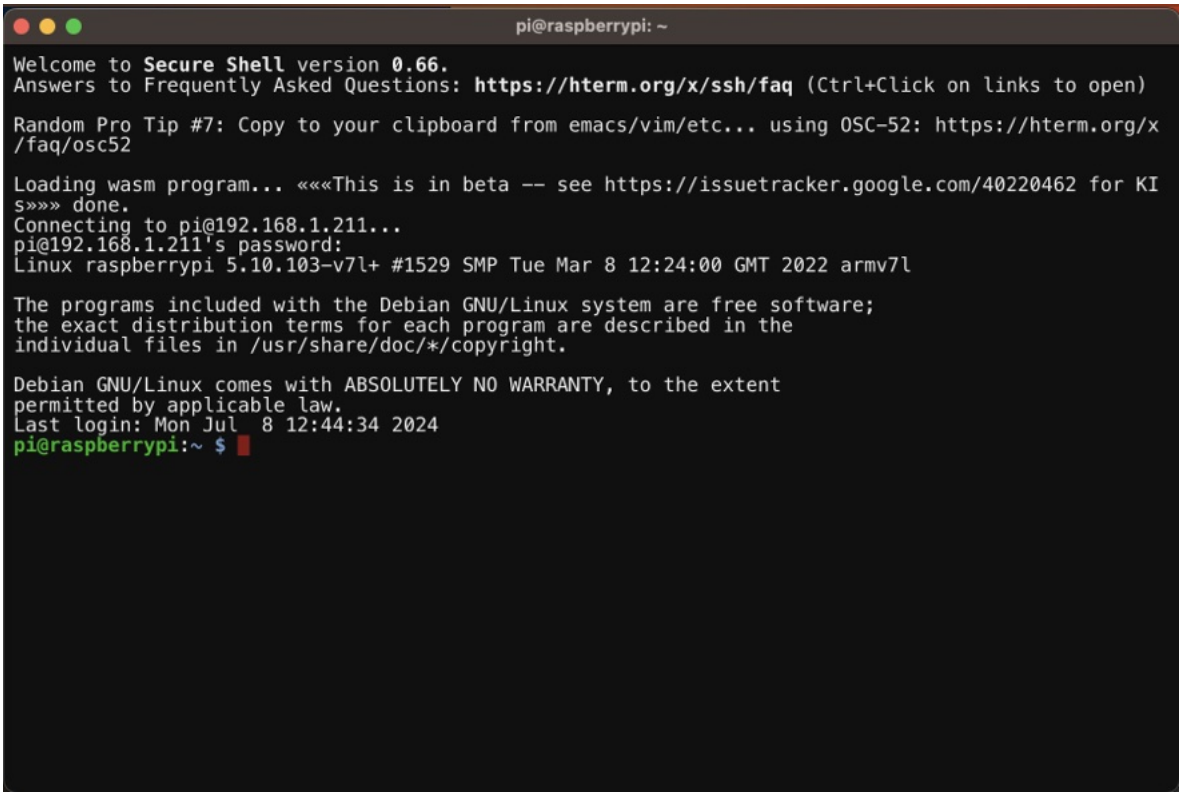


Figure 9: Success! You have just connected your computer to the RPi over the internet!

work (LAN) in the Undersea Collaboration and Technical Outreach Center (UCTOC) to "remotely" (wirelessly) connect to the RPis from a laptop. All one needs to remotely control any of the RPis being used in the UCTOC is the username, password, and IP address. There are 256 possible IP addresses on the .XXX subnetwork and 2 of them are 192.168.1.0 (the gateway to the outside world) and 192.168.1.1 (the address of the router providing the other 254 addresses). The remaining 254 are a combination of dynamic addresses (the router chooses) and static (the user/machine chooses). There are a number of IP address sniffing tools that can quickly locate what is connected to a LAN. The point is, it may be a good idea to change your password. This can easily be accomplished once you log in to RPi. Simply type `passwd` and enter your current password. It will then prompt you for a new password and a repeat of your new password (figure 10). This way, no other teams (or mentors) can sabotage your work ... or at least it will be a lot harder for them to sabotage your work.

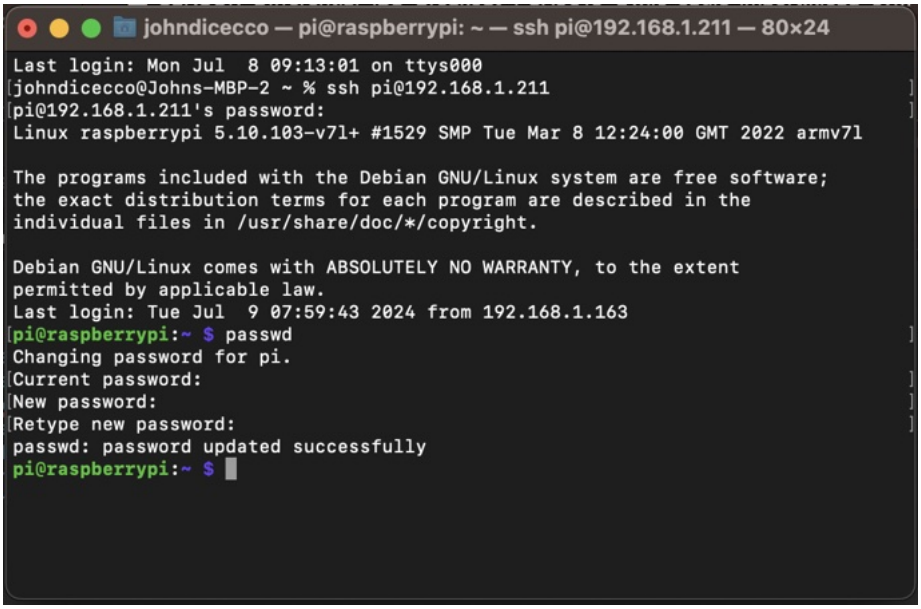


Figure 10: Changing the password on the RPi.

The UTAP course is 3 weeks of very intensive engineering experience. It was designed to give you a realistic glimpse into one small aspect of the type of work engineers do everyday. Hopefully, you'll leave this apprenticeship with a better understanding of the ingenuity it takes to be an engineer (yes, both words have the same root, from the Latin *ingeniare* - to devise). The Raspberry Pi was chosen for this internship due to its ease of use, powerful computing capabilities and vast user support as well as relatively low cost. There are many online resources for projects that can be run on the Raspberry Pi, the primary being the Raspberry Pi Foundation website provided earlier (`https://www.raspberrypi.org/`). Here you can find downloads for the many different operating systems that can currently run on the RPi as well as links to many affiliates and an extensive network of forums and collaboration groups.

3 Preloaded Software

As mentioned, the RPis that have been assigned to the teams have already been loaded with a copy of Raspian Jessie, a Linux distribution based on Debian, which is itself a Linux distribution, based on the Linux kernel, or operating system. While that may sound a bit convoluted, consider this - Windows 95, Windows 98, Windows ME, Windows NT, Windows 2000, Windows XP, Windows Vista, Windows 7, Windows 8, Windows 8.1, Windows 10, Windows 11 ... Windows 392547 one day maybe.

In addition to the operating system, other software and libraries were installed to speed the development of the software we will use to control the Remotely Operated Vehicle (ROV). YOU DO NOT HAVE TO DOWNLOAD ANY SOFTWARE - IT HAS ALREADY BEEN DOWNLOADED AND INSTALLED ON YOUR RASPBERRY PI. The following steps WERE taken:

- Download and install Raspberry Pi OS
- `sudo apt-get update && sudo apt-get upgrade` (updates the distribution and other packages installed on the Pi)
- Modify Internet/Router communication files to setup network for static IP
  - `/etc/network/interfaces`
  - `/etc/wpa_supplicant/wpa_supplicant.conf`
  - `/etc/dhcpcd.conf`
- `sudo apt-get install joystick`
- `sudo apt-get install python-dev python-rpi.gpio` (python-rpi.gpio is part of the OS distribution)
- `sudo apt-get install python-smbus`(System Management Bus needed to communicate over I2C, also included with the OS download)
- uncomment `/boot/config.txt dtparams` (for hardware i2c/spi configuration)
- `sudo pip3 install --upgrade adafruit-python-shell`
  `wget https://raw.githubusercontent.com/adafruit/Raspberry-Pi-Installer-Scripts/r`
  `sudo python3 raspi-blinka.py` (the baseline libraries from Adafruit that all the other sensor software is based upon)
- `spip3 install adafruit-circuitpython-pca9685` (this installs the Adafruit Python code we need for the PWM driver)
- `sudo pip3 install adafruit-circuitpython-servokit` (this installs the OTHER Adafruit Python code we need for the PWM driver)
- `pip3 install adafruit-circuitpython-ssd1306` (this installs the Adafruit Python code we need for the OLED screen)
- `sudo pip3 install adafruit-circuitpython-bme280` (this installs the Adafruit Python code we need for the temperature, humidity and pressure sensor)
- `sudo pip3 install adafruit-circuitpython-fxos8700`

`sudo pip3 install adafruit-circuitpython-fxas21002c` (This installs the software to use the NXP 9 Degree of Freedom (DoF) inertial navigation system (INS ... also known as IMU - inertial measurement unit)

4 The "Sensors"

We will be using two sensors with our UTAP project - a BME280 Temperature, Humidity, and Pressure (THP) sensor, and the NXP 9 DoF IMU, from which we'll use the FXOS8700 magnetometer and the FXAS21002C gyroscope (Fig. 11). Each of these sensors communicates with the Raspberry Pi using a serial communication protocol called I2C (or I^2C, or IIC, for Inter-Integrated Circuit). We'll be discussing I2C in the lecture portion of the program. For now, it's enough to know that it's just a way for one integrated circuit to talk to another integrated circuit.

The other two "sensors" aren't really sensors at all but function as peripheral devices to the Raspberry Pi. One is an Organic LED (OLED) screen for heads-up display and the other is a Pulse Width Modulation (PWM) Board that extends the Raspberry Pi's ability to produce multiple PWM signals. (There is a section later (Sec. 6) in the document that explains exactly what this means for motor control.)

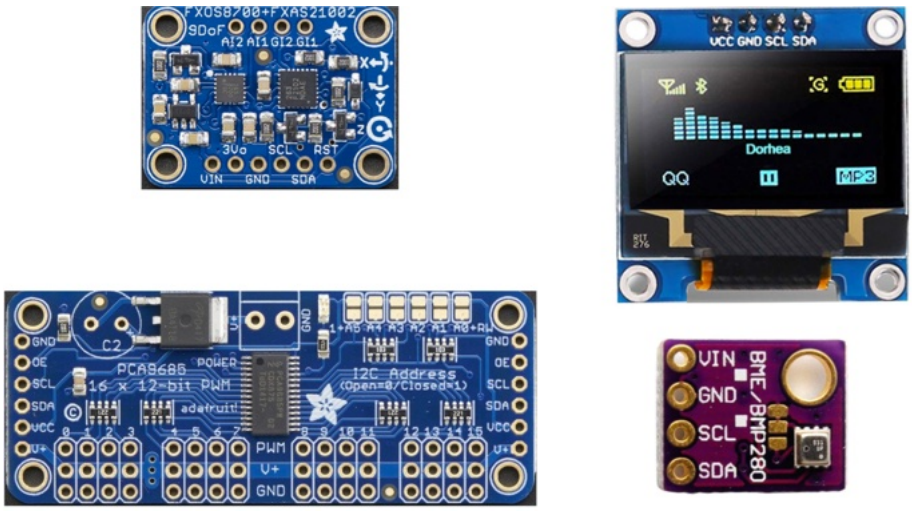


Figure 11: Clockwise from upper left, the IMU, the OLED screen, the BME280, and the PWM board (not really a sensor, but does use the same communication protocol (I2C) as the sensors.).

A quick blurb about the IMU. Since this sensor will be mounted to your ROV to give you attitude (yes, there is another definition of attitude) information, it is positioned a very long distance away from the RPi which reads the data. Communications over this length of cable (over 35 feet) is challenging

as the signal loses integrity - think about the static as you get further away from your favorite radio station (wait, do people still listen to the radio?). This is overcome by using a protocol that provides a differential pair of wires for the data and a differential pair of wires for the synchronizing clock. The noise on these pair of wires will be identical which means taking the differential will yield only the underlying signal. This is the way these long distances are achieved in aircraft, trains and automobiles as the sensor information must be relatively noise free in order to get an accurate value. In order to establish this differential I2C, a special chip (PCA9615) is installed on your topside PCB as well as a tiny PCB inside the housing that holds the IMU. The I2C communications are split into a differential pair at the topside PCB and recombined at the sensor PCB - we'll talk more about this during the I2C lecture when we see a demonstration of what happens when you try to communicate over I2C WITHOUT this type of technique.

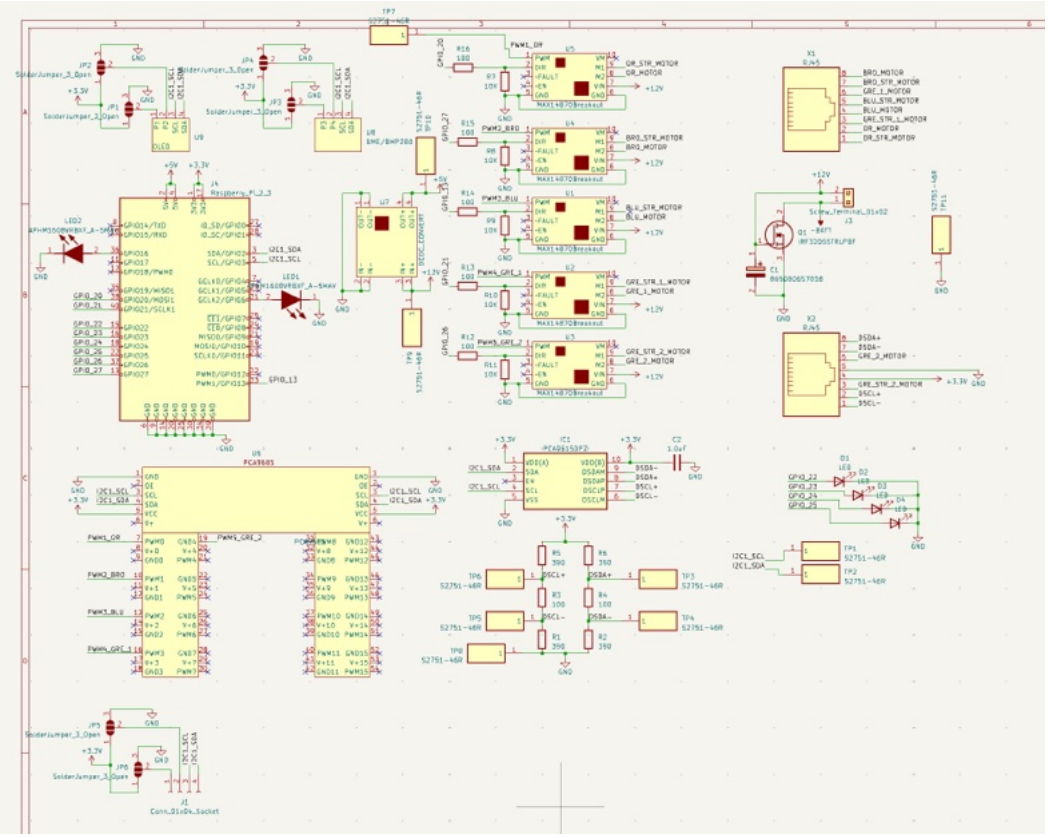
5 The Circuit

The goal of this project is to control UP TO 5 individual motors on an ROV with a USB gamepad controller (you may find you need less than 5). There is no ready-made circuit or software that will accommodate this task while simultaneously maintaining the SeaPerch standard output from an RJ45 Ethernet jack. Therefore, we needed to design and build our own printed circuit board (figures 12(a) and 12(b)) as well as develop our own software. And remember that little blurb about the IMU? We had to design a separate board to handle the differential signals (Figures 13(a) and 13(b)).

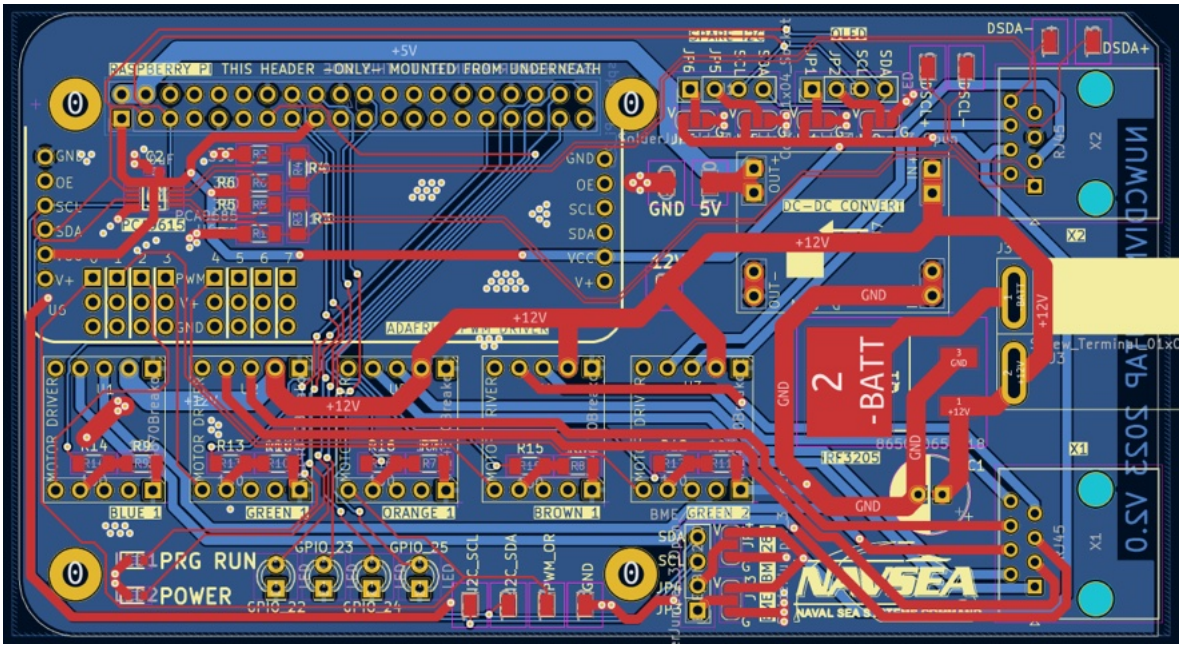
Since we're using the RPi, it seemed rather obvious to design a circuit board that would mate to the 40 General Purpose Input/Output (GPIO) pins on the RPi (see figure 18). This allows each of the pins to be routed to other devices on the PCB (the PWM driver and the motor drivers) to control the power to the motors. On this newly designed board, header pins (essentially sockets) are mounted that will then accept the motor drivers and pulse width modulation (PWM) board as well as the sensors that we will use to display information about the environment.

5.1 Pinout

The Raspberry Pi has a somewhat confusing pinout structure. The header pins have several different numbering systems, depending on how a user intends to access them. For instance, the pins can be numbered in an intuitive way from the top row such that the first pin is 1 and the one next to it is 2. The next row is 3 and 4 and so on. Makes perfect sense, right? However, to use the pins as General Purpose Input Output (GPIO), the user needs to access the Broadcom BCM2837 system-on-chip (SoC) processor, which includes the four high-performance ARM Cortex cores. This chip handles all the input/output for the RPi. Still makes sense, right? Well, the pin numbers on the Broadcom chip DO NOT correspond to the actual pin numbers of the header pin. Referring to figure 18, it's clear that care must be taken when defining a pin to use. For the UTAP project, the Broadcom naming convention is used, which is the *NAME* column in figure 18, NOT the *Pin#*. Notice, for instance, that GPIO19 isn't

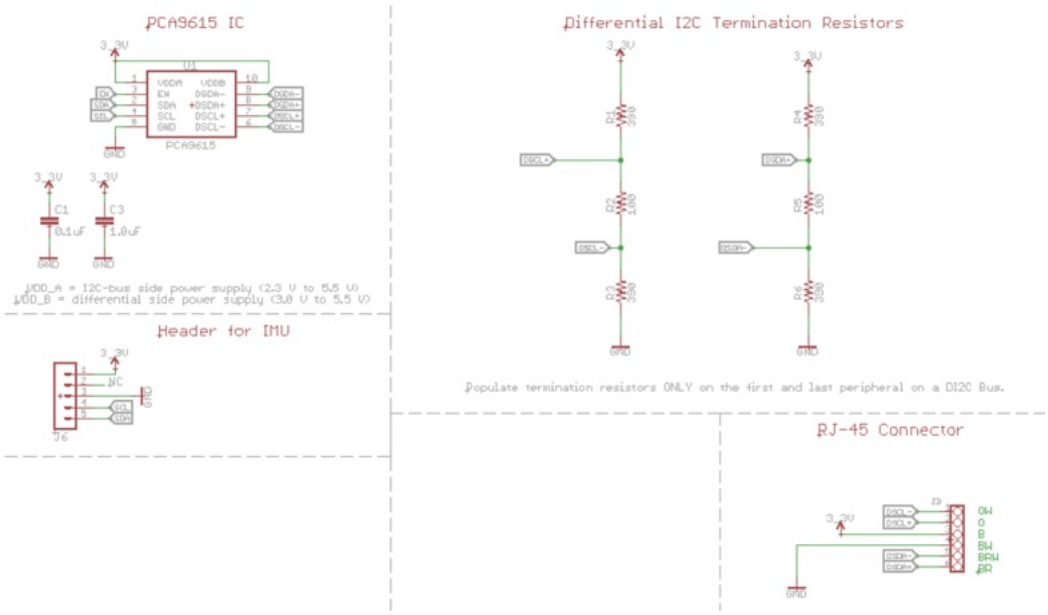


(a) Circuit diagram (schematic) for the daughterboard.

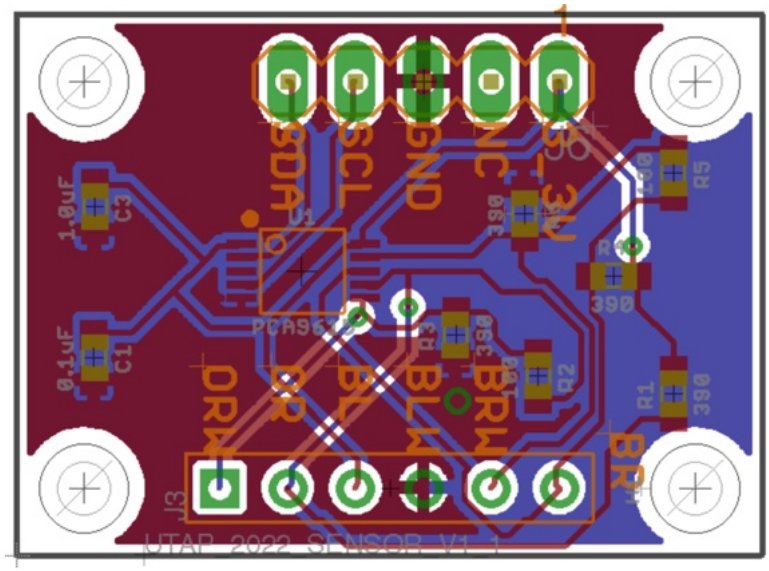


(b) The PCB that the schematic produces.

Figure 12: Daugterboard schematic and printed circuit board.



(a) Circuit diagram (schematic) for the daughterboard.



(b) The PCB that the schematic produces.

Figure 13: Daugterboard schematic and printed circuit board.

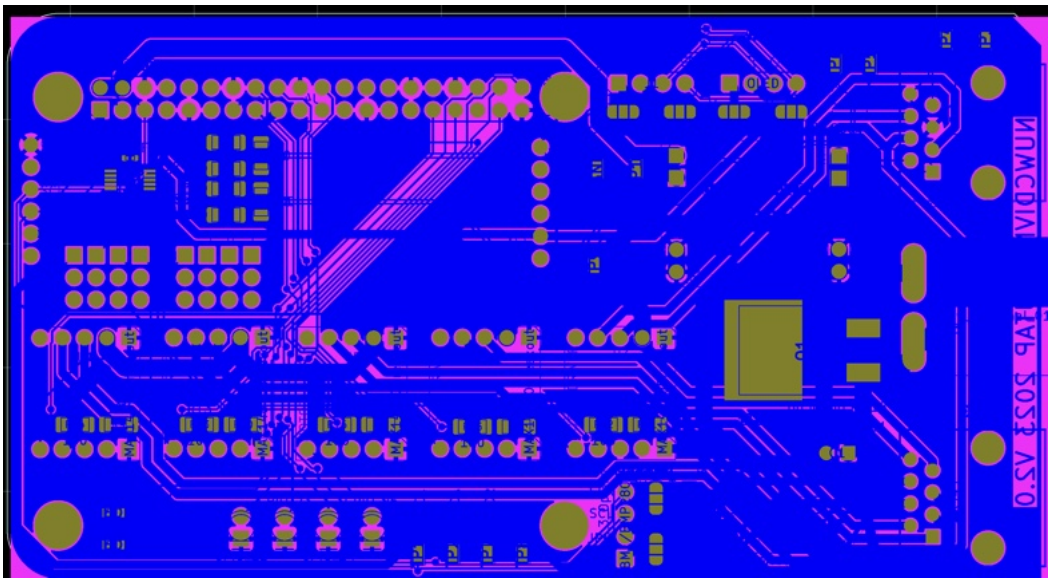


Figure 14: Superimposed layers of the PCB sent to the manufacturer for fabrication.

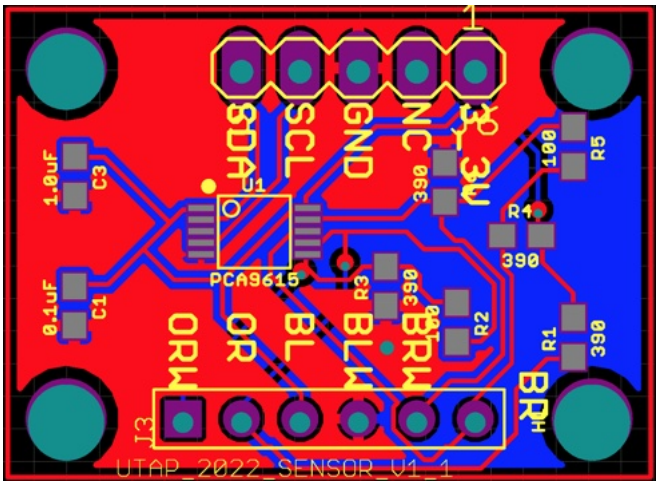
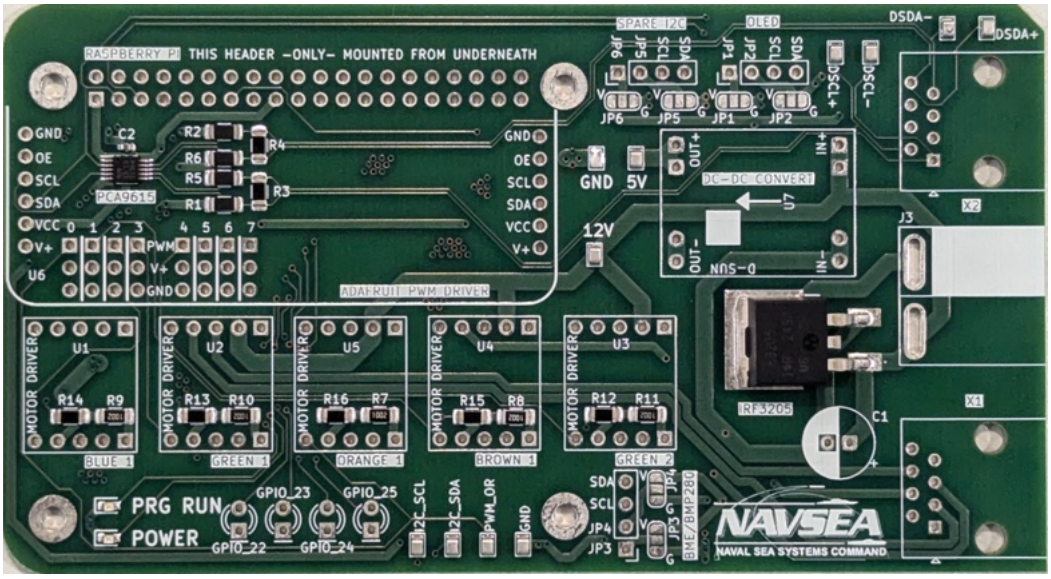
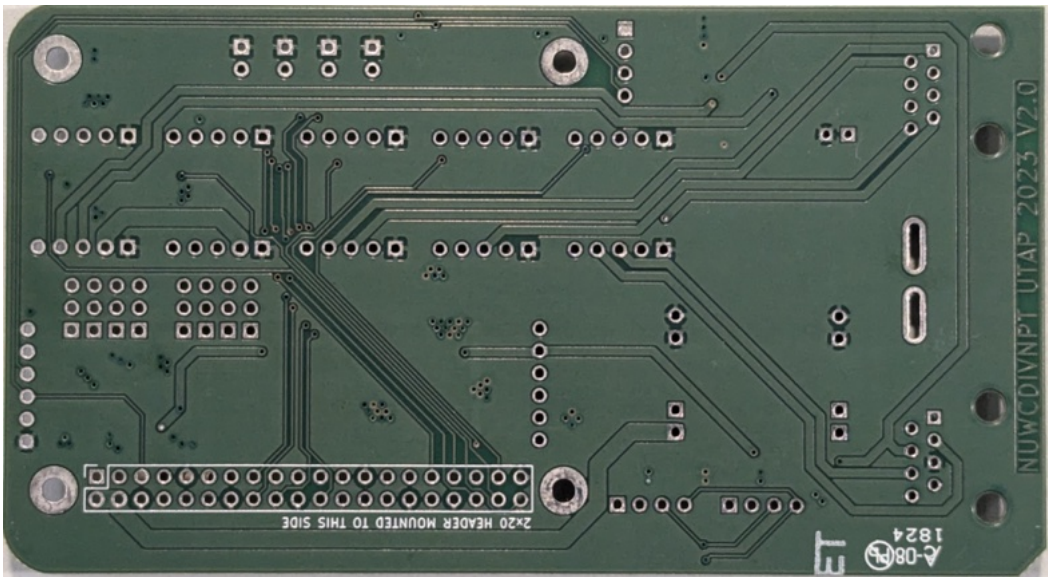


Figure 15: Superimposed layers of the PCB sent to the manufacturer for fabrication.

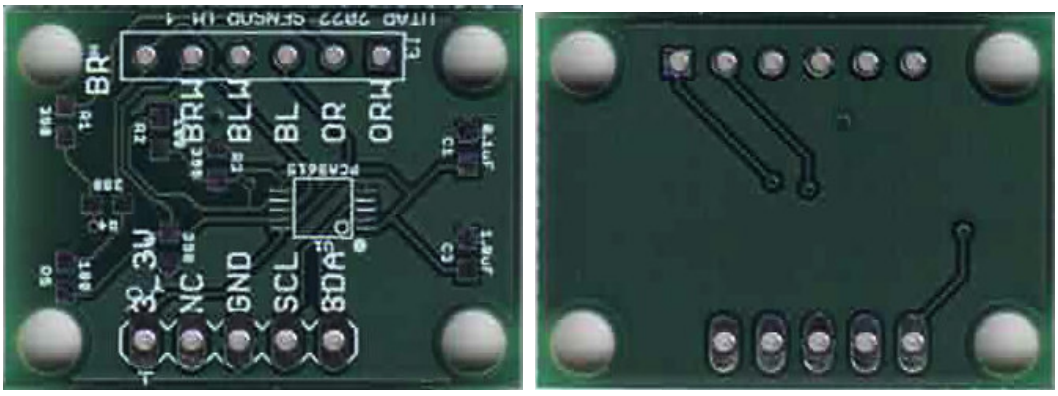


(a) Front side of the PCB.



(b) Back side of the PCB.

Figure 16: UTAP Printed Circuit Board (PCB).



(a) Front side of the sensor PCB. (b) Back side of the PCB.

Figure 17: UTAP Sensor Printed Circuit Board (PCB).

on pin#19 but rather on pin#35. Overall, this has limited effect on our build, since all the connections are already hardwired into the printed circuit board. However, should you choose to continue working with the RPi after the apprenticeship, it is something to keep in mind.

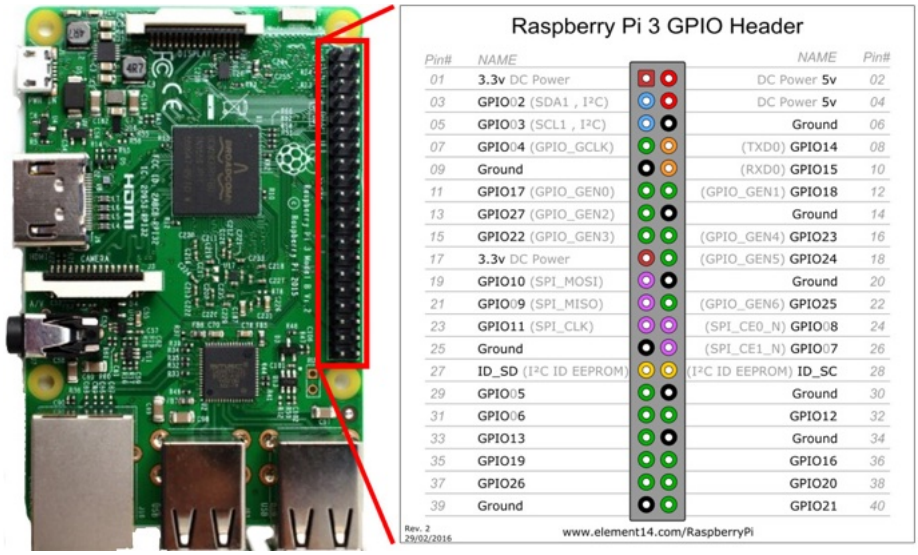


Figure 18: This pinout is valid for RPi versions 2+ through 5.

6 Adafruit PWM Driver

For all the power the RPi has as a computer, the GPIO pins lack several functions common on most microcontrollers. This isn't really surprising - after all, the RPi is a computer not a microcontroller. However, because it's a computer, it can overcome most, if not all, of these shortcomings. The most immediate limitation for our project is that the RPi has only 1 hardware PWM pin. Since each motor needs to be speed controlled, and there are 8 motors, it would be really helpful if the RPi had 8 PWM

pins.

Pulse Width Modulation (PWM) is a technique for delivering variable power to devices that require a certain amount of voltage (say, 12 v) to activate. For instance, to profvide half power to a 12V motor that only turns on at 12V, supplying 6V won't work - the motor wouldn't even spin, let alone spin at half power. What is done instead is to provide 12V and 0V in equal proportions (pulses), effectively providing 12V only half the time. This is called a 50% duty cycle and it is accomplished by modulating (changing) the width of the 12V pulse. For full power, the pulse is infinitely long, i.e. it stays at 12V. The PWM can be tied to the joystick to match the PWM width to the amount of movement in the joystick, hence joystick controlled variable speed motors.

The RPi does have hardware PWM available. As previously mentioned, the problem is there is only one *hardware* PWM pin. This means the pin is configured with hardware (transistors) to provide a variable pulse width through the pin. But it's also perfectly reasonable to make pulses using software, by setting a pin high (5V or 3.3V for instance) for a period of time, then setting it low (0V) for a period of time. While this is a reasonable solution (in fact it's done all the time), it's not ideal - there are delays and interrupts involved and it makes the code more complex than it is already. And since the RPi is a computer, it is often busy manging its own internal timing requirements.

Problem: we need as many as 8 PWM signals (for potentially 8 motors) but we have only one PWM pin on the RPi. Solution: Adafruit's PWM Driver Board, which can control 16 PWM signals over a data transfer protocol called I2C[2]. The RPi supports I2C and the PWM driver supports I2C. We simply program the RPi to read the USB gamepad joystick and send that information to the PWM driver, which will tell it how to generate the 8 PWM signals we need. Each PWM signal has its own address (conceptually no different than an Internet address) so we can send specific commands to specific PWM addresses to control them individually. Mind blown? Don't worry, we'll go over this during one of our instruction sessions.

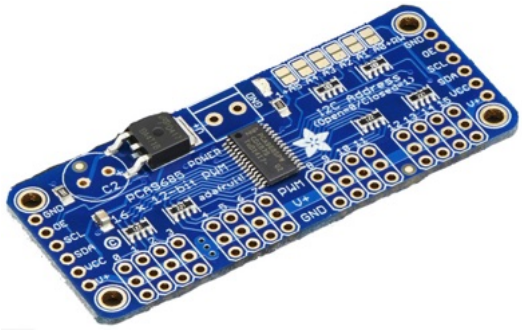


Figure 19: The Adafruit 16 channel PWM Driver Board. We will only be using 8 channels.

[2]Inter Integrated Circuit communication protocol was developed by Philips in 1982. It is an easy way to communicate information between 2 different devices on the same circuit or communication bus.

7 Motor Drivers

The motor drivers are responsible for controlling the amount of power delivered to the motors, as well as which direction they spin. The MAX14870 (Maxim Integrated Co.) accepts a number of different inputs but our application only uses four: the PWM input, the direction input and power and ground for the motor (figure 20). The PWM comes from the Adafruit PWM driver, while the direction control comes from one of the pins (GPIO) on the RPi. When the joystick is moved from the dead zone[3], a value proportional to the displacement of the joystick is generated. Since the PWM is has 12 bits of resolution (meaning it has 4096 (2^{12}) levels from off to on), numbers from 0 to 4095 represent the levels from fully off (0) to fully on (4095). For instance, if the gamepad joystick is moved halfway between the center position and fully engaged, the value from the joystick will be 2048, and the pulse width proportion will be 50%, or half power. This pulse waveform goes into the MAX14870 where the H-bridge[4] pulses 'on' half the time and 'off' half the time, creating the effect of half power. You most likely performed this exact same technique when driving your SeaPerch for the first time, essentially toggling the switches back and forth (pulsing) to change the speed of the vehicle to gain a bit more fine control.

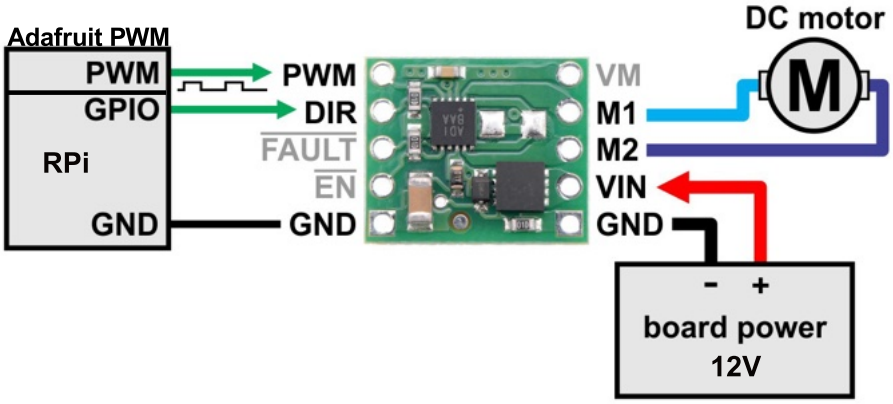


Figure 20: The Motor Driver board and PWM interface diagram. From `https://www.pololu.com/product/2961`

It is instructive to note that this technique for driving a motor via PWM is necessary for motors that have a so-called turn-on voltage. As previously suggested, many motors need a minimum of 12V before they are even able to start spinning. Therefore, delivering 6V will not spin the motor at half speed. This is the critical distinction between the PWM method of varying power versus reducing the voltage value available to the motors.

IT IS CRITICAL THAT THE MOTOR DRIVER BE INSTALLED IN THE RIGHT

[3] A dead zone is an area around the center position of the joystick that insures an "OFF" state. It's necessary because a joystick will never be able to return to the EXACT center position. To insure the center position is truly off, an area around the center large enough to accommodate any slight shift from the center is established.

[4] This is the circuit that controls how power is delivered to the motor. It's called an H-bridge because of the shape of the transistor layout.

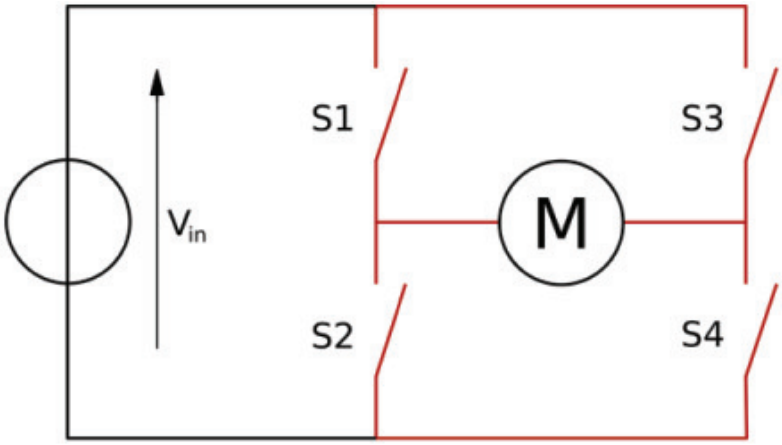
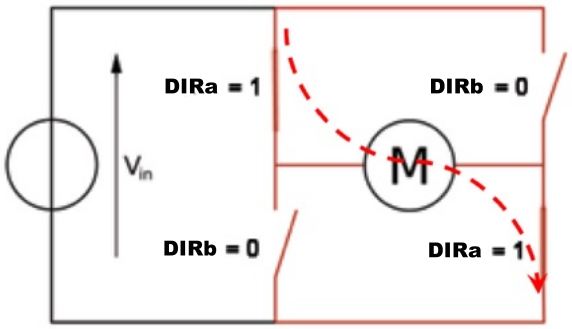
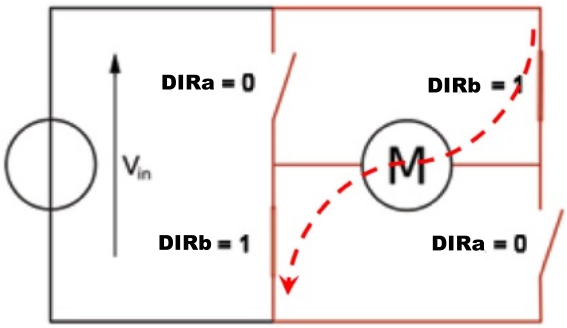


Figure 21: H-Bridge schematic. The center portion with the motor resembles an 'H'.



(a) When DIRa switches are set to 1, they close and the motor spins in the forward direction. The other switches (DIRb) receive the NOT ($1 \rightarrow 0$,$0 \rightarrow 1$) of DIRa.



(b) When DIRb switches are set to 1, they close and the motor spins in the reverse direction. Likewise, the other switches (DIRa) receive the NOT ($1 \rightarrow 0$,$0 \rightarrow 1$) of DIRb.

Figure 22: Note when two switches are closed, the other two MUST be open, otherwise the motor won't be able to spin at all.

ORIENTATION. IF IT IS INSTALLED UPSIDE DOWN, IT WILL BURN UP!!!!

When it's time to mount the motor driver onto the female header pins, refer to figure 37.

8 The Gamepad

The Logitech F310 USB gamepad looks like most modern gamepad controllers. It has all the components we need (quite a bit more than we need, actually) with a fairly straightforward interface. That is, the code to pull information from the gamepad (the joystick position, which buttons have been pressed, etc.) is readily available.

Figure 23: The Logitech F310 USB gamepad.

The code to read the gamepad for this project has been cobbled together from a number of different sources, the most principal of which is from (js_dev.py) available from `https://gist.github.com/rdb/8864666`. While this may seem like cheating, it's actually a pretty common way of writing code. There is no need to rewrite code that has already been written and made available for sharing under the concept of *Open Source*. You may need to modify it (slightly or heavily) but it's often more efficient to modify already existing code than it is to write code from scratch. However, it's always good form to reference the sources when you use someone else's work. It's the least you can do for the resulting workload reduction. Plus, since no one is getting compensated for the work they did to write the code, this is the only form recognition they receive.

9 Pin Mapping

The RPi GPIO pinout was discussed earlier. Figure 24, however, shows which GPIO pins control which components on the PCB. For instance, GPIO pin 16 turns on an LED (furthest to the right) when the program is running, and GPIO4 is the direction pin for the ORANGE wire connected to RJ1. Recall from the SeaPerch build that each Ethernet cable has 4 twisted pair of wires: one orange pair, one brown pair, one blue pair, and one green pair. This is how we will identify which motor to turn off and on.

*3.3V	5V
SDA	5V
SCL	GND
4	14/TX
GND	15/RX
17	18
27	GND
22	23
3.3V	24
SPI MOSI	GND
SPI MISO	25
SPI CLK	8/SPI CS0
GND	7/SPI/CS1
RESERVE	RESERVE
5	GND
6	12
13	GND
19	16
26	20
GND	21

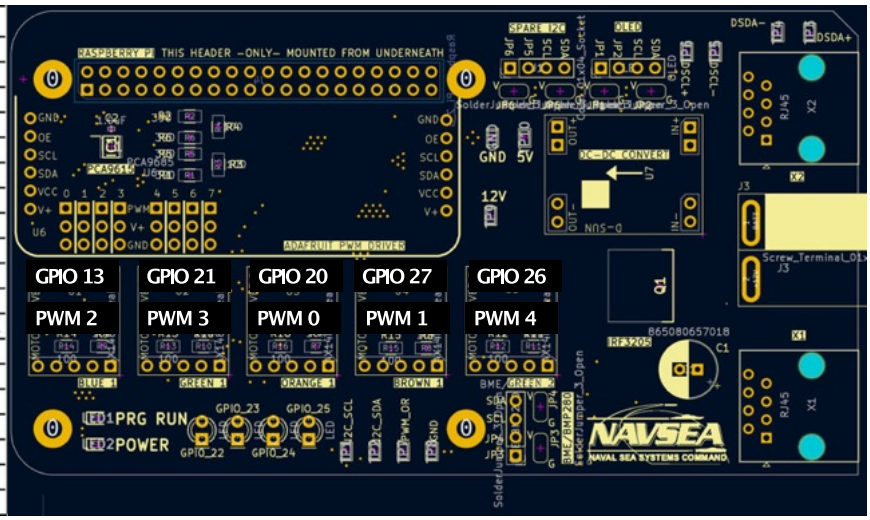


Figure 24: Mapping of the GPIO pins and PWM pins to the motor drivers. The asterisk marks the location of pin 1.

10 Assembling the Circuit

It turns out that the assembly of this circuit board is really pretty straightforward. For the most part, it's just a number of female header pins, some LEDs, a capacitor, a couple RJ45 jacks, and a power terminal. Because changes were made to accommodate the underwater inertial measurement unit, some of the board is already pre-populated (Fig. 10).

The principal reason these parts are pre-installed is due to the small chip that requires special mounting techniques (Fig. 26). This type of mounting is called Surface Mounting and any device mounted in this way is an SMD - Surface Mounted Device. And in order to make sure it was mounted correctly, the other components were added to verify that the chip would perform correctly. There are still a number of headers and parts that need to be installed, so you'll have plenty to solder.

The printed circuit boards have what is known as a silk screen (white printing on the board) to guide you to put the right part in the right place. **PAY ATTENTION TO THE TEXT!** In addition to text, the outline of the part is printed. Simply put the leads, or legs, of the part in the holes of the PCB. For some things like resistors, the orientation of the part is not important - it can't be put in backward. Other things, like electrolytic capacitors, as well as integrated circuits (chips) need to be put in the correct way because they have a polarity - meaning a 'plus' side and a 'minus' side. Think of it as putting batteries in a device with the plus and minus sides lined up with the pictures on the battery tray. Start by verifying that you have the parts in figure 10.

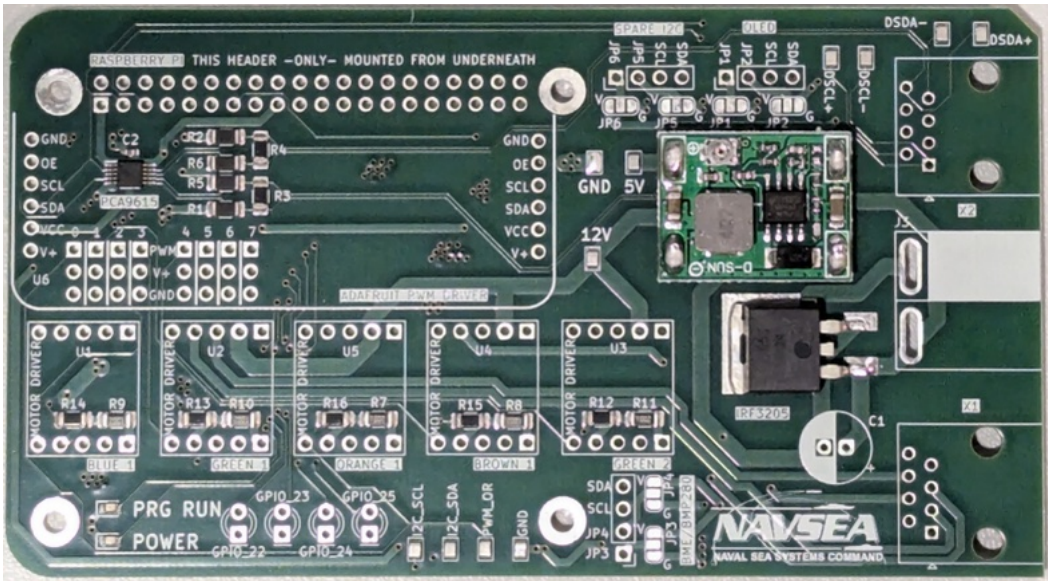


Figure 25: The board will look like this when you receive it.

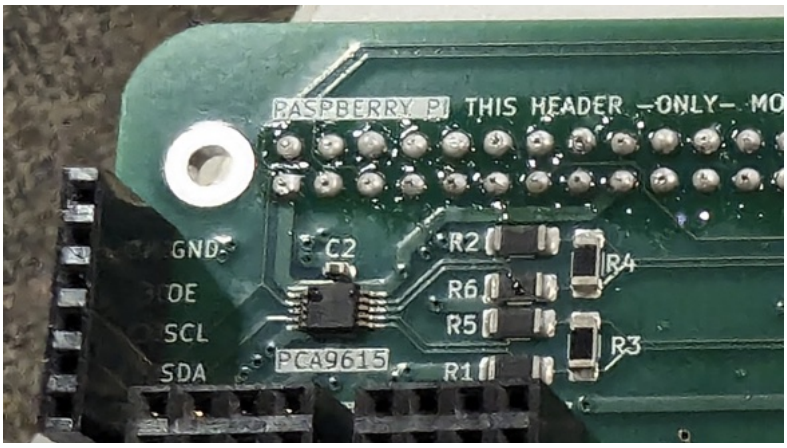


Figure 26: The PCA9615, differential I2C.

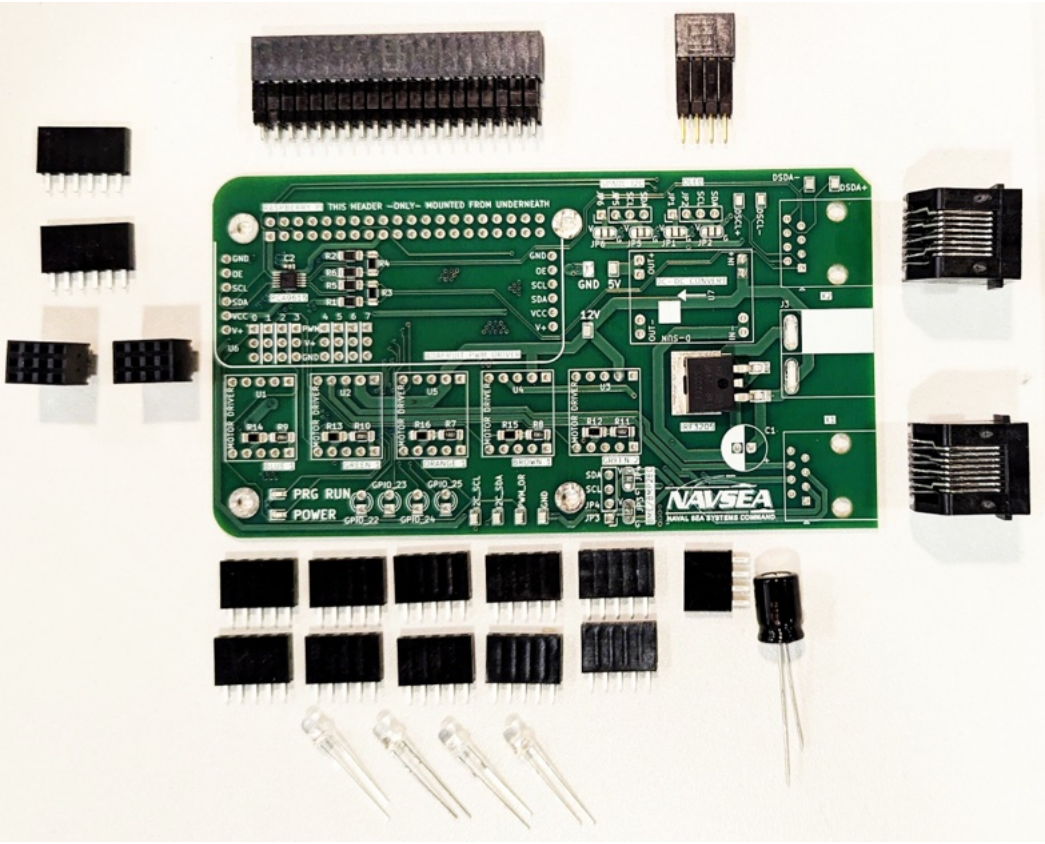


Figure 27: You should have all the parts in this figure except the LEDs - you will select the colors you'd prefer.

When installing the headers, as long as you follow the pictures and the silkscreen on the printed circuit boards, the order in which you install the headers is not very important. The issue is that some times when you add headers, it creates a little bit of a challenge to install the next one if the previous one is in the way. The following sequence is only one way to install the headers in a particular order. It is not necessarily the best order but it will be relatively straightforward to install one after the next.

Start by installing the 6 pin headers for the PWM board (figure 28).

Next add the two 3x4 block headers that support the PWM signals (figure 29)).

Next add the 5 motor driver headers - note there are 5 sets (2 headers per set) (figure 30)).

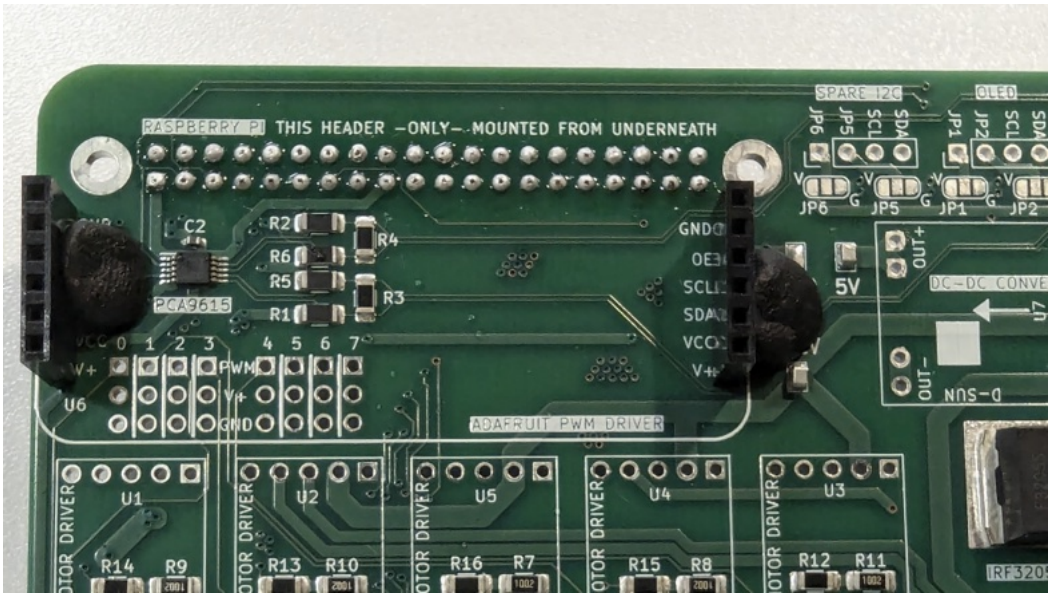


Figure 28: Mount the female header pins for the PWM breakout board. You can use a tiny piece of the butyl rubber to hold the headers in place when you turn upside down to solder.

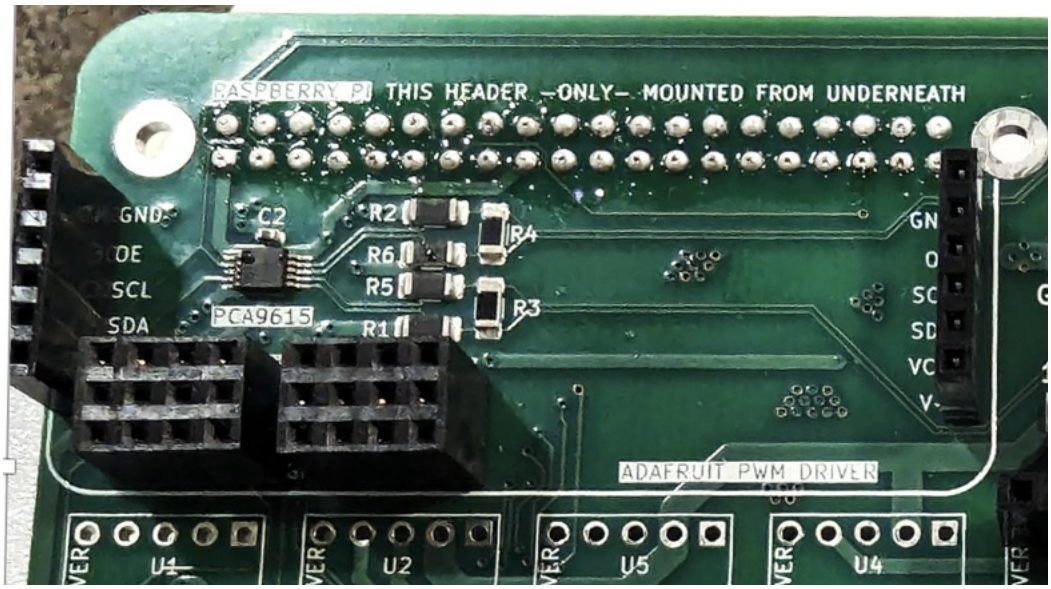


Figure 29: The 3X4 headers are mounted just below the PCA9615 chip.

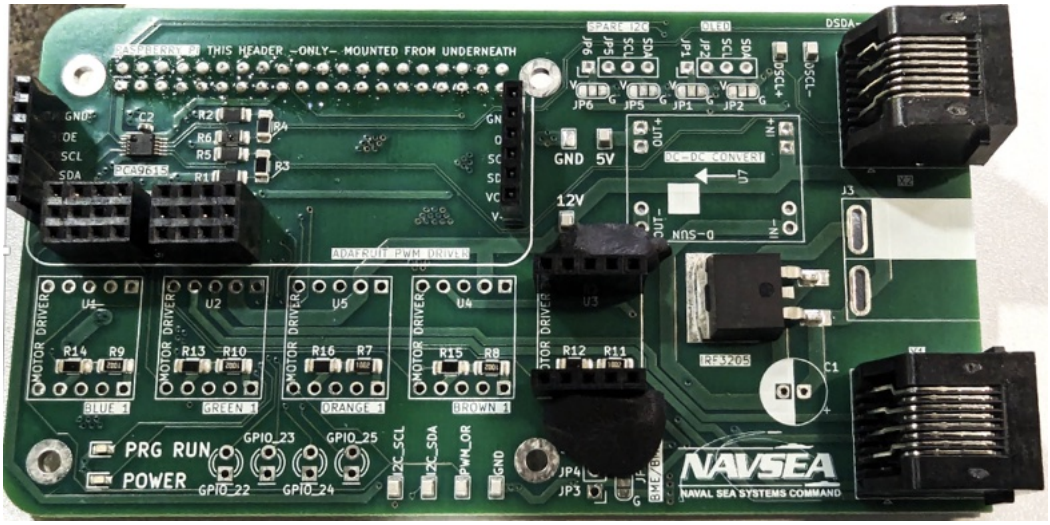


Figure 30: Mount the female header pins for the motor drivers ... there are 5 sets.

Now add the 4 pin header for the BME280 temperature, humidity and pressure sensor. This is the shorter 4 pin header - the taller 4 pin header is for the OLED screen (figure 31)).

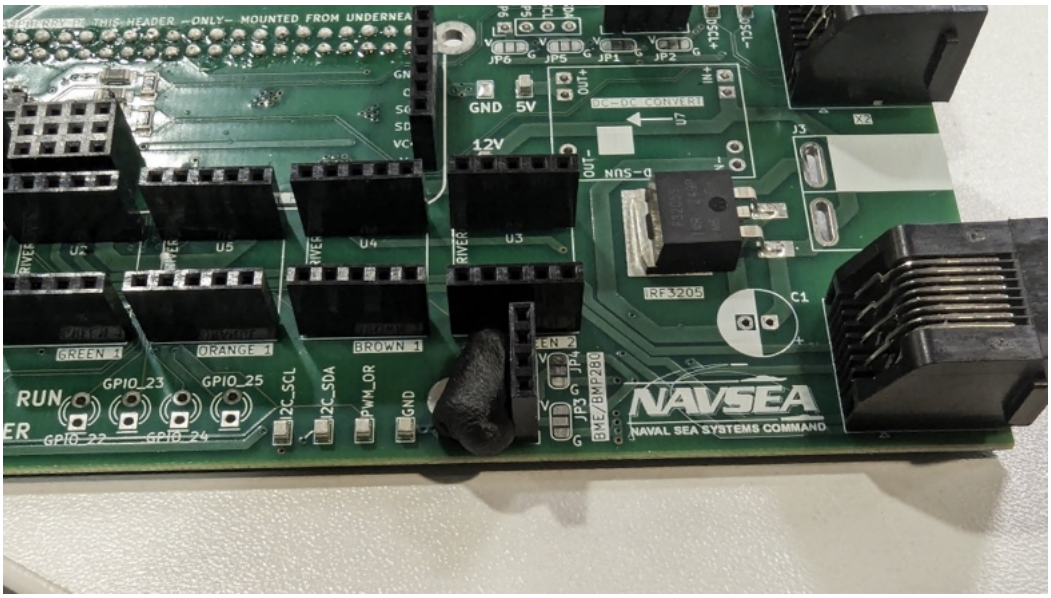


Figure 31: Mount the header for the temperature, humidity and pressure sensor. Note the motor driver headers above.

Now add the 4 pin header for the OLED screen (figure 32)). Hopefully you didn't use the taller header for the temperature sensor.

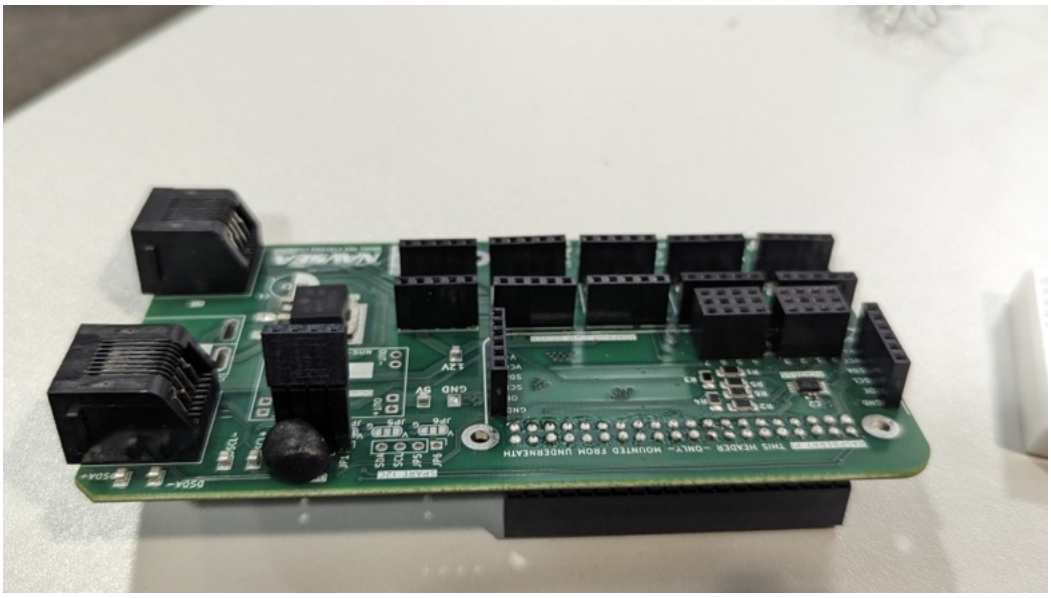


Figure 32: The OLED header is a double stack to allow the OLED to sit higher on the board.

The next headers aren't really headers but rather RJ45 Ethernet jacks. They click in place so you won't need to use butyl rubber. Just be careful that the little metal pins are lined up before you click the jacks in place (figure 33)).

Now add the capacitor (figure 34). **TAKE NOTE OF THE PICTURE!!!! DO NOT INSTALL THIS CAPACITOR BACKWARDS - IT WOULD BE BAD!!!**

Finally, you may select 4 LEDs to use as indicator lights - while not strictly necessary, they do provide critical functions to make sure your code is running the way you think it should. Like capacitors, they need to be installed with a particular orientation. Please note figure 35. The longer leg goes into the round solder pad and the shorter leg goes into the square solder pad.

You can now attach the Adafruit PWM board to the PCB (figure 36) and the motor drivers (FOLLOW THE GUIDELINES ON THE NEXT PAGE!!!!!).

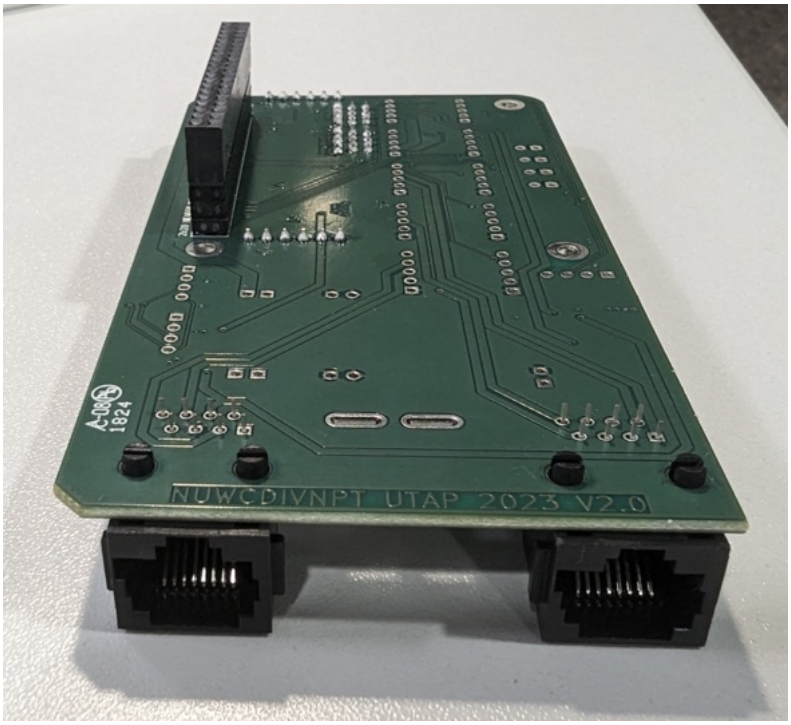


Figure 33: RJ

Figure 34: The capacitor has to be mounted correctly - note the picture and make sure the gold band is pointed away from the RJ45 connectors.

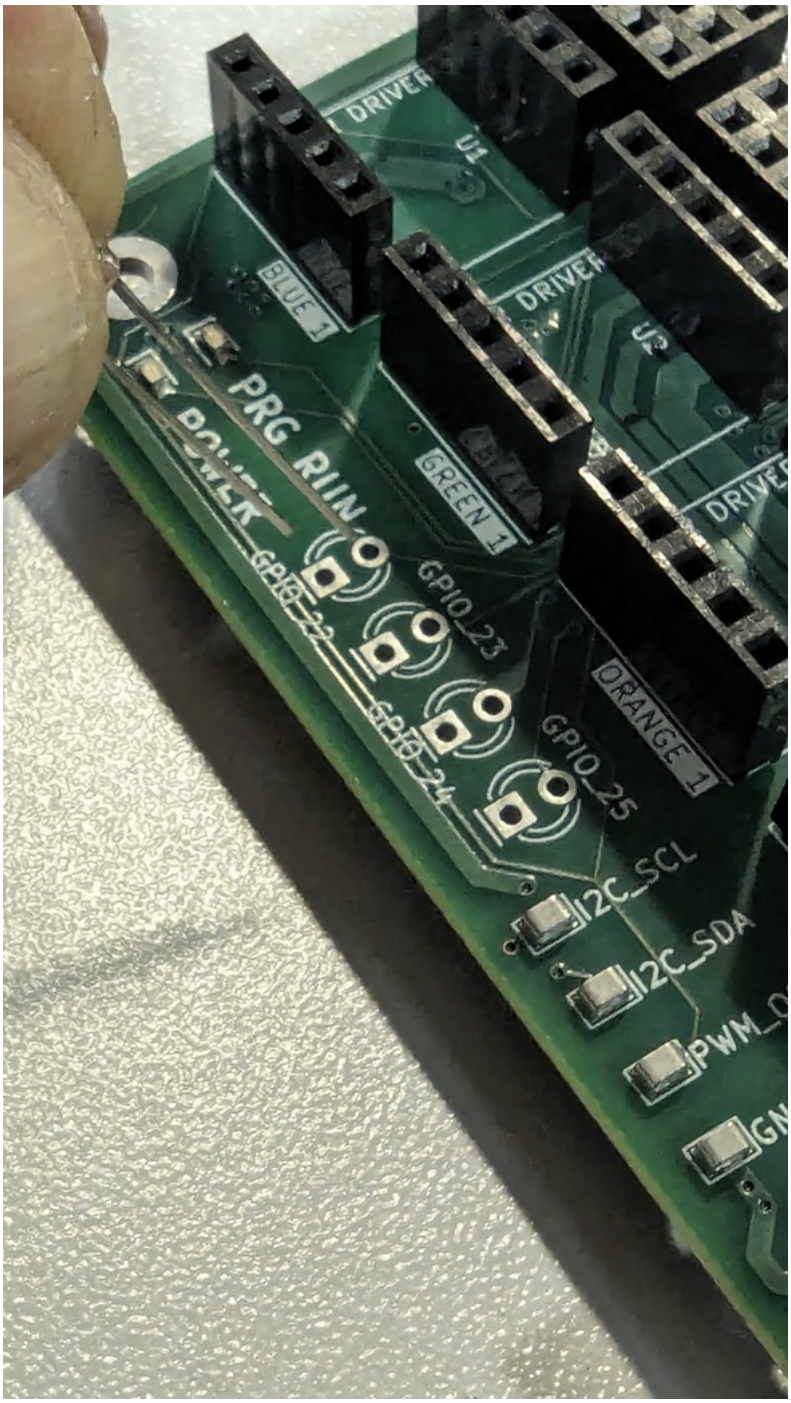


Figure 35: The long leg is the positive leg and the short leg is the ground (negative) leg. The short leg goes in the hole with the square solder pad.

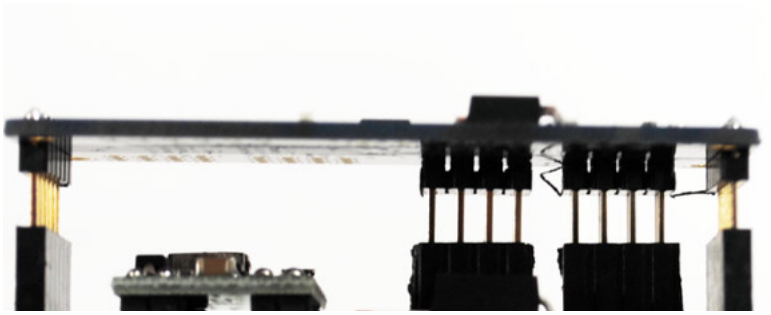


Figure 36: The PWM driver board mounted to the PCB.

DO NOT INSTALL THE MOTOR DRIVERS UPSIDE DOWN!!!!!

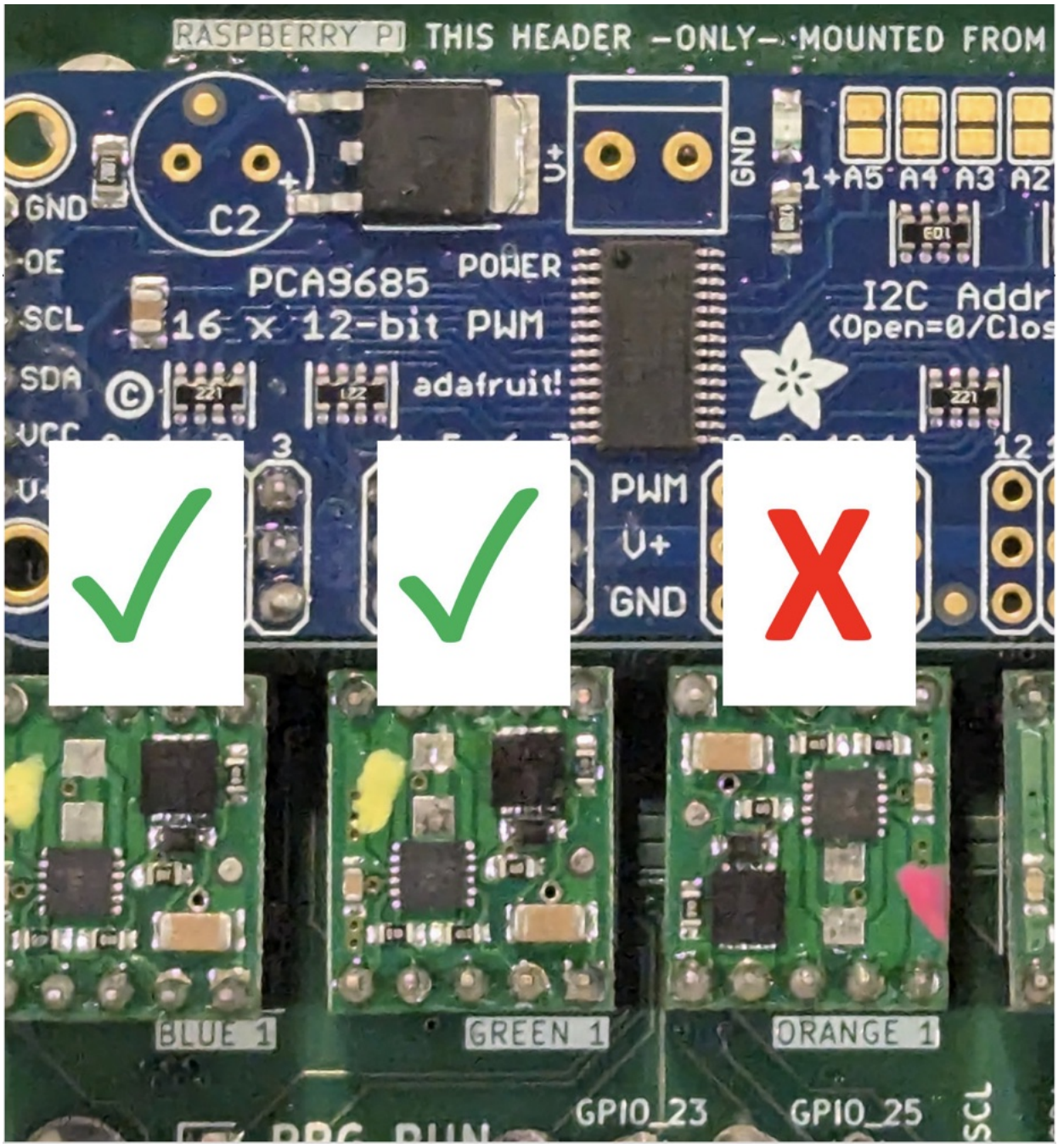


Figure 37: Look closely!!! The driver should be inserted so the colored marker (yellow in this case) is in the upper left NOT the lower right.

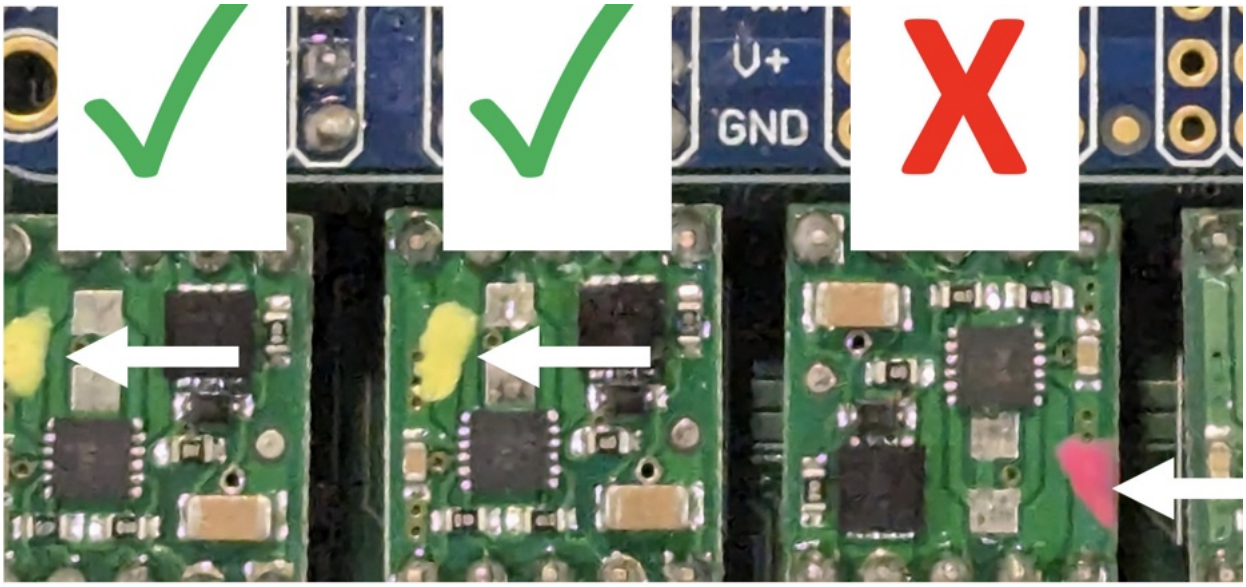


Figure 38: Look even MORE closely!!!!

11 Power Cord

Power cords with matching Anderson connectors to those you mounted to the board will be provided.

12 Completed Daughterboard

When the circuit board is completed it should look like figure 39. Yours may or may not have the compass display when powered up - it depends on the software condition on start-up. More to come on that topic.

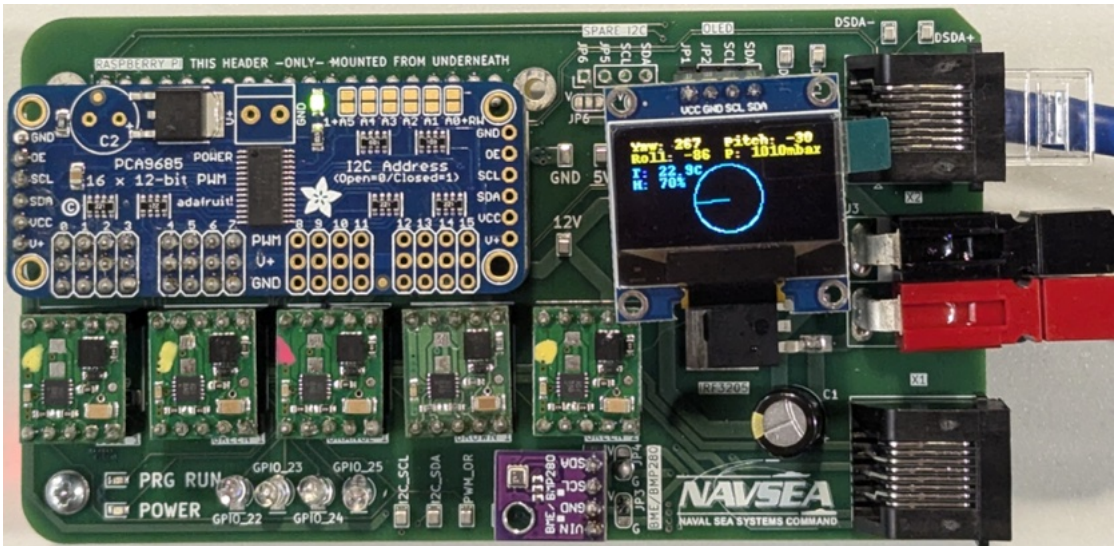


Figure 39: The completed circuit board.

13 Power

Before attaching the daughterboard to the RPi, it is a good idea to verify that the step down voltage regulator is actually producing a 5V reference voltage for the RPi. This was initially checked for all boards to prevent the RPi from being destroyed by providing too much voltage. However, things change so it's a good habit to make sure the voltage is correct. Anywhere in the range of 5.0 to about 5.3 is perfectly fine. A series of test points are located throughout the board - the 5V test point is seen in figure **??**. You measure with the black lead from the multimeter on GND and the red lead on 5V (figure 41).

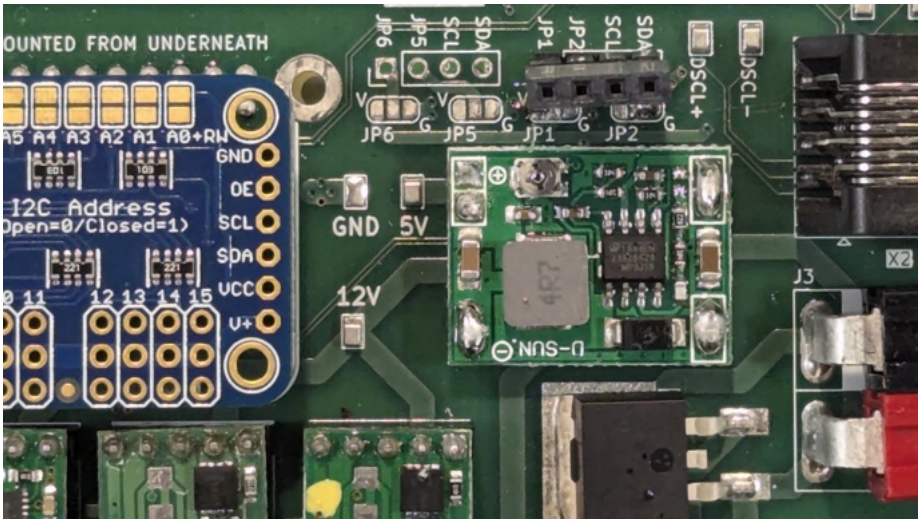


Figure 40: Test points for measure the 5V output needed for the Raspberry Pi.

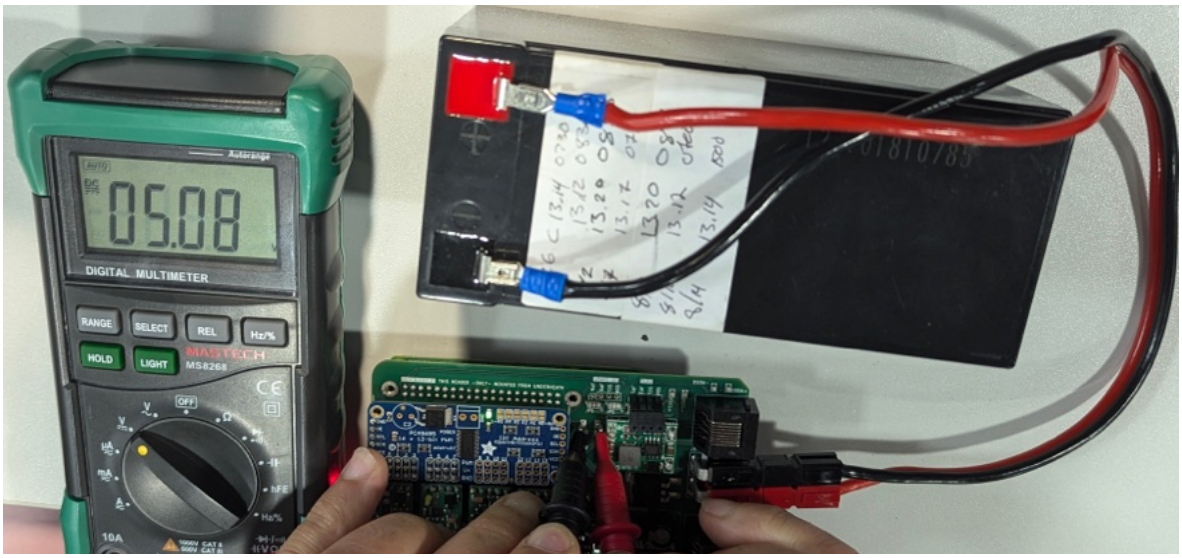


Figure 41: Use a multimeter to test the voltage. Ask a mentor if you need assistance.

14 Connecting Daughterboard to the Raspberry Pi

Now that power to the RPi has been verified to be a stable 5V, the daughterboard can be connected to the RPi. First, make sure there is no power connected to either the RPi or the board. Then, simply line up the female header pins under the daughterboard with the male header pins on the RPi. To keep them from coming apart, insert spacers between the two boards. Note that there are holes that are aligned between the two boards.

Insert the supplied spacers between the holes and attach with the supplied screws from the top only (figure 42). You'll need to remove the temperature sensor board temporarily to access the top of the spacer so do not connect the bottom at this point.

Figure 42: Attach 2 spacers with small screws - one is under the temperature sensor board (the little purple one) so you'll need to remove the board.

We have designed and fabricated custom boxes for this program (figure 43.) The entire circuit fits nicely into the box. **Be sure to slide the circuit all the way forward so the RJ45 jacks fit snugly in the openings.** This will make it much easier to insert the screws in the bottom spacer. Additionally, there is an acrylic shelf that prevents the USB from the gamepad from touching the underside of the RPi and shorting out the whole circuit.

First, you'll need to detach the power cable.
Now, attach the provided long screw through the bottom of the box and into the spacer (figure 45). If the board is well aligned, the screw should be able to be hand tightened. If not, you may need to wiggle

Figure 43: The entire circuit is placed inside the custom box.

it, just a little bit.

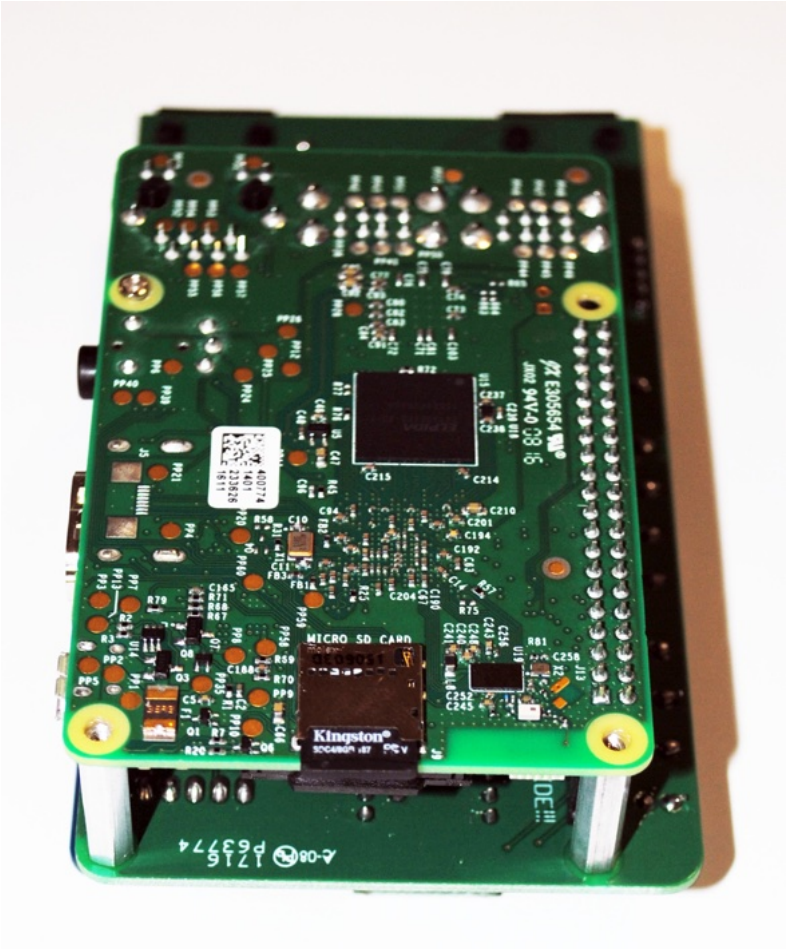


Figure 44: The board will be mounted to the box do not install screws at this point.

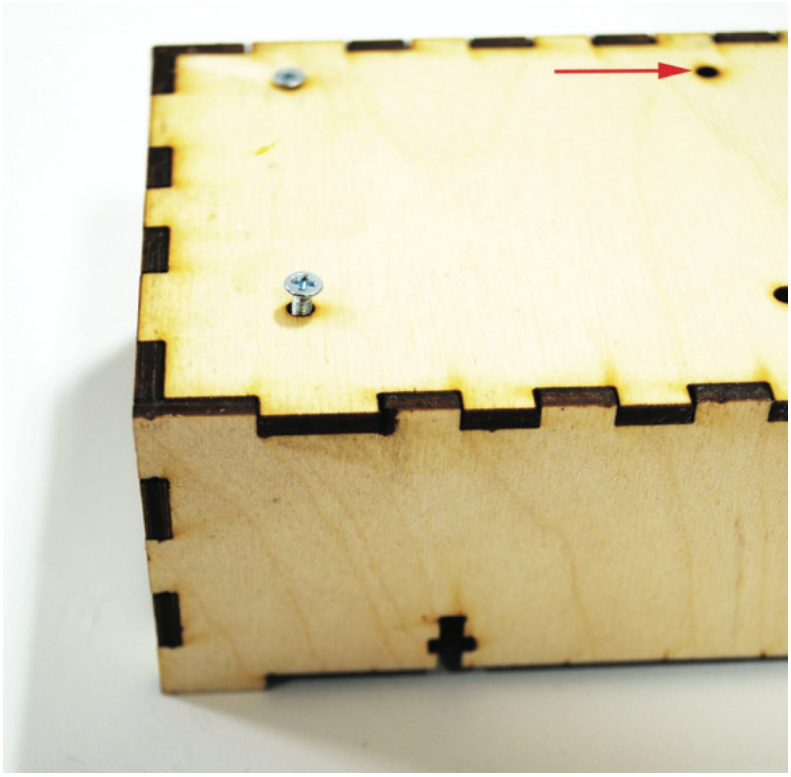


Figure 45: Once the board is aligned in the box (the RJ45 jacks will be all the way forward) attach screws through the box, into the spacer.

15 Code Overview

While we'll be spending some time during the apprenticeship going over the code, there are a few items to address here for reference. First, a copy of the BASE code is available on a GitHub repository at `https://github.com/jdicecco/UTAP`. (We'll be going over how to get a copy of the code onto your Raspberry Pi in the lecture). Note the use of the word BASE. This code has been constructed to provide you basic joystick control of you newly designed vehicle. The code assumes that the vehicle maintains the same motor structure as the basic SeaPerch, i.e. the motors attached to the green wires are on the right of the vehicle, the motors attached to the blue wires are on the left of the vehicle and orange is used for ascent and descent. Of course, if you've chosen to use all 6 motors (4 on one Ethernet cable, two on the other) you'll have two motors connected to green wires, two motors connected to blue wires, and one motor each connected to the orange and brown wires. Usually, one should avoid assuming how to place the motors on the vehicle. However, in this case, it really doesn't matter because this is all controlled in software. And software can be changed very easily.

The code accounts for reading the values from the gamepad - the joystick values, the button values, and the had values. Currently, only the joystick values are being used in the code. But again, this is software, so that can, and should, change. The code reads the value from the joystick, in the range of -32767 to 32767. But the software library that controls our PWM board is expecting an input in the range of -65535 to 65535. In hexidecimal representation 65535 would be 0xFFFF. You'll see this in the code in some spots - don't worry, we'll be going over all this in the lecture portion.

You have been provided base code that will allow you to control your vehicle in what it referred to as `Tank Drive`. That is, the left joystick controls the left side, the right joystick controls the right side, just like the SeaPerch controller. Now, of course, it has speed control.

The up and down thrusters are controlled by the trigger joysticks in the front of the controller and are also speed controlled.
Here's the really fun part: you can map any pin on the joystick to control any motor you want on your vehicle. If you want to use the hat for strafing (move from side to side) for instance, no problem. Since you're writing the code, you can make it do anything you want ... within reason.

16 Running the Code on Raspberry Pi

You can power the Raspberry Pi at this point with EITHER the battery or the USB port from your PC but NOT both!!!! The 5V regulator on the board is NOT at the same exact value as the 5V from your USB port.

In order to modify and run the code while developing the design, some simple steps are necessary

to open the code, change it, and run it. Of course, you need to supply power to your RPi board first.

First, we need to make a directory on the desktop of the Raspberry Pi. You are no doubt used to right clicking and making a folder. It turns out you can do it with a command as well. In Linux, that command is `mkdir name`, where name is the folder name you'd like to use. Change your working directory with the `cd` command to the desktop. Once there, make a directory called RPi (figure 46).

Next, we need to make sure the code has been downloaded from `https://github.com/jdicecco/UTAP`. The code we need is titled `UTAP_MAIN.py`. If you didn't follow along during the lecture and you still need to get the software, it's pretty simple. Make sure your current working directory is Desktop/RPi, then type `git clone https://github.com/jdicecco/UTAP.git`. You will find there is a new folder in your RPi directory called UTAP. In that folder is a Python script called `UTAP_MAIN.py`. Using either FileZilla or the terminal, make a copy of the script and place it in the RPi folder. Using the terminal, simply change directory to the new folder (`cd UTAP`) then type `cp UTAP_MAIN.py /home/pi/Desktop/RPi/`, which will copy the file from the UTAP folder that you just changed into to the RPi folder. (See figure 47). Now do the same for two other files, `launcher.sh` and `9dof_calibration.py`. Or, you can simply do them all at the same time ... `cp launcher.sh UTAP_MAIN.py 9dof_calibration.py /home/pi/Desk top/RPi/` (figure 47).

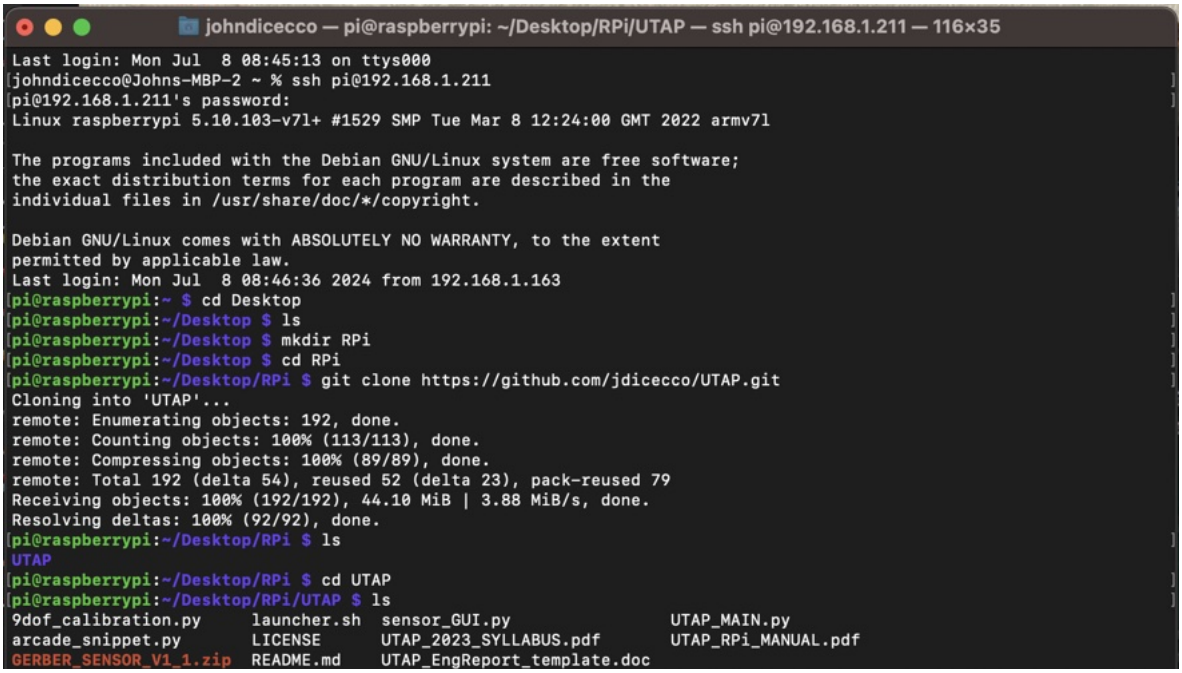


Figure 46: Terminal commands to get code from the Git repository.

As previously indicated, the primary interface between your laptop PC and the RPi will be a command line SSH connection. First, navigate to the folder that is on the RPi desktop. Once you're logged

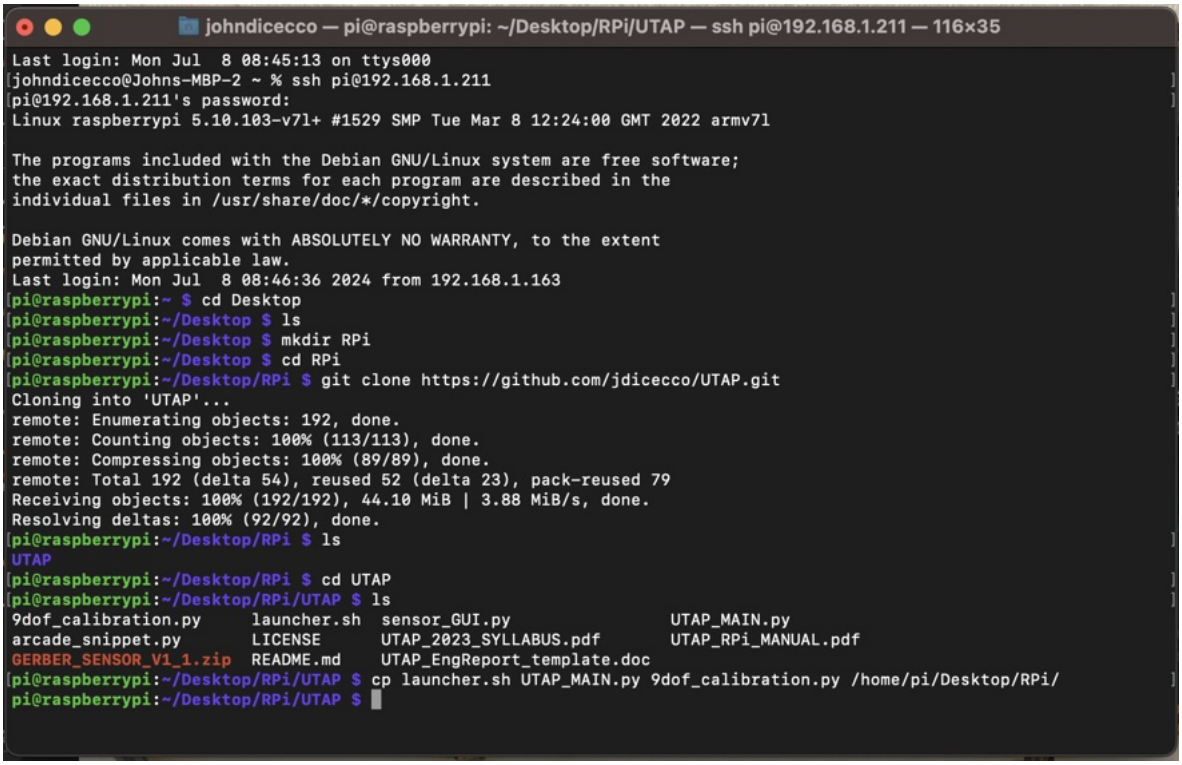


Figure 47: After downloading the git files, copy files from the downloaded folder (UTAP) to the folder you created on the Desktop called RPi.

into the RPi, enter `cd Desktop/RPi`(figure 48). This folder contains the base code that needs to be modified. To verify the files and other folders in the `RPi` folder, simply type `ls`. This will list the files (white) and folders (blue) as well as executable files (green) in the folder. The `ls` command works on all folders. Appendix B lists a handful of helpful commands for navigating a command line interface in Linux.

You should see a file titled `UTAP_MAIN.py`. This is the file that contains the base software for your RPi to control the vehicle. To modify it, you'll need to open the file for editing. The easiest way to do this is to open it right in the command line window. Using the editor `nano`, simply type `nano UTAP_MAIN.py` (figure 49) and you'll see the text appear in the window. **Remember, the mouse will not work for navigation!** You'll need to use the keyboard navigation (arrows, page up/down, home, space, etc.) to move from line to line - you won't be able to mouse-click to a new location.

Figure 48: Change the directory to the `Desktop` of the RPi.

Using only the keyboard to navigate through a file does take some getting used to. For the slightly more advanced student, the `UTAP_MAIN.py` file can be transferred to your laptop where you can use a word processing application (such as Notepad++) to modify the code. Then, when you are finished modifying the code, simply transfer it from the laptop back to the RPi. There are a number of ways this can be accomplished. One of the more popular methods is to use a program called FileZilla (engineers don't get out much).

FileZilla is a file sharing application that is used to transfer files between two computers that are connected through a SSH. Of course there are other types of connections besides SSH through which FileZilla can send files, but since we are already using SSH, that's what we'll use.

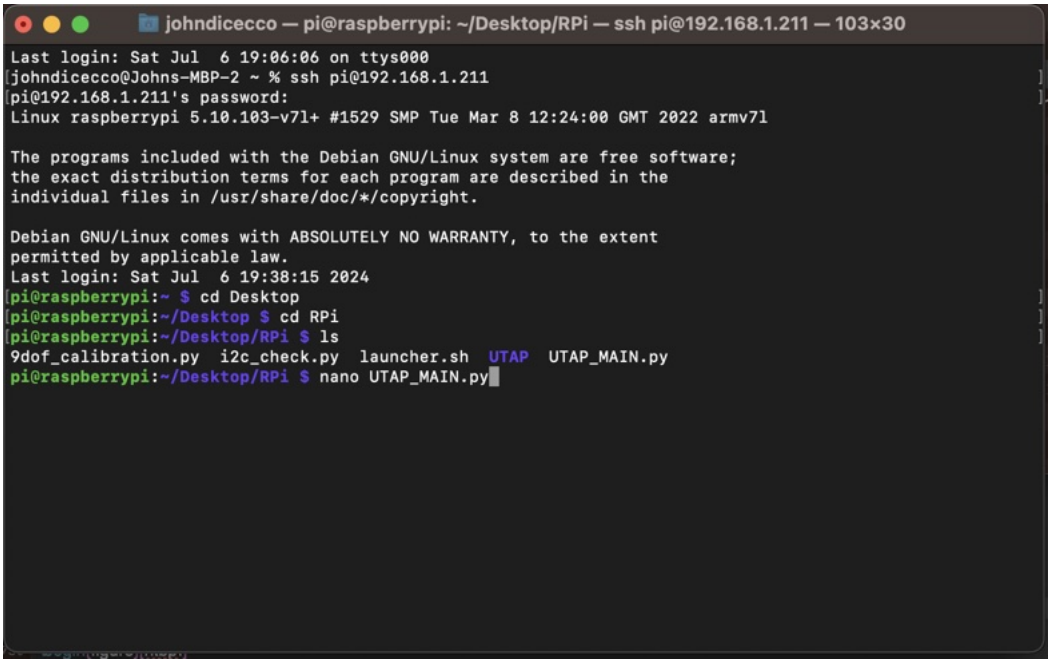


Figure 49: Use the "nano" text editor to modify your code.

Open the application FileZilla (it is already installed on the laptop). A window will appear like figure 51. You'll need to fill out the fields indicated by the red arrows and click `Quickconnect`. The `Local site:` (yellow highlight) is the current directory on your laptop where files will be transferred to/from. By default, you'll see the entire computer directory in a directory tree so you can pick where you'd like to transfer files to/from (figure 51). You could create a folder on your desktop, for instance, with your team name and select that as the `Local site:`. On the other side, you'll see `Remote site:` (also yellow highlight). This is your Raspberry Pi. Navigate to `Desktop/RPi` and you'll see all the files listed in the folder.

Simply click on the `UTAP_MAIN.py` file and drag it over to the folder displayed in the `Local site:` window. Now that file is on your laptop and you can edit it with an application like Notepad, or even better, Idle, which is a Python-specific editor. (This should already be on your laptop) (figure 52). Because Python is unforgiving when it comes to indent spacing, it's probably best to use an editor that preserves indent spacing. Also, note the color change of certain words/commands (figure 53). These are keywords and the editor is telling you they have specific functions associated with the words.

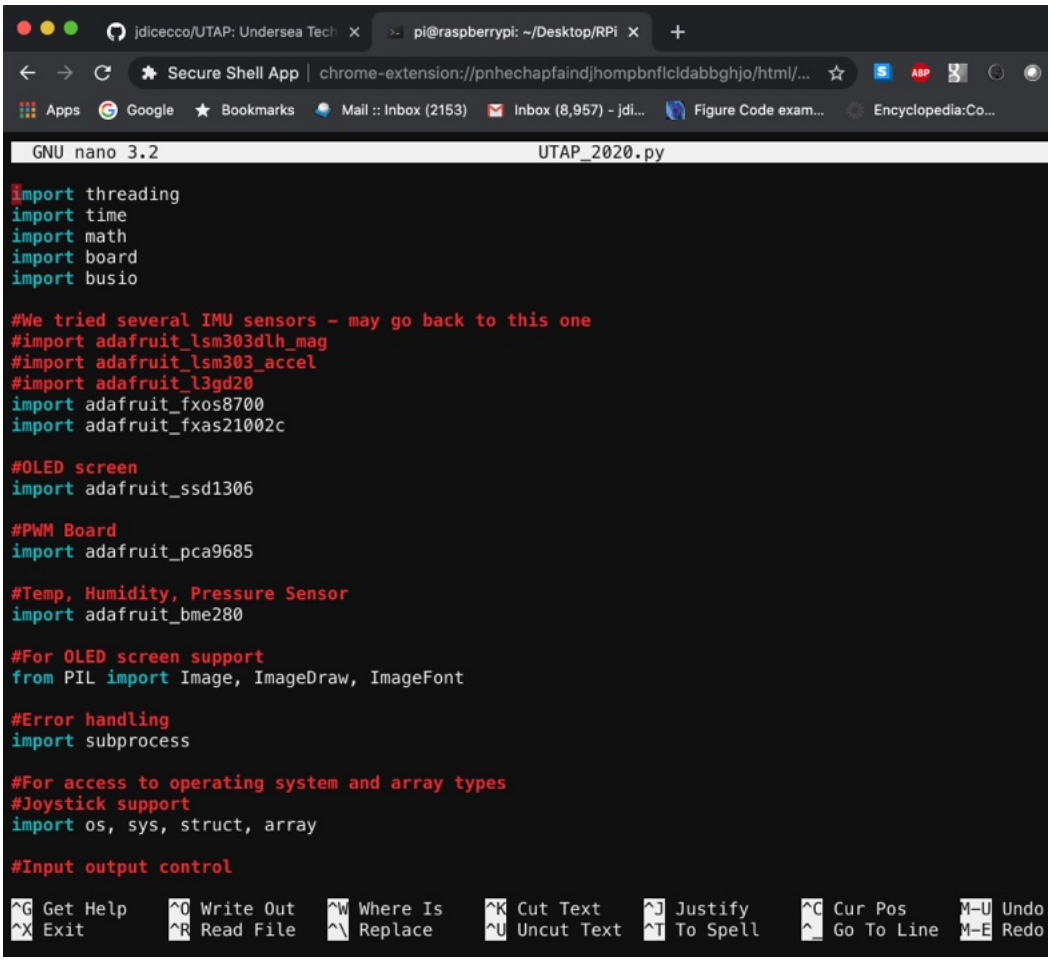


Figure 50: You can type and modify your code from here. Follow the commands at the bottom of the screen to exit and save your modified code.

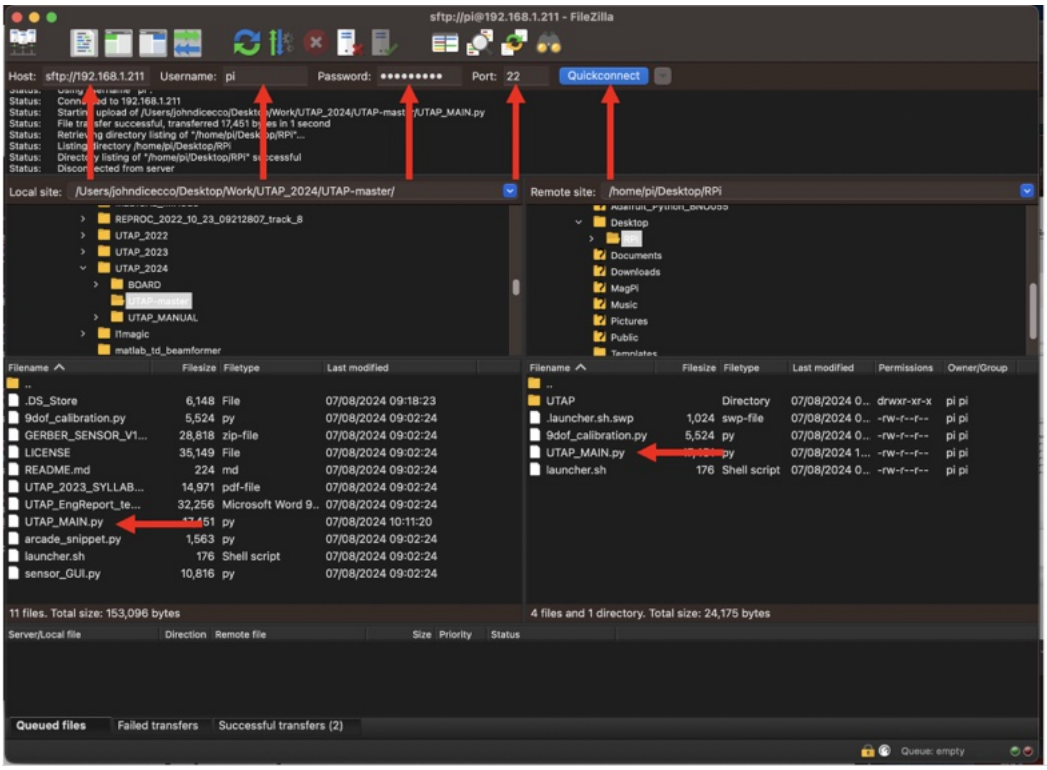


Figure 51: Filezilla.

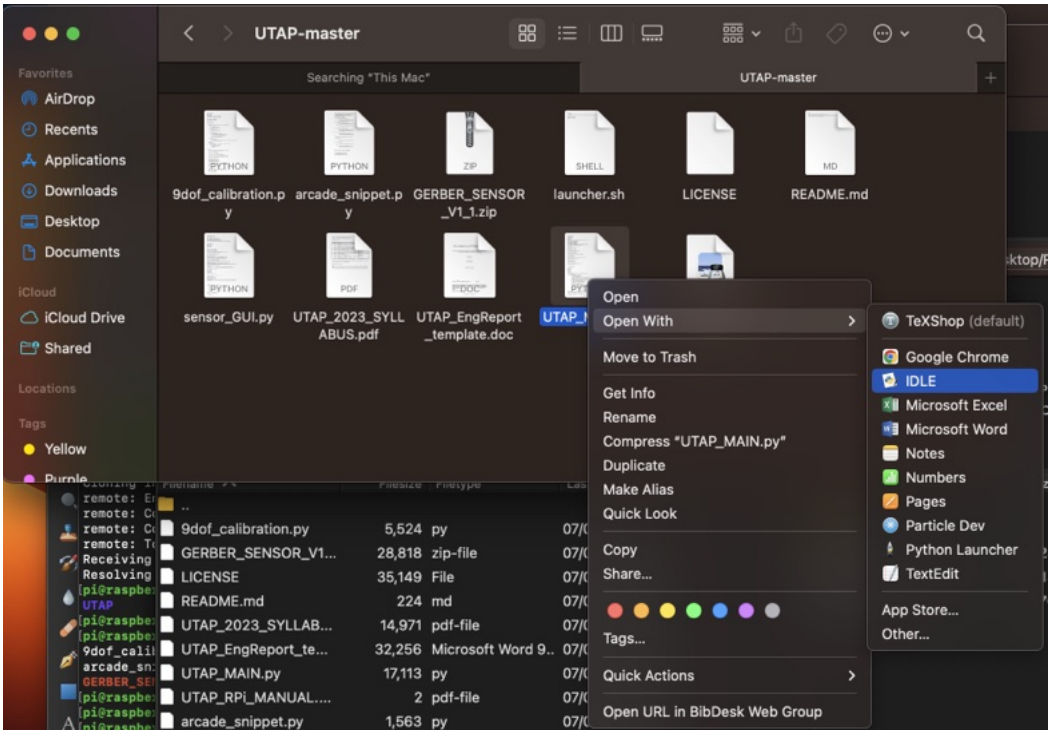


Figure 52: `Edit with Idle` should appear in the task window.

UTAP_MAIN.py - /Users/johndicecco/Desktop/Work/UTAP_2024/UTAP-master/UTAP_MAIN.py (3.11.4)

```
import threading
import time
import math
import board
import busio
#copied
#We tried several IMU sensors - may go back to this one
import adafruit_lsm303dlh_mag
import adafruit_lsm303_accel
import adafruit_l3gd20
import adafruit_fxos8700
import adafruit_fxas21002c
from adafruit_lsm6ds.lsm6dsox import LSM6DSOX as LSM6DS
from adafruit_lis3mdl import LIS3MDL

#OLED screen
import adafruit_ssd1306

#PWM Board
import adafruit_pca9685

#Temp, Humidity, Pressure Sensor
import adafruit_bme280

#For OLED screen support
from PIL import Image, ImageDraw, ImageFont

#Error handling
import subprocess

#For access to operating system and array types
#Joystick support
import os, sys, struct, array

#Input output control
#Joystick support
from fcntl import ioctl
```

Figure 53: The code as it appears in the Idle editor.

16.1 Running Code at Boot-Up

In order to make sure the software will run when the RPi is initially powered (without being connected to a computer), we need to make a shell script that will execute the code at boot time. This is necessary because it may not always be feasible to have a computer connected to it. In fact, when we are going to the base, we will NOT be able to bring computers. The shell script is placed in a root folder of the RPi and points to the UTAP Python code on boot. To make sure we are not running two instances of the program, we will only do one or the other. That is, we will either be modifying code during development and running it for testing purposes or we will run the program after development to execute the final code.

The shell script (like an executable file) is already in your `RPi` folder on the RPi's desktop. A line in the file `launcher.sh`[5] that points to the `UTAP_MAIN.py` program has been commented out with the # character. Simply remove the # from the line (figure 54) and your program will execute at boot time. There will be no need to run the command `python UTAP_MAIN.py` from the command line.

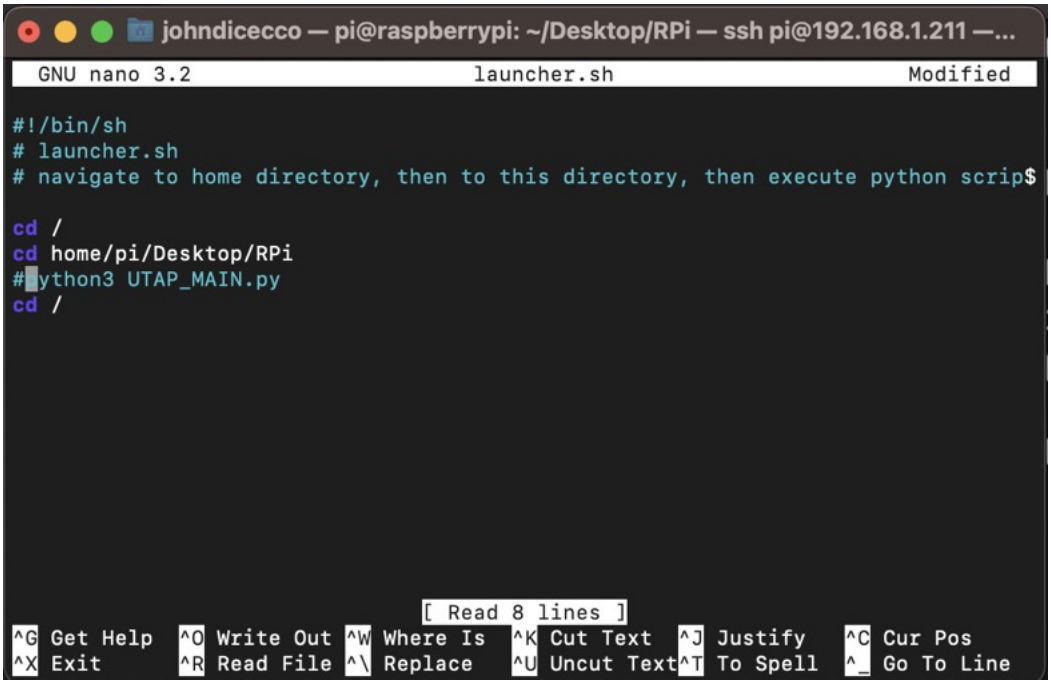


Figure 54: The launcher script.

17 Thruster Control

Recall from the SeaPerch you previously constructed, the color of the twisted pair of wires from the CAT-5 Ethernet cable was mapped to a specific motor, i.e. the blue twisted pair provided power to

[5]Incidentally, the `.sh` extension identifies this file as a **sh**ell script.

the left thruster, the green twisted pair provided power to the right thruster and the orange twisted pair provided power to the up/down thruster. The brown twisted pair was not used in the original SeaPerch design. We will now be using the brown wire but we'll also be using an additional Ethernet cable which will provide power for an additional four thrusters. This is where things get complicated, since the Ethernet cables will appear to be identical. Referring to figure 24, notice that the Ethernet jacks were labeled as RJ1 and RJ2 to help you keep track of which motors are being controlled by the software. So, OR1 in the software controls the thruster connected to the orange twisted pair belonging to the Ethernet cable plugged into RJ1. You might want to label you Ethernet cables so you'll know which motors are being controlled by the software.

A The Code

The code is maintained on GitHub at `https://github.com/jdicecco/UTAP`

References

[1] Raspberry Pi Foundation `https://www.raspberrypi.org/`

[2] Bohm, H. and Jensen, V. *Build Your Own Underwater Robot and Other Wet Projects.* Westcoast Words; 6th Rep edition (January 1997).

[3] Adafruit® Online documentation `https://www.adafruit.com`

[4] SparkFun® Online documentation `https://www.sparkfun.com`

[5] Pololu® Online documentation `https://www.pololu.com/product/2961`

[6] `https://github.com/`