## Raspberry Pi Zero W

You will have to solder the pins on to the board yourself, [Nolop](https://nolop.org/solder/) has access to several soldering stations. To interface with the Servo Hat, solder MALE Pin headers to your board.

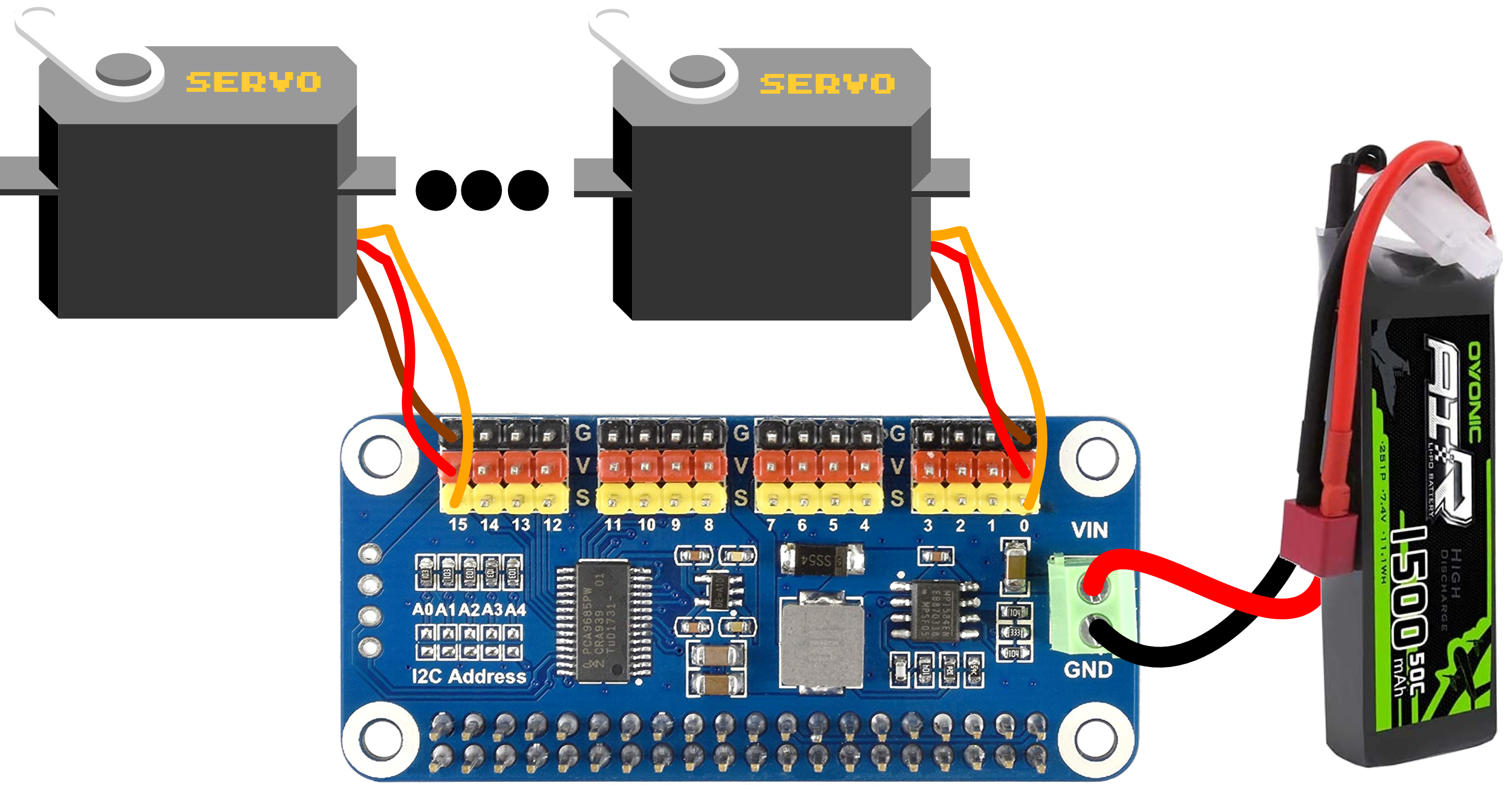
### Getting Started (With a Monitor)

1. Plug your SD Card into the Raspberry Pi
2. Connect your peripherals to the Raspberry Pi Zero W.
   * Plug the power cable in last
   * The EPDC located in the SEC has computers, physically re-route the peripherals from one of the computers there to your Raspberry Pi Zero
3. Wait - This part may take a bit of time, if it takes longer than 10 minutes, disconnect the power cable and then reconnect.
4. Follow the instructions in the dialog box, keep in mind if you plan on connecting to Wi-Fi on Tufts Campus…
   * Connect to **Tufts\_Wireless**
   * Don’t Check for Updates (It won’t work as there is one more step to connect to the internet)
   * Use **ifconfig wlan0** to find the MAC Address of your Raspberry Pi and register it with Tufts Technology Services via the online form
5. Configure the following interface(s):
   * I2C (For Controlling the Servo Hat), Serial (Optional for Communication), VNC (Optional for Communication)
6. Update the packages on your Raspberry Pi by opening a Terminal Window and entering the following line... **sudo apt update -y && sudo apt-get update -y && sudo apt-get upgrade -y**
7. Now Reboot…  **sudo reboot**

### Getting Started ([Without a Monitor](https://howchoo.com/pi/raspberry-pi-gadget-mode))

1. Flash Raspberry PI OS onto your [SD Card](SD%20Card)
2. Edit config.txt on the boot partition and append **dtoverlay=dwc2**
3. Enable SSH, on the command line in the boot directory type **touch ssh**
4. Edit **cmdline.txt** Look for **rootwait,** and add **modules-load=dwc2,g\_ether** immediately after. Save and Exit
5. Connect over USB and boot the Pi
   * After the Pi boots up (this will take a while initially), the Pi should appear as a USB Ethernet device, and you can SSH into it using: ssh pi@raspberrypi.local

### Using the [Waveshare Servo Hat](https://www.waveshare.com/w/upload/1/1b/Servo_Driver_HAT_User_Manual_EN.pdf)

1. I2C should be enabled
2. ****Install the following libraries from the Terminal
   * **python-smbus, p7zip-full, RPi.GPIO**
3. Connect the hat to the top of the Raspberry Pi
4. Connect one or more servos to the hat, using this diagram.
   * The Wires should be connected in line parallel to the short side of the board
   * The Brown wire to Black G Pin
   * The Red wire to Red V Pin
   * The Orange Wire to the Yellow S Pin
5. There is not one consolidated library to use, instead the libraries you installed in Step 2, are there to help you unpack and run the code that can be found here: http://www.waveshare.net/w/upload/6/6c/Servo\_Driver\_HAT.7z
6. The LiPo Battery Pack is only necessary if you want to operate without connecting the Raspberry Pi to power.
   * To make a connection between the LiPo and green terminal I soldered two jumper cables to the end of the connector.
   * When attaching your LiPo Battery to the Waveshare Servo Hat make sure the **Red wire attaches to VIN and the Black Wire Attaches to GND**
   * **CAUTION: IF USING THE LIPO BATTERY SEE THE SECTION ON SAFETY PROCEDURES AT THE END OF THIS SHEET**

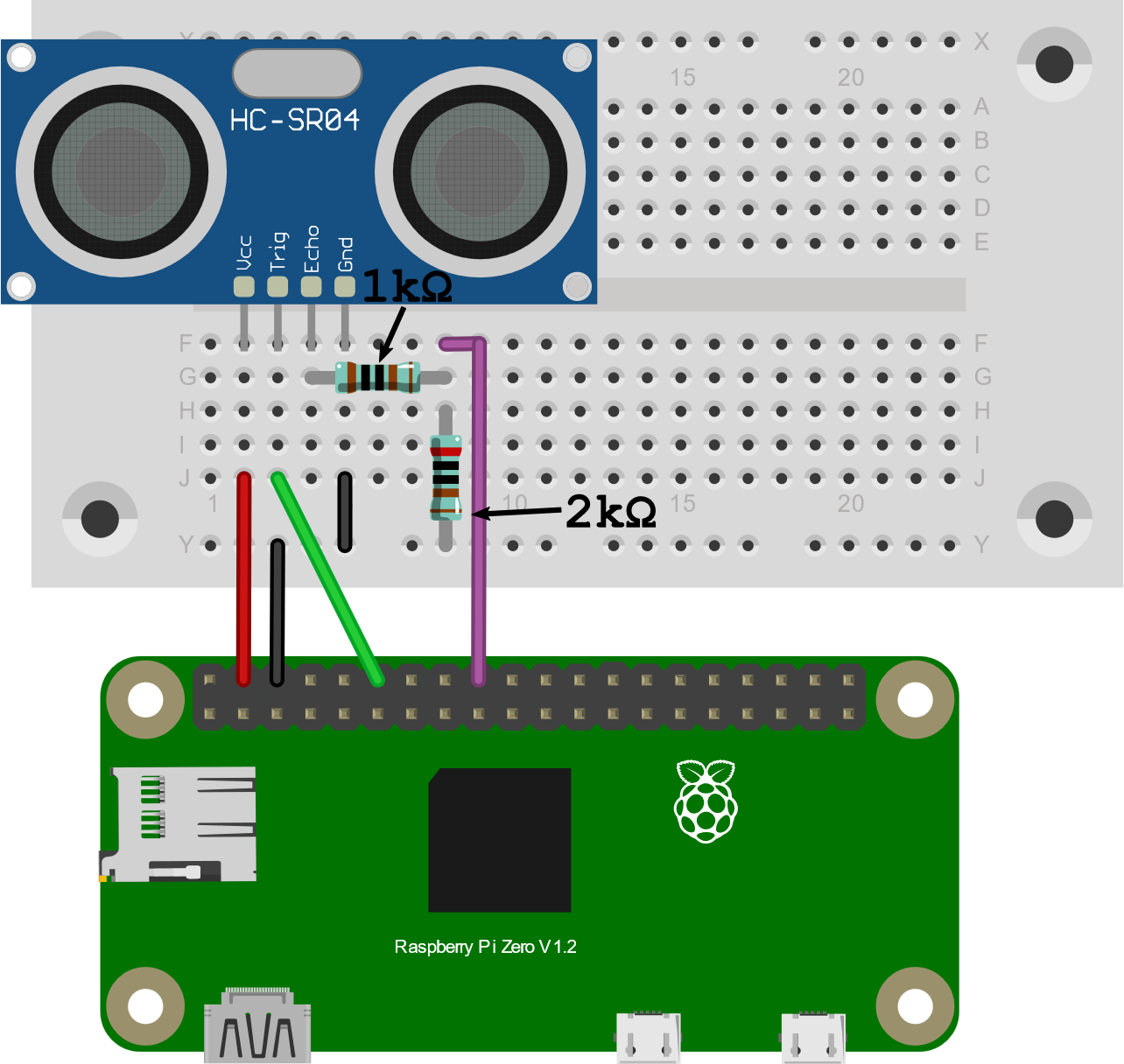
### Getting Readings from the Lidar Sensor

1. pip install the **adafruit-circuitpython-vl53l0x** package
2. To get readings, import adafruit\_vl53l0x. This package has a ton of documentation online if you are looking for specific commands.

|  |
| --- |
| import board, busio, adafruit\_vl53l0x  # Initialize I2C bus and sensor.  i2c = busio.I2C(board.SCL, board.SDA)  sensor = adafruit\_vl53l0x.VL53L0X(i2c)  # Get Data  print(sensor.range) |

1. Refer to the diagram on the right for wiring it up.
   * Pi 3V3 to sensor VIN (red wire on STEMMA QT version)
   * Pi GND to sensor GND (black wire on STEMMA QT version)
   * Pi SCL to sensor SCL (yellow wire on STEMMA QT version)
   * Pi SDA to sensor SDA (blue wire on STEMMA QT version)

### Getting Readings from the Ultrasonic Sensor HC-SR04

There is no specific package used to manage the Ultrasonic Sensor HC-SR04. Instead, you should wire it up as shown below and look for example code on the internet!

* + VCC Connects to Pin 2 (5v)
  + Trig Connects to Pin 7 (GPIO 4)
  + Echo Connects to R1 (1k Ω)
  + R2 (2k Ω) Connects from R1 to Ground
  + Wire from R1 and R2 connects to Pin 11
  + GND connects to Pin 6 (Ground)

## Coding on the Raspberry Pi

Graphical user interface, application

Description automatically generatedIf you are interested in coding directly on the Raspberry Pi, there are a ton of IDEs to choose from. To find them, open the main menu (top left of your screen), and click on programming. The two I recommend looking into are Geany and Thonny.

## [Thonny IDE](https://thonny.org/)

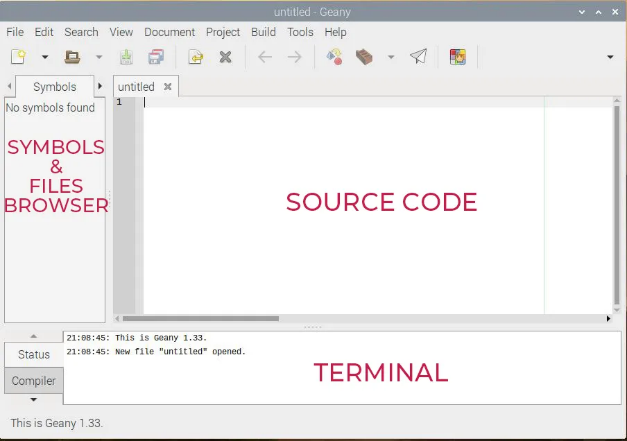
*Graphical user interface, text, application, email

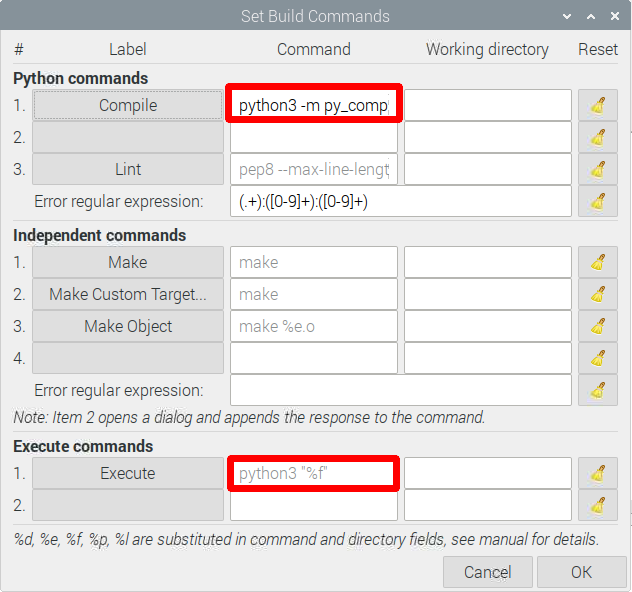
Description automatically generatedGreat for Python*

The Thonny IDE is a bit simpler, and their website has a ton of simple documentation, so I’ve linked it above. You’ll type source code in the main section, the bottom section is a terminal display, and the top are all the buttons to run your program. The biggest advantage is the built-in debugger, which allows you to step through your either line by line or by stopping at breakpoints in the code. This is essential when our code starts to get much larger. [Learn More](https://learn.sparkfun.com/tutorials/python-programming-tutorial-getting-started-with-the-raspberry-pi/hello-world)

## [Geany IDE](https://www.geany.org/)

The language of choice when programming a Raspberry Pi is usually python; however, this is not always the case, and the Geany IDE is a great multi-language IDE.

The Geany interface looks like this: This is standard for a code editor. You’ll type the source code in the main section. On the left there is a browser to see your files and variables. On the top you have the main menu, with essential shortcuts (you can customize this). And what’s interesting is the bottom part, where you can see the logs, and have a terminal on hand to run your scripts (perfect for Python for example).

***A picture containing graphical user interface

Description automatically generated***There is one small edit you will want to make in order to have your python programs run with python3. Create a new file by going to *File > New (Ctrl + N),* then save it by going to *File > Save As…* and name it ***test.py***. Now go to *Build > Set Build* *Commands* and edit the two boxes show in red to the right by replacing ‘python’ with ‘python3’. What this does is run python3 instead of python2 when executing the program. To run a program simply click the paper airplane shown below.

[Learn More](https://raspberrytips.com/use-geany-on-raspberry-pi/)

For even more information on other Raspberry Pi IDEs checkout this article by [Dzone](https://dzone.com/articles/best-raspberry-pi-ides).

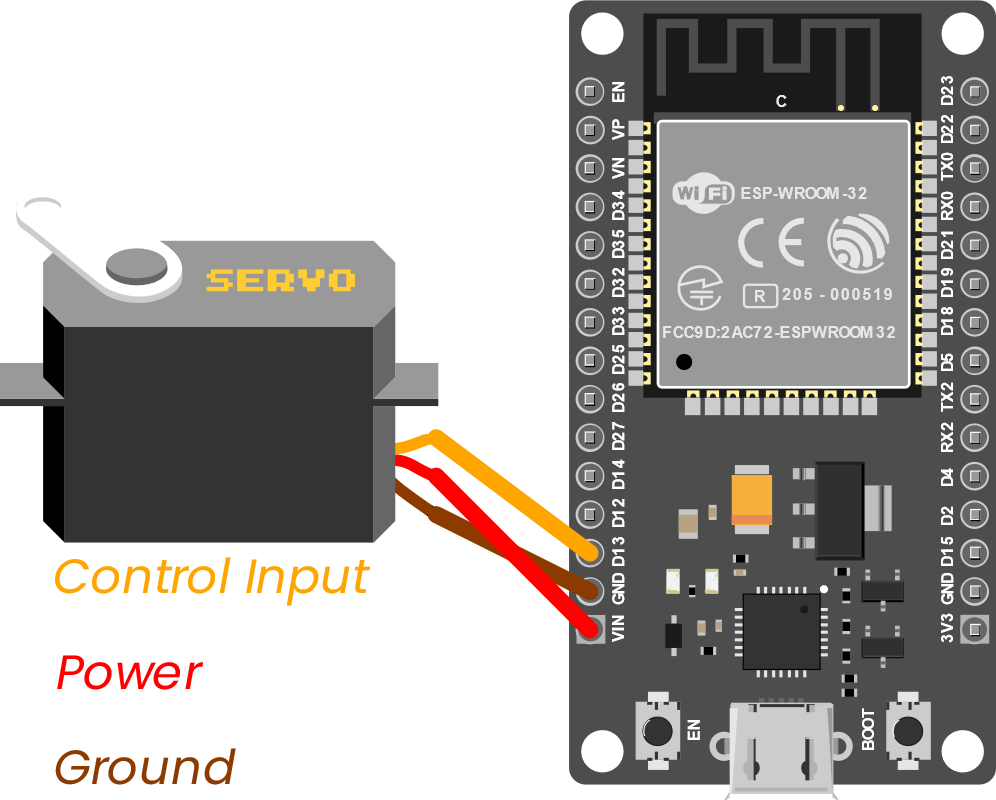
## ESP 32

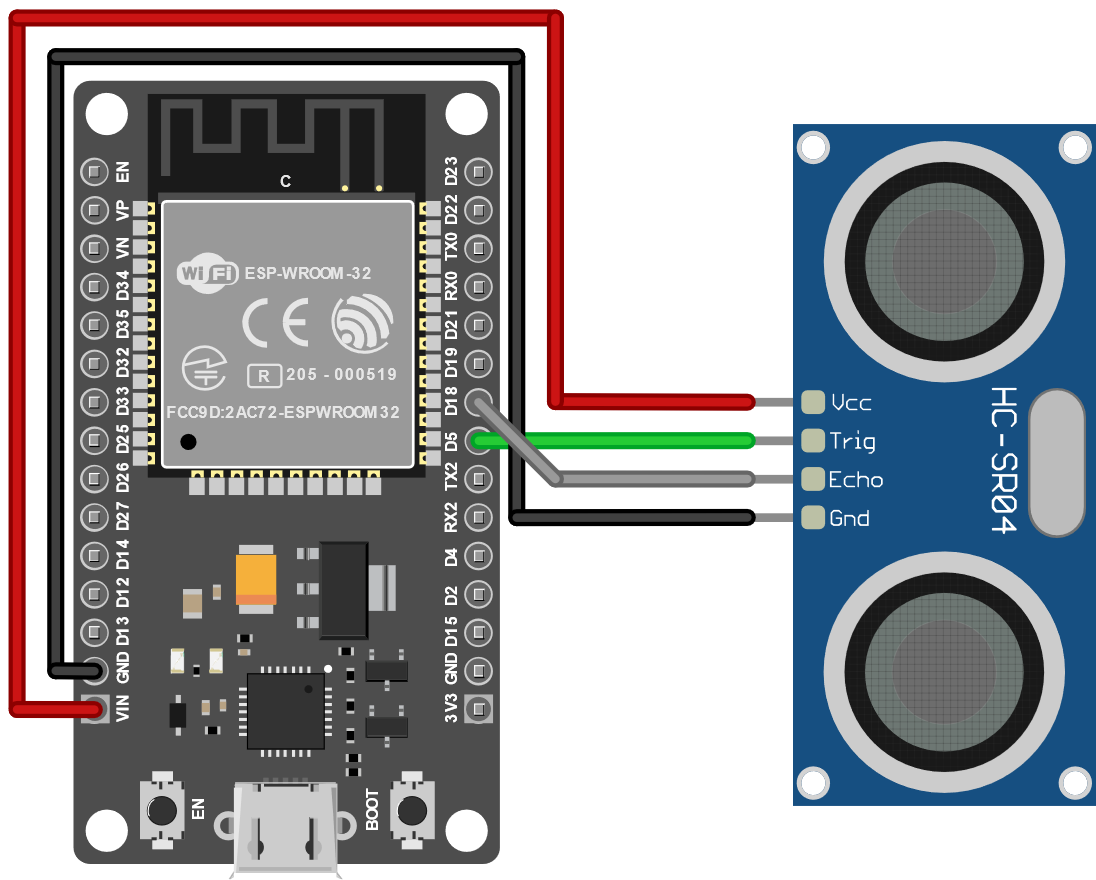
### Getting Started

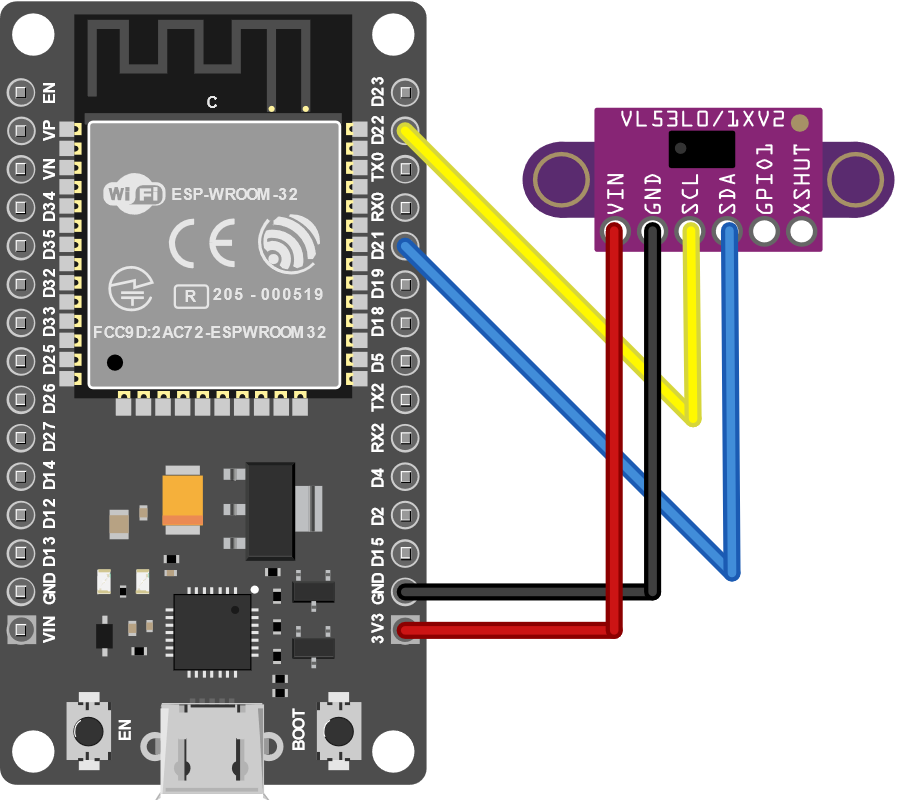
1. If you don’t have it already, download and install the [Arduino IDE](https://www.arduino.cc/en/software)
2. Add the ESP32 Board to the Arduino Board Manager by [URL](https://randomnerdtutorials.com/installing-the-esp32-board-in-arduino-ide-windows-instructions/). Use the Package Maintained by [Espressif](https://dl.espressif.com/dl/package_esp32_index.json)
3. Select the Board and Port in the **Tools > Board** menu (DOIT ESP32 DEVKIT V1)
   * If you don’t see the COM Port in your Arduino IDE, you need to install the CP210x USB to UART Bridge VCP Drivers)
4. If you are planning on using **Tufts\_Wireless** to connect this board to the internet, [find it’s MAC Address](https://randomnerdtutorials.com/get-change-esp32-esp8266-mac-address-arduino/) and register it with Tufts Technology Services.
5. There are a ton of great example codes to get you started programming in **File > Examples**

### Running the ESP32

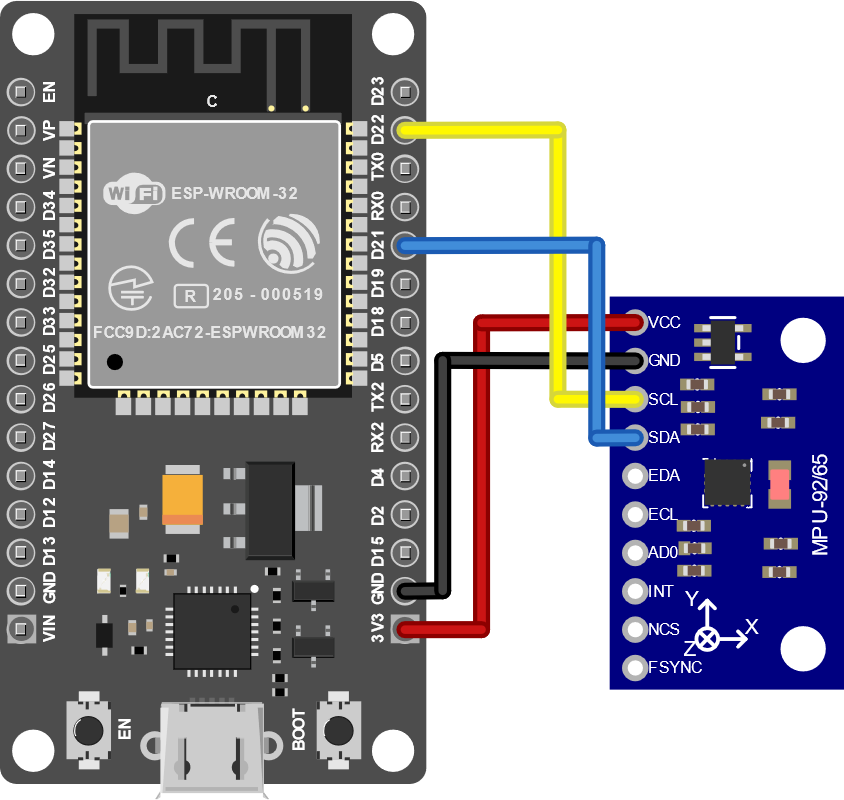
1. Connecting to WiFi from
   * **File > Examples > WiFi (ESP32) > WiFiScan**
   * **WiFi.macAddress()** returns the MAC Address of the device
2. Using the [ESP32 Dual Cores](https://randomnerdtutorials.com/esp32-dual-core-arduino-ide/)
   * The ESP32 comes with 2 Xtensa 32-bit LX6 microprocessors, so it’s dual core.
   * You can use the **xTaskCreatePinnedToCore()** function to pin a specific task to a specific core
3. The ESP32 Chip can run [MicroPython](https://docs.micropython.org/en/latest/esp32/tutorial/intro.html#powering-the-board); however, while running MicroPython you cannot take advantage of the Dual Cores
4. Using the Servo Motors
   * To run the [Servo Motors](https://dronebotworkshop.com/esp32-servo/) use ServoESP32 made by RoboticsBrno (Jaroslav Paral).
   * Find it in the Library Manager **Sketch > Include Libraries > Manage Libraries**
   * Refer to the wiring diagram. The orange wire **Control** can connect to any digital IO pin.



1. Getting Readings from the [Ultrasonic Sensor](https://www.sparkfun.com/products/15569) HC – SR04
   * No library or demo code for this device, but examples can be found online!
   * **Refer to this wiring Diagram
2. Getting Readings from the Lidar Sensor
   * To get readings from the Lidar Sensor use the [Adafruit\_VL53L0X Library](https://github.com/adafruit/Adafruit_VL53L0X)
   * To test if it is working open **File > Examples > Adafruit\_VL53L0X > vl53l0x** and upload to your Arduino wired up to the sensor
   * This sensor uses I2C communication. Refer to the Wiring Diagram (Bottom Left)

**

1. Getting Readings from MPU9250
   * To get readings from the ESP32 I used the library made by Bolder Flight
   * It can be found in the Arduino Library Manager called **Sketch > Include Libraries > Manage Libraries > Bolder Flight Systems MPU9250** Version 1.0.1
   * To test if it is working open **File > Examples > Bolder Flight Systems MPU9250 > Basic\_I2C** and upload to your Arduino wired up to the sensor. The numbers will appear in Serial.
   * This sensor uses I2C communication. Refer to this Wiring Diagram to connect things up



## Some Helpful Links

* [Datasheet for Lidar](https://cdn-learn.adafruit.com/assets/assets/000/037/547/original/en.DM00279086.pdf)
* [Datasheet for Ultrasonic Sensor](https://cdn.sparkfun.com/datasheets/Sensors/Proximity/HCSR04.pdf)
* [Datasheet for Raspberry Pi Zero W](https://www.electronicsdatasheets.com/datasheet/RPI-ZERO-V1_3_reduced.pdf)
* [Datasheet for ESP32](https://www.espressif.com/sites/default/files/documentation/esp32_datasheet_en.pdf)
* [Datasheet for Servo Driver Hat](https://www.waveshare.com/w/upload/1/1b/Servo_Driver_HAT_User_Manual_EN.pdf)
* [Data Sheet for Servo Motors](http://www.ee.ic.ac.uk/pcheung/teaching/DE1_EE/stores/sg90_datasheet.pdf)

## LIPO BATTERY INFORMATION

Lithium Polymer (aka "LiPo") batteries are a type of battery now used in many consumer electronic devices. In the past few years, they have become more and more popular in the radio control industry and are now the most popular choice for anyone looking for long runtimes and high power. In fact, most phones today will use a LiPo Battery. However, the LiPo Batteries we will be using do not have the same safety mechanisms built into it. All that means is you should treat this battery with a bit more caution.

**Kit Specific:**

* When attaching your LiPo Battery to the Waveshare Servo Hat make sure the red wire attaches to VIN and the Black Wire Attaches to GND.
* Always attach the wires of the LiPo Battery Connector to the green terminal FIRST and then attach the LiPo Battery itself.

**General:**

* Do not puncture your LiPo Battery
* Keep your LiPo Battery Away from sources of heat (Don’t leave it in the sun)
* Always charge LiPo batteries on surfaces that won’t catch on fire such as cement, steel, ceramic or stone. Wooden tables and carpeted floors are not recommended charging surfaces.
* Do not charge batteries near flammable products or liquids.
* Never charge a LiPo battery while inside your model or other electronic device
* More Information: <https://www.genstattu.com/blog/lipo-lithium-battery-safety-guide>

**Lidar Accuracy**

