|  |
| --- |
| ECSE 421 – Embedded Systems  Laboratory #1 |

|  |
| --- |
| **Introduction to Arduino** |

|  |
| --- |
| Zichen Gao (261055991)  Presented to Prof. Jeremy Cooperstock  Due January 31st 2025 |

**Exercise 1: Working with an RGB LED**

|  |
| --- |
| **Task:** Using the blink sketch as an example, blink your external LED in white color using delay(amount), where amount is measured in milliseconds. |

*The result of the above task is shown in one of the videos (see attached URLs). As well, see the attached Arduino file for the source code used.*

|  |
| --- |
| **Task:** Modify your previous code to make the LED cycle through 6 distinct colors every 500ms. |

*The result of the above task is shown in one of the videos (see attached URLs). As well, see the attached Arduino file for the source code used.*

**Exercise 2: Using the Rotary Encoder, the Buzzer, and the Microphone**

|  |
| --- |
| Task: Write a program that outputs a counter on the serial monitor. The counter is initialized at 0 and is incremented by 1 for each clockwise increment of the rotary encoder. Similarly, it is decremented by 1 for each counterclockwise increment. The counter should be reset to 0 when the encoder is pressed down like a button. |

*The result of the above task is shown in one of the videos (see attached URLs). As well, see the attached Arduino file for the source code used.*

|  |
| --- |
| **Task:** Use the rotary encoder to select the buzzing frequency of the buzzer. Using the serial plotter and the microphone, find the approximate frequency at which the buzzer appears to be the loudest from the microphone's point of view. |

*The result of the above task is shown in one of the videos (see attached URLs). As well, see the attached Arduino file for the source code used.*

***The approximate frequency at which the buzzer appears to be the loudest from the perspective of the ‘Big Sound’ sensor is around 4 kHz (see the attached video). Indeed, at this frequency the sensor reads a peak value of around 60. For higher frequencies the peak value decreases and tapers off.***

|  |
| --- |
| **Task:** Repeat the previous task with the other microphone module. How does the sensitivity change? |

*The result of the above task is shown in one of the videos (see attached URLs). As well, see the attached Arduino file for the source code used.*

***The approximate frequency at which the buzzer appears to be the loudest from the perspective of the ‘Small Sound’ sensor is around 4 kHz (see the attached video). It matches the same value found in the previous task.***

***Note: I adjusted the potentiometer values for this part since I thought my microphone wasn’t picking up noise correctly. This is why the values read are in the 500-600 range.***

|  |
| --- |
| **Question:** What is the smallest value of change in voltage that the analog inputs can report?  Explain your reasoning. |

***Using a 5V power supply, the voltage range that can be read by the sensor is 5V. The ADC of the sensor is a 10-bit ADC (we know this since the maximum value that can be output from analogRead() for this specific sensor is 1024). Therefore, the smallest voltage change that can be detected is:***

**Exercise 3: The Weather**

|  |
| --- |
| **Task:** Write a program that displays the temperature and humidity in the serial monitor. |

*The result of the above task is shown in one of the videos (see attached URLs). As well, see the attached Arduino file for the source code used.*

|  |
| --- |
| **Task:** Repeat the previous task but display the temperature and humidity on the LCD display. |

*The result of the above task is shown in one of the videos (see attached URLs). As well, see the attached Arduino file for the source code used.*

**Exercise 4: You Are a Pilot**

|  |
| --- |
| **Task:** Open the serial plotter from the **Tools -> Serial Plotter** menu. Shake the sensor around in each axis. Do you see the values changing? What do they mean? |

*The result of the above task is shown in one of the videos (see attached URLs). As well, see the attached Arduino file for the source code used.*

***The values are changing as the sensor shook. There are 7 values in total, the first three (from the left) represent the acceleration in X, Y, Z, respectively, and the next three represent the angular velocity measurements in X, Y, Z, respectively. The values of the accelerometer are in units of g, where 1g = 9.81 m/s^2, whereas the units of the angular velocity are given in rad/s.***

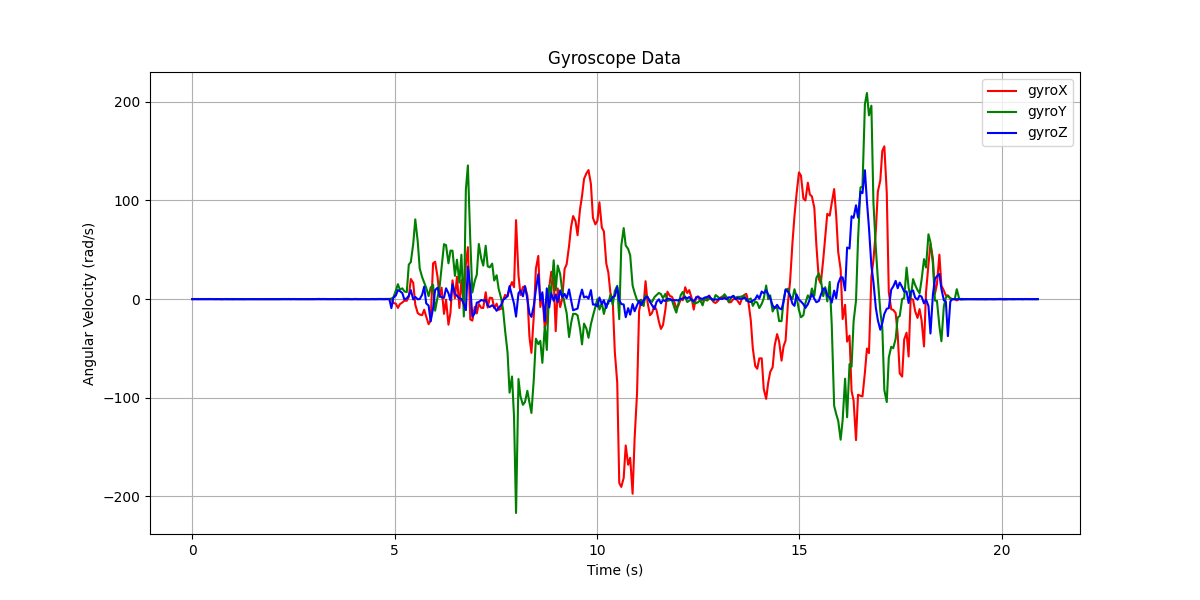
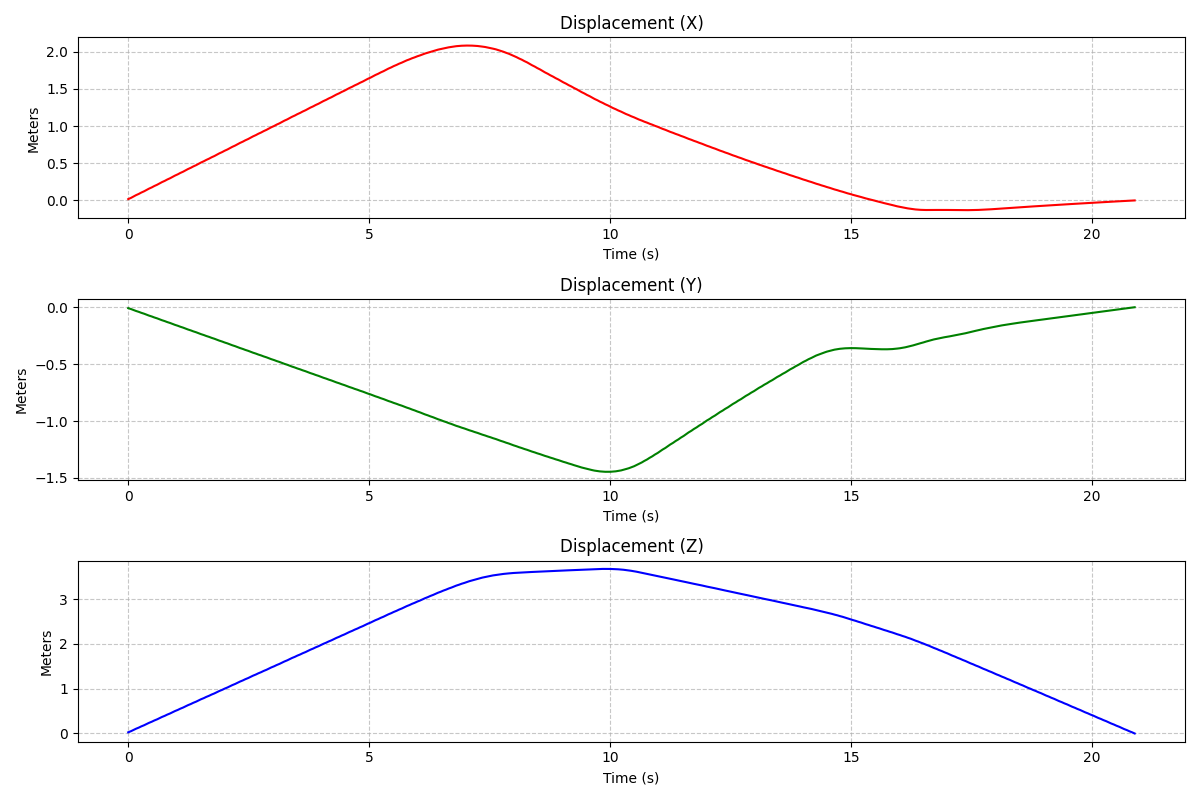
|  |
| --- |
| **Task:** Plot these data using any tool of your choice (Excel, Python, MATLAB, etc.). Use these values to estimate the displacement of the accelerometer and plot displacement on a separate graph. |

*The result of the above task is shown in one of the videos (see attached URLs). As well, see the attached Arduino file for the source code used.*

***After calibration, the values are recorded using PuTTY and are transformed into the csv format to be further processed. A python script is used (see code) to plot the raw acceleration/gyroscope data as well as the displacement plots (which are calculated using double integration and assuming zero initial velocity). The displacement plots do correctly illustrate the general direction in which I moved the sensor, however the distances plotted are incorrect (e.g. the max displacement in z-axis is shown to be 3 meters roughly, but I only moved the sensor up around 0.5 meters).***

***Note: An LLM (***[***Deepseek***](https://www.deepseek.com/)***) was used for the generation of the plotting Python script***

A graph showing a graph

Description automatically generated****** ******

|  |
| --- |
| **Task:** Choose a suitable value of N and write a moving average filter. Run your data through it. Plot this against the raw (unfiltered) signal. What are the benefits/drawbacks of having a high N compared to a low N? |

***Using Python (matplotlib, pandas) once again to calculate the moving filter. The pandas function*** [**pd.DataFrame.rolling()**](https://pandas.pydata.org/pandas-docs/stable/reference/api/pandas.DataFrame.rolling.html) ***was used to calculate the moving average for a given window size N. Plots for the filtered acceleration data are shown below for N=10, 50, 100, 200. The entire set of plots (filtered accel, filtered gyro, filtered displacement) can be seen in the 4\_imu\_calibrated folder. We can see from the inspection of the graphs below that the data gets progressively ‘smoother’ and hence the high frequency noise is eliminated. However, real acceleration data (the shape of the curve) gets distorted more and more as the value of N increases. For example, looking at the acceleration data with the filter N=200, we see that the overall shape does not look anything like the original data. From the plots below, to maintain the integrity of the data and reduce noise, a value of N of 10-50 is desirable.***

***Note: An LLM (***[***Deepseek***](https://www.deepseek.com/)***) was used for the generation of the plotting Python script***

A graph of a graph

Description automatically generatedA graph of a graph

Description automatically generated with medium confidenceA graph of different colored lines

Description automatically generatedA graph of different colored lines

Description automatically generated

|  |
| --- |
| **Task:** With your accelerometer and gyroscope calibrated, connect the LCD to your board. Design an airplane instrument panel. Display the yaw, pitch, and roll of your plane, as well as the elevation (estimated through the accelerometer readings). |

*The result of the above task is shown in one of the videos (see attached URLs). As well, see the attached Arduino file for the source code used.*

|  |
| --- |
| **Task:** Flesh out your cockpit. Use the buzzer as a [stall](https://simple.wikipedia.org/wiki/Stall_(flight)) warning for your plane. If the pitch goes above a certain threshold, the buzzer should be heard. |

*The result of the above task is shown in one of the videos (see attached URLs). As well, see the attached Arduino file for the source code used.*