ECE40862: Software for Embedded Systems

Fall 2021

Lab 5 – Real-Time Motion Detection System using ESP32, ThingSpeak and IFTTT

To be done individually; Due by 11:59pm, Sunday, December 5, 2021.

1. Overview

In this assignment, you are going to design a **Real-Time Motion Detection System** using the ESP32 Feather board and the Adafruit Sensor Featherwing. 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.

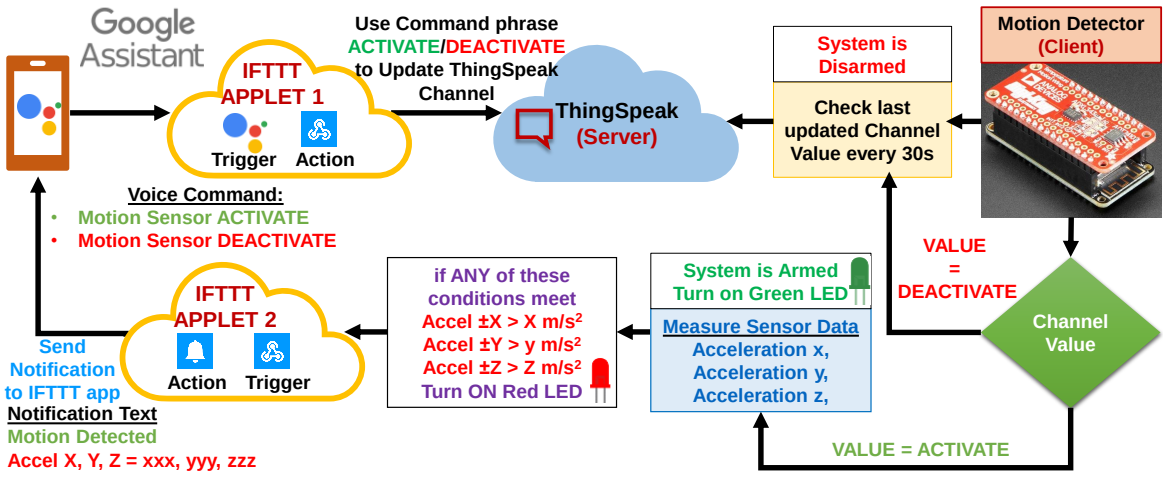

Fig 1. High level overview of Motion Detection System

2. Hardware Assembly

The featherwing consists of two specific sensors from Analog Devices: ADXL343 triple-axis Accelerometer and ADT7410 precision Temperature Sensor. Both sensors are connected over the **shared I²C bus**. Some of the features of the accelerometer sensor are described here.

- **ADXL343** is an I²C accelerometer sensor with 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.

To start with this assignment, you should solder *the Short Feather Male Headers (12 pin and 16 pins)* with the sensor Featherwing. As shown in Fig. 2(a), the 12 pin headers should be soldered to the 12 pin breakout pads and 16 pin headers should be soldered to the 16 pin breakout pads on the featherwing. You must solder the headers to the **outer** breakout pad.

Before you begin soldering, remember that the short pins of the male headers should poke through the featherwing **TOP SIDE**. It will be easier to solder if you insert both the headers into a breadboard - **LONG SIDE down**. Place the featherwing over the pins so that the short side poke through the breakout pads on both the sides (12 and 16 pins) and solder all the 28 pins. More details can be found on this link (check the section '*Soldering in Plain Headers*'): https://learn.adafruit.com/adafruit-huzzah32-esp32-feather/assembly#soldering-in-plain-headers-3-7. Although this link shows soldering of headers on the ESP32 board, the instructions should be same as pins and size of the ESP32 and sensor Featherwing are the same.

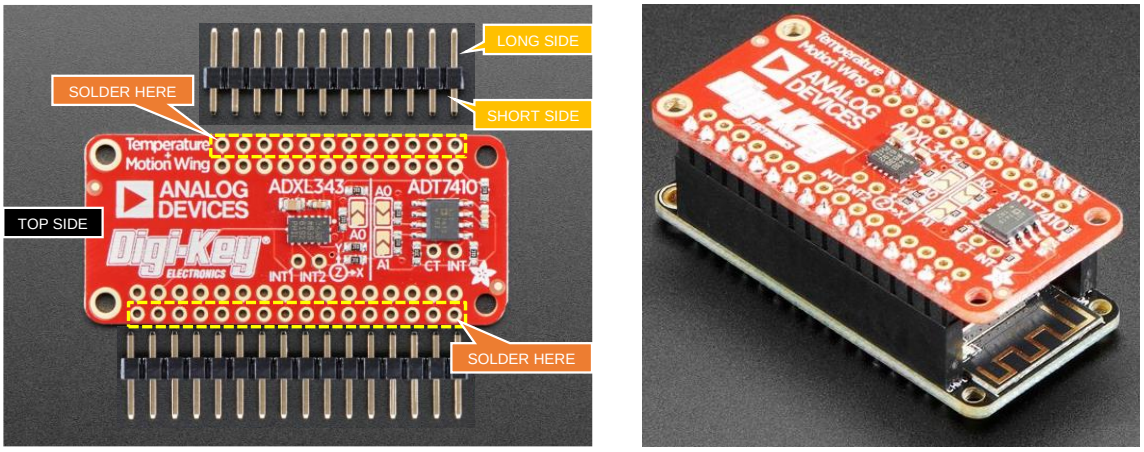

Fig 2. (a) Soldering Headers on Sensor Featherwing (b) ESP32 and Sensor Featherwing Assembly

Once you have successfully soldered both the headers on to the featherwing, you can directly insert it into female headers on your ESP32 board. The complete assembly is shown in Fig. 2(b). The 3V, GND, and I²C (SCL, SDA) pins of the ESP32 coincide with the same pins on the featherwing. You do not need to make additional connections between ESP32 and sensor.

3. ESP32 and Sensor Initialization

3.1. Hardware Interfacing

Interface the following components to the ESP32 board:

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

The featherwing already has *pull-up resistors* attached to the I^2C pins (**SCL** and **SDA**). You do not need to attach separate resistors to communicate with ESP32.

3.2. Software Initialization

Both the accelerometer sensor and temperature sensor are connected to the shared I^2C bus on the featherwing. You can use the I^2C driver in MicroPython (**machine.I2C**) to communicate with **ONLY THE ACCELEROMETER** sensors by constructing I^2C bus on your ESP32. You do not need to use the temperature sensor. Perform the following operations:

- 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 like ADXL343 expose data using memory or register addresses. This means you can interact with sensor using both its I^2C address and the address of a register or memory location in the device. Tutorials on I^2C communication are available here. Check ADXL343 datasheet to understand how they expose data and the memory or register addresses to use.

3.3.1. Initialize Accelerometer

Start the accelerometer initialization process by checking the device ID (*find out from the corresponding datasheet*). If the device ID is inaccessible or incorrect, your program should return an error message. Upon successful checking, configure the following settings in the ADXL343 using I^2C. *Your program should display adequate messages on the terminal after completion of all the initialization steps.* You can also use values other than recommended here but ensure that you are able to read sensor data properly.

- Set to **10-bit full-resolution mode** for output data (X-, Y-, and Z-axis)
- Set range to **±2g**
- Set output data rate (ODR) to **400 Hz (optional)**

3.3.2. Calibrate Accelerometer

The accelerometer data needs to be calibrated before use. For instance, if your ESP32 and Featherwing 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. Check datasheet for more details on **offset calibration**. *Your program should display appropriate 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 (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: 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!

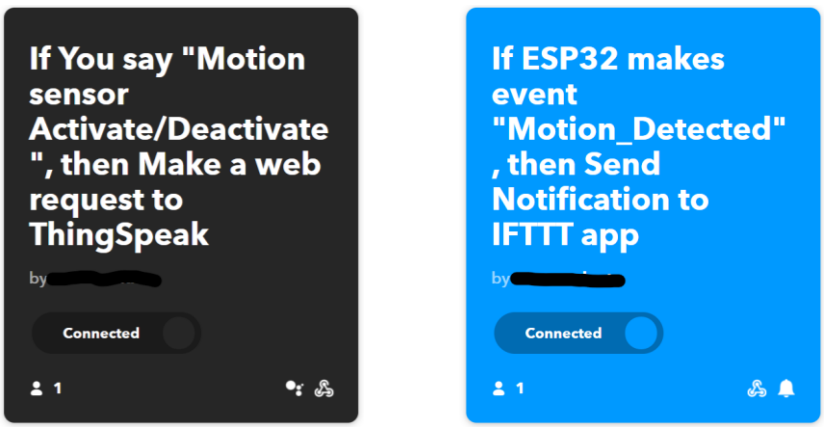

Fig 3. Example of (a) IFTTT Applet 1 (b) IFTTT Applet 2. Once you build the applets, they appear under 'My Applets' on the IFTTT website.

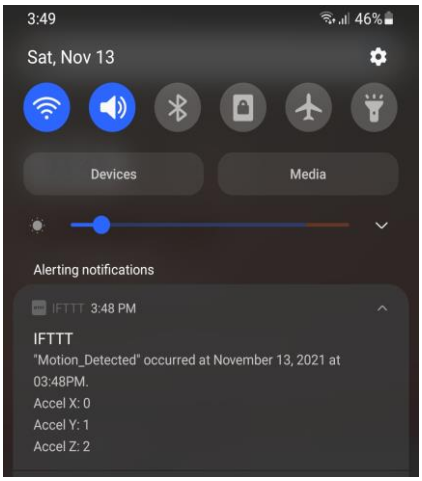

Fig 4. Notification received on IFTTT app on the phone

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 4 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 featherwing 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.

- 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.**
  The 3 different axes of motion: X, Y, Z, and corresponding motions are indicated in Fig. 5.

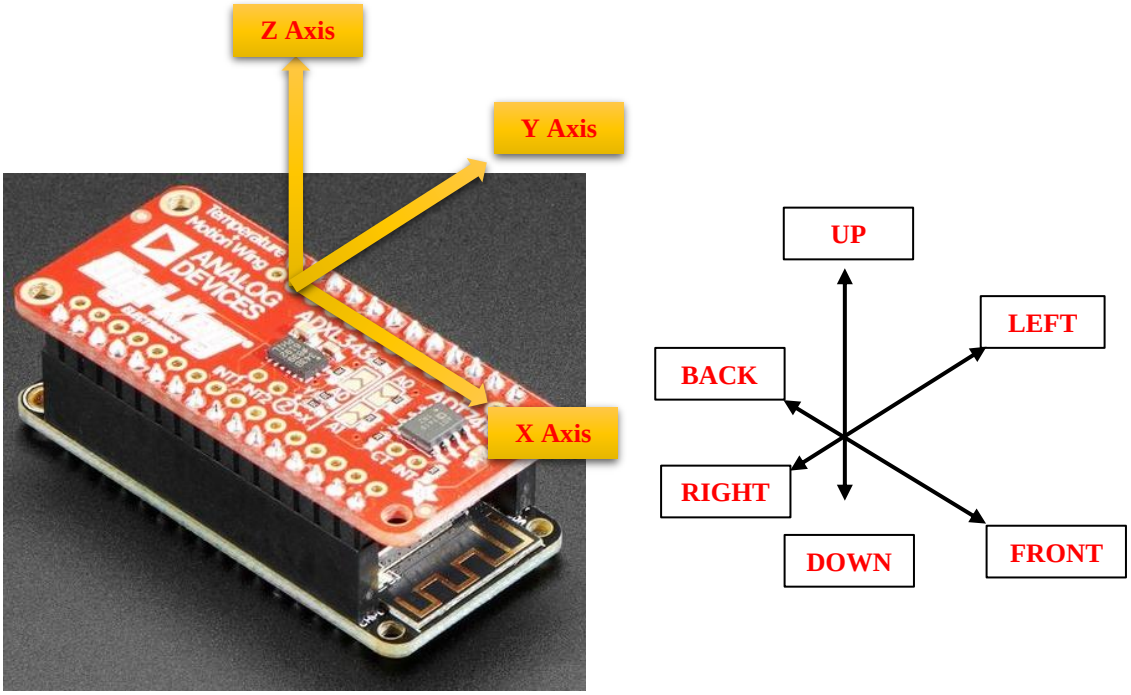

Fig 5. Motion Detector Axes Orientation and Motion Notations

3. Move the featherwing 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.**
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 featherwing assembly 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 FeatherWing 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. 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***: 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*_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 automatic grading scripts, manual code inspection, as well as 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] ADXL343 triple-axis Accelerometer: https://www.adafruit.com/product/4097

[10] ADXL343 datasheet: https://www.analog.com/media/en/technical-documentation/data-sheets/ADXL343.pdf

[11] Soldering Headers on Feather Instructions: https://learn.adafruit.com/adafruit-huzzah32-esp32-feather/assembly#soldering-in-plain-headers-3-7

[12] Guide to Soldering: https://learn.adafruit.com/adafruit-guide-excellent-soldering

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

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