

06/04/24  
Wesley  
Copa

# Revisiting the BNO055

33

For previous results and a little confusion, see pages 90-92 of Book 1.

→ Previously using the adafruit library.

## Objective

Write our own I2C code to get a heading from the BNO055. See if the error that occurs in the adafruit library still exists.

## Operating Modes

Table 3-3: Operating modes overview

Operating Mode	Available sensor signals			Fusion Data	
	Accel	Mag	Gyro	Relative orientation	Absolute orientation
CONFIGMODE	-	-	-	-	-
ACCONLY	X	-	-	-	-
MAGONLY	-	X	-	-	-
GYROONLY	-	-	X	-	-
ACCMAG	X	X	-	-	-
ACCGYRO	X	-	X	-	-
MAGGYRO	-	X	X	-	-
AMG	X	X	X	-	-
* <del>IMU</del>	X	-	X	X	-
COMPASS	X	X	-	-	X
M4G	X	X	-	X	-
NDOF_FMC_OFF	X	X	X	-	X
NDOF	X	X	X	-	X

We are going to use the IMU mode since we won't have to worry about noisy magnetic fields, and it provides a relative orientation.



The operating mode can be selected by writing to the OPR\_MODE register, possible register values and the corresponding operating modes are shown in the table below.

Table 3-5: operating modes selection

Parameter	Value	[Reg Addr]: Reg Value
CONFIG MODE	CONFIGMODE	[OPR_MODE]: xxx0000b
Non-Fusion Mode	ACCONLY	[OPR_MODE]: xxx0001b
	MAGONLY	[OPR_MODE]: xxx0010b
	GYROONLY	[OPR_MODE]: xxx0011b
	ACCMAG	[OPR_MODE]: xxx0100b
	ACCGYRO	[OPR_MODE]: xxx0101b
	MAGGYRO	[OPR_MODE]: xxx0110b
	AMG	[OPR_MODE]: xxx0111b
Fusion Mode	IMU	[OPR_MODE]: xxx1000b ←
	COMPASS	[OPR_MODE]: xxx1001b
	M4G	[OPR_MODE]: xxx1010b
	NDOF_FMC_OFF	[OPR_MODE]: xxx1011b
	NDOF	[OPR_MODE]: xxx1100b

OPR\_MODE is at address 0x3D

To set IMU mode, we will write 1<<3 to register 0x3D.

### Unit Selection

The measurement units for the various data outputs (regardless of operation mode) can be configured by writing to the UNIT\_SEL register as described in Table 3-9.

Table 3-11: unit selection

Data	Units	[Reg Addr]: Register Value
Acceleration, Linear	m/s <sup>2</sup>	[UNIT_SEL]: xxxxxx0b
Acceleration, Gravity vector	mg	[UNIT_SEL]: xxxxxx1b
Magnetic Field Strength	Micro Tesla	NA
Angular Rate	Dps	[UNIT_SEL]: xxxxx0xb
	Rps	[UNIT_SEL]: xxxxx1xb
Euler Angles	Degrees	[UNIT_SEL]: xxxxx0xb ←
	Radians	[UNIT_SEL]: xxxxx1xb
Quaternion	Quaternion units	NA
Temperature	°C	[UNIT_SEL]: xx0xxx b
	°F	[UNIT_SEL]: xx1xxx b

I want degrees

0<<2



06/04/24  
Wally  
Cade

# Revisiting the BNO055

35

## Important Notes From the data sheet:

Table 3-13: Rotation angle conventions

Rotation angle	Range (Android format)	Range (Windows format)
Pitch	+180° to -180° (turning clockwise decreases values)	-180° to +180° (turning clockwise increases values)
Roll	-90° to +90° (increasing with increasing inclination)	
Heading / Yaw	0° to 360° (turning clockwise increases values)	

Table 3-28: Compensated orientation data in Euler angles format

Parameter	Data type	bytes
EUL_Heading	Signed	2
EUL_Roll	Signed	2
EUL_Pitch	Signed	2

Table 3-29: Euler angle data representation

Unit	Representation
Degrees	1 degree = 16 LSB
Radians	1 radian = 900 LSB

Our headings will be from 0-360°.  
However, we will have to divide the value from the registers by 16.



```

1 #include <Wire.h>
2 #define BNOAddress 0x28 // Found using I2C scanner
3 #define OPR_MODE 0x3D
4 #define UNIT_SEL 0x3B
5
6 void setup()
7 {
8   Serial.begin(115200);
9   Wire.begin();
10
11   // I2C Scanner -> 0x28 found
12   for(int add=1; add<=127; add++)
13   {
14     Wire.beginTransmission(add);
15     int reply = Wire.endTransmission();
16     if (reply == 0)
17     {
18       Serial.print("0x");
19       Serial.print(add, HEX);
20     }
21   }
22
23   // Configure the BNO to IMU mode
24   // OPR_MODE is 0x3D
25   // IMU setting : xxxx1000b
26   Wire.beginTransmission(BNOAddress);
27   Wire.write(OPR_MODE);
28   Wire.write(1<<3);
29   Wire.endTransmission();
30
31   // Unit Select Register
32   // Set Degrees xxxxx0xxb
33   Wire.beginTransmission(BNOAddress);
34   Wire.write(UNIT_SEL);
35   Wire.write(0<<2);
36   Wire.endTransmission();
37 }

```

search for  
an address

Operating Mode

Unit Selection



06/04/24  
Wiley  
code

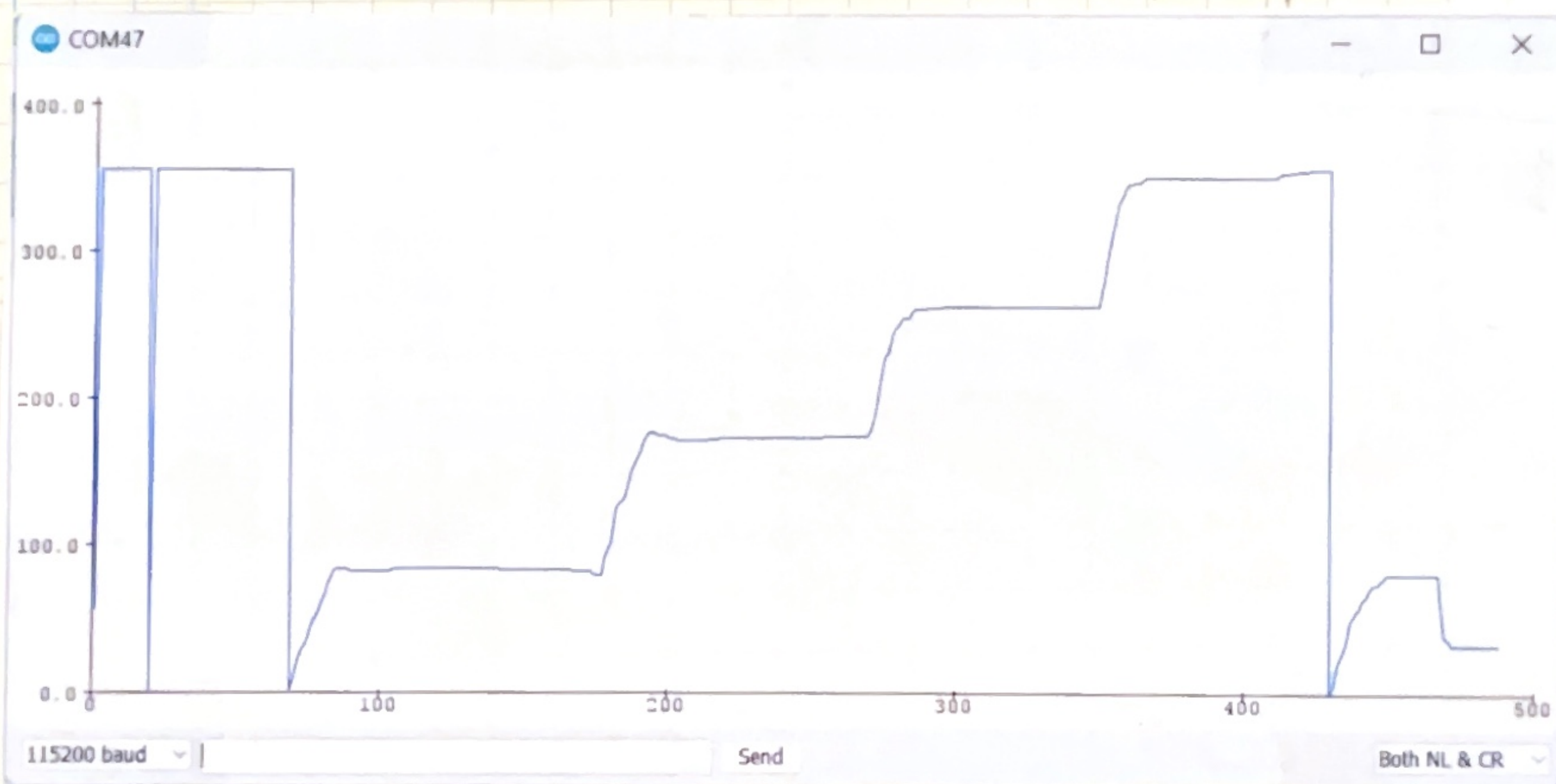
# Revisiting the BNO055

37

```
39 void loop()
40 {
41   // Heading Registers 1A LSB, 1B MSB
42   Wire.beginTransaction(BNOAddress);
43   Wire.write(0x1A);
44   Wire.endTransmission();
45   Wire.requestFrom(BNOAddress, 2); // Go Get 1A, 1B
46
47   uint8_t LSB = Wire.read();
48   uint8_t MSB = Wire.read();
49
50   // Combine the data and divide by 16
51   Serial.println(((int)MSB<<8 | LSB)>> 4);
52   delay(100); Or ((float)((int)MSB<<8 | LSB))/(16.0);
53 }
```

## Results

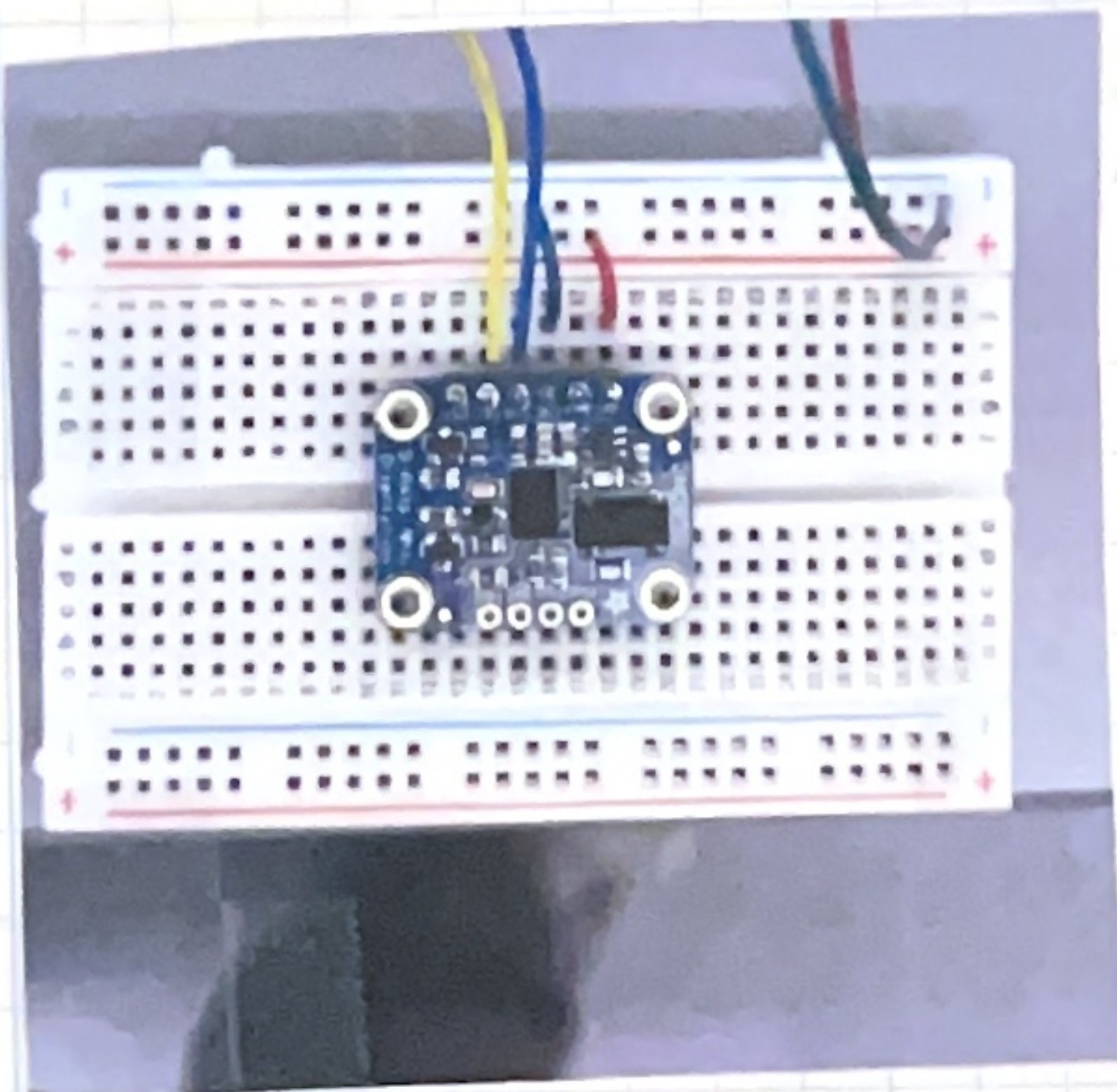
Rotating the board Clock wise? No weird jump!



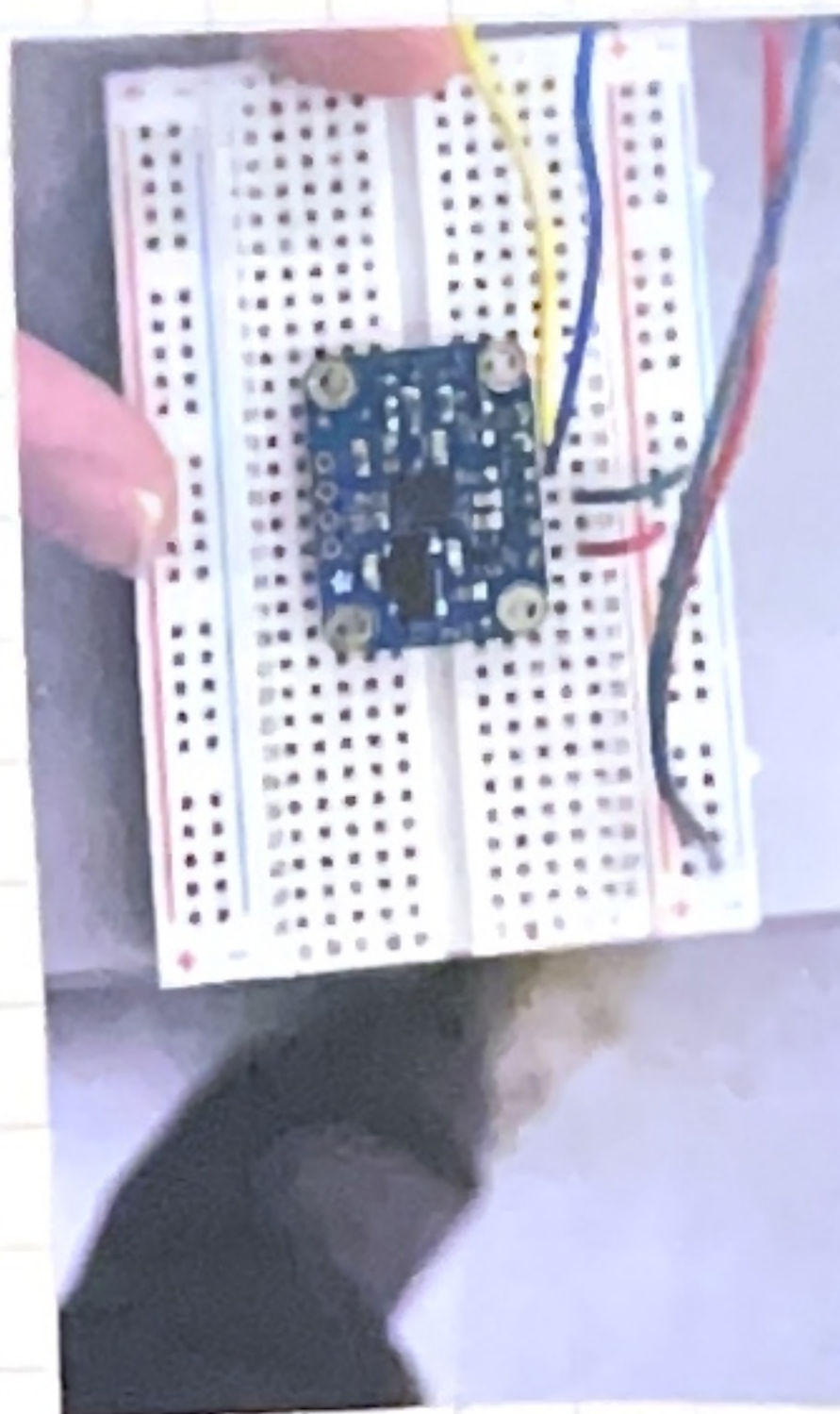


Procedure

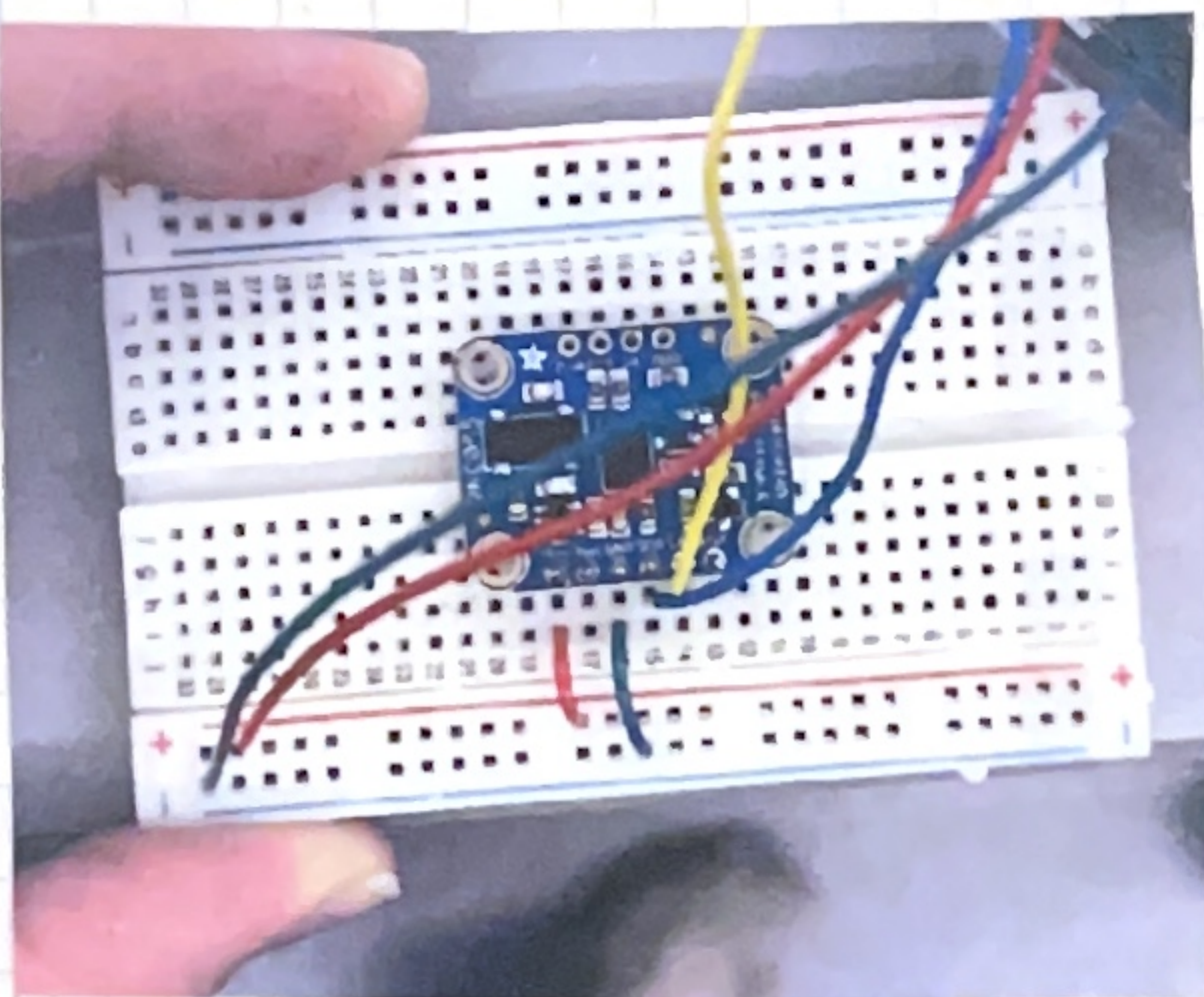
①



②



③



④

