Sebastian Lee

A12041475

02/09/18

**Lab 1: Python and Arduino Communication**

1. **Introduction**

This lab consisted of 8

1. **Objective 1: Polling Sensors from Arduino**

In this objective, data was sampled. The data was continuously readable so as to detect changes in the sensor output. Using delays, we attempted to poll at a rate of 100Hz (unable to get a perfect 100Hz rate though). Using Fig. 1, we can determine that this is not a good data collection strategy since our time difference changes.

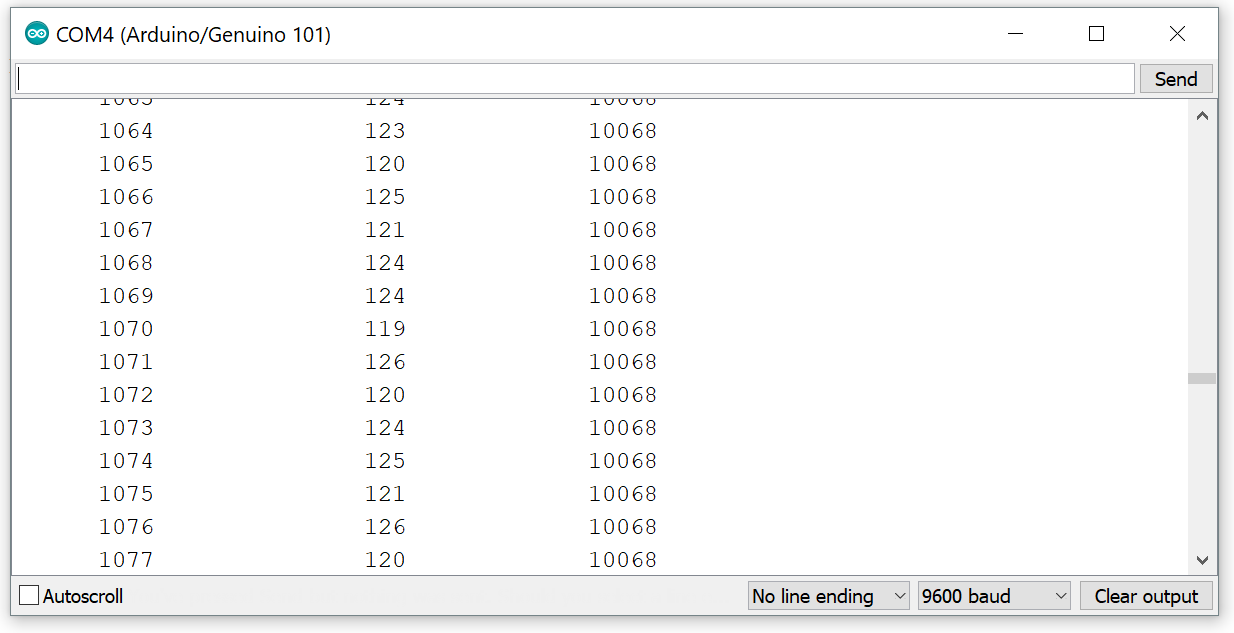


Fig. 1: Sample number (left), Votlage output (middle), Time Difference (output)

1. **Objective 2: ISR with Sensors and Arduino**

Here, we started using the Curie Timer One. This allowed us to capture information evenly spaced out by providing timer based interrupts, a very powerful tool for homogeneous sampling. This approach is better at data collection because by using a timer, we can guarantee that data will be sampled at a constant time we give it. This is also shown in Fig. 2.

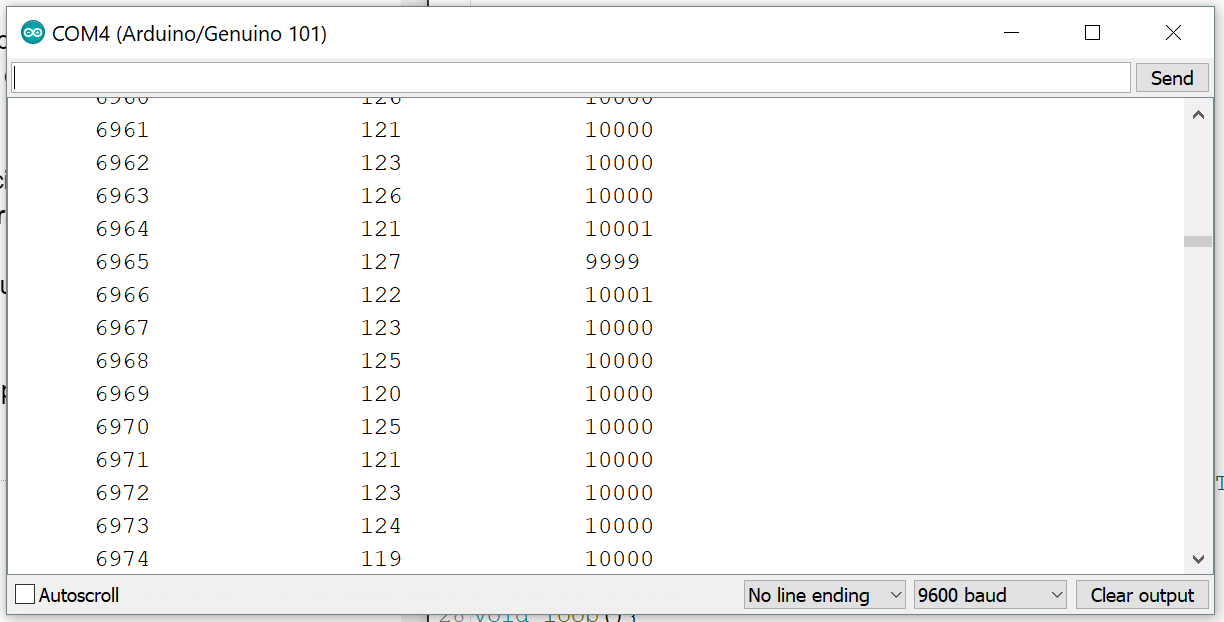


Fig. 2: Serial Monitor displaying messages sent by Arduino after resetting the board

1. **Objective 3: Timing sensor readings from the Arduino**

In Objective 3, we were introduced to the sensors that are embedded unto the Arduino: the Accelerometer and Gyroscope. To first be able to sample this data correctly, we printed out how fast it took for data to be read using the function micros(). This was done for each sensor, including an analog pin. Fig. 3 shows the recorded times.

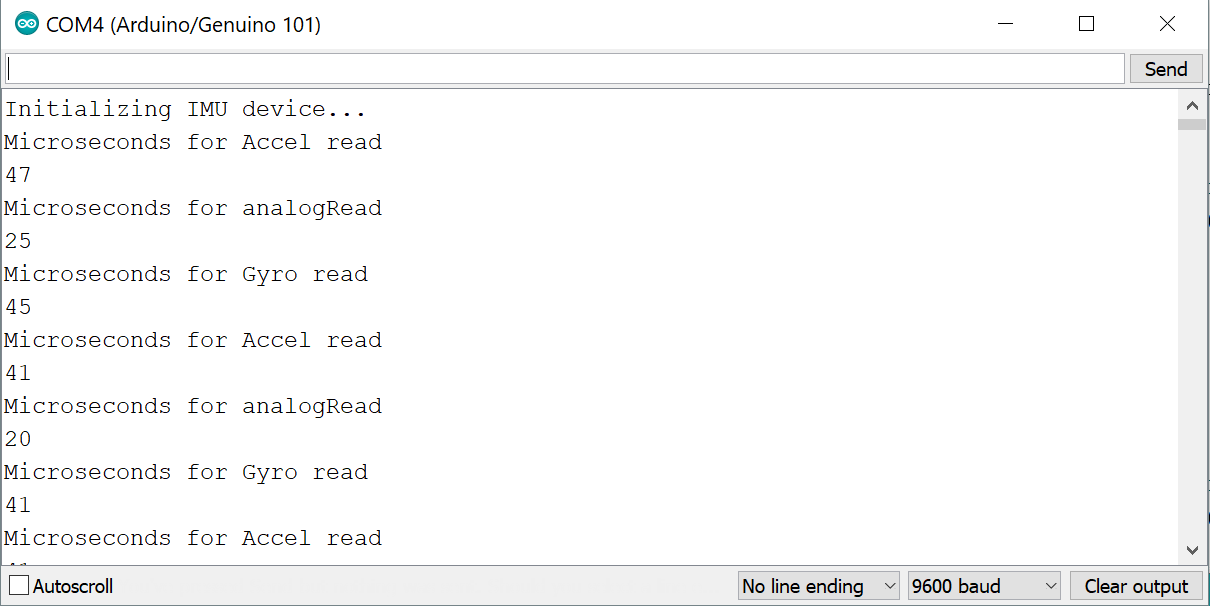
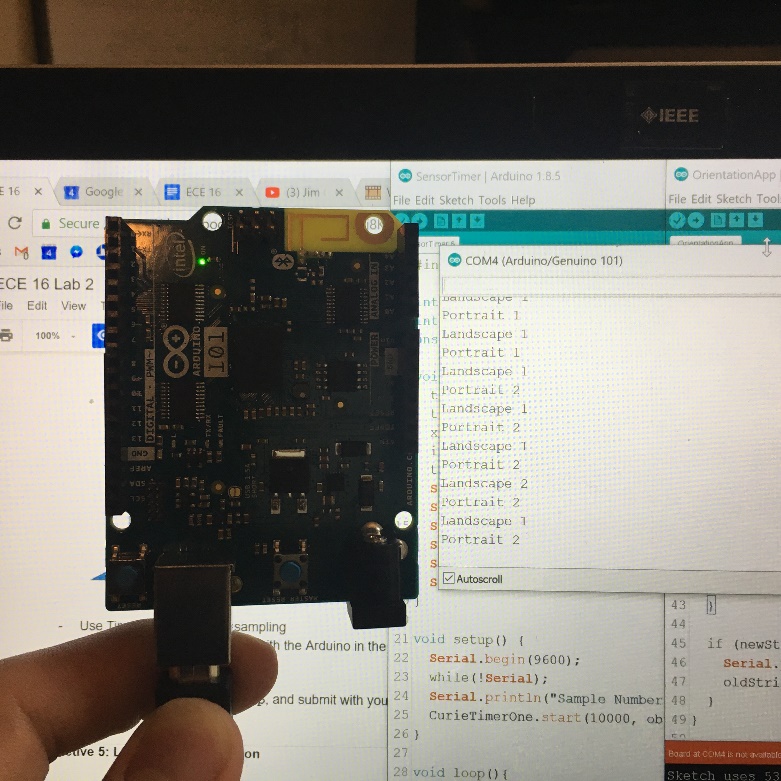
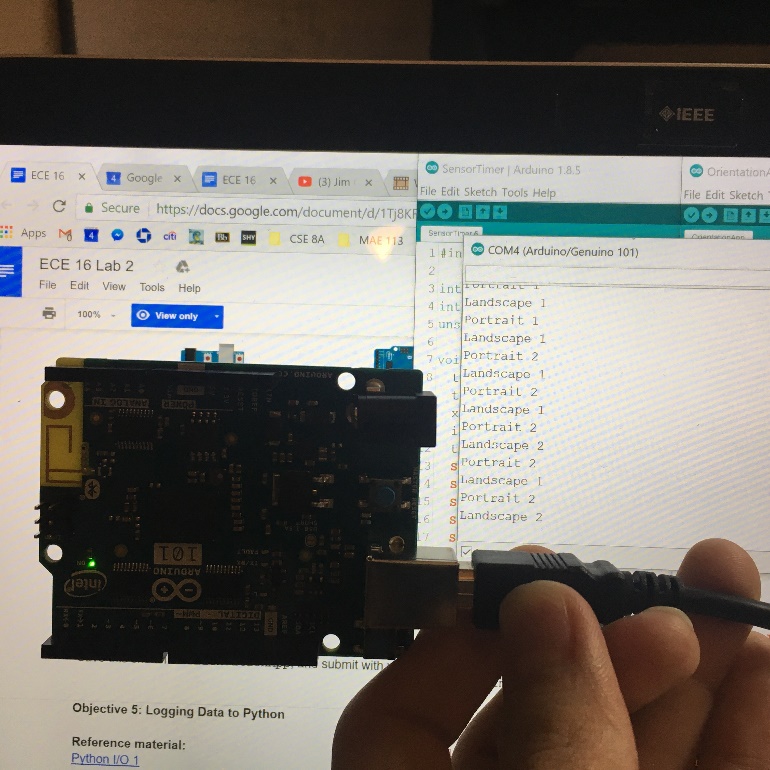


Fig. 3: Serial Monitor displaying messages after inputting a random message.

The maximum theoretical rate for each sensor would be the average of the times recorded (each sensor separate), inversed. If it took 47+41 microseconds for the accelerometer readings, then we could probably sample them at about 22500Hz to be safe (1/44microseconds). For Gyro and analog would be 23250Hz and 44400Hz respectively.

1. **Objective 4: Arduino Orientation applet**

Now we are using the onboard sensors, specifically the accelerometer, to determine the orientation of the Arduino. Using the data that I messed around with in Objective 3 (not shown), I determined the parameters needed to figure out what orientation the Arduino was in.



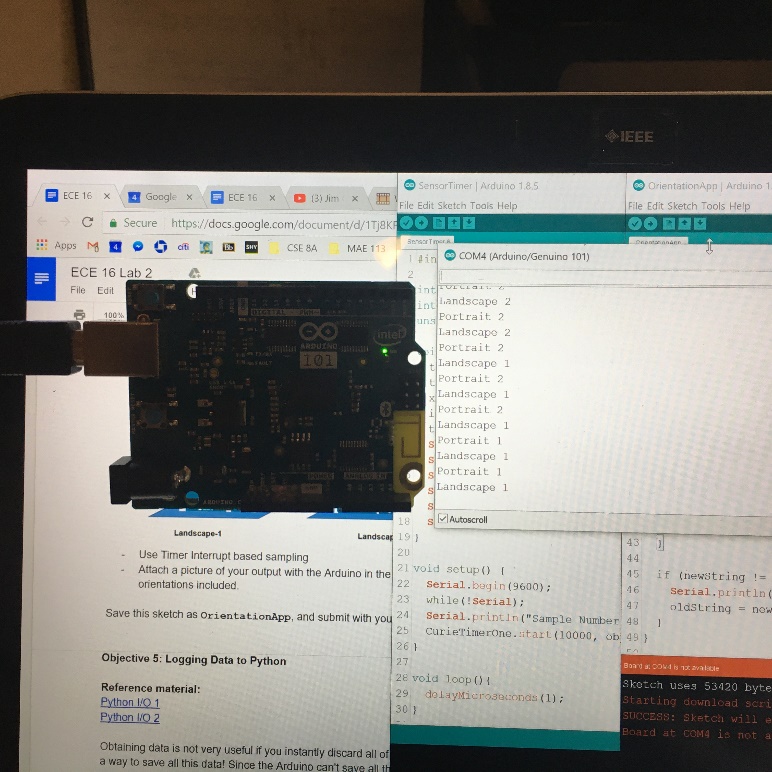


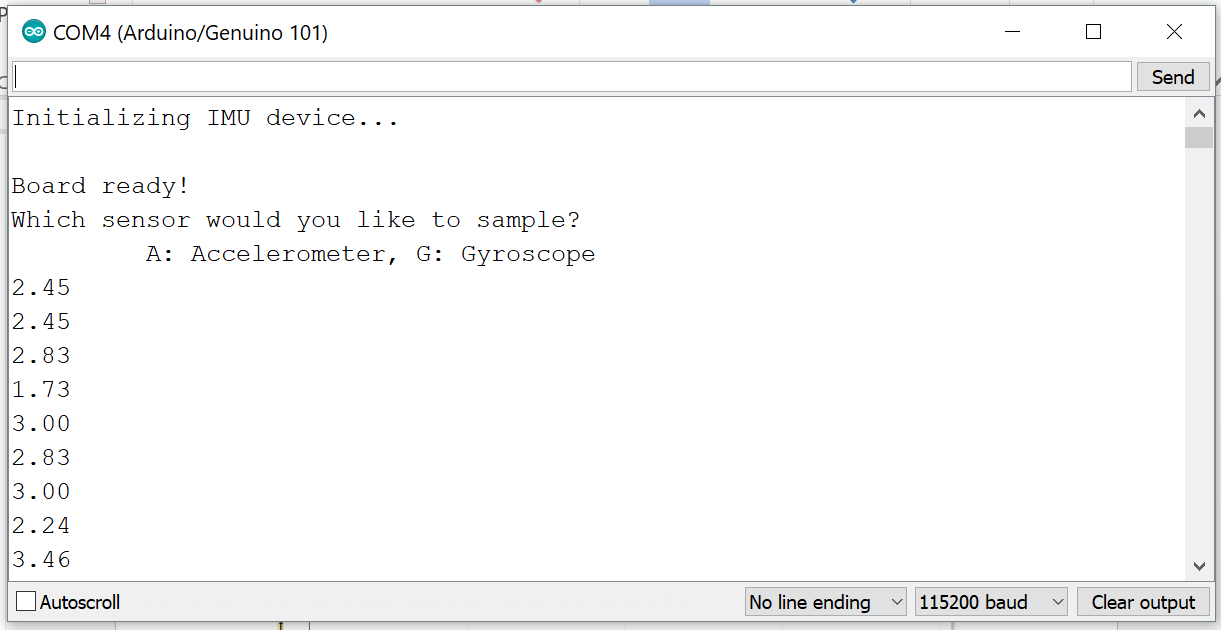
Fig. 4: s[0][0] = Portrait-1, s[1][0] = Landscape-1, s[0][1] = Portrait-2, s[1][1] = Landscape-2

1. **Objective 5: Logging Data to Python**

After playing with Arduino’s sensors and the data collected from these sensors, the next step was to write and save this data in real-time. To do so, we use Python to write into Serial, which runs an Arduino sketch (not turned in or shown). After inputting 1, 2, or 3, Arduino reads the input and prints out data depending on the input (accel, gyro, analog). The sketch samples at a fixed 200 Hz using timer based interrupts for 5000 samples. Python then reads the data in Serial and writes it into the respective filename.txt file. Inside the same git repo as this word doc are 3 text files.

1. **Objective 6: Arduino Serial Plotter**

In Objective 6, we use an Arduino sketch to ask the user which sensor’s magnitude they want to measure. We calculate magnitude using the function sq() and sqrt(). The Serial Monitor and Serial Plotter do not match in the pictures below.



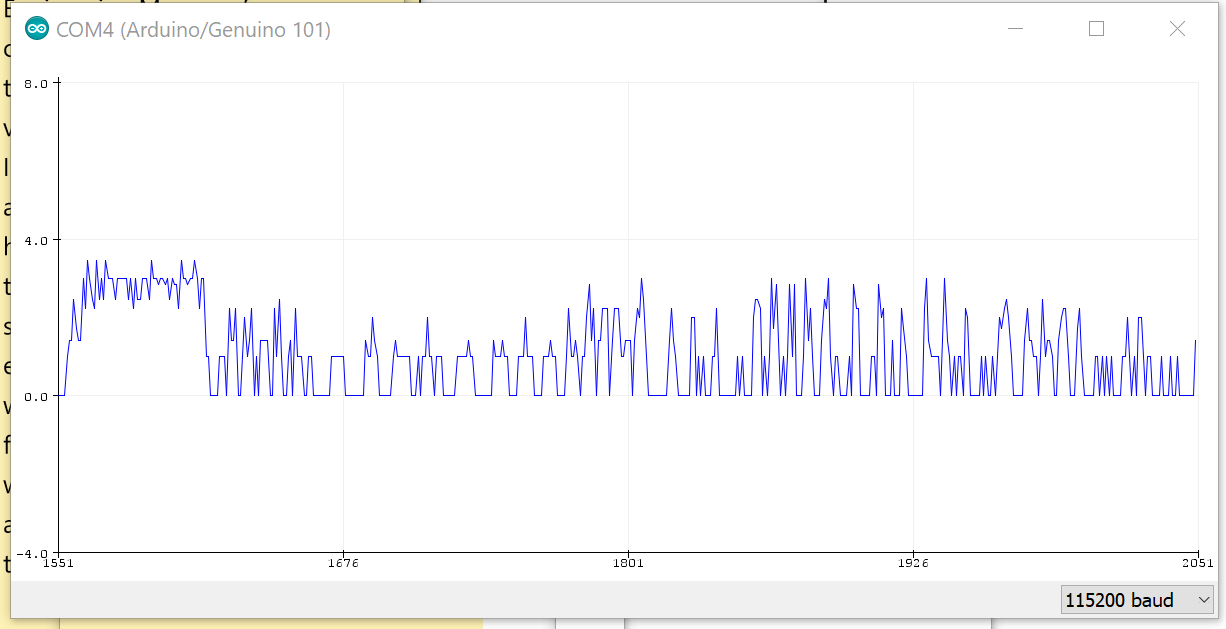
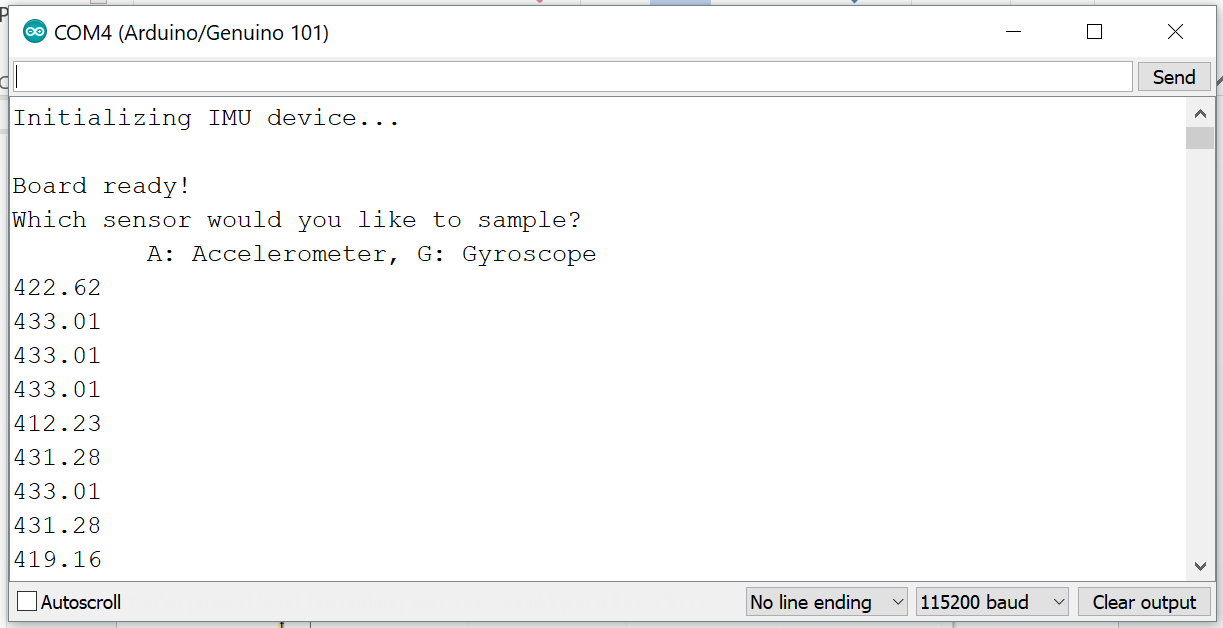


Fig. 5: Serial Plotter and Serial Monitor output for Accelerometer.



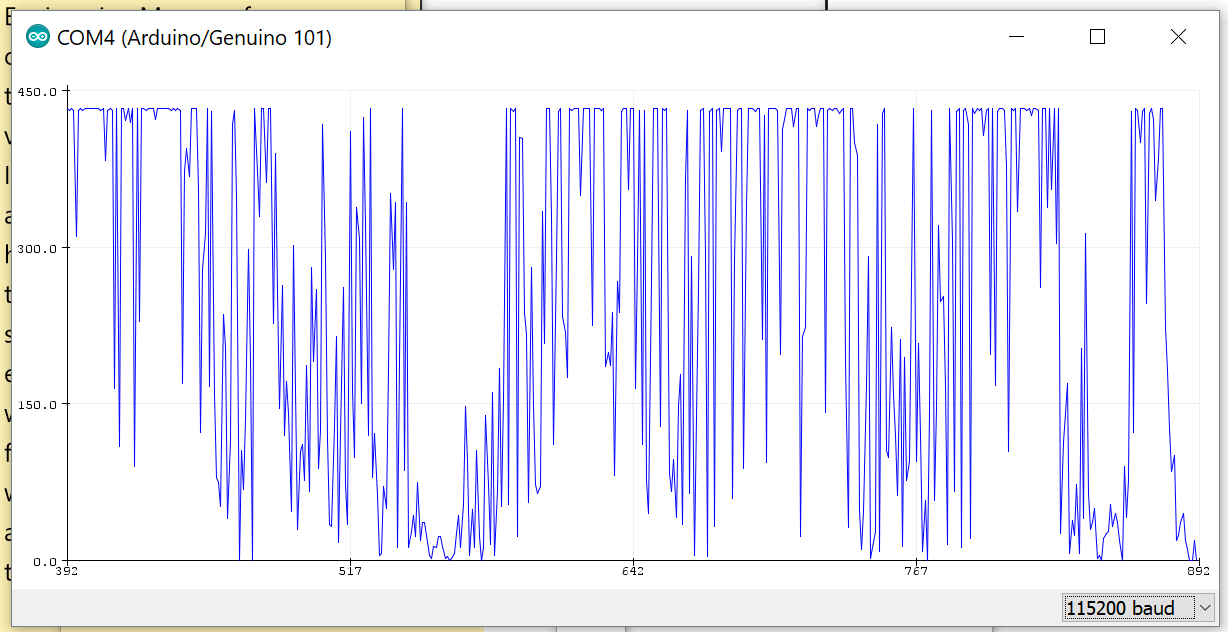


Fig. 6: Serial Plotter and Serial Monitor output for Gyroscope.

1. **Objective 7:**

Now it was time to put it all together. First, we wrote a sketch in Arduino that would make the user play a game. In this game, the Arduino measures the time it takes the user to move or rotate the Arduino, depending on whether they chose the accelerometer or the gyroscope. We use Python as the user interface, in which we can control how many times we want to play the game by specifying the number of iterations n. Along with the Word doc, files response\_time.py, responseTime.ino, gyro\_response\_time.txt, and accel\_response\_time.txt are within the repository.

1. **Objective 8: Data Analysis**

For the last challenge, objective 8 asked us to write a python script that reads and analyzes data in the text files from Objective 5 and 7. In the Obj5 text files, we determined:

* If any samples were missing.
* Number of samples analyzed
* Mean time difference
* Standard deviation of time difference
* Maximum time difference
* Minimum time difference

In the Obj7 text files, we determined:

* Mean response time
* Standard deviation
* Max response time
* Min response time

1. **Problems and Issues**

Most problems came down to not kno

The hardest

1. **Conclusion**

In this lab, I’ve gained a greater understanding on how to use Arduino and Python together in ways that can be reall