ECE40862: Software for Embedded Systems

Fall 2020

Lab 5 – Implementing a Flying Car using Accelerometer Sensors

To be done individually; Due by 11:59pm, Monday, November 16, 2020.

In this lab, you, the students, are free to make decisions on your own. Here, we have given you a task and a minimal set of instructions to help you in getting the task accomplished. You must decide how to place the components/what library to use etc. You have 3 weeks for this lab, so read the datasheet and user manuals and figure out steps needed to accomplish the task, while sticking to the basic instructions. As always, you are welcome to clarify any doubts in Piazza.

1. Overview

In this assignment, you are going to program a very basic "spinner". A spinner is a flying car that is featured in the movie Blade runner (More details are available on this wiki: spinner). You will be implementing the spinner using the ESP32 Feather board and the MPU6050 Accel and Gyro sensor. This consists two specific sensors: 6-axis accelerometer and a gyro sensor (and an additional temperature sensor). The sensors are connected over the **shared I^{2}C bus**.

You can use the JST SH 4-pin to Premium Male Headers Cable to connect this to your ESP32 feather board, as shown here.

This link will give you a good understanding of how to connect the sensors to your board and start getting data out of the MPU6050.

2. Programming Exercises

2.1. Hardware Interfacing

Interface the following components to the ESP32 board:

- Three external **LEDs** (red, yellow, green) as GPIO OUTPUTs.
- Two external **push buttons** as GPIO (DIGITAL) INPUTs.
- MPU6050 sensor board.

2.2. Software Implementation

2.2.1. Initialization

All the three sensors (accelerometer, gyro, temperature) are connected to the shared I^{2}C bus on the MPU6050. You can use the I^{2}C driver in MicroPython (**machine.I2C**) to communicate with both of the sensors by constructing I^{2}C bus on your ESP32. Perform the following operations:

- Initialize LEDs, Switches, Interrupts.
- Use the schematics and ESP32 manuals to create the I^{2}C bus using proper pins.
- Set the I^{2}C communication speed to 400 kHz (optional).
- Detect switch presses using **interrupts**.
    - If you press SWITCH1, use **onboard LED** (NOT external LEDs) as GPIO output and turn it ON and implement the actions outlined in section 2.2.2 (*Interfacing Sensors*). LED should stay ON unless you press SWITCH2.
    - If you press SWITCH2, turn off **the onboard LED**, use it as PWM output and implement the actions outlined in section 2.2.3 (Spinner Demo). This onboard PWM controlled LED should now behave as described in section 2.2.3.3 (*Detect Temperature*) unless you press SWITCH1.

2.2.2. Interfacing Sensors

I^{2}C sensors expose data using memory or register addresses. Normally, you must interact with sensor using both its I^{2}C 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.

2.2.2.1. 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/)

The accelerometer and gyroscope need 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/s^2** or **0 g** and for Z should be **9.8 m/s^2** or **1 g** which is the default acceleration of gravity.

Note1: Fix the sensor firmly on to the ESP32 (or to the breadboard). You can use whichever method you like, but make sure that you do not damage the components!)

Note2: Your program should display appropriate message on the terminal after completion of the calibration.

2.2.2.2 Spinner Demo

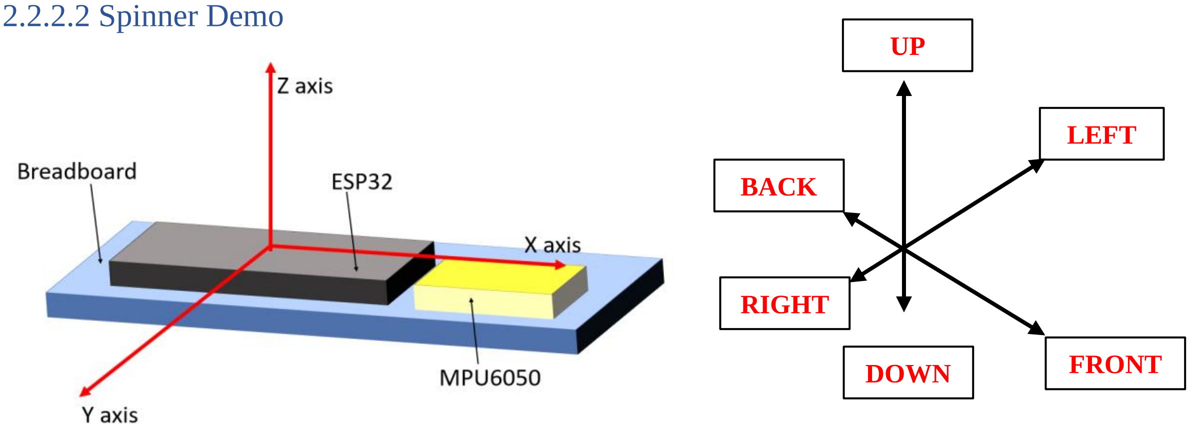

Figure 1. Spinner Axes Orientation and Motion Notations

Read sensor data from the MPU6050 using the I^{2}C protocol and implement the following functionalities, so that your spinner behaves as mentioned. You should calculate **velocities** and **tilt angles** in 3 axes from your sensor data, as well as measure temperature. Display all this information on the terminal continuously. The 3 different axes of motion: X, Y, Z and corresponding motions are indicated in Fig. 1.

2.2.2.2. Detect Spinner Velocity

- **Detect velocity in X axis**: The max speed it can move in **±X** direction i.e., **back** and **front** is X m/s. When this is violated, turn on the external **Red LED**.
- **Detect velocity in Y axis**: The max speed it can move in **±Y** direction i.e., **left** and **right** is Y m/s. When this is violated, turn on the external **Red LED**.
- **Detect velocity in Z axis**: The max speed it can move in **+Z** direction (**up**) is Z m/s. When this is violated, turn on the external **Red LED**. However, the spinner can move downwards i.e., in **-Z** direction at any speed. Hence, Red LED should NOT be turned on regardless of the downward velocity.
- **Display velocity**: Display all three velocities on the terminal.
  NOTE: The accelerometer measures the acceleration along the 3 axes. In order to calculate the speed, you can use the information given in this link.

2.2.2.3. Detect Spinner Tilt Angles

- **Measure tilt angles (pitch, roll, and theta)**: In order to detect the angles of the sensor board in three dimensions, the pitch, roll and theta are sensed using the outputs of the sensor. Three tilt angles are:
    - **Pitch (ρ)** is defined as the angle of the X-axis relative to ground.
    - **Roll (φ)** is defined as the angle of the Y-axis relative to the ground.
    - **Theta (θ)** is the angle of the Z-axis relative to ground.
- Maximum permissible value of any of the tilt angles is ±30°. If during its motion, your spinner tilts by more than ±30° in any axis (pitch, roll, theta), turn on the **Yellow LED**. Turning *ON* or *OFF* this yellow LED should be independent of the velocity of the spinner, i.e., in a scenario where your spinner is moving at a high velocity in a tilted fashion, both red and yellow LED might be turned on.
- **Display tilt angles**: Display all three tilt angles on the terminal.
- When the spinner is *motionless/completely still*, turn on the **Green LED**. Any motion in the spinner should turn it off.

2.2.2.4. Detect Temperature

- **Create PWM Output**: Initialize onboard LED as *PWM* output with duty cycle of 512 and a frequency of 10 Hz.
- **Detect temperature change**: If the change in measured temperature is more than 1°C, change the led PWM frequency by ±5 Hz for every ±1°C change in temperature. The onboard led blinking frequency should now increase/decrease with temperature change.
- **Display temperature**: Display the current temperature on the terminal.

2.2.3. Overall Workflow

Fig. 2 explains the state transition diagram you need to follow for this assignment. Start your program with the initialization step (2.2.1). This section needs to run only once. Afterwards, use switch1 and switch2 presses to change your state between interfacing sensors to spinner demo. It is obvious that you need to interface your sensors before demo for the first time.

You need to move the board by hand to mimic the motion of the spinner. You are free to choose the values for X, Y, Z m/s as the maximum velocities in 3 axes. However, make sure that the values chosen should distinguish the given states (e.g. the difference of velocities below +X m/s and velocities above +X m/s should be clearly visible to the observer's eyes).

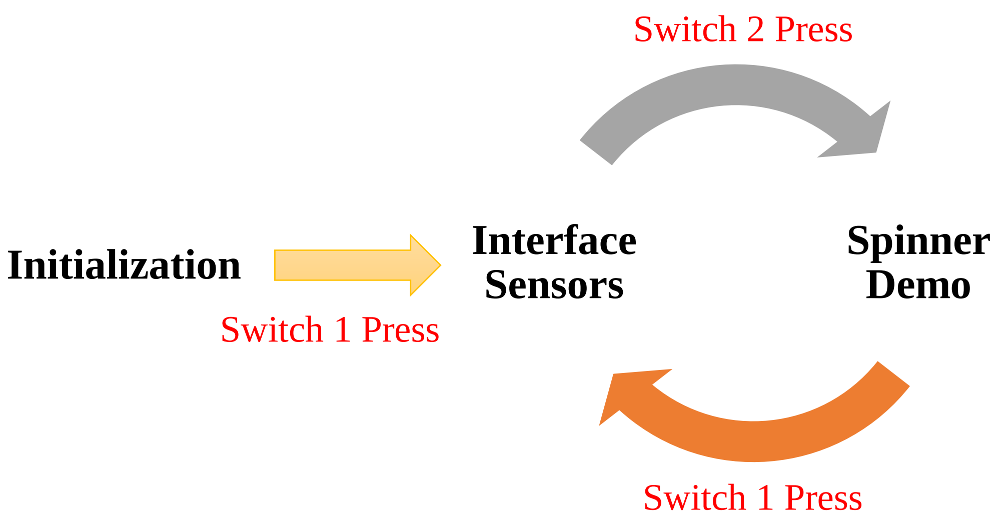

Figure 2. State Transition Diagram of Spinner using ESP32 and Sensor

3. Submission

Make sure you follow these instructions precisely. Points will be deducted for any deviations. You need to turn in your code on Blackboard. Please create a zip file named ***username*_lab5.zip**, where username is your CAREER account login ID. *This should contain only the following files, i.e., no executables, no temporary files, etc.*

1. ***spinner.py***: your program for the Spinner.
2. ***myDecisions***.txt: As mentioned in the beginning, you should design your own spinner. Please include a brief description of what you did and why in this text file.

Upload the .zip file to Brightspace. Make sure the zip file contains no subfolders and only these two files as specified.

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

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

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

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

[10] How to convert Acceleration to Tilt Angles: http://aitendo3.sakura.ne.jp/aitendo_data/product_img/sensor/MMA7260Q/MMA7260QT_AN3461.pdf

[11] How to calculate Speed/Velocity from Acceleration: https://physics.stackexchange.com/questions/153159/calculate-speed-from-accelerometer