ECE 40862: Software for Embedded Systems

Fall 2023

Labs 5 & 6 – Real-Time Motion Detection System using ESP32, ThingSpeak and IFTTT

To be done individually; Due by 11:59pm, Tuesday, December 12, 2023.

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 **ThingSpeak IoT platform** and **IFTTT** service (If-This-Then-That) and integrate with ESP32-Sensor Assembly 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 accleromenter 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.

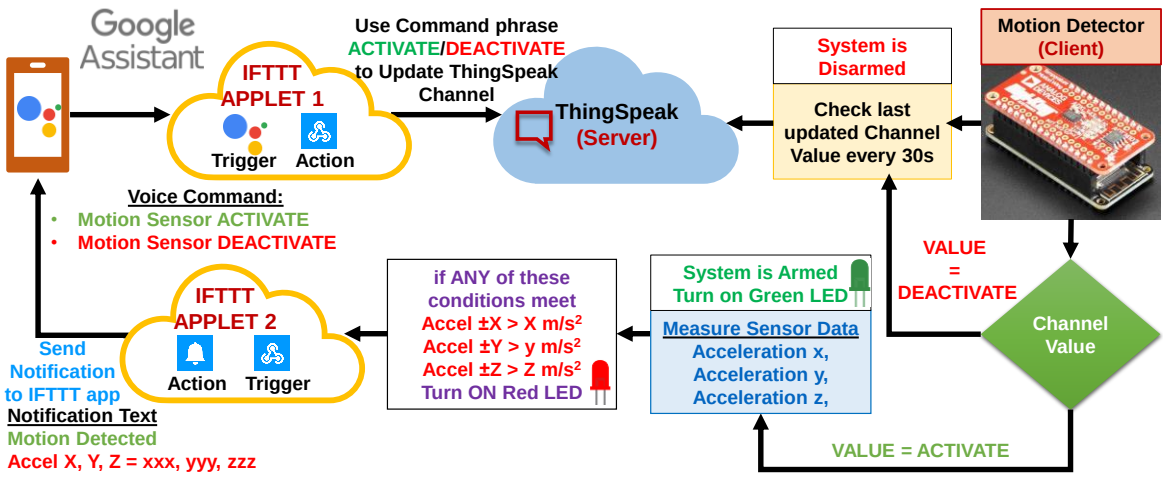

Fig 1. High level overview of Motion Detection System

2. Hardware Assembly

The MPU-6050 6-DoF Accel and Gyro Sensor board consists of an accelerometer and gyro sensor (and an additional temperature sensor) from Invensense: MPU-6050. Both sensors are connected over the **shared I²C bus**. The board already has *pull-up resistors* attached to the I²C pins (**SCL** and **SDA**), so you do not need to attach separate resistors to communicate with ESP32. You can use the STEMMA QT cable to connect this board to your ESP32 board. This link will give you a good understanding of how to connect the sensors to your board and start getting data out of the MPU-6050.

In addition to the sensor board, you will use the onboard red LED and the onboard NeoPixel as you have done in previous labs.

3. ESP32 and Sensor Initialization

3.1. Software Initialization

At the most basic level, you can use the I²C driver in MicroPython (**machine.I2C**) to communicate with the accelerometer from your ESP32 (you do not need to use the gyro sensor at all in this lab). However, for this lab, you are allowed to use the adafruit-circuitpython-mpu6050 library, which makes interactions with the sensor far easier.

3.1.1.1. Initializing and Calibrating the Accelerometer and Gyroscope

Initializing the sensor can be done as below:

```
import board
import busio
import adafruit_mpu6050

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

(This is only an example. You are welcome to use any other library other than adafruit_6050.)

Show an appropriate message after this.

(Source: https://circuitpython.readthedocs.io/projects/mpu6050/en/latest/)

3.1.2. Calibrate the Accelerometer

The accelerometer data needs to be calibrated before use. For instance, if your sensor board *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. Check datasheet for more details on **calibration**. *Your program should display a message on the terminal after completion of the calibration.*

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 (it's 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.

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**

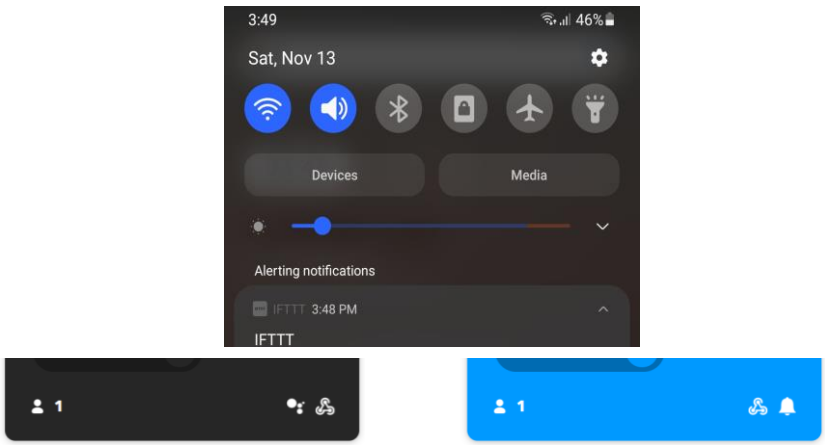

Fig 3. Example of (a) IFTTT Applet 1 (b) IFTTT Applet 2. Once you build the applets, they appear under 'My Applets' on the IFTTT website.

in applet 2. Create the IFTTT applets accordingly. You can use the following tutorial as a guidance. Make sure to keep things simple!

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 previously 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 lab4 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 and you are able to read the 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 to activate the motion sensor. 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 onboard NeoPixel to **GREEN** 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.
   - 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.**
3. Move the sensor board. The ESP32 should now trigger IFTTT applet 2 and send a notification to your phone (IFTTT app).
4. ESP32 should check ThingSpeak channel every 30 seconds. If channel value is '**ACTIVATE**', continue detecting motion and send notifications for around 1 minute.
5. **You should show that you are receiving notifications in your video.**

6. Give the voice command to Google Assistant to deactivate the motion sensor. The IFTTT applet 1 gets triggered and sends the value '**DEACTIVATE**' to the ThingSpeak channel.
7. ESP32 now sees that the channel value is '**DEACTIVATE**'. Stop measuring sensor values and turn OFF the NeoPixel to indicate that system is disarmed now. Moving the sensor SHOULD NOT SEND any notification to your phone. You also have to show this in your video.

NOTE: You need to move the ESP32 and sensor board 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. Points will be deducted for any deviations. You need to turn in your code on Brightspace. Please create a directory named ***username*_lab5**, 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***: Please include a brief description of your system in this text file (any assumptions, any special conditions needed for your system to work, any limitations, etc.)
3. Additional python files used by the motion_detector.py file (*if any*)

Zip the files and name it as *username*_lab5.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_lab5.zip
   |---username_lab5 (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 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.

8. Grading

We will use a combination of manual code inspection and your submitted video for evaluation. If your code is unusually complex or different from expected solutions, you may be asked to attend an office hour and explain your code.

NOTE: Follow the lab document strictly when using different peripherals/modules/packages. Points will be deducted if you fail to follow the lab instructions. If anything is NOT mentioned explicitly, you can use package/module to write your program.

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

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

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