ECE40862: Software for Embedded Systems

Fall 2019

Lab 5 – Implementing a Flying Car using Accelerometer Sensors

To be done individually; Due by 11:59pm, Wednesday, October 9, 2019.

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 Adafruit Sensor Featherwing. The featherwing consists two specific sensors from Analog Devices: an ADXL343 triple-axis Accelerometer and an ADT7410 precision Temperature Sensor. Both sensors are connected over the **shared I²C bus**. Some of the features of these two sensors 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 range gives more resolution for slow movements, the higher range is good for high speed tracking.
- **ADT7410** is an I²C temperature sensor, with 16-bit 0.0078°C temperature resolution and 0.5°C temperature tolerance.

2. Hardware Assembly

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. 1, 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 (Fig. 1).

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.

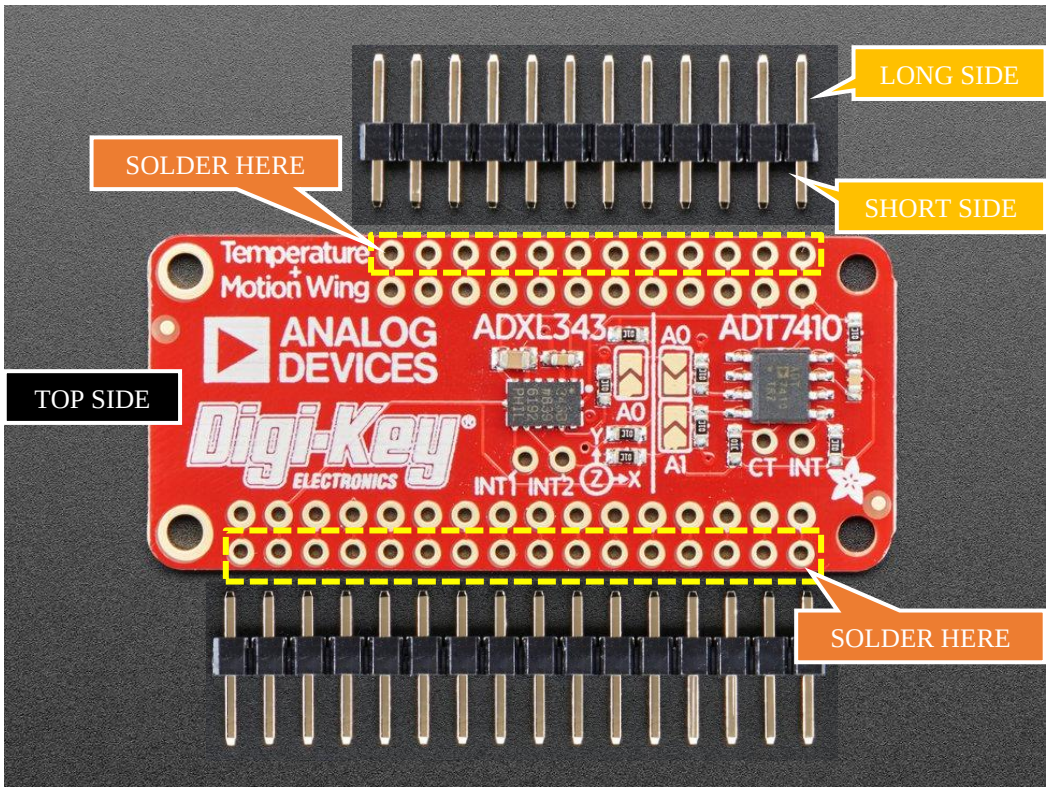

Figure 1. Soldering Headers on Sensor Featherwing

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. The 3V, GND, and I^2C (SCL, SDA) pins of the ESP32 coincide with the same pins on the featherwing. You do not need to make any additional connections between ESP32 and sensor.

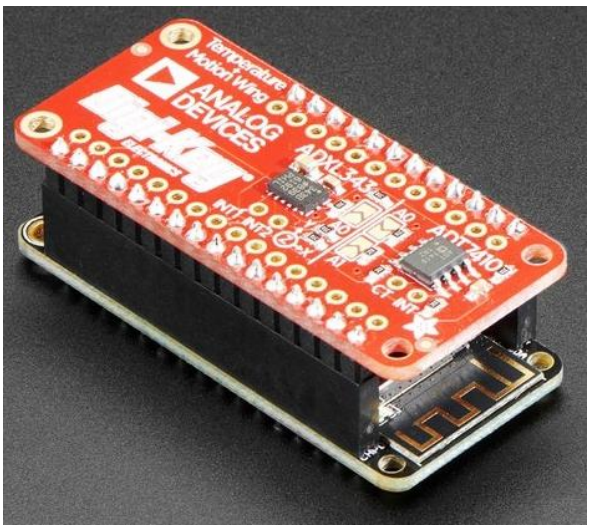

Figure 2. ESP32 and Sensor Featherwing Assembly

3. Programming Exercises

3.1. Hardware Interfacing

Interface the following components to the ESP32 board:

- Three external **LED**s (red, yellow, green) as GPIO OUTPUTs.
- Two external **push buttons** as GPIO (DIGITAL) INPUTs.
- I^2C pins on your ESP32 (**SCL** and **SDA**) do not have *pullup resistors*. You must add them to communicate with the featherwing.

3.2. Software Implementation

3.2.1. Initialization

Both the accelerometer sensor and temperature sensor are connected to the shared I^2C bus on the featherwing. You will be using the I^2C driver in MicroPython (**machine.I2C**) to communicate with both of the sensors by constructing I^2C bus on your ESP32. Perform the following operations:

- Initialize LEDs, Switches, Interrupts.
- Use the schematics and ESP32 manuals to create the I^2C bus using proper pins.
- Set the I^2C communication speed to 400 kHz.
- 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 3.2.2 (*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 3.2.3 (Spinner Demo). This onboard PWM controlled LED should now behave as described in section 3.2.3.3 (*Detect Temperature*) unless you press SWITCH1.

3.2.2. Interfacing Sensors

I^2C sensors like ADXL343 and ADT7410 expose data using memory or register addresses. This means you interact with sensor using both its I^2C address and the address of a register or memory location in the device. You can find tutorials on I^2C communication on this link. Check ADXL343 datasheet and ADT7410 datasheet to understand exactly how they expose data and the memory or register addresses to use.

3.2.2.1. 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*.

- Set to **13-bit full-resolution mode** for output data (X-, Y-, and Z-axis)
- Set range to **±2g**
- Set output data rate (ODR) to **800 Hz**

3.2.2.2. 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/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. Check datasheet for more details on **offset calibration**. *Your program should display appropriate message on the terminal after completion of the calibration.*

3.2.2.3. Initialize Temperature Sensor

Start the temperature sensor 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 ADT7410 using memory/register operations using I^{2}C. *Your program should display adequate message on the terminal after completion of the initializations.*

- Set to **16-bit high resolution** for temperature data

3.2.3. Spinner Demo

Read accelerometer and temperature sensor data from the featherwing using 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 accelerometer 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. 3.

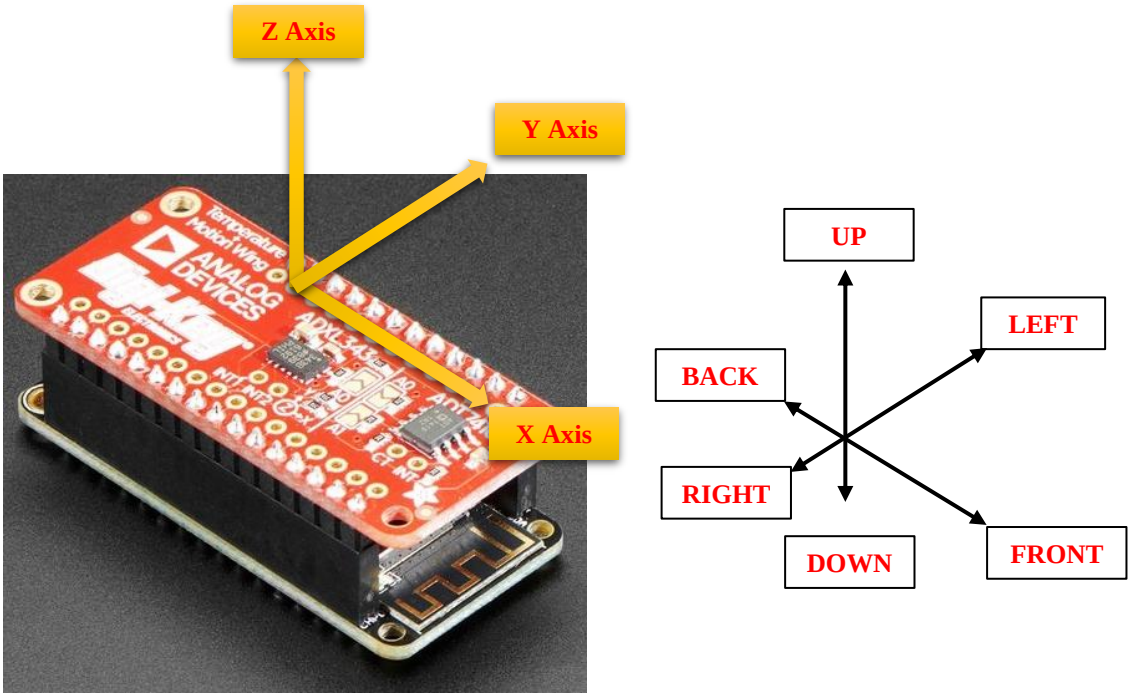

Figure 3. Spinner Axes Orientation and Motion Notations

3.2.3.1. 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.

3.2.3.2. Detect Spinner Tilt Angles

- **Measure tilt angles (pitch, roll, and theta)**: In order to detect the angles of the accelerometer in three dimensions, the pitch, roll and theta are sensed using all three outputs of the accelerometer. Use X, Y, and Z axis readings to calculate these 3 tilt angles. You can use this application note from freescale semiconductor for understanding relation between tilt and accelerometer readings. 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 gravity.
- 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.

3.2.3.3. 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.

3.2.4. Overall Workflow

Fig. 4 explains the state transition diagram you need to follow for this assignment. Start your program with the initialization step (3.2.1). 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).

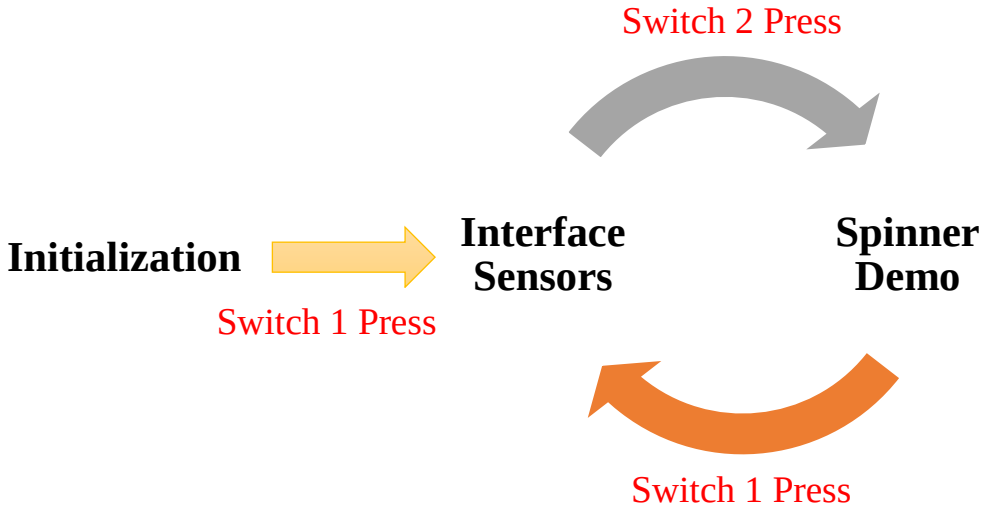

Figure 4. State Transition Diagram of Spinner using ESP32 and Sensor Featherwing

4. 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 directory named ***username*_lab5**, where username is your CAREER account login ID. *This directory should contain only the following files, i.e., no executables, no temporary files, etc.*

1. ***spinner.py***: your program for Spinner using ESP32 and Sensor Featherwing.

Zip the files and name it as *username*_lab5.zip and **upload the .zip file to Blackboard**.

5. Grading

20 students will be randomly selected for lab demo, they will be informed on Thursday, 10th October, 10 AM *via your Purdue email id*. If you are selected for the demonstration, you will need to login to any of the computers in the **EE217 lab**, download your submitted source codes from Blackboard, copy them to your ESP32 board *using rshell* and show the TA that your code works correctly. You will also be asked to show the TA your code listing and answer conceptual questions about your code. Marks are only awarded if you satisfy all requirements.

In addition, automatic evaluation scripts will be executed on your uploaded source codes on Blackboard and you will be graded according to the evaluation results.

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] ADT7410 precision Temperature Sensor: https://www.adafruit.com/product/4089

[12] ADT7410 datasheet: https://www.analog.com/media/en/technical-documentation/data-sheets/adt7410.pdf

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

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

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

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