ECE 568: Embedded Systems

Spring 2022

Lab 3 – Real-Time Motion Detection System using ESP32, ThingSpeak and IFTTT

To be done individually; Due by 11:59pm, Saturday, May 7, 2022.

1. Overview

In this assignment, you are going to design a **Real-Time Motion Detection System** using the ESP32 Feather board and the Adafruit MPU-6050 sensor board. In addition, you will be using the **ThingSpeak IoT platform** and IFTTT service (If-This-Then-That) to integrate with the ESP32-Sensor board to build this motion detection system.

One application of this system is *bag theft detection*. Consider that you have put the system inside your bag, kept the bag in a stationary place and you are going to workout/gym. You put the system in Armed state via Google Assistant voice commands, and the ESP32 starts detecting motion using the acclerometer sensor. As soon as any motion is detected, the system sends an IFTTT notification to your phone (on the IFTTT app), thereby alerting you of possible theft. On the other hand, you might want to disarm the system using Google Assistant when you are ready to take the bag yourself or when you are at home, as you might not want to receive unnecessary notifications. Fig. 1 shows a high-level overview of this system.

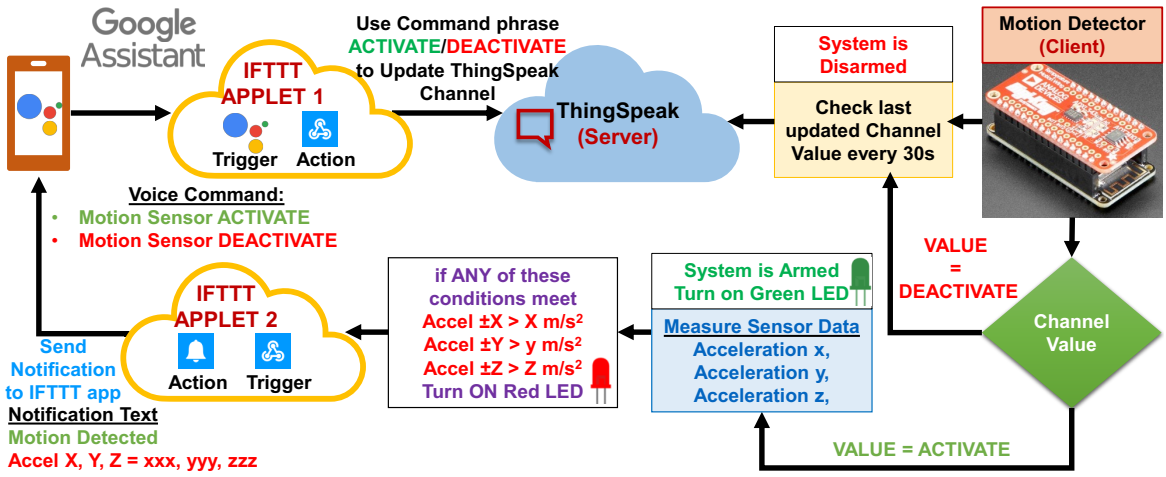

Fig 1. High level overview of Motion Detection System

2. Hardware Assembly

The Adafruit MPU-6050 6-DoF Accel and Gyro Sensor - STEMMA QT sensor board consists of an MPU-6050 triple-axis accelerometer and gyroscope. There is also an integrated temperature sensor on the board. Both sensors are connected over the **shared I²C bus**. The I²C accelerometer sensor has three axes of measurements, X, Y, Z. You can set the sensitivity level to either ±2g, ±4g, ±8g or ±16g. The lower ranges give more resolution for slow movements, so are more sensitive, whereas the higher ranges are good for high-speed tracking as they have wider measurement range.

The best thing about this board is that you DO NOT NEED TO SOLDER. You can use the JST SH 4-pin to Premium Male Headers Cable to connect this to your ESP32 feather board, as shown in Fig. 2. Follow this link to get a better understanding on the exact connections needed between the ESP32 and the sensor board.

For this lab, you are NOT required to use the gyroscope or temperature sensor present in the sensor board. However, feel free to experiment with it if you would like.

3. ESP32 and Sensor Initialization

3.1. Hardware Interfacing

In addition to the sensor board, interface the following components to the ESP32 board:

- Two external **LED**s (red, green) as GPIO OUTPUTs.

The MPU board already has 10K *pull-up resistors* attached to the I²C pins (**SCL** and **SDA**). You do not need to attach separate resistors to communicate with ESP32.

3.2. Software Initialization

All the three sensors (accelerometer, gyro, temperature) are connected to the shared I²C bus on the MPU6050. You can use the I²C driver in MicroPython (**machine.I2C**) to communicate with the accelerometer by constructing I²C bus on ESP32. Perform the following operations:

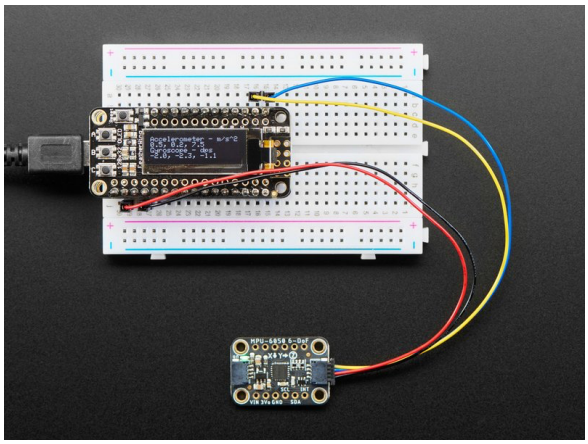

Fig 2. Connecting MPU-6050 to ESP32 using JST SH Cable

- Initialize LEDs, Timers (also Interrupts if you need).
- Use the schematics and ESP32 manuals to create the I^2C bus using proper pins.

3.3. Interfacing Sensors

I^2C sensors expose data using memory or register addresses. Normally, you are supposed to interact with an I^2C sensor using both its I^2C address and the address of a register or memory location in the device. However, for this lab, you are allowed to use the adafruit-circuitpython-mpu6050 library, which makes interactions with the sensor easier.

However, if you are unable to use this library for some reason, you are free to use I^2C protocol to communicate with the sensor board. Tutorials on I^2C communication are available here. Check MPU-6050 datasheet to understand how they expose data and the memory or register addresses to use. Note that the earlier method is the easier one.

3.3.1. Initialize and Calibrate the Accelerometer

You can initialize the sensor in the following way. This is only an example. You are welcome to use any other library other than adafruit_6050.
(Source: https://circuitpython.readthedocs.io/projects/mpu6050/en/latest/)

```
import board
import busio
import adafruit_mpu6050

i2c = busio.I2C(board.SCL, board.SDA)
mpu = adafruit_mpu6050.MPU6050(i2c)
```

The accelerometer needs to be calibrated before use. For instance, if your ESP32 and MPU6050 assembly *remains flat and stands still*, output data for X and Y should be **0 m/s²** or **0 g** and for Z should be **9.8 m/s²** or **1 g** which is the default acceleration of gravity. For this lab, if the output data for each axis is close to these mentioned values, that is acceptable. *Your program should display an appropriate message on the terminal after completion of the calibration*. For calibration, you might want to fix the sensor to the breadboard.

4. Setup ThingSpeak and IFTTT

4.1. IFTTT Applets Setup

IFTTT stands for "If This Than That", and it is a free web-based service to create chains of simple conditional statements called **applets**. This means you can trigger an event when something happens. In this lab, you need to create two different applets.

4.1.1. Create IFTTT Account

The first step is to setup an account in IFTTT. You can easily do this by signing up at www.ifttt.com using your email address (its free). You also need to install the IFTTT app on your phone and login, for the applets to work.

4.1.2. Applet 1

The purpose of the 1st applet is to arm and disarm (activate and deactivate) your motion detection system, as shown in Fig. 1. Creating an IFTTT applet is simple: You simply pick a **trigger**, or an "if this," then pick a "then that" **action**. For the 1st applet, you should use '**Google Assistant**' as the *trigger* and write a phrase to control your motion detection system. For example, you can use '**Motion sensor $**' as the phrase, where $ can be either **Activate** or **Deactivate**. For the *action*, you should use '**Webhooks**' to send data (in this case $) to ThingSpeak channel.

4.1.3. Applet 2

The purpose of the 2nd applet is to receive a web request from your ESP32 if any motion is detected and send a notification to your phone. In this case, you should use '**Webhooks**' as the *trigger* and **Notifications** as the *action*. Whenever any motion is detected, the ESP32 should make a web request to Webhooks and pass on the accelerometer sensor values (X, Y, Z axis), which in turn should send a notification to the IFTTT app on your phone. Fig. 3 shows examples of these two applets. Fig. 4 shows an example notification received on the phone. Note that the accelerometer data for all 3 axes are shown in the notification.

NOTE: You must find out how you can use Webhooks to send data to ThingSpeak (applet 1) and receive data from ESP32 (applet 2). Webhooks is used as an **action** in applet 1 while as a **trigger** in applet 2. Create the IFTTT applets accordingly. You can use the following tutorial as a guidance. Make sure to keep things simple!

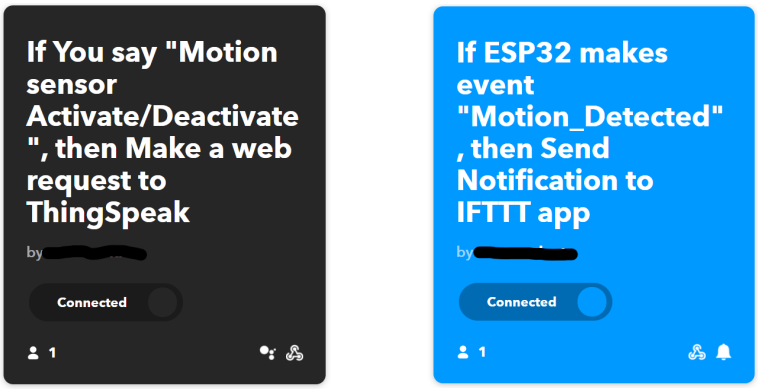

Fig 3. Example of (a) IFTTT Applet 1 (b) IFTTT Applet 2. Once you build the applets, they appear under 'My Applets' on the IFTTT website.

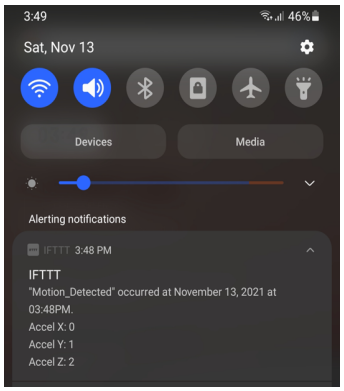

Fig 4. Notification received on IFTTT app on the phone

4.2. ThingSpeak Setup

The purpose of ThingSpeak is to act as the server, a medium of communication between ESP32 and your Google Assistant. You have used ThingSpeak in lab 2 and should already have an account. Login into the account and create a new channel "**Google Assistant Data**" and enable one field "sensor state" to receive data from Webhooks you create in IFTTT applet 1. Unlike lab 2 where you posted data to channel using ESP32, here you will read data from channel using ESP32 every 30 seconds (use a *Hardware Timer*). You need to use your specific **Read API Key** to read data from your channel. *You can use any MicroPython package to read data from ThingSpeak server.*

4.3. Google Assistant Setup

Generally, Android phones have **Google Assistant** installed, however, if your phone doesn't have it preinstalled (e.g., in iPhone), you need to download it from your App/Play store, install it and login with your Google account.

5. Overall Application Workflow (Fig. 1)

The overall flow of tasks executed by your motion detection system is described here. These steps can be only performed after you have completed Section 3 and Section 4. Before proceeding forward, ensure that ESP32 has been properly interfaced with the sensor board and you are able to read accelerometer readings in all 3 axes. **It is highly recommended that you test each of the individual components of your system, *viz.*, ESP32 and sensor assembly, IFTTT applets and ThingSpeak server separately before integrating everything together.**

1. Give the voice command to Google Assistant: **Ok Google, Motion Sensor ACTIVATE**. The IFTTT applet 1 gets triggered and sends the value '**ACTIVATE**' to the ThingSpeak channel.
2. ESP32 is working as a client and reads the last updated channel value from ThingSpeak server. As the last value is '**ACTIVATE**', turn on the **GREEN** Led to indicate that system is in armed state. You can consider that your system has detected **MOTION** if any of the following conditions occur. Also turn on **RED** Led to indicate motion. The 3 different axes of motion: X, Y, Z, and corresponding motions are indicated in Fig. 5
   - Acceleration in **±X** direction > X m/s^2 i.e., **back** and **front**
   - Acceleration in **±Y** direction > Y m/s^2 i.e., **left** and **right**
   - Acceleration in **±Z** direction > Z m/s^2 i.e., **up** and **down.**

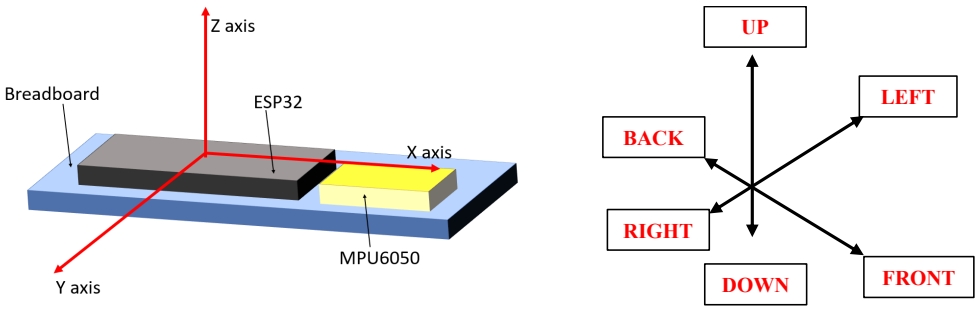

Fig 5. Motion Detector Axes Orientation and Motion Notations

3. Move the sensor board assembly. ESP32 should now trigger IFTTT applet 2 and send Notification to your phone (IFTTT app) like Fig. 4.
4. ESP32 should check ThingSpeak channel every 30 seconds. If channel value is '**ACTIVATE**', continue detecting motion and send notifications for around 1 minute. **You should show that you are receiving notifications in your video. You must show the values for each axis in your notification. Just posting 'Motion Detected' in your notification is not sufficient for this lab.**
5. Give the voice command to Google Assistant: **Ok Google, Motion Sensor DEACTIVATE**. The IFTTT applet 1 gets triggered and sends the value '**DEACTIVATE**' to the ThingSpeak channel.
6. ESP32 now sees that the channel value is '**DEACTIVATE**'. Stop measuring sensor values and Turn OFF Green Led to indicate that system is disarmed now. Moving the sensor board assembly SHOULD NOT SEND any notification to your phone. You also must show this in your video.

NOTE: You need to move the ESP32 and Sensor board assembly by hand. You are free to choose the values for X, Y, Z m/s^2 as the threshold acceleration in 3 axes to decide if there is any motion or not. You are also free to choose the **Time Interval** between two consecutive readings from the sensor. However, make sure to choose values such that you do not end up getting hundreds of notifications on your phone every minute.

6. Submission

Make sure you follow these instructions precisely. You need to turn in your code on Brightspace. Please create a directory named ***username*_lab3**, where username is your CAREER account login ID. *This directory should contain the following files, i.e., no executables, no temporary files, no sub-directories, etc.*

1. ***motion_detector.py***: your main program for the motion detection system.
2. ***README.txt***: As mentioned in the beginning, you should design your own system. Please include a brief description of what you did and why in this text file. You should also describe hardware connections (which pins you used, exactly what they were connected to, etc.).
3. Additional python files used by the motion_detector.py file (*if any*)

Zip the files and name it as *username*_lab3.zip and **upload the .zip file to Brightspace**.

NOTE: In this lab, you can break up your code in multiple python files if you need. For reference, the directory and file structure of your submission should look something like this. Note the spellings, spaces, etc. in the file/directory names.

```
username_lab3.zip
   |---username_lab3 (directory)
      |---motion_detector.py
      |---misc1.py
      |---misc2.py
      |---README.txt
```

7. Video Submission

Create a short video that shows you demonstrating your working solution (you can show your ThingSpeak channel and IFTTT applets before proceeding to the actual demo). Please do not upload video files directly on Brightspace. Instead, upload the video to YouTube (or other such video hosting platform) and include the link to the video in your README file above.

REFERENCES

[1] Getting started with MicroPython on the ESP32
https://docs.micropython.org/en/latest/esp32/tutorial/intro.html

[2] ESP32 WROOM-32 Datasheet
https://www.espressif.com/sites/default/files/documentation/esp32-wroom-32_datasheet_en.pdf

[3] ESP32 Technical Reference Manual
https://www.espressif.com/sites/default/files/documentation/esp32_technical_reference_manual_en.pdf

[4] Adafruit HUZZAH32 – ESP32 Feather Online Manual
https://learn.adafruit.com/adafruit-huzzah32-esp32-feather

[5] Adafruit ESP32 Feather Schematics https://cdn-learn.adafruit.com/assets/assets/000/041/630/original/feather_schem.png?1494449413

[6] MicroPython GitHub https://github.com/micropython/micropython

[7] ESP32 specific functionalities in MicroPython
http://docs.micropython.org/en/latest/library/esp32.html

[8] MPU6050: https://learn.adafruit.com/mpu6050-6-dof-accelerometer-and-gyro

[9] Learn how to talk to I^2C devices with MicroPython:
https://learn.adafruit.com/micropython-hardware-i2c-devices/i2c-master

[10] ThingSpeak: https://thingspeak.com/

[11] IFTTT: https://ifttt.com/