E

Project 4: MPU6050 Accelerometer and LCD Display

Overview

This project will introduce to you the concepts of reading sensor values, data storage inside a structure, and displaying recorded values on an LCD display. You will read the accelerometer and gyroscope readings from the MPU6050 sensor (also referred to as the GY-521 module) and store them in an internal structured data packet for reporting to the LCD module. Your LCD1602 module will have several different pages: one page will show the accelerometer values, another will show the gyroscope values, a third will show roll and pitch data, and others will show any additional information you desire. A button will be used to cycle between the different pages and a buzzer will be used to provide auditory feedback on every press.

See the Extra Credit section for more details.

Graduate Students

You will have some additional work for this project. Since an accelerometer provides accelerations, it will be your task to extrapolate (integrate) velocity and position as well as the other tasks outline above. You will be required to store this information in your storage packet and display the velocity and position values on separate pages on the LCD module. Since the MPU6050 module lacks a magnetometer, it will also be your task to integrate the heading angle and save it to the telemetry packet. You will also program a button to act as velocity, position, and heading reset after you hold the button down for a certain amount of time.

Requirements

For completion of this project, you must demonstrate the following:

- ▶ Successful wiring of the MPU6050 IMU, LCD1602 module, piezoelectric buzzer, and button input (with appropriate debounce filtering)
- ▶ Reading and calculating a variety of values from the MPU6050 sensor and storing them in a telemetry packet.

    - Acceleration X,
    - Acceleration Y,
    - Acceleration Z,
    - Gyroscope X,

- Gyroscope Y,
- Gyroscope Z,
- Roll,
- Pitch

▶ Display the values on distinct pages on the LCD display with appropriate labels and units

- Page 1: Acceleration values
- Page 2: Gyroscope values
- Page 3: Orientation values

▶ Use a button tied to an interrupt service routine on the Arduino to cycle between the different LCD pages. Use a buzzer to provide auditory feedback on each button press.

Submission

You will be required to submit the following on Canvas:

▶ a well-organized and documented schematic of the project setup
▶ the source code file

Grading

You will be graded on the following criteria:

Criterion	Points
Efficacy	50
Well organized and neat schematic	20
Well organized and neat source code	30
Extra Credit Challenge 1 *	25
Extra Credit Challenge 2	25
Extra Credit Challenge 3	25
Extra Credit Challenge 4	25

Extra Credit

There exists many opportunities for extra credit on this assignment. Ultimately, this is your chance to really explore and learn how to interact with sensors on a low level and gain an in-depth understanding of the relationship between microcontrollers and sensors. To that end, it is highly encouraged to pursue the following challenges and earn as many points as possible. A successful demonstration of *all* of the challenges below will earn you **an exemption from the midterm exam**. If you fail to finish all the

---

* Graduate students: this will count towards your normal score

challenges, the appropriate amount of points will still be given to you and used as standard extra credit for this assignment.

In your submission, please include a note of which challenges you have completed. Your source code will also need to broken into specific files for each challenge, or have each challenge highlighted clearly in different sections. *If it is not clear where your challenge-specific code is, it may not be counted.*

Challenge 1: Sensor Fusion - Attitude and Heading Reference System

Note: This challenge is *mandatory* for graduate students.

In this challenge, you are tasked with combining the accelerometer and gyroscope readings from the MPU6050 into a unified AHRS. There are a multitude of different ways to accomplish this task, but at the end, you should be able to store the values into a telemetry packet and display them on separate pages on your LCD module. You will be required to capture and store:

- ▶ Acceleration (X, Y, Z)
- ▶ Gyroscope (X, Y, Z)
- ▶ Orientation (Roll, Pitch, Yaw [heading])
- ▶ Velocity (X, Y, Z)
- ▶ Displacement (X, Y, Z)

A button will be used to reset the velocity, position, and heading when held for a certain amount of time.

FOR CREDIT: You must scroll through the different pages of the LCD module and show that you are capturing the appropriate data. You must also demonstrate resetting the velocity, position, and heading values with the reset button.

Challenge 2: AHRS Filtering

For information, consult the lecture notes Chapter 5, Section 5.1.5 and This excellent tutorial (focus on the 1-dimensional application for now).

You should have noticed from the AHRS challenge that the velocity, displacement, and orientation values are extremely inaccurate and drift significantly over time. This is due to the inherent inaccuracies and drift built into the accelerometer and gyroscope readings. In that challenge, you were working directly with the raw values without filtering them out. These values accumulate over time, causing the massive divergence.

Therefore, to combat these inaccuracies, we will implement some filters to increase the estimation accuracy. First, you will implement a Kalman filter to "smooth" out your raw acceleration and gyroscope data to give you better readings to use. Here, it will be beneficial to a 6-axis Kalman filter which can be created using the Arduino Kalman libary by *rfetick*. It is strongly recommended to study Kalman filtering and the basic theory before attempting this. It will make a lot more sense after some studying!

You will then feed the filtered acceleration and gyroscope data into a Mahony Filter that can generate the appropriate orientation data. You will not need to understand the theory of this filter, you may just use the example code provided by the Arduino Mahony library.

If you established and tuned the Kalman filter correctly, you should have some more accurate readings of velocity, position, and orientation. Again, update the telemetry packet with these values and display them on the LCD module on separate pages.

FOR CREDIT: you must submit a capture of a plot created from your data that shows one accelerometer axis (X, Y, or Z). One line must be the method you used previously, and one line must be the filtered estimate. Quickly discuss the results in your submission.

Challenge 3: Removing Gravity

For this challenge, we will convert the acceleration signals into linear acceleration, put that information into the telemetry packet, and display it on a separate page of the LCD module. For this challenge, it is strongly recommended to use a quaternion rotation as this method is immune to gimbal lock - a phenomenon where rotations require additional euler transformations at certain orientations.

1: You may also want to look into the Adafruit IMU maths helper libraries

To assist you, the code to convert from Euler angles to Quaternion is below:
1

```cpp
struct Quaternion
{
    double w, x, y, z;
};

// yaw (Z), pitch (Y), roll (X)
Quaternion ToQuaternion(double yaw, double pitch, double roll)
{
    // Abbreviations for the various angular functions
    double cy = cos(yaw * 0.5);
    double sy = sin(yaw * 0.5);
    double cp = cos(pitch * 0.5);
    double sp = sin(pitch * 0.5);
    double cr = cos(roll * 0.5);
    double sr = sin(roll * 0.5);

    Quaternion q;
    q.w = cr * cp * cy + sr * sp * sy;
    q.x = sr * cp * cy - cr * sp * sy;
    q.y = cr * sp * cy + sr * cp * sy;
    q.z = cr * cp * sy - sr * sp * cy;

    return q;
}

```

From there, you will define a Quaternion vector for gravity defined in the North, East, Down (NED) reference frame. [2] Then, you will rotate the gravity quaternion from the NED reference frame to the body reference frame using:

$$Q_{g,b} = Q_b^{-1} * Q_{g,NED} * Q_b$$

Finally, subtract the X,Y,Z components of the rotated Quaternion vector for gravity from the recorded body accelerations and you will have linear acceleration. *Easy right!?*

As always, display these values on a new page of the LCD module!

Challenge 4: MPU6050 Motion Detect Interrupt

The interrupt pin on the MPU6050 can be configured to serve as a motion detection trigger. This challenge's task will be to put the Arduino into a sleep mode after the initialization and use the motion detection interrupt to wake the Arduino up and read sensor data for a defined period of time before going back to sleep.

FOR CREDIT: You must demonstrate the Arduino not capturing any data (asleep), then you jolting the Arduino awake by moving the accelerometer, and the values changing on the LCD module. After a set time, these readings should stop updating as the Arduino goes back to sleep.