QT Py S2 Round Display Compass

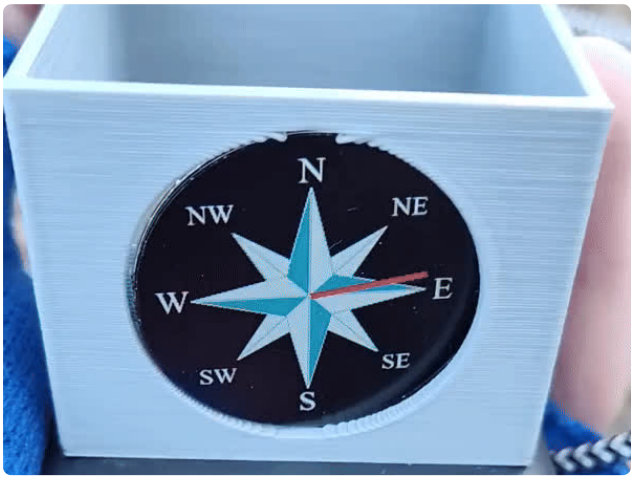
Created by Tim C

https://learn.adafruit.com/qt-py-s2-round-display-compass

Last updated on 2025-02-28 09:28:29 AM EST

Table of Contents

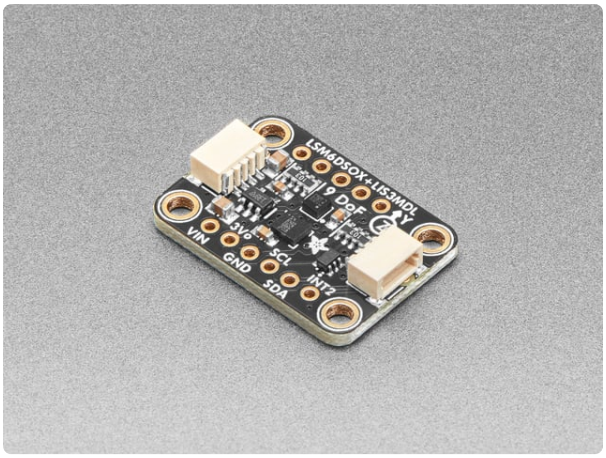
Overview



The GC9A01A round displays (http://adafru.it/6178) are great for displaying a compass face. This project uses a magnetometer & accelerometer/gryo breakout to obtain the cardinal direction, then plot it onto the display with a red pointer needle overlaid on top of a beautiful compass rose.

The EYESPI ribbon and STEMMA QT cable connections make this project a breeze to assemble with only minimal soldering.

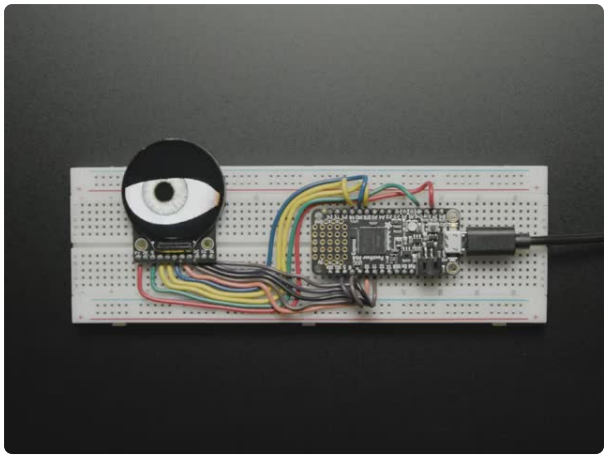
Parts



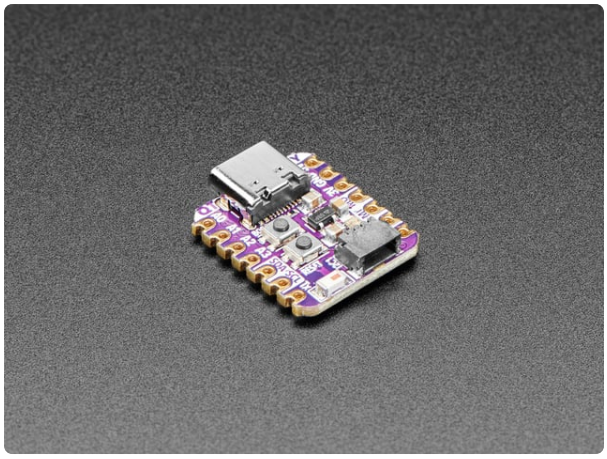
Adafruit LSM6DSOX + LIS3MDL - Precision 9 DoF IMU

Add high-quality motion, direction, and orientation sensing to your Arduino project with this all-in-one 9 Degree of Freedom (9-DoF) sensor with sensors from ST. This little...

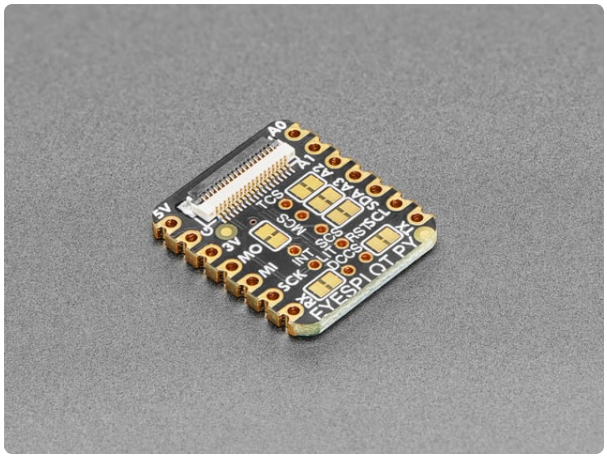
https://www.adafruit.com/product/4517



[Adafruit 1.28" 240x240 Round TFT LCD Display with MicroSD]
'Round these parts we enjoy unusually-shaped displays. And this one certainly fits the description - it's a 1.28" diagonal TFT that comes in a round shape and contains a...
https://www.adafruit.com/product/6178

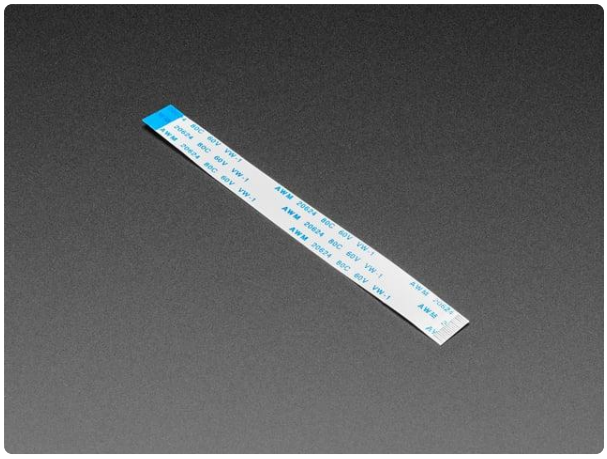


[Adafruit QT Py ESP32-S2 WiFi Dev Board with STEMMA QT]
What has your favorite Espressif WiFi microcontroller, comes with our favorite connector - the STEMMA QT, a chainable I2C port, and has...
https://www.adafruit.com/product/5325



[Adafruit EYESPI BFF for QT Py or Xiao - 18 Pin FPC Connector]
Our QT Py boards are a great way to make very small microcontroller projects that pack a ton of power - and now we have a way for you to add a small, colorful, and bright display to...
https://www.adafruit.com/product/5772

[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



EYESPI Cable - 18 Pin 100mm long Flex PCB (FPC) A-B type

Connect this to that when a 18-pin FPC connector is needed. This 25 cm long cable is made of a flexible PCB. It's A-B style which means that pin one on one side will match...

https://www.adafruit.com/product/5239

1 x USB Type A to Type C Cable

https://www.adafruit.com/product/4474

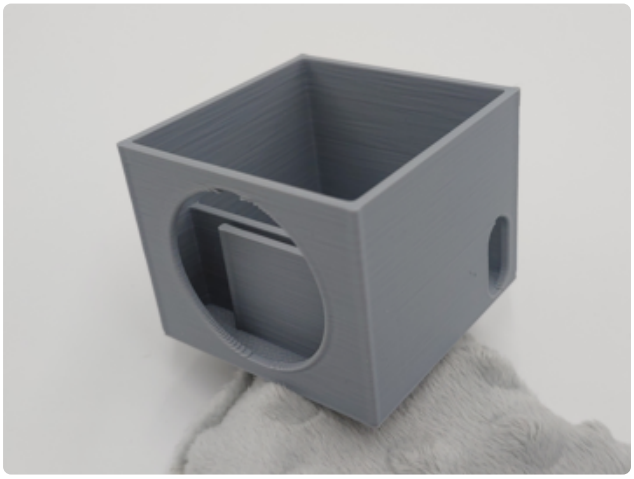
Approx 1 meter / 3 ft long

---

1 x Mounting Putty / Sticky Tack

Pea sized glob

3D Printing



3D Printed Box

STL files for 3D printing will need to be oriented for print using either FDM or SLS machines. Parts were built with PLA filament.

STL file for printing may be downloaded using the link below.

The box was sliced with CURA using the settings below

PLA filament 205c extruder
0.35 layer height
No supports

round_eyespi_display_box.stl

https://adafru.it/1aev

Original Design File

The box was designed using OpenScad (https://adafru.it/XD4), find the source code for it below.

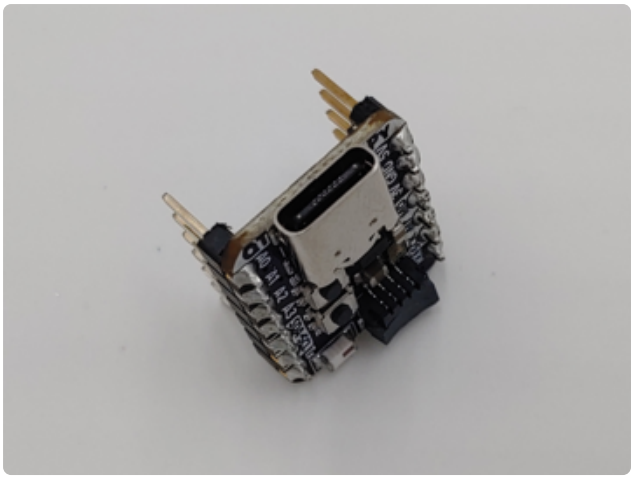
```
wall_thickness = 1.6;
short_wall_height = 25;

difference(){

    cube([50+(wall_thickness*2),50+(wall_thickness*2),42+1], center=true);
    translate([0,0,1]){
        cube([50,50,42+1], center=true);
    }

    translate([0,20,0])
    rotate([90,0,0]){
        cylinder(r=36/2, h=30, center=true, $fn=80);
    }

    translate([
        -10,
        -(50+(wall_thickness*2))/2 + 9/2 + 4,
        -(42+1)/2 + 18/2 + 1]){

        hull(){
            translate([0,0,-4.5])
            rotate([0,90,])
            cylinder(r=9/2, h=40, center=true, , $fn=30);

            translate([0,0,4.5])
            rotate([0,90,0])
            cylinder(r=9/2, h=40, center=true, $fn=30);
        }
        //cube([40,9,18], center=true);
    }
}


translate([-(50+(wall_thickness*2))/2 + 40/2,
    ((50+(wall_thickness*2))/2) - wall_thickness/2 - 8,
    -(43/2)+(short_wall_height/2)])
cube([40,wall_thickness,short_wall_height], center=true);


translate([(50+(wall_thickness*2))/2 - 40/2,
    ((50+(wall_thickness*2))/2) - wall_thickness/2 - 8 - 4,
    -(43/2)+(short_wall_height/2)])
cube([40,wall_thickness,short_wall_height], center=true);


/*
translate([0,-5,0])
cube([22, 32, 18], center=true);
*/
```

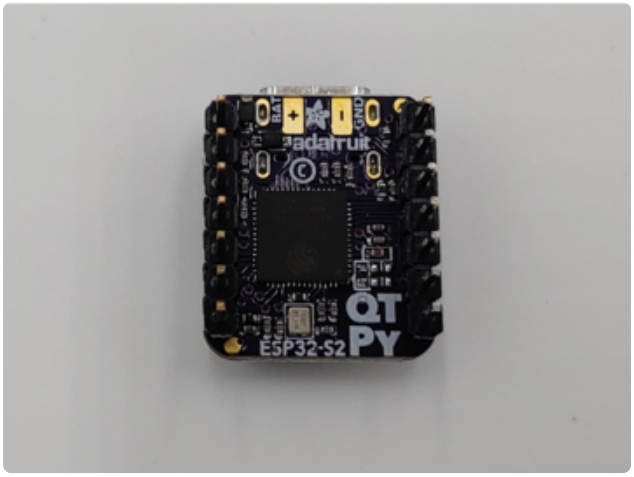
Assembly

The only soldering required for this project are the headers on the QT Py and EYESPI BFF. After that you just need a few quick connections and to fit everything into the box.

Solder Headers

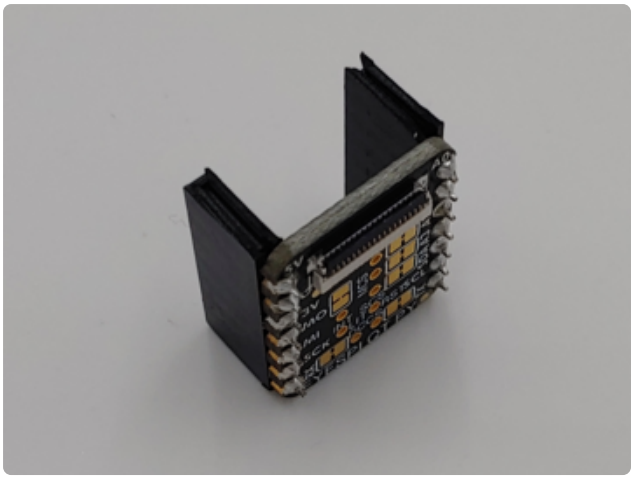
First solder the following header types to the BFF and QT Py. Make sure that you have the appropriate header paired with the correct board and that the header is facing the proper direction relative to the board. We have a learn guide (https://adafru.it/1a1h) that covers soldering headers if you're new to the process or want a refresher.

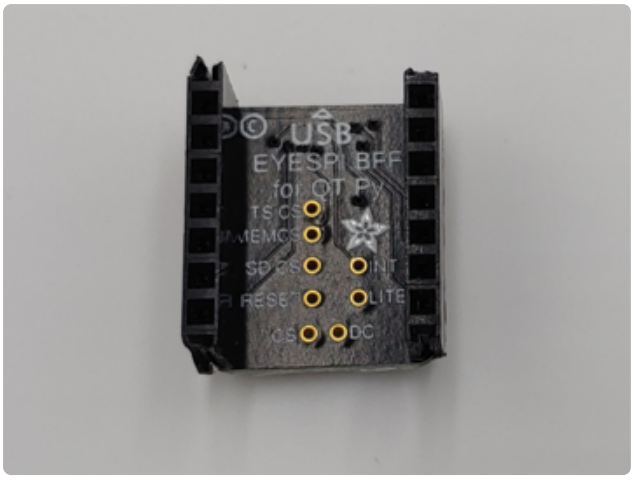


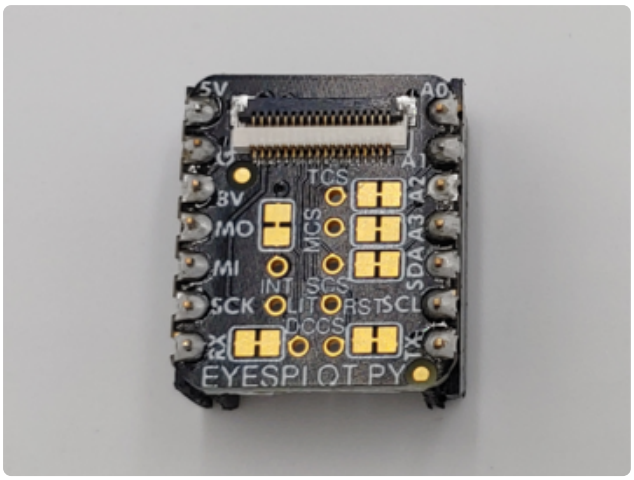


QT Py ESP32-S2

Standard male header with the pins going down on the side opposite the USB-C connector.



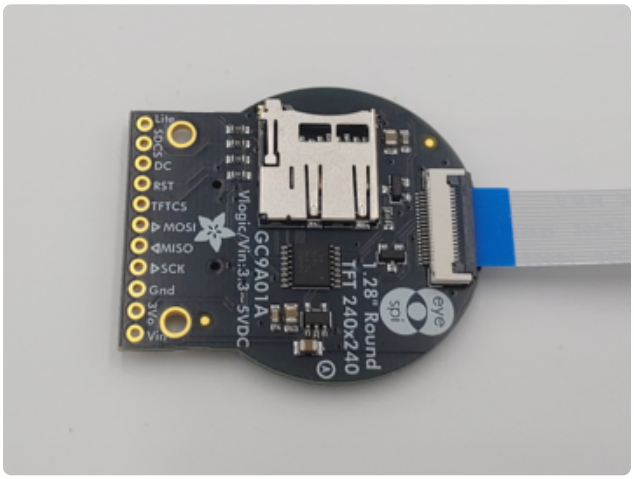




EYESPI BFF

Standard female header with the female header on the opposite side of the board from the EYESPI ribbon cable connector.

Connect Display

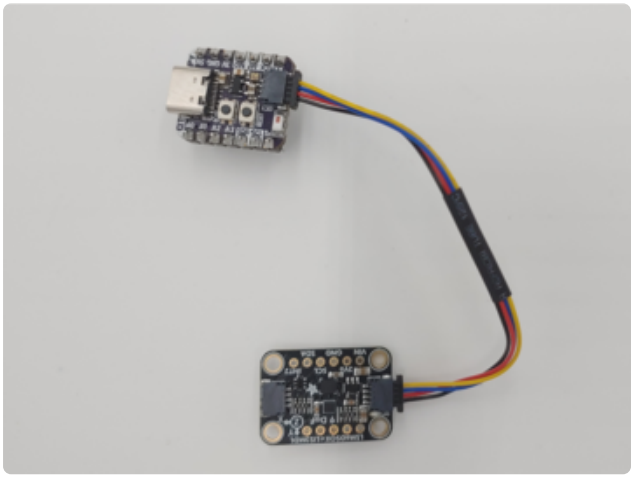


First you must carefully flip up the dark grey flap that secures the ribbon. Gently slip your fingernail or a small flat head screw driver under it, then carefully pry upward. The flap is delicate so be careful, it should not require much force once you've got something up underneath it.

Connect one end of the EYESPI ribbon cable to the EYESPI BFF.
Connect the other end of the EYESPI ribbon cable to the GC9A01A breakout.

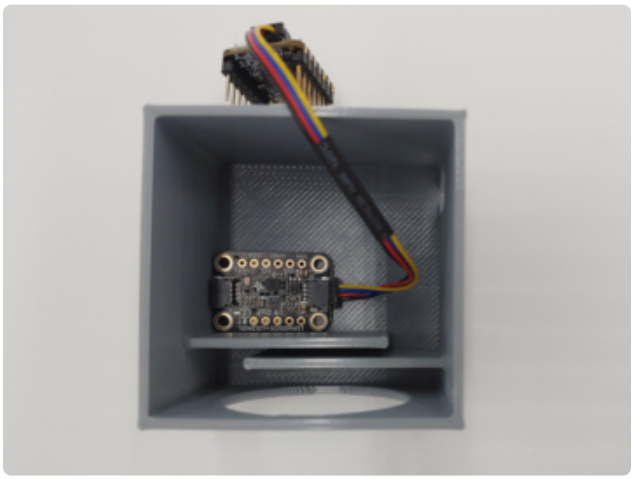
Both connections should have the blue tab facing upward. Insert the ribbon cable, then push down the flap to secure it.

Connect LSM6DSOX + LIS3MDL Breakout



Connect one end of the STEMMA QT cable to the QT Py ESP32-S2, and the other end to the LSM6DSOX + LIS3MDL breakout. Be sure to use the same STEMMA QT connector that is pictured on the LSM6DSOX + LIS3MDL breakout. The wire should be coming out of the right side, when the text on the board is facing away from you.

Mount LSM6DSOX + LIS3MDL Breakout Into Box



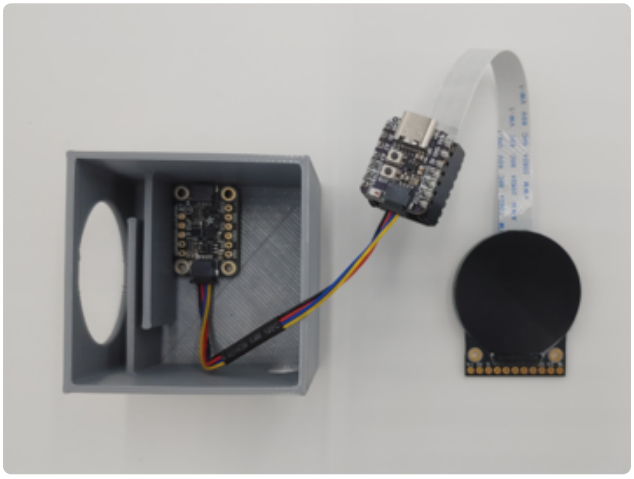
Take a pea sized glob of mounting putty and smash it around in your fingers until it's tacky, then squish it onto the back of the LSM6DSOX + LIS3MDL breakout. Place the breakout into the front left corner of the back cavity of the box behind offset walls that make the display slot.

The text on the breakout should be upside down or facing away from you if viewed from the top of the box with the circle cutout in the front as shown in the photo.

Make sure the edges of the breakout are snug against the walls of the box, it needs to be in the proper orientation for the compass code to function.

Once it's in place press it down firmly to stick it to the bottom of the box with the mounting putty that you put on it's bottom side.

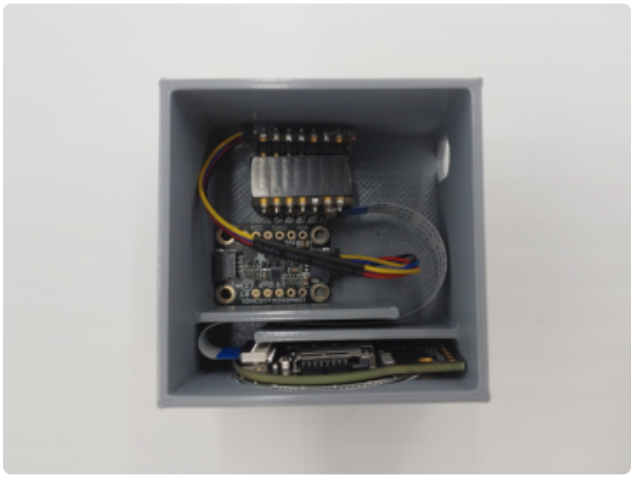
Stack QT Py and EYESPI BFF



Plug in the male header pins on the QT Py to the female headers on the EYESPI BFF. Be sure they are in the proper orientation relative to each other! You can check the pin labels on the silk screen as well as the USB pointer label on the BFF. They should also match the photos shown here.

Do this without the USB cable plugged in so they are unpowered.

Place Into Box



Carefully place the display into its slot at the front of the box, feeding the EYESPI ribbon cable down into its channel made by the two offset walls.

Set the QT Py and BFF stack into the open space behind the offset walls with the USB C connector pointing towards the USB Cable hole cut out of the side of the box. Plug in your desired USB C cable to power the project, feeding it through the hole.

Calibrate

Project Setup

Are you new to using CircuitPython? No worries, there is a full getting-started guide here (https://adafru.it/cpy-welcome).

Plug the device into your computer with a known good USB cable (not a charge-only cable). The device will appear to your computer in File Explorer or Finder (depending on your operating system) as a flash drive named **CIRCUITPY**. If the drive does not appear, you can install CircuitPython (https://adafru.it/Amd) on your device and then return here.

Download the project files with the Download Project Bundle button below. Unzip the file and copy/paste the **code.py** and other project files to your **CIRCUITPY** drive using File Explorer or Finder (depending on your operating system).

Run Calibrate Script

In order for the compass function most effectively you must obtain some calibration values from the sensor breakout by running the calibration script on this page. The script is included in the project bundle with the name **calibrate.py**.

To run it, connect to the serial console (for example in the Mu program Serial window), press Ctrl-C on your keyboard to enter the REPL. Then type `import calibrate` and press enter. Example output is shown below.

Watch the serial output and follow the directions shown there. The script will first prompt you to set the sensor down on a flat surface and leave it perfectly still. After it takes several readings it will then ask you to start moving the sensor around. Pick up the box and carefully wave it around in a figure eight motion twisting and turning it as you go. The goal is to move the sensor into every possible orientation while the calibration code keeps track of the extremes of the values in each axis.

Once the script is finish select the calibration values printed at the end and copy them. They need to be pasted into the **code.py** script of the compass on the next page.

```
&gt;&gt;&gt; import calibrate
Preparing gyroscope calibration. Keep board perfectly still on flat surface.
Starting gyroscope calibration..

Uncalibrated gyro:  (-0.00473421, -0.00290161, 0.00122173)
Calibrated gyro:  (-0.00946841, -0.00580322, 0.00244346)
...

Gyroscope calibrated!
Preparing magnetometer calibration. Move board around in 3D space.
```

```
Starting magnetometer calibration..

Uncalibrated: -56.3578 29.0558 -37.3283
Calibr:   -1.0 1.0 1.0
MAG_MIN = [-56.3578, 28.1935, -37.8398]
MAG_MAX = [-55.9047, 29.0558, -37.3283]
...

MAG_MIN = [-75.1973, -22.5665, -34.5221]
MAG_MAX = [-1.2131, 68.1379, 20.8126]
GYRO_CAL = [-0.0038, -0.0026, -0.0011]
```

The code for the **calibrate.py** script is shown below.

```
# SPDX-FileCopyrightText: 2025 Tim Cocks for Adafruit Industries
# SPDX-License-Identifier: MIT
#
# Adapted from Liz Clark's calibrate.py in the QualiaS3 Compass learn guide
# https://learn.adafruit.com/qualia-s3-compass/code-the-compass
#
# Which was adapted from Gamblor21's calibrate.py in the
Gamblor21_CircuitPython_AHRS library
# https://github.com/gamblor21/Gamblor21_CircuitPython_AHRS/blob/master/examples/
calibrate.py
#
# Gyro will be calibrated first, followed by magnetometer
# Keep the board still for gyro, move around for magnetometer

import time

import board
from adafruit_lsm6ds.lsm6dsox import LSM6DSOX
import adafruit_lis3mdl


i2c = board.STEMMA_I2C()
accel_gyro = LSM6DSOX(i2c)
magnetometer = adafruit_lis3mdl.LIS3MDL(i2c)
MAG_MIN = [1000, 1000, 1000]
MAG_MAX = [-1000, -1000, -1000]

def map_range(x, in_min, in_max, out_min, out_max):
    """
    Maps a number from one range to another.
    :return: Returns value mapped to new range
    :rtype: float
    """
    mapped = (x - in_min) * (out_max - out_min) / (in_max - in_min) + out_min
    if out_min <= out_max:
        return max(min(mapped, out_max), out_min)

    return min(max(mapped, out_max), out_min)

def calibrate_gyro():
    """
    Calibrates gyroscope
    Gyroscope values are in rads/s
    """
    gx, gy, gz = accel_gyro.gyro
    min_gx = gx
    min_gy = gy
    min_gz = gz

    max_gx = gx
    max_gy = gy
    max_gz = gz
```

```
    mid_gx = gx
    mid_gy = gy
    mid_gz = gz

    for _ in range(10):
        gx, gy, gz = accel_gyro.gyro

        min_gx = min(min_gx, gx)
        min_gy = min(min_gy, gy)
        min_gz = min(min_gz, gz)

        max_gx = max(max_gx, gx)
        max_gy = max(max_gy, gy)
        max_gz = max(max_gz, gz)

        mid_gx = (max_gx + min_gx) / 2
        mid_gy = (max_gy + min_gy) / 2
        mid_gz = (max_gz + min_gz) / 2

        print("Uncalibrated gyro: ", (gx, gy, gz))
        print("Calibrated gyro: ", (gx + mid_gx, gy + mid_gy, gz + mid_gz))
        print("Gyro calibration: ", (mid_gx, mid_gy, mid_gz))

        time.sleep(1)
    mid_gx = float(f"{mid_gx:.4f}")
    mid_gy = float(f"{mid_gy:.4f}")
    mid_gz = float(f"{mid_gz:.4f}")
    _CAL = [mid_gx, mid_gy, mid_gz]
    return _CAL

def calibrate_mag():
    """
    Calibrates a magnometer
    """
    countavg = 0
    x, y, z = magnetometer.magnetic
    mag_vals = [x, y, z]
    for i in range(3):
        MAG_MIN[i] = min(MAG_MIN[i], mag_vals[i])
        MAG_MAX[i] = max(MAG_MAX[i], mag_vals[i])

    for _ in range(10):
        x, y, z = magnetometer.magnetic
        mag_vals = [x, y, z]

        for i in range(3):
            MAG_MIN[i] = min(MAG_MIN[i], mag_vals[i])
            MAG_MAX[i] = max(MAG_MAX[i], mag_vals[i])

        countavg += 1
        print("Uncalibrated:", x, y, z)
        cal_x = map_range(x, MAG_MIN[0], MAG_MAX[0], -1, 1)
        cal_y = map_range(y, MAG_MIN[1], MAG_MAX[1], -1, 1)
        cal_z = map_range(z, MAG_MIN[2], MAG_MAX[2], -1, 1)
        print("Calibrate:  ", cal_x, cal_y, cal_z)
        print("MAG_MIN =", MAG_MIN)
        print("MAG_MAX =", MAG_MAX)

        time.sleep(1)
    return MAG_MIN, MAG_MAX

print("Preparing gyroscope calibration. Keep board perfectly still on flat
surface.")
time.sleep(5)
print("Starting gyroscope calibration..")
print()
GYRO_CAL = calibrate_gyro()
print("Gyroscope calibrated!")
```

```
print("Preparing magnetometer calibration. Move board around in 3D space.")
time.sleep(5)
print("Starting magnetometer calibration..")
print()
MAG_MIN, MAG_MAX = calibrate_mag()
print("Magnetometer calibrated!")
print()
print("MAG_MIN =", MAG_MIN)
print("MAG_MAX =", MAG_MAX)
print("GYRO_CAL =", GYRO_CAL)
```

Code the Compass

You need to run the calibration script and copy the values that it outputs into the compass script before it will be able to work properly. If you have not already done so, follow the instructions on the Calibrate page (https://adafru.it/1af9).

Drive Structure

After copying the files, your drive should look like the listing below. It can contain other files as well, but must contain these at a minimum.

Code

The **code.py** for the project is shown below.

Each section of code contains comments about its purpose.

Be sure to replace the calibration value variables near the top of the code with the ones you obtained by following the steps on the calibrate page (https://adafru.it/1af9).

```
# SPDX-FileCopyrightText: 2025 Tim Cocks for Adafruit Industries
# SPDX-License-Identifier: MIT
#
# Adapted from QualiaS3 Compass Learn Guide by Liz Clark (Adafruit Industries)
```

```
# https://learn.adafruit.com/qualia-s3-compass/

import time
from math import atan2, degrees, radians
import adafruit_lis3mdl
import board
from adafruit_lsm6ds.lsm6dsox import LSM6DSOX
from gamblor21_ahrs import mahony
import bitmaptools
from adafruit_gc9a01a import GC9A01A
import adafruit_imageload
import displayio
from fourwire import FourWire


# change these values to your calibration values
MAG_MIN = [-75.1973, -22.5665, -34.5221]
MAG_MAX = [-1.2131, 68.1379, 20.8126]
GYRO_CAL = [-0.0038, -0.0026, -0.0011]


# use filter for more accurate, but slightly slower readings
# otherwise just reads from magnetometer
ahrs = True
center_x, center_y = 120, 120

i2c = board.STEMMA_I2C()
accel_gyro = LSM6DSOX(i2c)
magnetometer = adafruit_lis3mdl.LIS3MDL(i2c)
# Create the AHRS filter
ahrs_filter = mahony.Mahony(50, 5, 100)

# Variable to account for the offset between raw heading values
# and the orientation of the display.
offset_angle = 90


def map_range(x, in_min, in_max, out_min, out_max):
    """
    Maps a value from one range to another.

    :param x: The value to map
    :param in_min: The minimum value of the input range
    :param in_max: The maximum value of the input range
    :param out_min: The minimum value of the output range
    :param out_max: The maximum value of the output range

    :return: The value mapped to the output range
    """
    mapped = (x - in_min) * (out_max - out_min) / (in_max - in_min) + out_min
    if out_min <= out_max:
        return max(min(mapped, out_max), out_min)

    return min(max(mapped, out_max), out_min)

last_heading = offset_angle
heading = offset_angle
last_update = time.monotonic()  # last time we printed the yaw/pitch/roll values
timestamp = time.monotonic_ns()  # used to tune the frequency to approx 100 Hz

# Display Setup
spi = board.SPI()
tft_cs = board.TX
tft_dc = board.RX
displayio.release_displays()
display_bus = FourWire(spi, command=tft_dc, chip_select=tft_cs, reset=None)
display = GC9A01A(display_bus, width=240, height=240)
display.rotation = 90
```

```
# group to hold all of our visual elements
main_group = displayio.Group()
display.root_group = main_group

# load the compass rose background image
rose_bmp, rose_palette = adafruit_imageload.load("compass_rose.png")
rose_tg = displayio.TileGrid(bitmap=rose_bmp, pixel_shader=rose_palette)

# bitmap for the pointer needle
pointer = displayio.Bitmap(5, 90, 2)

# bitmap for erasing the pointer needle
pointer_eraser = displayio.Bitmap(5, 90, 1)

# pointer needle palette, red foreground, transparent background
pointer_palette = displayio.Palette(2)
pointer_palette[0] = 0x000000
pointer_palette[1] = 0xFF0000
pointer_palette.make_transparent(0)
pointer.fill(1)

# display sized bitmap to paste the rotated needle into
rotated_pointer = displayio.Bitmap(240, 240, 2)

# tilegrid for the rotated pointer needle
pointer_tg = displayio.TileGrid(rotated_pointer, pixel_shader=pointer_palette)

# add rose then pointer to the displaying group
main_group.append(rose_tg)
main_group.append(pointer_tg)

while True:
    # if it's time to take a compass reading from the mag/gyro
    if (time.monotonic_ns() - timestamp) > 6500000:
        # read magnetic data
        mx, my, mz = magnetometer.magnetic

        # map it to the calibrated values
        cal_x = map_range(mx, MAG_MIN[0], MAG_MAX[0], -1, 1)
        cal_y = map_range(my, MAG_MIN[1], MAG_MAX[1], -1, 1)
        cal_z = map_range(mz, MAG_MIN[2], MAG_MAX[2], -1, 1)

        # if using ahrs filter
        if ahrs:
            # get accel/gyro data
            ax, ay, az, gx, gy, gz = accel_gyro.acceleration + accel_gyro.gyro

            # apply callibration offset
            gx += GYRO_CAL[0]
            gy += GYRO_CAL[1]
            gz += GYRO_CAL[2]

            # update filter
            ahrs_filter.update(gx, gy, -gz, ax, ay, az, cal_x, -cal_y, cal_z)

            # get yaw
            yaw_degree = ahrs_filter.yaw

            # convert radians to degrees
            heading = degrees(yaw_degree)

        else:  # not using ahrs filter
            # calculate heading from calibrated mag data
            # and convert from radians to degrees
            heading = degrees(atan2(cal_y, cal_x))

        # save time to compare next iteration
        timestamp = time.monotonic_ns()
```

```
    # if it's time to update the display
    if time.monotonic() > last_update + 0.2:
        # wrap negative heading values
        if heading < 0:
            heading += 360

        # if the heading is sufficiently different from previous heading
        if abs(last_heading - heading) >= 2:
            #print(heading)

            # erase the previous pointer needle
            bitmaptools.rotozoom(rotated_pointer, pointer_eraser,
                                 ox=120, oy=120,
                                 px=pointer.width // 2, py=pointer.height,
                                 angle=radians(last_heading + offset_angle))

            # draw the new pointer needle
            bitmaptools.rotozoom(rotated_pointer, pointer,
                                 ox=120, oy=120,
                                 px=pointer.width // 2, py=pointer.height,
                                 angle=radians(heading + offset_angle))

            # set the previous heading to compare next iteration
            last_heading = heading

        # set the last update time to compare next iteration
        last_update = time.monotonic()
```