E

Project 4: MPU6050 Accelerometer and LCD Display

Overview

This project will introduce to you the concepts of reading sensor values and storing them in a packet, and displaying those 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 the button to act as 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 video of the project working with narration of what is occurring
- a well-organized and documented schematic of the project setup
- the source code file

Please package all of these items into a compressed (zipped) folder and upload them to Canvas.

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

* Graduate students: this will count towards your normal score

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 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 and videos of the challenge working. You will also need to include source code highlighting the specific challenge 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)

The button used to change the display page will also act as a position and velocity reset when held for a certain amount of time.

FOR CREDIT: in your submission video scroll through the different pages of the LCD module and show that you are, in fact, capturing the appropriate data. When you reach the position page, reset the position values back to 0 after a couple of seconds.

Challenge 2: AHRS Filtering

For information, consult the lecture notes Chapter 5, Section 5.1.5, TJKElectronics' excellent brief, TJKElectronics' example code, 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 Mahony filter to convert the IMU readings to orientation values. Do not worry, you will not need to research the theory or algorithm behind this filter, there is a handy library that is easy to use and has some example code.

Second, you will use a Kalman filter to estimate the velocity and displacement values. Realistically, this would be done with complex matrix math and with a single 9-dimensional Kalman filter object, but for sake of ease, we will just use six 1-dimensional Kalman filters; three for velocity X/Y/Z, and three for displacement X/Y/Z.

Again, the telemetry packet will be updated with these values and displayed on the LCD module.

FOR CREDIT: you must submit a capture of a plot created from your data that shows either roll, pitch, yaw values. One line must be the method you used previously, and one line must be the Kalman-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 points.

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

```
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;
}

```

Hint: it's just a normal gravitational acceleration vector with another term for the 'w' component of the Quaternion.

From there, you will define a Quaternion vector for gravity defined in the North, East, Down (NED) reference frame. 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: In your submission video, 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.