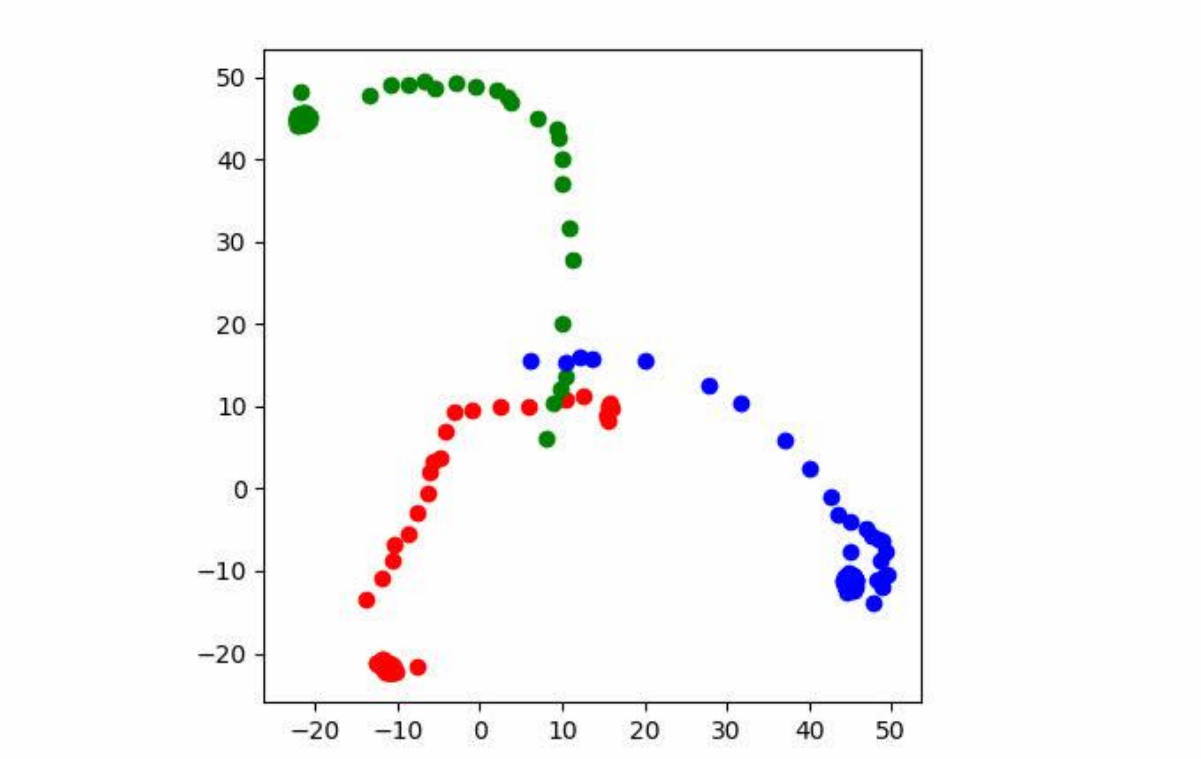
Adafruit SensorLab - Magnetometer Calibration

Created by lady ada

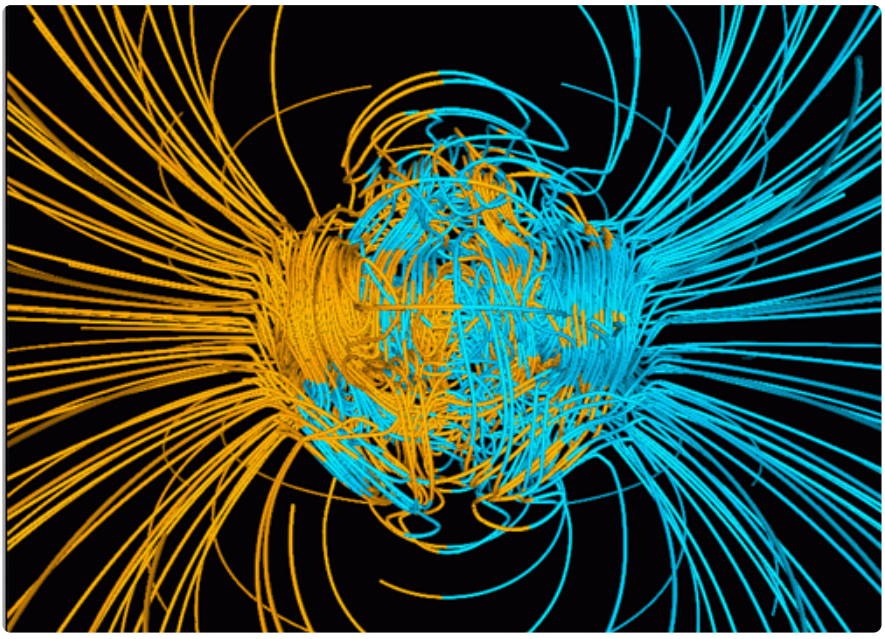


https://learn.adafruit.com/adafruit-sensorlab-magnetometer-calibration

Last updated on 2021-11-15 07:57:30 PM EST

Table of Contents

Magnetometer Calibration



Magnetometers can be used to detect orientation with respect to the Earths magnetic field. Basically, like a compass! We can tell which way is North, and thus correct for motion calculation errors and 'absolute orientation'

Good stuff! But, magnetometers have to measure a very small magnetic field of 35-65 uTesla, in a world full of magnets. And there's some offset when they are manufactured and pick and placed.

Of all the sensors that need calibration, magnetometers are the most essential to calibrate! Unless you're detecting strong magnets, there's no way for a magnetometer to work unless you perform a hard iron offset calculation. Once this is done, you will get rid of any strong magnetic offset values and be able to find magnetic North!

Install SensorLab

Since there's dozens of different sensor manufacturers out there, and we don't want to have a ton of #ifdef's in our code to manage each kind, we'll be using Adafruit SensorLab to manage detecting the various magnetometers, accelerometers, pressure sensors... etc!

Adafruit SensorLab automatically detects a wide range of sensors, over I2C, no matter what I2C address it's on. It will return an Adafruit Unified Sensor (https://adafru.it/aZm) object that we can query for events. You can't do advanced stuff like manually setting ranges or internal filters, but for many projects the basics will do just fine!

We'll be assuming you have the sensor on the main I2C port, and of course use the matching Adafruit library to verify the sensor is working and powered right before you continue!

A list of supported sensors is available here (https://adafru.it/IBg)

Remember, only supported sensors on I2C will be detected!

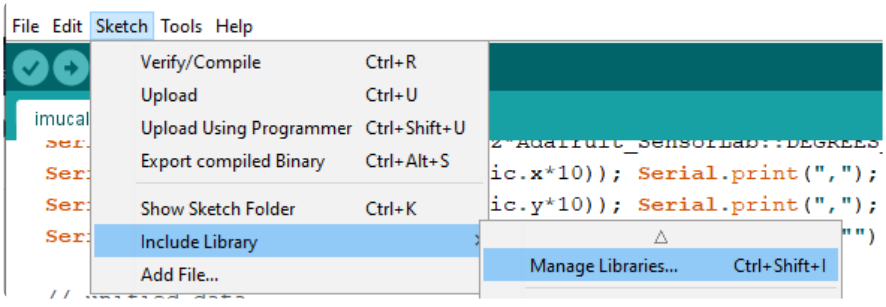
Install SensorLab

Since there are a ton of sensors, and we also use Arcada in a few examples (https://adafru.it/EUk), there's a lot of libraries to install

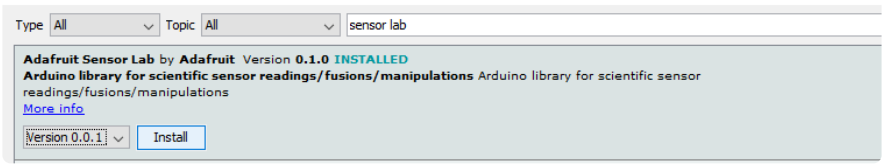
No really, we have a lot of software involved here - probably 20 or so libraries total!

For that reason we really strongly recommend you use Arduino 1.8.10 or greater which handles automatic library dependency installation. Otherwise you will be frustrated...

Select the Sketch -> Include Library -> Manage Libraries...



Search for Sensor Lab and install the Adafruit library you see



Simple Magnetic Calibration

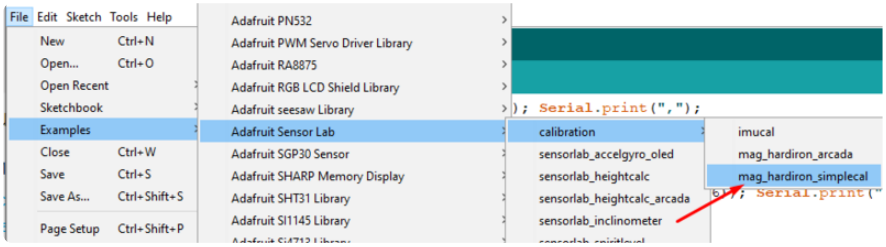
If you don't want to set up a graphical interface for calibration a magnetic sensor, you can do a simple hard iron calibration using just the serial interface. The nice thing about this is it will work for any and all boards, and does not require any additional software installation!

This example can be run by any Arduino compatible, from a Arduino UNO/ ATmega328 or better

Step 1 - Upload the SensorLab `hardiron simplecal` Example

We have a simple sketch that will repeatedly read magnetometer data and calculate hard iron offsets

Open up the `Adafruit_SensorLab->calibration->mag_hardiron_simplecal`



Step 2 - Open Serial Port

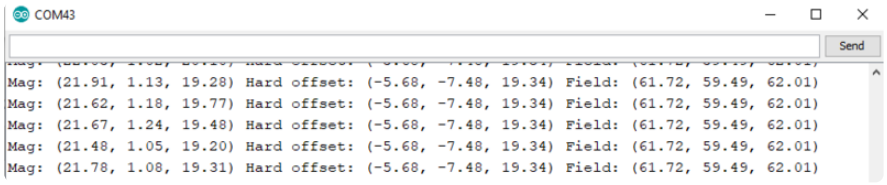
```
Adafruit LIS3MDL test!
Sensor Lab - Magnetometer Calibration!
Looking for a magnetometer
Found addr 0x1C
Found a LIS3MDL IMU
------------------------------------
Sensor:       LIS3MDL
Type:         Magnetic (uT)
Driver Ver:   1
Unique ID:    0
Min Value:    -1600.00
Max Value:    1600.00
Resolution:   0.01
------------------------------------
```

Open the serial port to launch the SensorLab calibration. You should see your magnetometer detected

Spin the board around until you see the last three numbers settle closer to each other and range from 25uT to 65uT (https://adafru.it/IBb)

The middle three numbers are the hard offsets in uTesla.

In this case shown below, the screenshot indicates `x = -5.68, y = 7.48, z = 19.34`



Magnetic Calibration with MotionCal

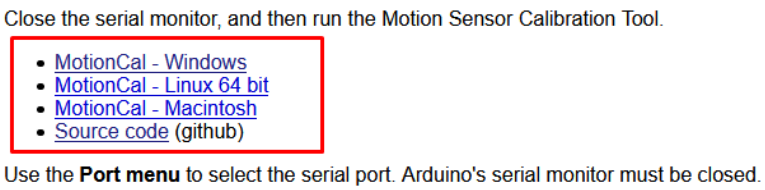
Paul Stoffregen of PJRC (https://adafru.it/IAa) wrote a really awesome cross-platform calibration helper that is great for doing both soft and hard iron magnetometer calibration. What's nice about it is you get a 3D visualization of the magnetometer output and it also tosses outliers and tells you how much spherical coverage you got!

This example runs on chips with at least 64 KB of flash, and will not fit on an UNO (Atmega328) or Leonardo (Atmega32u4) - try the simple calibration instead!

Step 1 - Download MotionCal Software

MotionCal is available for Mac, Windows and Linux, you can download it from clicking here (https://adafru.it/vAH).

Look for this section in the website:

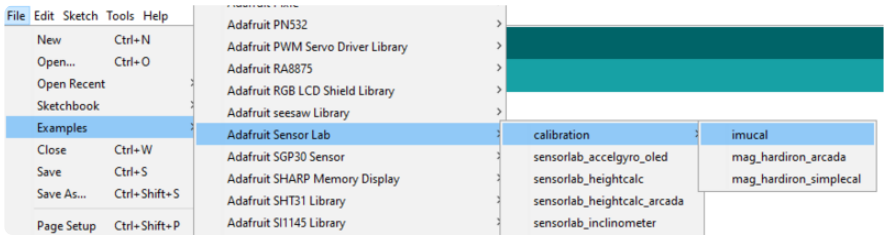


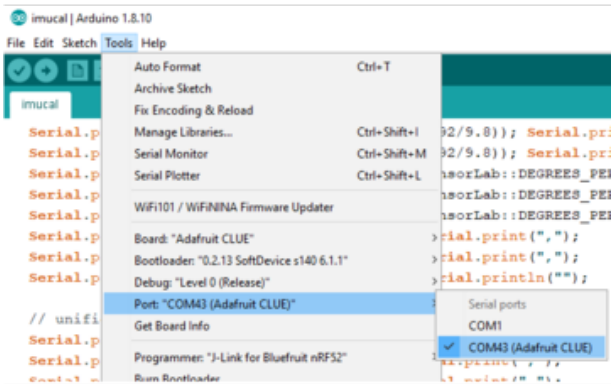
And click the one that matches your computer the best.

Step 2 - Upload the SensorLab `imucal` Example

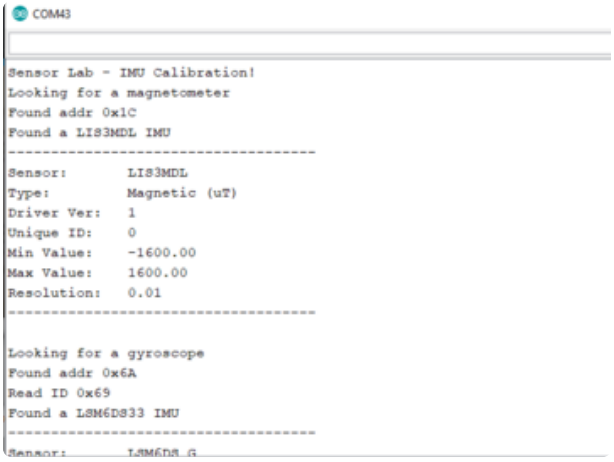
Next we have to tell the microcontroller board to send the magnetometer (and, if there is one, accelerometer and gyroscope) data out over serial in the right format.

Open up the `Adafruit_SensorLab->calibration->imucal`





Select your desired board & port from the Tools menu then click Upload



Open up the serial console, you'll see SensorLab initialization and detection of whatever magnetometer is available. In this case is a LIS3MDL, but any magnetometer can be calibrated!

```
Uni:-0.05,-0.98,10.03,0.0886,-0.1368,-0.1060,-38.03,-16.25,-2.09
Raw:-54,-825,8363,74,-119,-94,-380,-160,-23
Uni:-0.07,-0.99,10.01,0.0809,-0.1300,-0.1034,-38.06,-16.02,-2.32
Raw:-85,-816,8291,66,-112,-92,-382,-163,-20
Uni:-0.10,-0.98,9.92,0.0730,-0.1222,-0.1013,-38.21,-16.33,-2.09
Raw:-49,-821,8345,74,-119,-95,-378,-159,-24
Uni:-0.06,-0.98,9.98,0.0814,-0.1303,-0.1042,-37.88,-15.92,-2.47
Raw:-37,-812,8392,75,-117,-94,-379,-157,-22
Uni:-0.04,-0.97,10.04,0.0823,-0.1286,-0.1032,-37.91,-15.73,-2.28
Raw:-51,-812,8357,56,-94,-85,-377,-163,-27
```

You'll then see a stream of data that looks like:

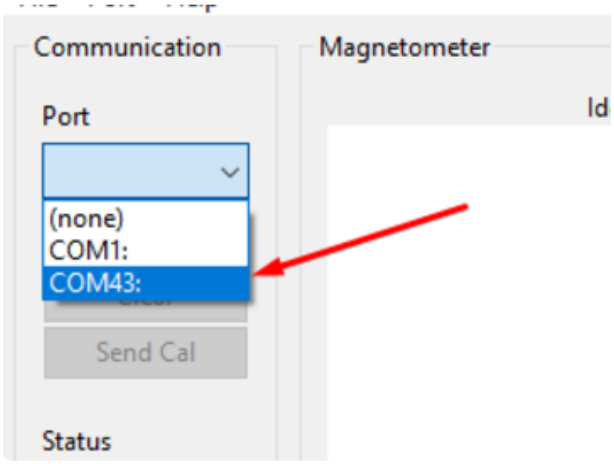
```
Raw:-58,-815,8362,76,-121,-95,-375,-159,-24
Uni:-0.07,-0.98,10.00,0.0832,-0.1327,-0.104
```

The first three numbers are accelerometer data - if you don't have an accelerometer, they will be 0

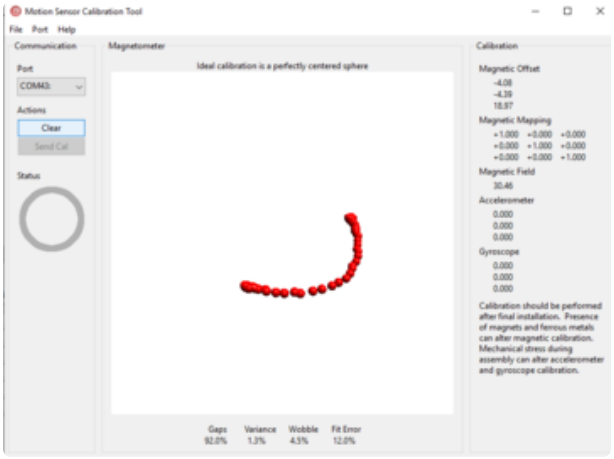
The middle three numbers are gyroscope data - if you don't have an gyroscope, they will be 0

The last three numbers are magnetometer, they should definitely not be zeros!

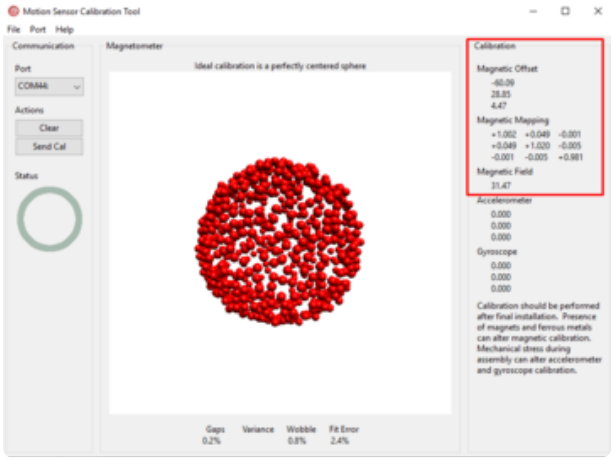
Close the serial port, and launch MotionCal



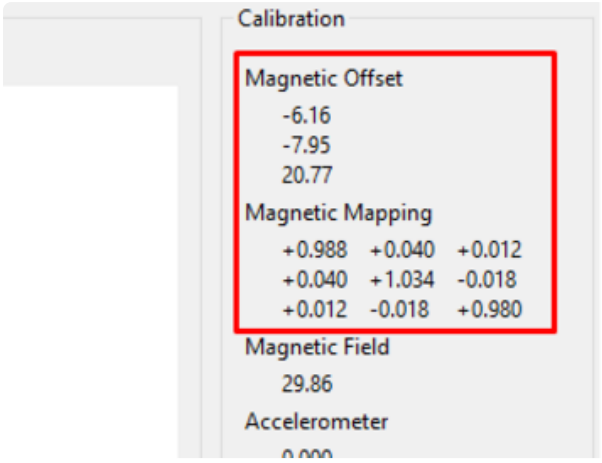
Select the same COM / Serial port you used in Arduino



Twist the board/sensor around. Make sure its not near any strong magnets (unless that's part of the installation)



Keep twisting until you get a complete 'sphere' of red dots. At this point you are calibrated!



In the top right you'll see the hard magnetic offsets at the top, the soft offsets in the middle and the field strength at the bottom.

In this case, the hard iron offsets are `[-6.16, -7.95, 20.77]`

Take a screenshot of this display, so you can refer to these numbers later!

Magnetic Calibration with Jupyter

Jupyter Notebooks are a powerful cross-platform method for analyzing data using Python (https://adafru.it/lBd)

You can definitely use Jupyter to plot, analyze and calibrate your sensor data. This method is the most powerful because you can do plotting and calculations. However,

we assume you already have Jupyter installed (either desktop or thru Anaconda) and have some familiarity with running 'notebook' style Python!

This example runs on chips with at least 64 KB of flash, and will not fit on an UNO (Atmega328) or Leonardo (Atmega32u4) - try the simple calibration instead!

Step 1 - Download Calibration Notebook

The gyro/magnetometer notebook lives in the SensorLab Arduino library, in the `notebooks` folder. (https://adafru.it/lBe)

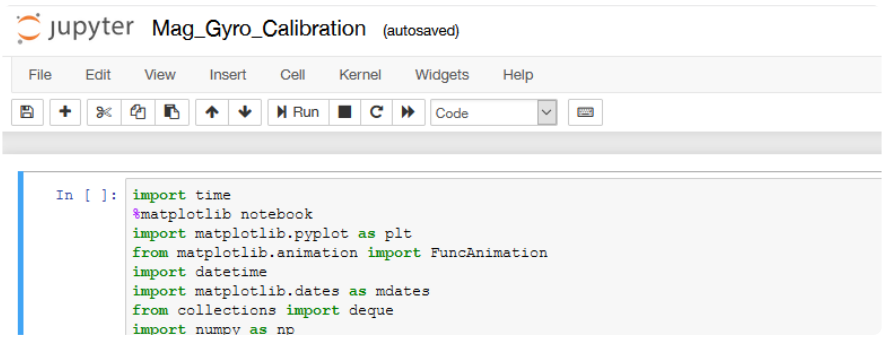
Look for this section in the website:

Click to download the iPython/ Jupyter Notebook

https://adafru.it/lBf

You must open this notebook within Jupyter - you cannot run it direct from github or from the command line as a text file!

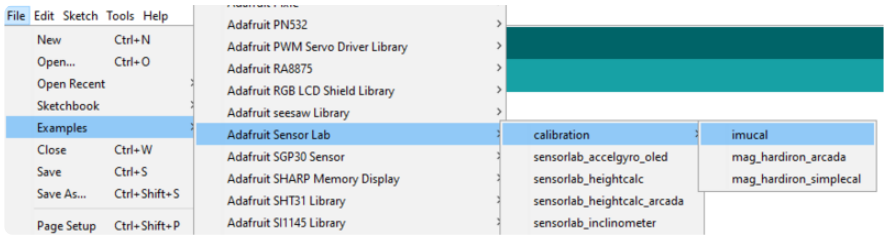
Once open, your browser will look like this:

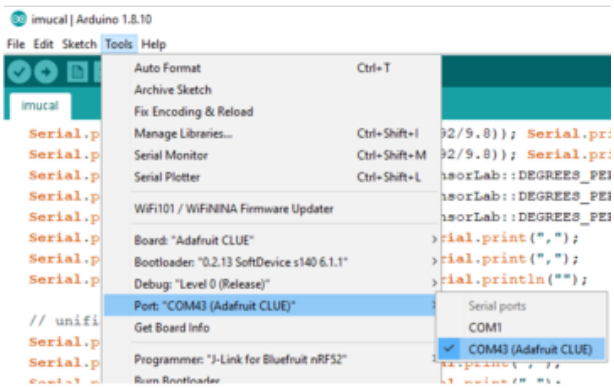


Step 2 - Upload the SensorLab `imucal` Example

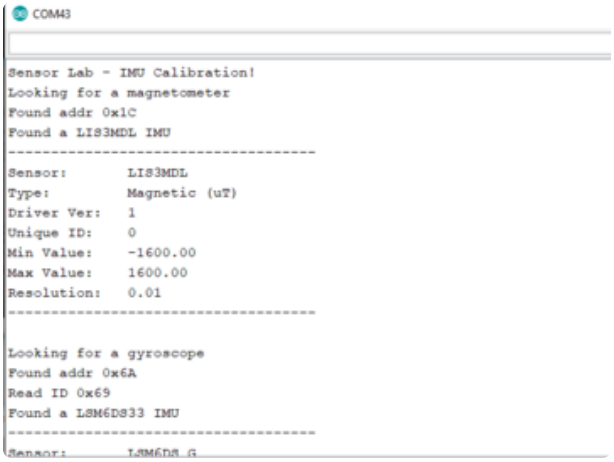
Next we have to tell the microcontroller board to send the magnetometer (and, if there is one, accelerometer and gyroscope) data out over serial in the right format.

Open up the `Adafruit_SensorLab->calibration->imucal`





Select your desired board & port from the Tools menu then click Upload



Open up the serial console, you'll see SensorLab initialization and detection of whatever magnetometer is available. In this case is a LIS3MDL, but any magnetometer can be calibrated!

```
Uni:-0.05,-0.98,10.03,0.0886,-0.1368,-0.1060,-38.03,-16.25,-2.09
Raw:-54,-825,8363,74,-119,-94,-380,-160,-23
Uni:-0.07,-0.99,10.01,0.0809,-0.1300,-0.1034,-38.06,-16.02,-2.32
Raw:-85,-816,8291,66,-112,-92,-382,-163,-20
Uni:-0.10,-0.98,9.92,0.0730,-0.1222,-0.1013,-38.21,-16.33,-2.09
Raw:-49,-821,8345,74,-119,-95,-378,-159,-24
Uni:-0.06,-0.98,9.98,0.0814,-0.1303,-0.1042,-37.88,-15.92,-2.47
Raw:-37,-812,8392,75,-117,-94,-379,-157,-22
Uni:-0.04,-0.97,10.04,0.0823,-0.1286,-0.1032,-37.91,-15.73,-2.28
Raw:-51,-812,8357,56,-94,-85,-377,-163,-27
```

You'll then see a stream of data that looks like:

```
Raw:-58,-815,8362,76,-121,-95,-375,-159,-24
Uni:-0.07,-0.98,10.00,0.0832,-0.1327,-0.104
```

The first three numbers are accelerometer data - if you don't have an accelerometer, they will be 0

The middle three numbers are gyroscope data - if you don't have an gyroscope, they will be 0

The last three numbers are magnetometer, they should definitely not be zeros!

Configure the notebook

```
import time
%matplotlib notebook
import matplotlib.pyplot as plt
from matplotlib.animation import FuncAnimation
import datetime
import matplotlib.dates as mdates
from collections import deque
import numpy as np

import serial
import re

PORT = "COM43"

# How many sensor samples we want to store
HISTORY_SIZE = 2500

# Pause re-sampling the sensor and drawing for INTERVAL s
INTERVAL = 0.01
```

Close the serial port, and go back to Jupyter. In the first cell, find where we define the PORT and change the port to match your serial/COM port. For windows it'll be something like COM4 for Mac/Linux it'll be like /dev/cu.USBSERIAL or something

```
for _ in range(20):
    print(get_imu_data())
```

```
Opened COM43
[0.92, 1.12, -9.45, 0.0728, -0.1165, -0.0988, -8.16, -8.4, 71.02]
None
[0.91, 1.14, -9.46, 0.0767, -0.1251, -0.1016, -8.17, -8.77, 67.48]
None
[0.91, 1.16, -9.46, 0.0779, -0.1222, -0.1006, -7.99, -8.7, 66.97]
None
[0.93, 1.12, -9.46, 0.0817, -0.1261, -0.1014, -8.01, -7.98, 66.0]
None
[0.92, 1.13, -9.45, 0.0845, -0.1303, -0.1034, -8.2, -8.42, 67.25]
None
[0.91, 1.13, -9.46, 0.0562, -0.0991, -0.091, -8.43, -8.18, 66.72]
None
[0.92, 1.15, -9.48, 0.0814, -0.1274, -0.1009, -8.2, -8.55, 66.66]
None
[0.93, 1.11, -9.45, 0.0819, -0.1284, -0.1017, -8.62, -8.33, 66.92]
None
```

Run the first cell so the serial port is set

Then run the second cell, you should see output like this - the serial port is opened and IMU raw data is output as numbers

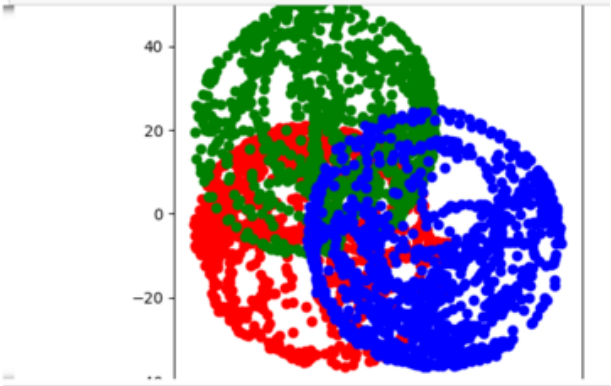
If you get errors or no numbers, hard-reset the board (click the reset button once) then try re-running the cell again.

Magnetometer data capture

```
In [ ]: # Deque for axes
        mag_x = deque(maxlen=HISTORY_SIZE)
        mag_y = deque(maxlen=HISTORY_SIZE)
        mag_z = deque(maxlen=HISTORY_SIZE)

        fig, ax = plt.subplots(1, 1)
        ax.set_aspect(1)

        # close port in case its open
        if serialport:
            try:
                serialport.close()
            except NameError:
                pass
```

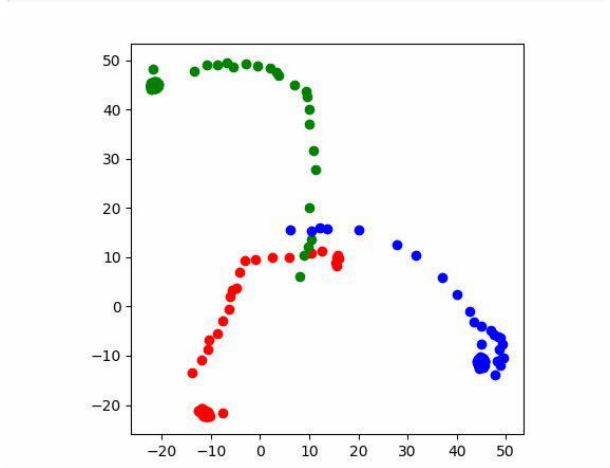
At the next cell we will perform the data capture. Move the board away from any strong magnets and run this cell

```
fig.canvas.mpl_connect('button_press_event', onClick)
anim = FuncAnimation(fig, animate)
```



At the bottom of the cell you should start seeing a live plot of 3 circles - each one represents the X, Y and Z offsets. Keep spinning the board in various directions until you get 3 spheroids

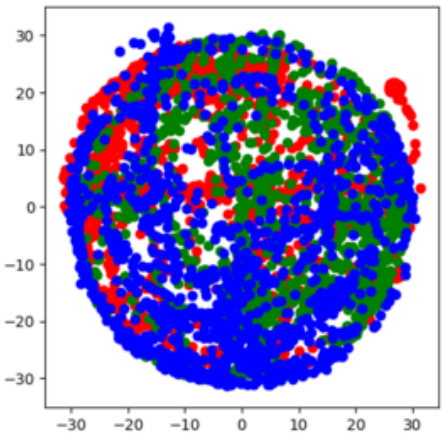
Once you're happy, use the mouse to click on the graph. This will cause the data capture to stop



Run the next cell to perform the analysis. You'll get your X/Y/Z magnetic ranges and the final hard-offset calibration values.

In this case, the calibration is X=-5.21, Y=-7.7 and Z= 20.86

```
X range:  -36.55 26.13
Y range:  -36.38 20.97
Z range:  -9.47 51.18
Final calibration in uTesla: [-5.209999999999999, -7.705000000000002, 20.855]
```



You'll also see the results of removing the offset, this should be 3 nearly-perfectly-superimposed circles with centers at 0,0

Calibration with Raspberry Pi using Blinka

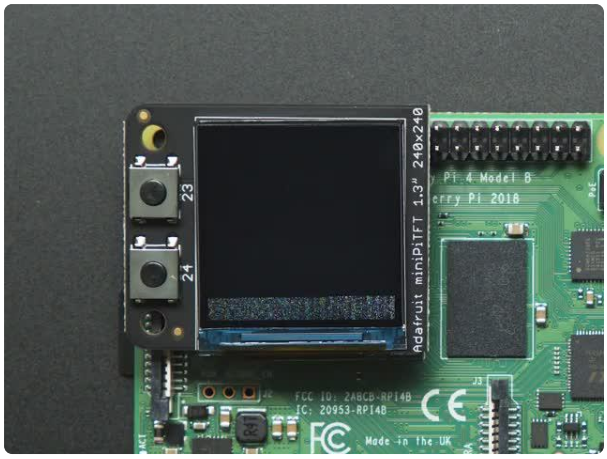
You can easily calibrate a sensor using the Raspberry Pi using our calibration script. It runs from the command line using Blinka. This page assumes you have already set up Blinka on the Raspberry Pi, but if not, be sure to follow our CircuitPython on Linux and Raspberry Pi (https://adafru.it/BSN) guide.

The easiest way to connect a 9-DoF sensor to the Raspberry Pi is to use a STEMMA QT/Qwiic (https://adafru.it/JqB) connector which many of our sensors include.

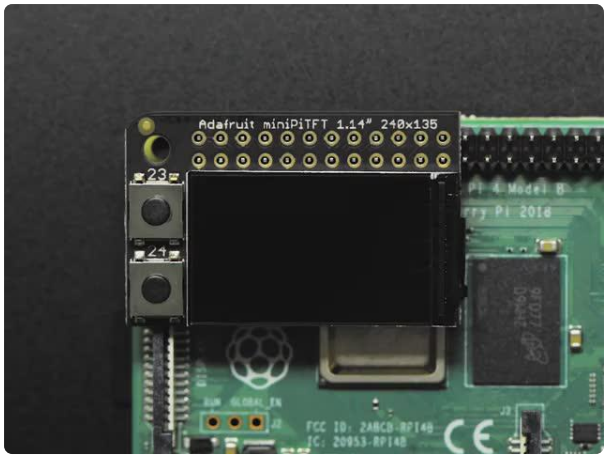
Using a STEMMA QT Cable

To add a STEMMA connector to your Pi, the easiest way is to use one of our Raspberry Pi add ons that feature the STEMMA QT connector such as a display:

Adafruit 2.23" Monochrome OLED Bonnet for Raspberry Pi
If you're looking for a bright, readable OLED display for a Raspberry Pi (most likely a https://www.adafruit.com/product/4567

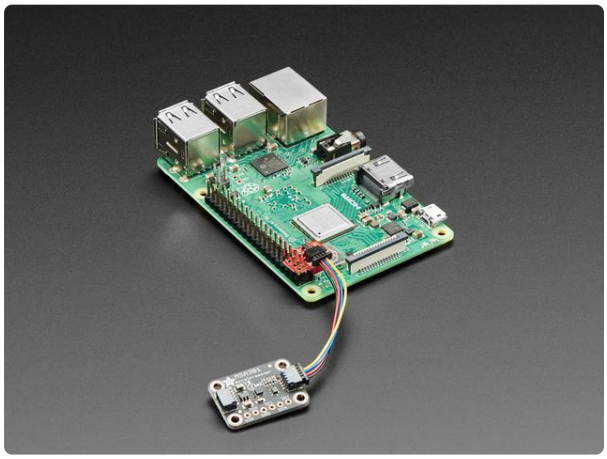


[Adafruit Mini PiTFT 1.3" - 240x240 TFT Add-on for Raspberry Pi]
If you're looking for the most compact li'l color display for a Raspberry Pi (most likely a
https://www.adafruit.com/product/4484



[Adafruit Mini PiTFT - 135x240 Color TFT Add-on for Raspberry Pi]
If you're looking for the most compact li'l color display for a Raspberry Pi (most likely a
https://www.adafruit.com/product/4393

Or if you'd prefer something more minimal, you could even use a STEMMA QT SHIM:



[SparkFun Qwiic or Stemma QT SHIM for Raspberry Pi / SBC]
The SparkFun Qwiic or Stemma QT SHIM for Raspberry Pi is a small, easily removable breakout that easily adds a 4-pin JST...
https://www.adafruit.com/product/4463

You'll need a STEMMA cable as well.

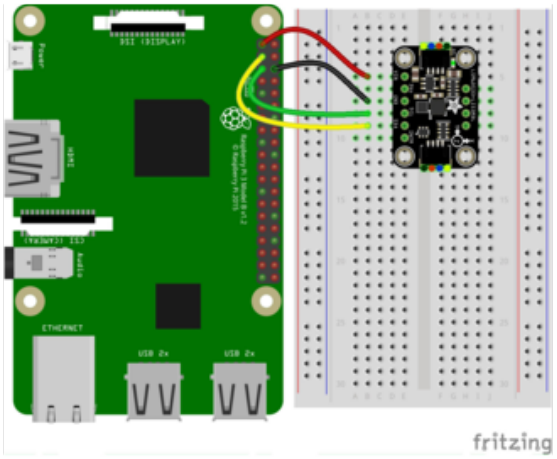
STEMMA QT / Qwiic JST SH 4-pin Cable - 100mm Long

This 4-wire cable is a little over 100mm / 4" long and fitted with JST-SH female 4-pin connectors on both ends. Compared with the chunkier JST-PH these are 1mm pitch instead of...

https://www.adafruit.com/product/4210

Wiring the Sensor

If your sensor does not have a STEMMA connector, you could just wire it up directly to the Pi. The sensors typically have an I2C interface and connecting them up is easy. Here's an example using the LIS3MDL+LSM6DS33 sensor:



- Pi 3V to sensor VCC (red wire)
- Pi GND to sensor GND (black wire)
- Pi SCL to sensor SCL (green wire)
- Pi SDA to sensor SDA (yellow wire)

Download Fritzing Object

https://adafru.it/LQb

For more details on wiring up other sensors, be sure to check out the Python page in our ST 9-DoF Combo Breakouts and Wings (https://adafru.it/LQc) guide.

Install the libraries

The calibration script uses the the Adafruit_CircuitPython_LIS3MDL and Adafruit_CircuitPython_LSM6DS libraries. To install, run the following commands:

- `pip3 install adafruit-circuitpython-lis3mdl`

- `pip3 install adafruit-circuitpython-lsm6ds`

If your default Python is version 3 you may need to run 'pip' instead. Just make sure you aren't trying to use CircuitPython on Python 2.x, it isn't supported! On some boards, you may need to add `sudo` before `pip3`.

If that complains about pip3 not being installed, then run this first to install it:

- `sudo apt-get install python3-pip`

Full Example Code

Here's the full script to run the calibration. Go ahead and save that to your Pi as 9dof _calibration.py.

```
import threading
import time
import board
import busio
from adafruit_lsm6ds import LSM6DSOX
from adafruit_lis3mdl import LIS3MDL

SAMPLE_SIZE = 500


class KeyListener:
    """Object for listening for input in a separate thread"""

    def __init__(self):
        self._input_key = None
        self._listener_thread = None

    def _key_listener(self):
        while True:
            self._input_key = input()

    def start(self):
        """Start Listening"""
        if self._listener_thread is None:
            self._listener_thread = threading.Thread(
                target=self._key_listener, daemon=True
            )
        if not self._listener_thread.is_alive():
            self._listener_thread.start()

    def stop(self):
        """Stop Listening"""
        if self._listener_thread is not None and self._listener_thread.is_alive():
            self._listener_thread.join()

    @property
    def pressed(self):
        "Return whether enter was pressed since last checked" ""
        result = False
        if self._input_key is not None:
            self._input_key = None
```

```
            result = True
        return result


def main():
    # pylint: disable=too-many-locals, too-many-statements
    i2c = busio.I2C(board.SCL, board.SDA)

    gyro_accel = LSM6DSOX(i2c)
    magnetometer = LIS3MDL(i2c)
    key_listener = KeyListener()
    key_listener.start()

    ############################
    # Magnetometer Calibration #
    ############################

    print("Magnetometer Calibration")
    print("Start moving the board in all directions")
    print("When the magnetic Hard Offset values stop")
    print("changing, press ENTER to go to the next step")
    print("Press ENTER to continue...")
    while not key_listener.pressed:
        pass

    mag_x, mag_y, mag_z = magnetometer.magnetic
    min_x = max_x = mag_x
    min_y = max_y = mag_y
    min_z = max_z = mag_z

    while not key_listener.pressed:
        mag_x, mag_y, mag_z = magnetometer.magnetic

        print(
            "Magnetometer: X: {0:8.2f}, Y:{1:8.2f}, Z:{2:8.2f} uT".format(
                mag_x, mag_y, mag_z
            )
        )

        min_x = min(min_x, mag_x)
        min_y = min(min_y, mag_y)
        min_z = min(min_z, mag_z)

        max_x = max(max_x, mag_x)
        max_y = max(max_y, mag_y)
        max_z = max(max_z, mag_z)

        offset_x = (max_x + min_x) / 2
        offset_y = (max_y + min_y) / 2
        offset_z = (max_z + min_z) / 2

        field_x = (max_x - min_x) / 2
        field_y = (max_y - min_y) / 2
        field_z = (max_z - min_z) / 2

        print(
            "Hard Offset:  X: {0:8.2f}, Y:{1:8.2f}, Z:{2:8.2f} uT".format(
                offset_x, offset_y, offset_z
            )
        )
        print(
            "Field:        X: {0:8.2f}, Y:{1:8.2f}, Z:{2:8.2f} uT".format(
                field_x, field_y, field_z
            )
        )
        print("")
        time.sleep(0.01)

    mag_calibration = (offset_x, offset_y, offset_z)
```

```
    print(
        "Final Magnetometer Calibration: X: {0:8.2f}, Y:{1:8.2f}, Z:{2:8.2f}
uT".format(
            offset_x, offset_y, offset_z
        )
    )

    ##########################
    # Gyroscope Calibration #
    ##########################

    gyro_x, gyro_y, gyro_z = gyro_accel.gyro
    min_x = max_x = gyro_x
    min_y = max_y = gyro_y
    min_z = max_z = gyro_z

    print("")
    print("")
    print("Gyro Calibration")
    print("Place your gyro on a FLAT stable surface.")
    print("Press ENTER to continue...")
    while not key_listener.pressed:
        pass

    for _ in range(SAMPLE_SIZE):
        gyro_x, gyro_y, gyro_z = gyro_accel.gyro

        print(
            "Gyroscope: X: {0:8.2f}, Y:{1:8.2f}, Z:{2:8.2f} rad/s".format(
                gyro_x, gyro_y, gyro_z
            )
        )

        min_x = min(min_x, gyro_x)
        min_y = min(min_y, gyro_y)
        min_z = min(min_z, gyro_z)

        max_x = max(max_x, gyro_x)
        max_y = max(max_y, gyro_y)
        max_z = max(max_z, gyro_z)

        offset_x = (max_x + min_x) / 2
        offset_y = (max_y + min_y) / 2
        offset_z = (max_z + min_z) / 2

        noise_x = max_x - min_x
        noise_y = max_y - min_y
        noise_z = max_z - min_z

        print(
            "Zero Rate Offset:  X: {0:8.2f}, Y:{1:8.2f}, Z:{2:8.2f} rad/s".format(
                offset_x, offset_y, offset_z
            )
        )
        print(
            "Rad/s Noise:       X: {0:8.2f}, Y:{1:8.2f}, Z:{2:8.2f} rad/s".format(
                noise_x, noise_y, noise_z
            )
        )
        print("")

    gyro_calibration = (offset_x, offset_y, offset_z)
    print(
        "Final Zero Rate Offset: X: {0:8.2f}, Y:{1:8.2f}, Z:{2:8.2f} rad/s".format(
            offset_x, offset_y, offset_z
        )
    )
    print("")
```

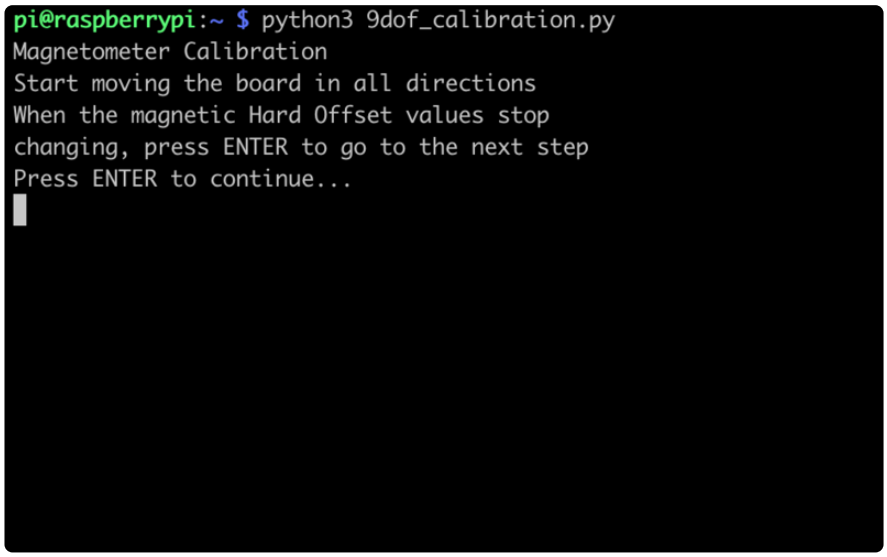
```
print("------------------------------------------------------------------------")
    print("Final Magnetometer Calibration Values: ", mag_calibration)
    print("Final Gyro Calibration Values: ", gyro_calibration)


if __name__ == "__main__":
    main()
```

Using the Script

Start the script by typing:

- `python3 9dof_calibration.py`

The script will first start by letting you know that it wants to calibrate the magnetometer.



Press ENTER to continue.

It will start measuring the magnetometer and scrolling the values. Start moving the sensor around in every direction.

```
Field:        X:    36.12, Y:   64.88, Z:   68.17 uT

Magnetometer: X:     4.44, Y:  -20.33, Z:  -54.06 uT
Hard Offset:  X:     3.10, Y:   -2.16, Z:   -1.78 uT
Field:        X:    36.12, Y:   64.88, Z:   68.17 uT

Magnetometer: X:     4.57, Y:  -20.34, Z:  -54.21 uT
Hard Offset:  X:     3.10, Y:   -2.16, Z:   -1.78 uT
Field:        X:    36.12, Y:   64.88, Z:   68.17 uT

Magnetometer: X:     4.55, Y:  -20.34, Z:  -53.96 uT
Hard Offset:  X:     3.10, Y:   -2.16, Z:   -1.78 uT
Field:        X:    36.12, Y:   64.88, Z:   68.17 uT

Magnetometer: X:     4.69, Y:  -19.92, Z:  -54.25 uT
Hard Offset:  X:     3.10, Y:   -2.16, Z:   -1.78 uT
Field:        X:    36.12, Y:   64.88, Z:   68.17 uT
```

The magnetic Hard Offset values should stop changing after a bit. After it does, press ENTER again.

```
Magnetometer: X:     7.91, Y:    0.89, Z:  -53.73 uT
Hard Offset:  X:     7.63, Y:    0.91, Z:  -53.17 uT
Field:        X:     0.75, Y:    0.52, Z:    0.91 uT

Magnetometer: X:     7.50, Y:    0.80, Z:  -53.42 uT
Hard Offset:  X:     7.63, Y:    0.91, Z:  -53.17 uT
Field:        X:     0.75, Y:    0.52, Z:    0.91 uT

Magnetometer: X:     7.57, Y:    0.67, Z:  -53.00 uT
Hard Offset:  X:     7.63, Y:    0.91, Z:  -53.17 uT
Field:        X:     0.75, Y:    0.52, Z:    0.91 uT

Magnetometer: X:     7.35, Y:    0.77, Z:  -52.95 uT
Hard Offset:  X:     7.63, Y:    0.91, Z:  -53.17 uT
Field:        X:     0.75, Y:    0.52, Z:    0.91 uT


Final Magnetometer Calibration: X:     7.63, Y:    0.91, Z:  -53.17 uT


Gyro Calibration
Place your gyro on a FLAT stable surface.
Press ENTER to continue...
```

The next step is to calibrate the Gyroscope. Place the sensor on a flat surface like a desk or table. Once it is lying still, press ENTER.

```
Gyroscope: X:     0.02, Y:   -0.01, Z:   -0.00 rad/s
Zero Rate Offset:  X:     0.00, Y:    0.01, Z:   -0.02 rad/s
Rad/s Noise:       X:     0.05, Y:    0.05, Z:    0.03 rad/s

Gyroscope: X:     0.02, Y:   -0.01, Z:   -0.00 rad/s
Zero Rate Offset:  X:     0.00, Y:    0.01, Z:   -0.02 rad/s
Rad/s Noise:       X:     0.05, Y:    0.05, Z:    0.03 rad/s

Gyroscope: X:     0.02, Y:   -0.01, Z:   -0.00 rad/s
Zero Rate Offset:  X:     0.00, Y:    0.01, Z:   -0.02 rad/s
Rad/s Noise:       X:     0.05, Y:    0.05, Z:    0.03 rad/s

Gyroscope: X:     0.02, Y:   -0.01, Z:   -0.00 rad/s
Zero Rate Offset:  X:     0.00, Y:    0.01, Z:   -0.02 rad/s
Rad/s Noise:       X:     0.05, Y:    0.05, Z:    0.03 rad/s

Gyroscope: X:     0.02, Y:   -0.01, Z:   -0.00 rad/s
Zero Rate Offset:  X:     0.00, Y:    0.01, Z:   -0.02 rad/s
Rad/s Noise:       X:     0.05, Y:    0.05, Z:    0.03 rad/s

Final Zero Rate Offset: X:     0.00, Y:    0.01, Z:   -0.02 rad/s

-------------------------------------------------------------------------
Final Magnetometer Calibration Values:  (3.098509207833967, -2.1631102016954067, -1.7757965507161657)
Final Gyro Calibration Values:  (0.000152716309549503, 0.013897184169004848, -0.015577063574049339)
```

It will run through the numbers and then give you your final calibration values.

Using a different Sensor

If you are using a different sensor, you will need to install the appropriate library and then make a few changes to the script. The easiest way to find the correct library for your sensor is to look at the associated learn guide for that sensor. This can usually be found on the product page or by searching the Adafruit Learn System.

Here are the changes you will need to make depending on the sensor you have:

LIS3MDL+LSM6DSOX

No changes are necessary for this sensor.

LIS3MDL+LSM6DS33

You will need to change the import line from

```
from adafruit_lsm6ds import LSM6DSOX
```

to

```
from adafruit_lsm6ds import LSM6DS33
```

You will also need to change the declaration line from

```
gyro_accel = LSM6DSOX(i2c)
```

to

```
gyro_accel = LSM6DS33(i2c)
```

LSM9DS1

You will need to change the import lines from

```
from adafruit_lsm6ds import LSM6DSOX
from adafruit_lis3mdl import LIS3MDL
```

to

```
from adafruit_lsm9ds1 import LSM9DS1_I2C
```

You will also need to change the declaration lines from

```
gyro_accel = LSM6DSOX(i2c)
magnetometer = LIS3MDL(i2c)
```

to

```
magnetometer = gyro_accel = LSM9DS1_I2C(i2c)
```

FXOS8700 + FXAS21002

You will need to change the import lines from

```
from adafruit_lsm6ds import LSM6DSOX
from adafruit_lis3mdl import LIS3MDL
```

to

```
from adafruit_fxos8700 import FXOS8700
from adafruit_fxas21002c import FXAS21002C
```

You will also need to change the declaration lines from

```
gyro_accel = LSM6DSOX(i2c)
magnetometer = LIS3MDL(i2c)
```

to

```
gyro_accel = FXAS21002C(i2c)
magnetometer = FXOS8700(i2c)
```

Also, change any instance of `gyro_accel.gyro` to `gyro_accel.gyroscope` and any instance of `magnetometer.magnetic` to `magnetometer.magnetometer`.