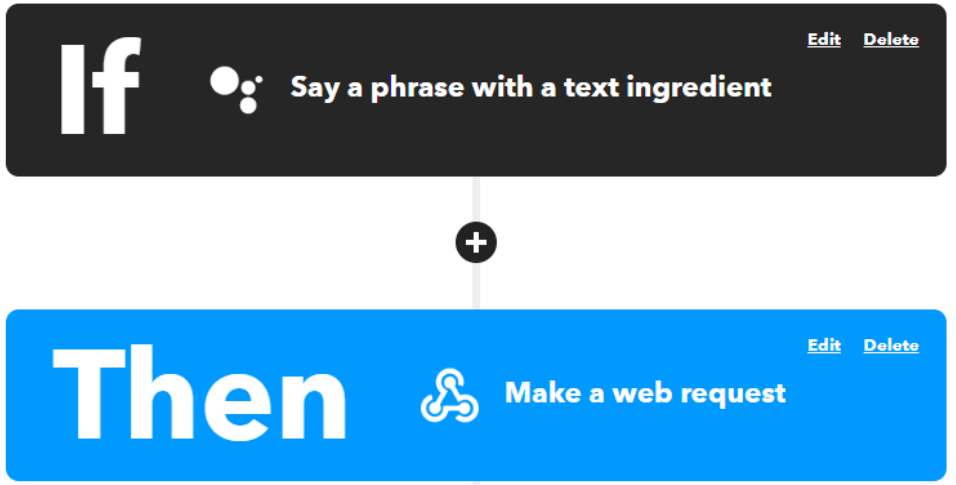
# Lab 3

Motion Detector

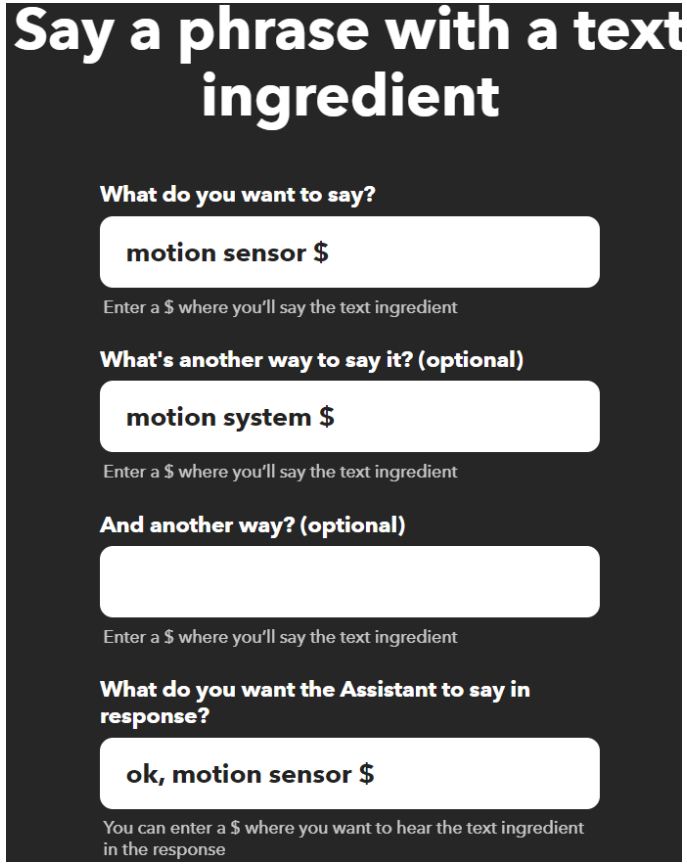
# IFTTT Design

- The motion sensor applet receives a google voice command from google assistant and transmit the data to thingspeak

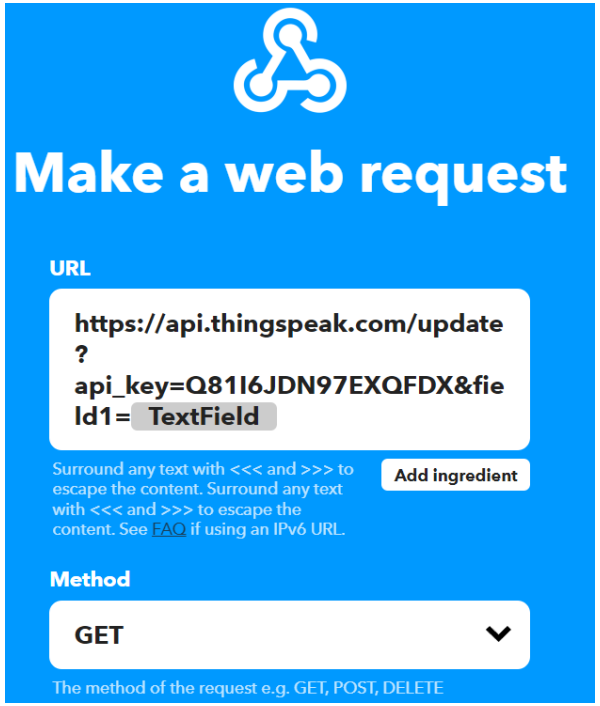
# IFTTT Design

- The google assistant setup is shown below
- The user can say ‘motion sensor’ + ‘activate/deactivate’
- The google assistant will respond with ‘ok, motion sensor activate/deactivate’



# IFTTT Design

- The webhooks can receive the data from the google assistant and send the data to thingspeak through a URL

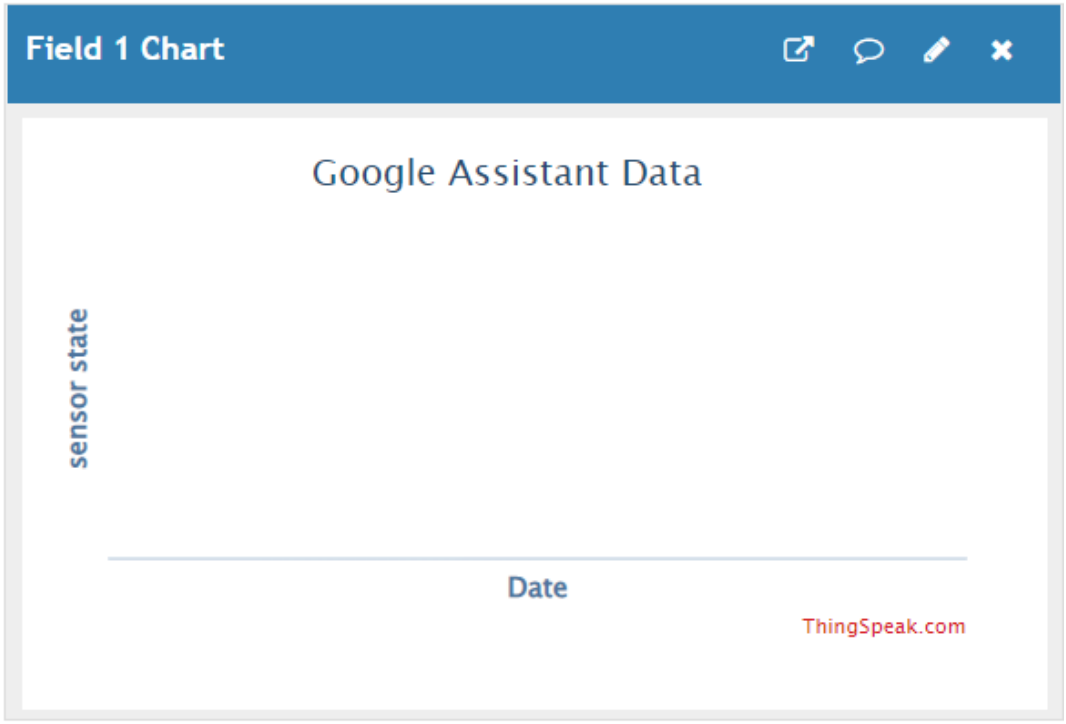


# ThingSpeak Design

- I created a channel in ThingSpeak
- There is a Field 1 to receive data from IFTTT

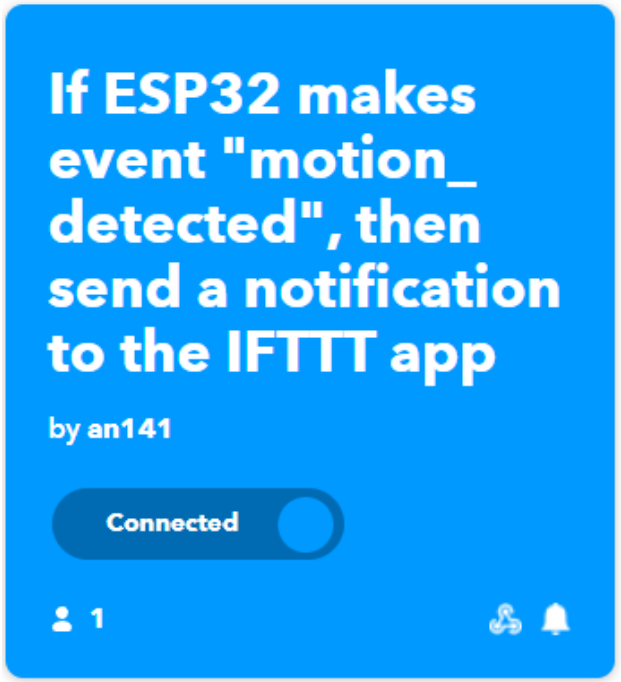
## Channel Stats

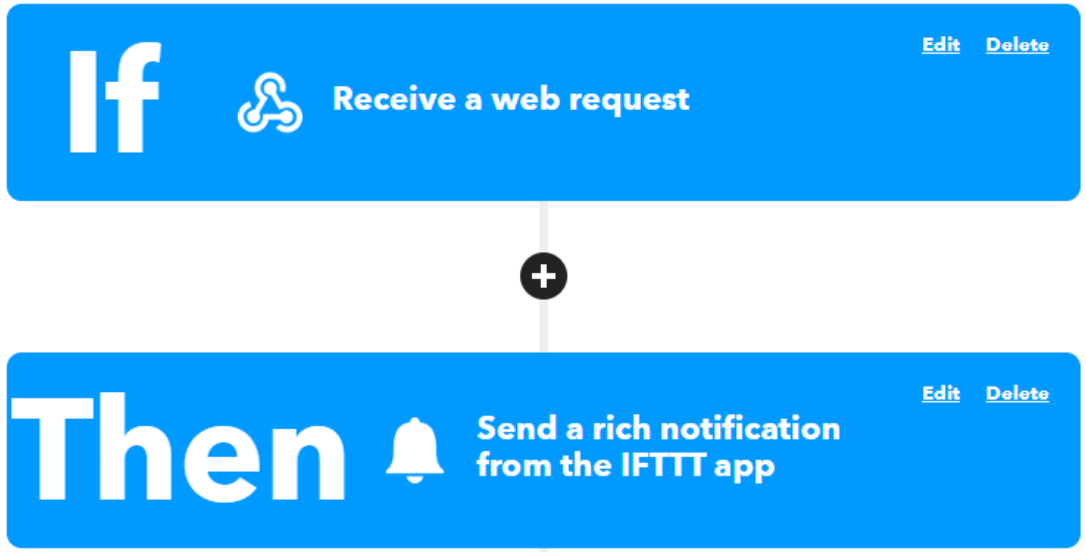
Created: 2 days ago
Last entry: about 5 hours ago
Entries: 10



# IFTTT Design

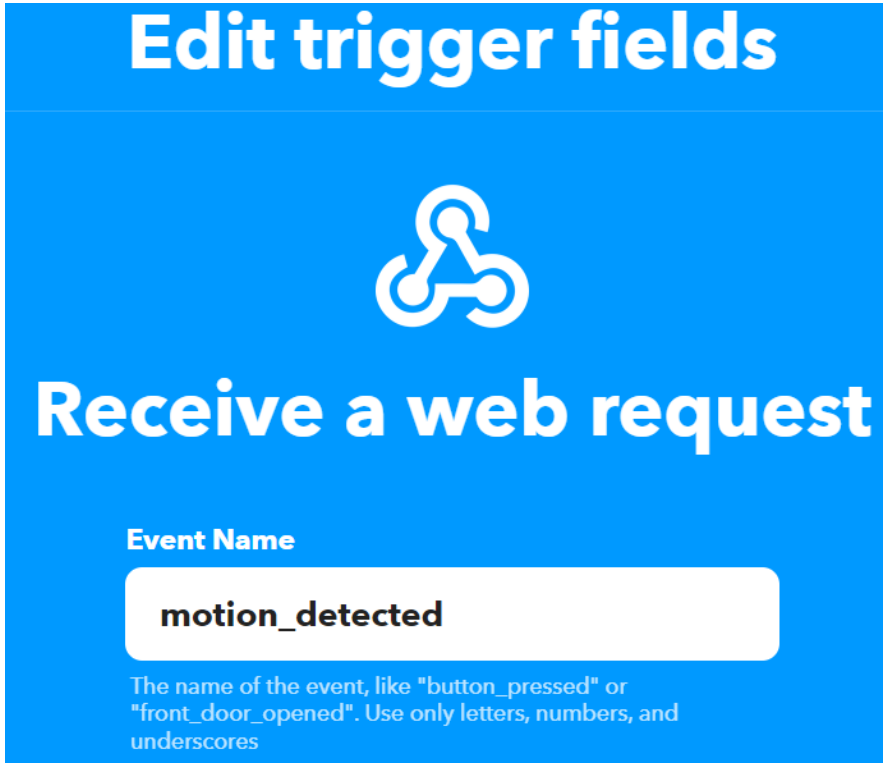
- The notification applet uses webhooks to received data
- The notification applet uses google notification to notify my phone



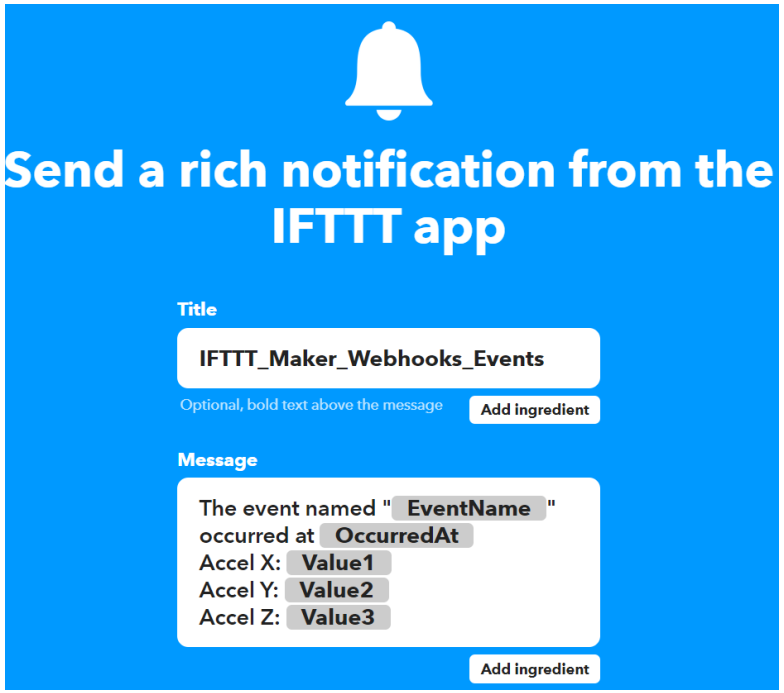
# IFTTT Design

- The webhooks set up is shown below
- The event name is ‘motion_detected’
- The webhooks has a special url. ESP32 can use this url to send data to the IFTTT



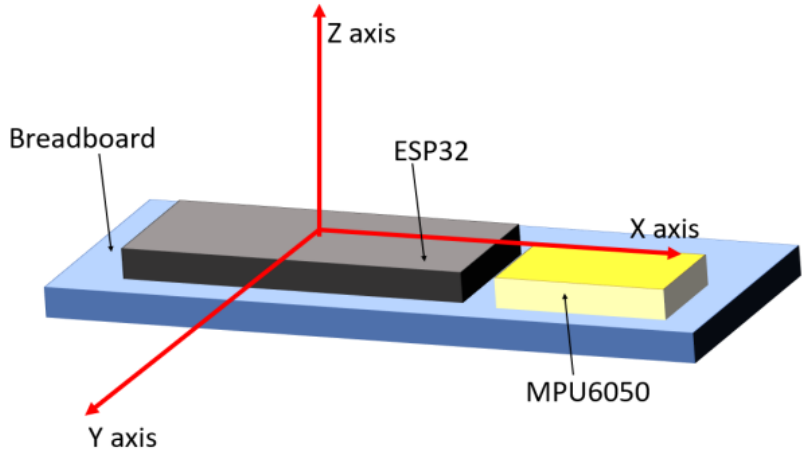
# IFTTT Design

- The notification set up is shown below
- The phone notification will show the current sensor reading, the event name, and the time when the notification is triggered



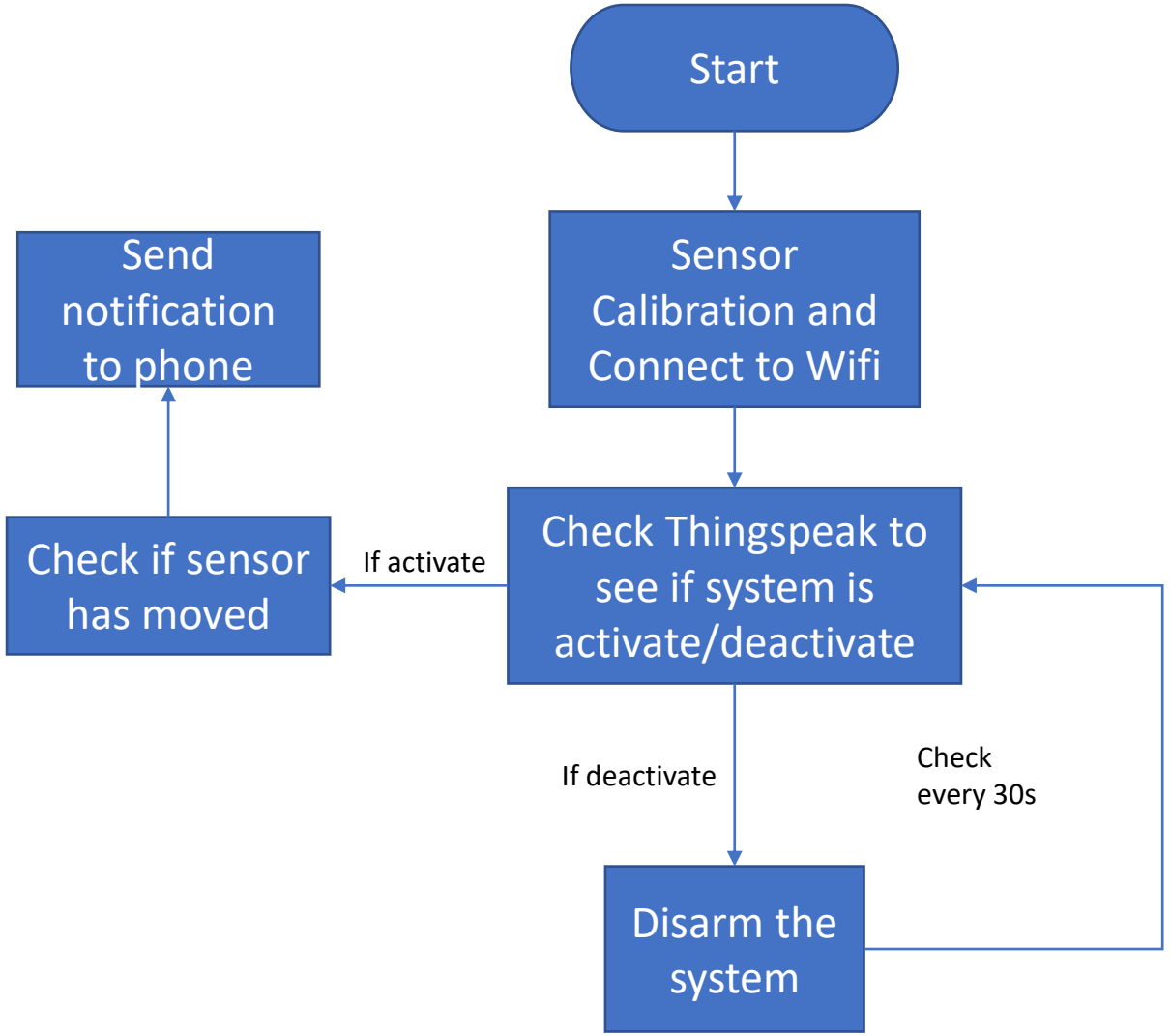
# Sensor Calibration

- I read sensor data when the sensor facing upwards
  - In this case, AcZ_faceup should be -9.8 and the other readings should be zero
- Then, I read sensor data when the sensor facing downwards
  - In this case, AcZ_facedown should be 9.8 and the other reading should be zero
- Therefore, the sensor gain K = 9.8*2/(AcZ_facedown - AcZ_faceup)
- The sensor reading at zero acceleration is also known
  - AcX_zero = (AcX_faceup + AcX_facedown)/2
  - AcY_zero = (AcY_faceup + AcY_facedown)/2
  - AcZ_zero = (AcZ_faceup + AcZ_facedown)/2
- The real acceleration reading is
  - AcX_real = (AcX - AcX_zero)*K
  - AcY_real = (AcY - AcY_zero)*K
  - AcZ_real = (AcZ - AcZ_zero)*K



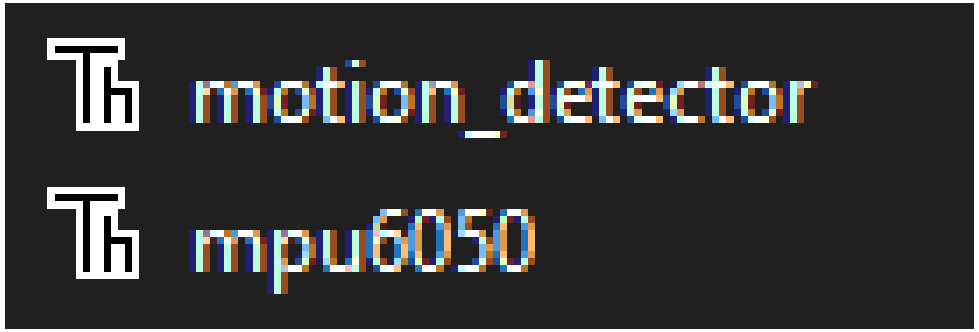
# Software Flowchart

- At the start point, ESP32 will calibrate the sensor
- Once the sensor is calibrated and Wifi is connected, ESP32 will check with Thingspeak to see if the system should be activated or not.
- If the system should be activated, ESP32 will use MPU6050 to check the acceleration
- If the acceleration has changed, ESP32 will send a phone notification



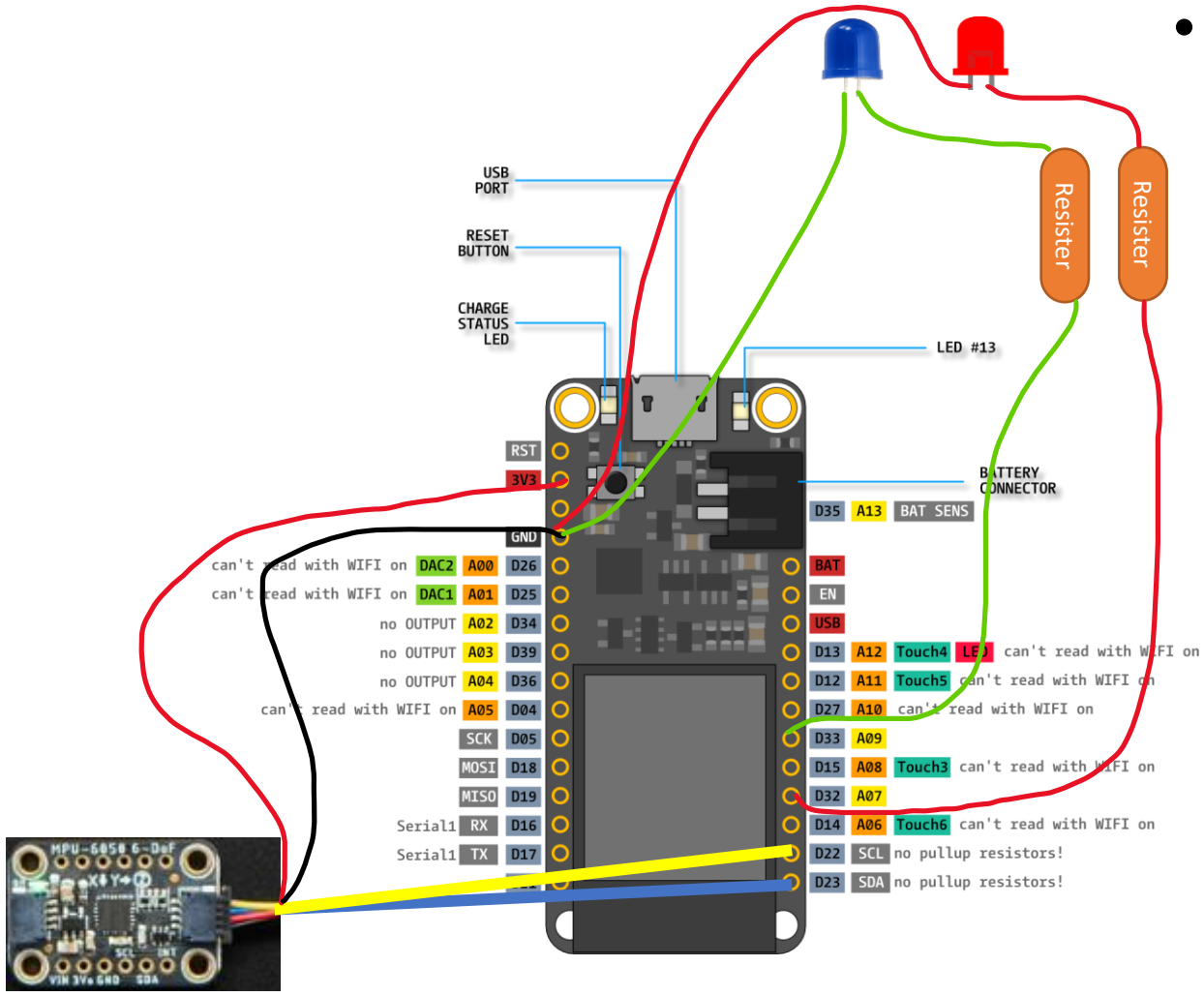
# File Structure

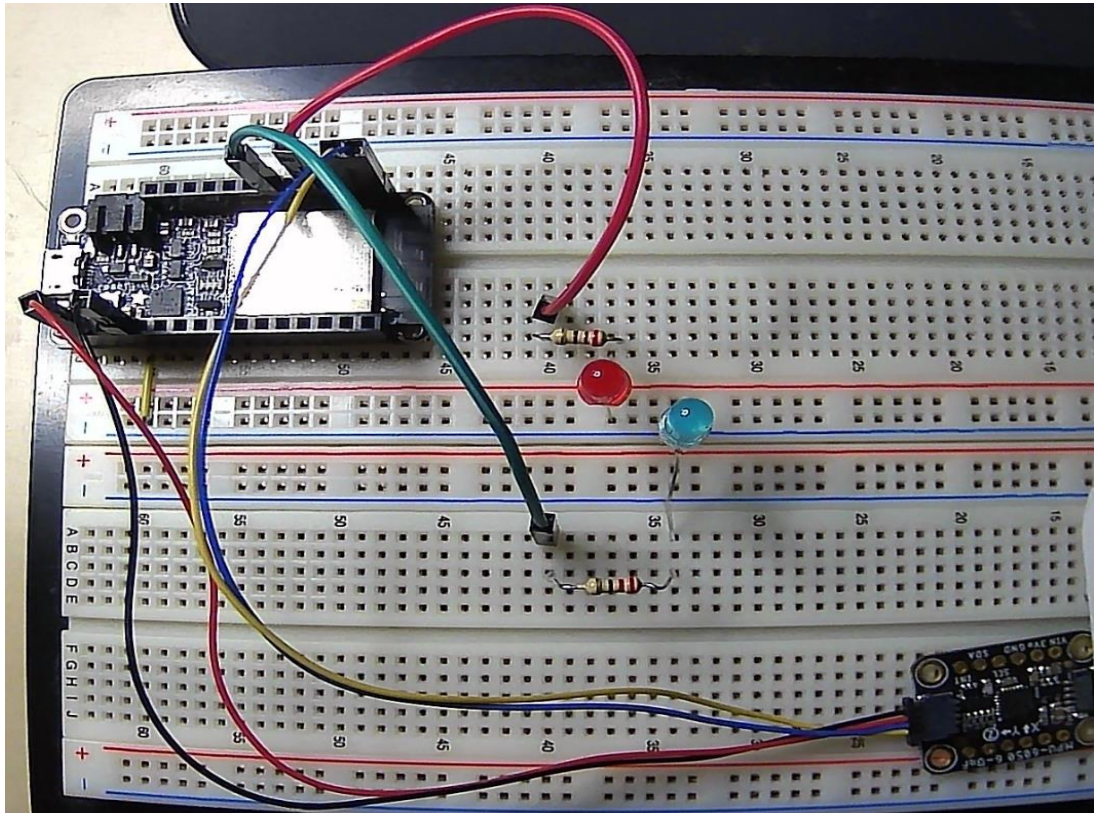
- motion_detector.py is the main code to run on the ESP32
- mpu6050.py is the MPU 6050 driver library that ESP32 used to communicate with MPU6050 through $I^2C$



# Hardware Connection

- The red led is connected to pin 32
- Blue led is connected to pin 33
- $I^2C$ connection is shown blow
  - Yellow goes to SCL
  - Blue goes to SDA
  - Red and black is 3V and ground



# Video

- https://youtu.be/zLkY6vPAUUY
- In the video, I showed that
  - I calibrated the sensor with ESP32
  - I can use google assistant from my phone to activate and deactivate the system
  - I can receive a notification with the sensor reading sent from ESP32 when the sensor detects motion
- Below are some external reference I used for my design
  - https://esp32io.com/tutorials/esp32-ifttt
  - https://www.youtube.com/watch?v=X-_25tzo8Cw
  - https://makeblock-micropython-api.readthedocs.io/en/latest/public_library/Third-party-libraries/urequests.html