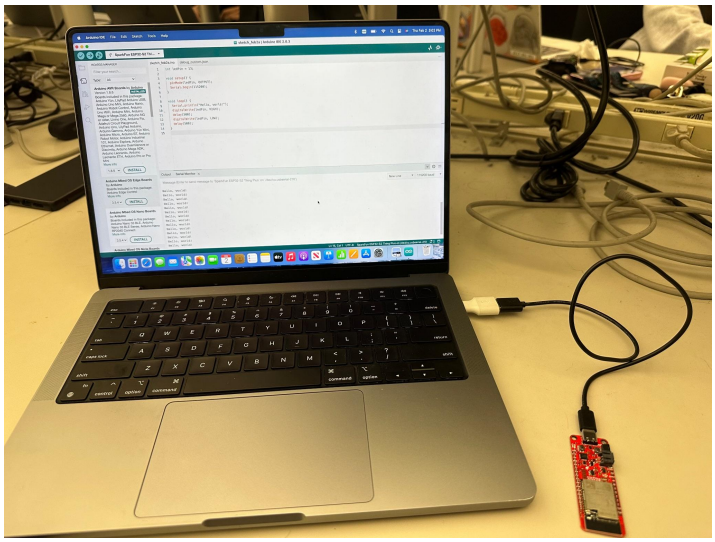
Zeid Solh
205323715
EC ENGR 180DA
Lab 4

Task 1:
MCU and IMU working pictures.



Blink

To verify the toolchain and board are properly set up on their computer, users can upload the simplest of sketches -- Blink! The LED attached to **GPIO 13** is perfect for this test. Plus, with the ESP32 attached to your computer, this is a good time to test out serial communication. Copy and paste the example sketch below into a fresh Arduino sketch:

COPY CODE

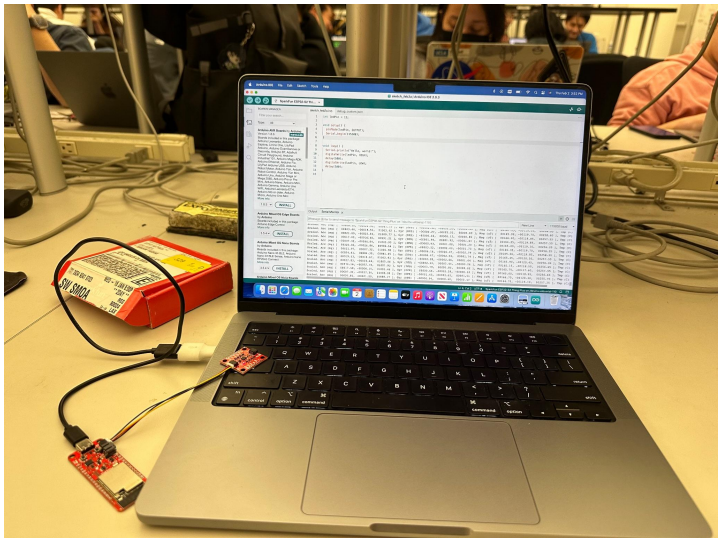
```
int ledPin = 13;

void setup()
{
    pinMode(ledPin, OUTPUT);
    Serial.begin(115200);
}

void loop()
{
    Serial.println("Hello, world!");
    digitalWrite(ledPin, HIGH);
    delay(500);
    digitalWrite(ledPin, LOW);
    delay(500);
}
```

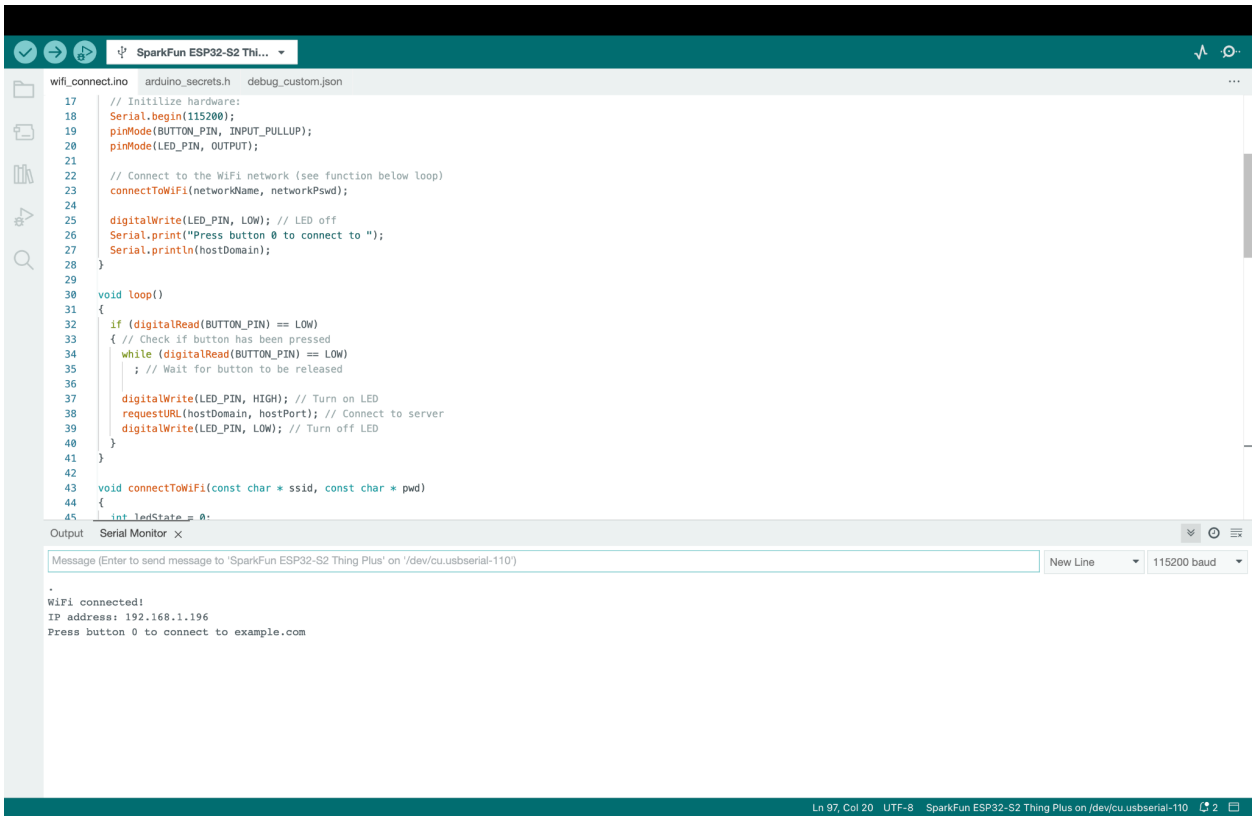
With everything setup correctly, upload the code! Once the code finishes transferring, **open the serial monitor** and set the baud rate to **115200**. Users should see `Hello, world!` 's begin to fly by.

I used the above blink code for MCU.



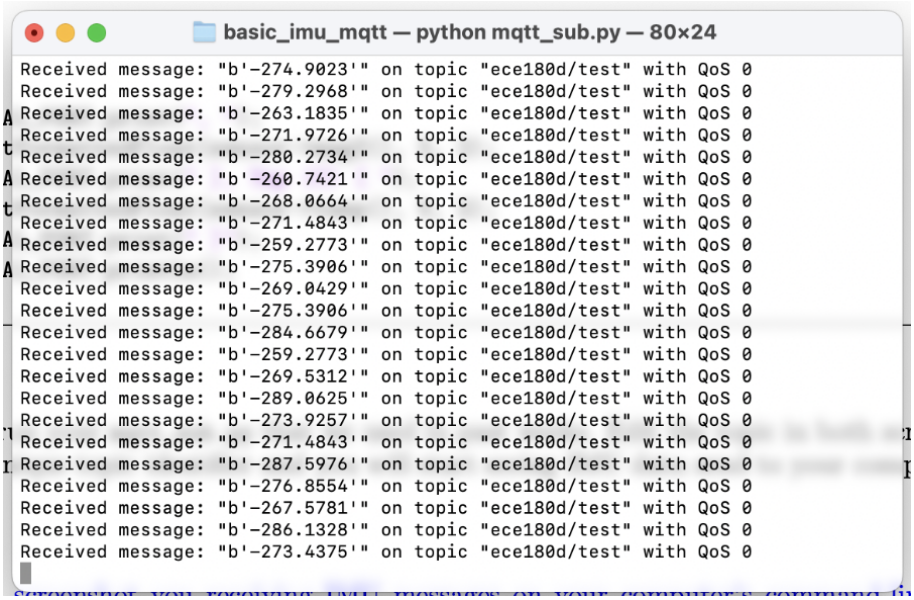
For the IMU, I used Example/Sparkfun 9DoF…/Arduino/Example1_Basics.

Task 2:
Wifi Connected successfully.



Used wifi_connect.ino to do this task.

Task 3:
Receiving IMU messages on my computer's command line. Using basic_imu_mqtt.ino and mqtt_sub.py. The messages represent the X acceleration.

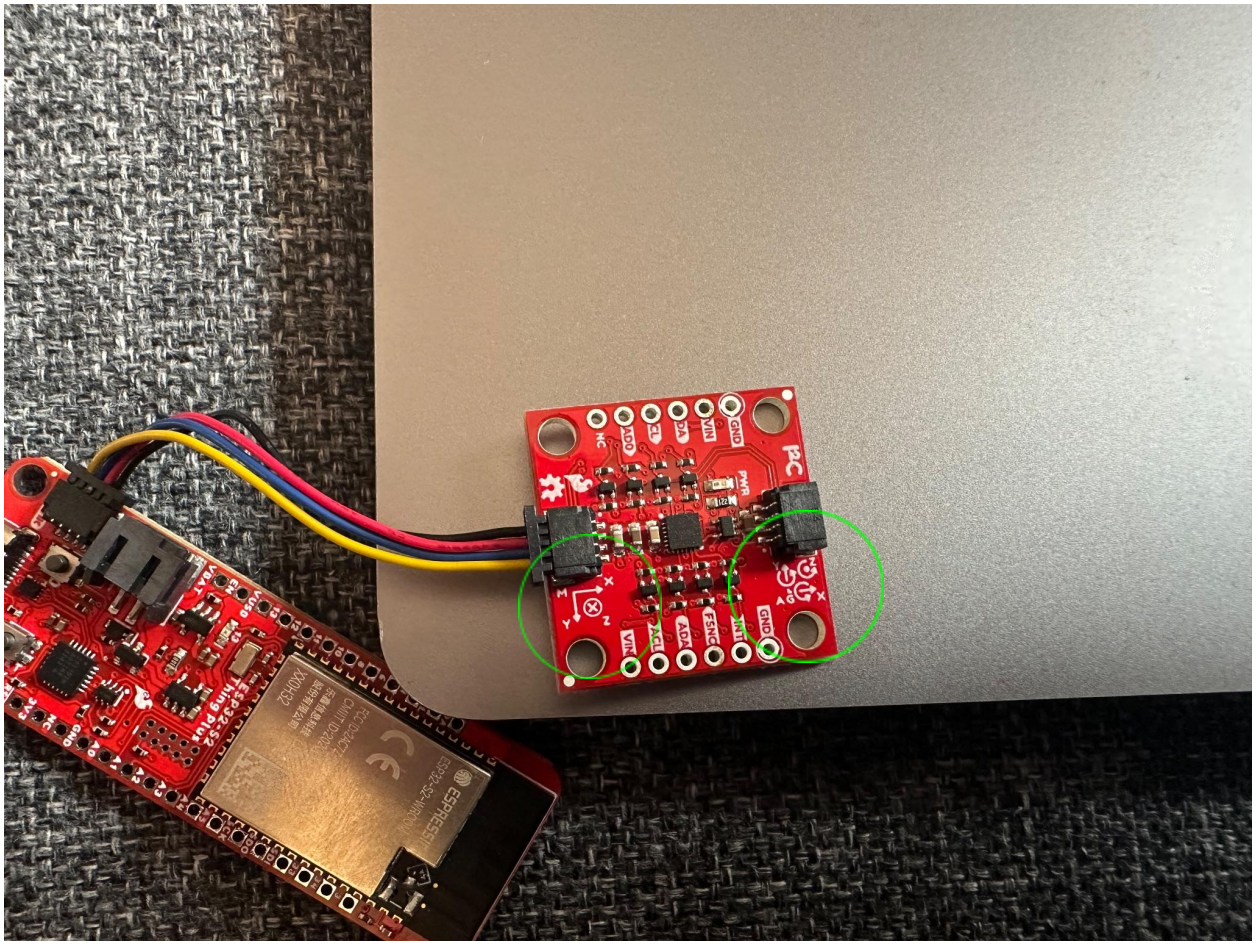


There is a small amount of lag present given that we try to send data at a really fast rate (30 ms). If we reduce for instance the rate at which we send messages, for instance from 30 ms to 100 ms or 500 ms, we get much better results and less lag.

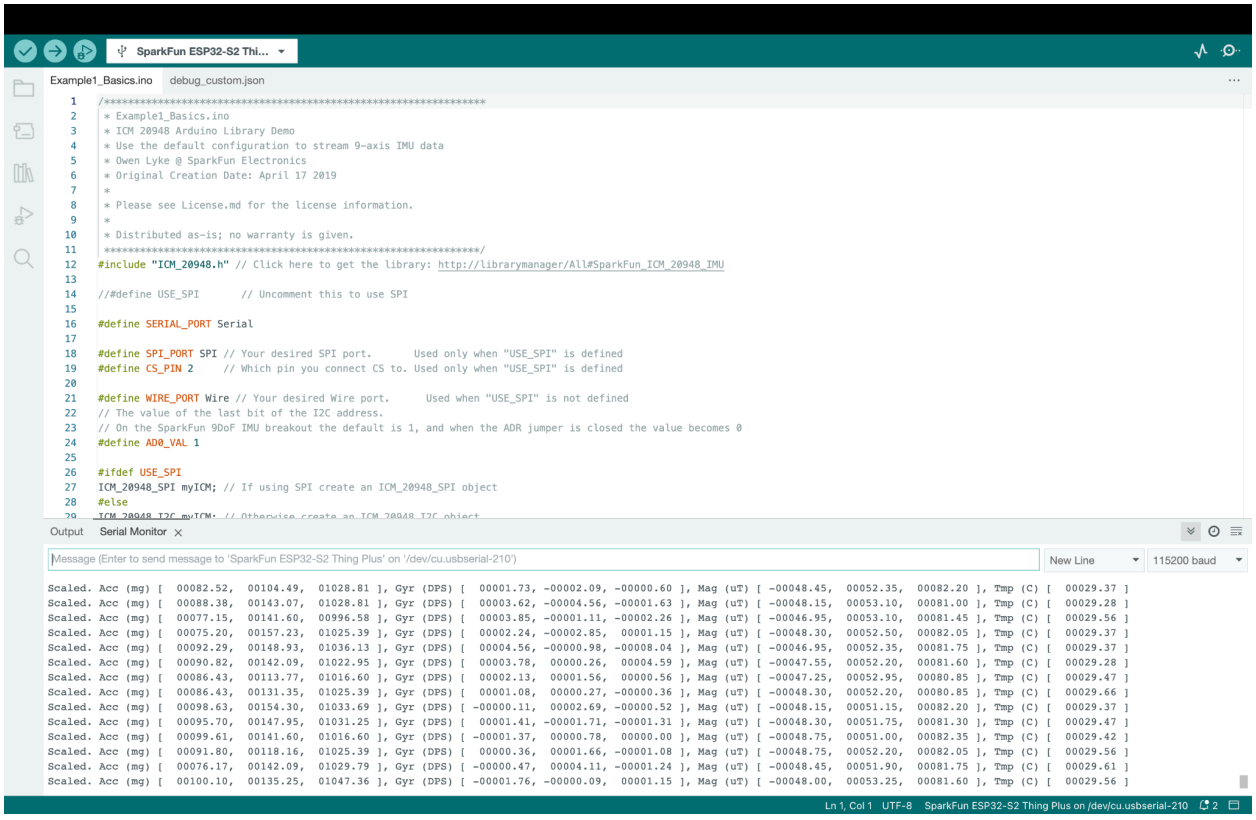
By doing processing on the IMU itself so that only a single message is sent when something is recognized, we get better results as we are no longer sending messages at 30 ms intervals whether we detected something or not. Above is an example of me doing so (combining my classifier.ino and basic_imu_mqtt.ino).

I think the best way to get around lag time, if we have to have lag, is to do some processing on the board itself and as soon as we have interesting results we send it back. We can also reduce the rate at which we are sending messages. Alternatively, we can use a cable, if wifi connectivity is not a must!

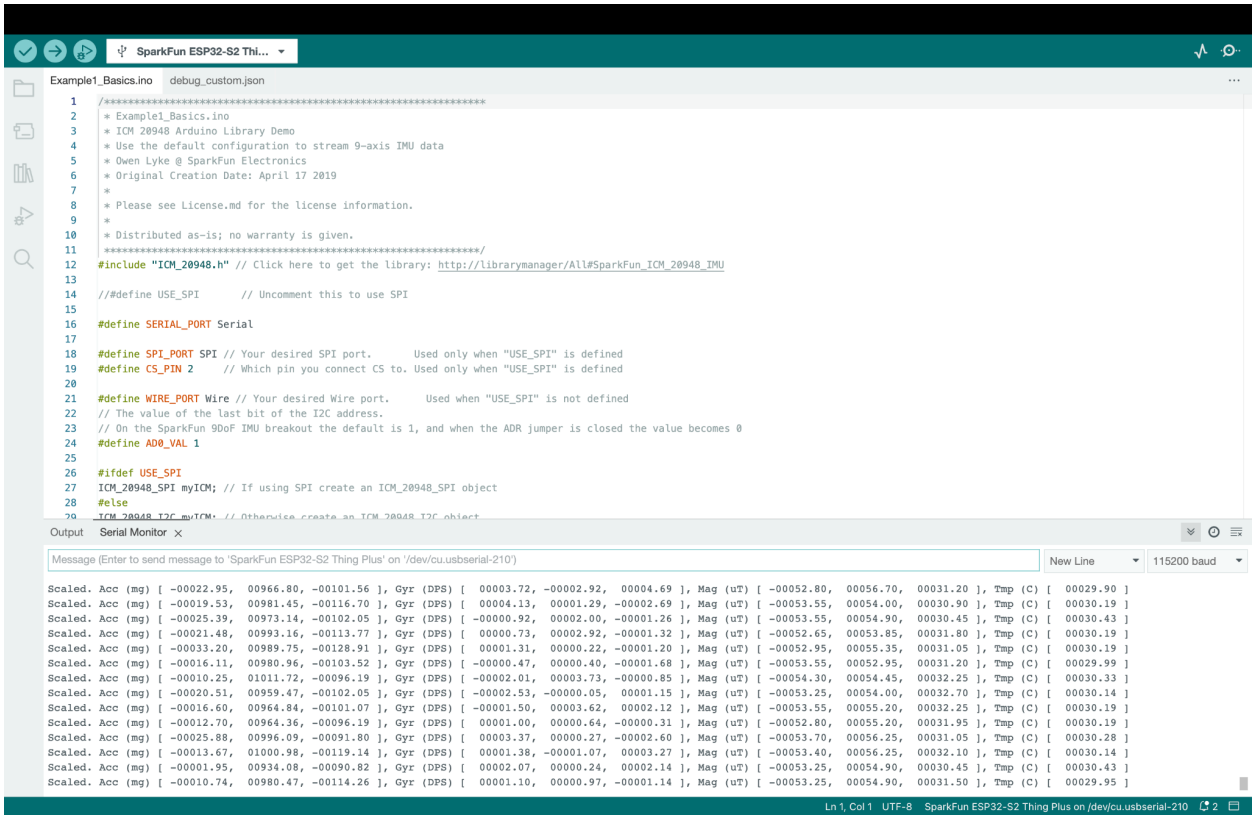
Task 4:
4.1



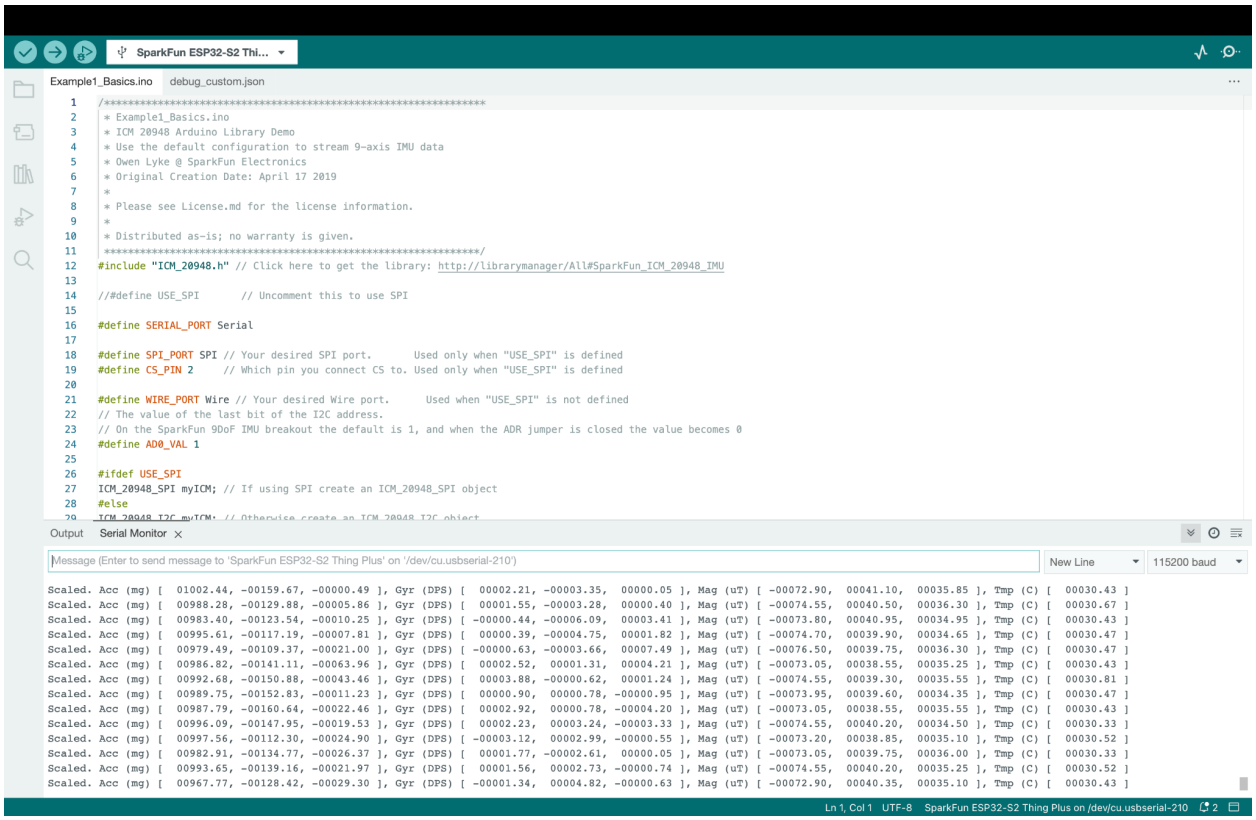
In the picture above, we can see the directions of x, y, and z and can confirm the roll, pitch, yaw values. I circled them in the picture.
Yes, when idle, we can see the gravity acceleration in one or more of the components (x, y, z) which takes values around ~ 980-1000 (which seems to be in cm/s^2).
Using the test file Example/Sparkfun 9DoF…/Arduino/Example1_Basics:



We can see gravity here in the z component.



We can see gravity here in the y component.



We can see gravity in the x component.
It just depends on the orientation of our IMU.

4.2

Roughly, when in idle state, the values of acceleration and gyro tend to be around 0 (except for acc in the direction of gravity). They tend to drift a tiny bit in idle state if the IMU isn't 100% steady. A good way to classify idle vs non-idle is by checking if a certain variable changed by a lot.

4.3

In order to build my classifier to detect forward push and upward lift, I used the accelerations in the x and z directions to detect any sudden changes, by comparing the values stored 30 ms before.

4.4

In order to build a classifier for 3 actions, I had to use the gyro as well to detect circular motions. I detected the gyro in the z direction. I think it does a fairly decent job, although it is not perfect. I think just moving in one direction, along x, y, or z direction is a little bit easier than detecting circular motions. It's important to note that when moving the IMU upward or forward and due to the sensitivity of the IMU, it will always detect circular motion with it.