Lab #3: Data Visualization and Signal Processing

Introduction

In this lab we were introduced to the EMG sensors, plotting, filtering, and real time processing. We specifically used matplotlib for plotting in Python, scipy for the filters, and numpy for array creation.

Objective 1: Reading EMG

In this objective we firstly read raw data from the EMG shield at a constant rate of 200 Hz. In the pictures below, the red line represents the frequency in microseconds (200 Hz=5000 us) and the blue line represents the signal read when flexing my arm.

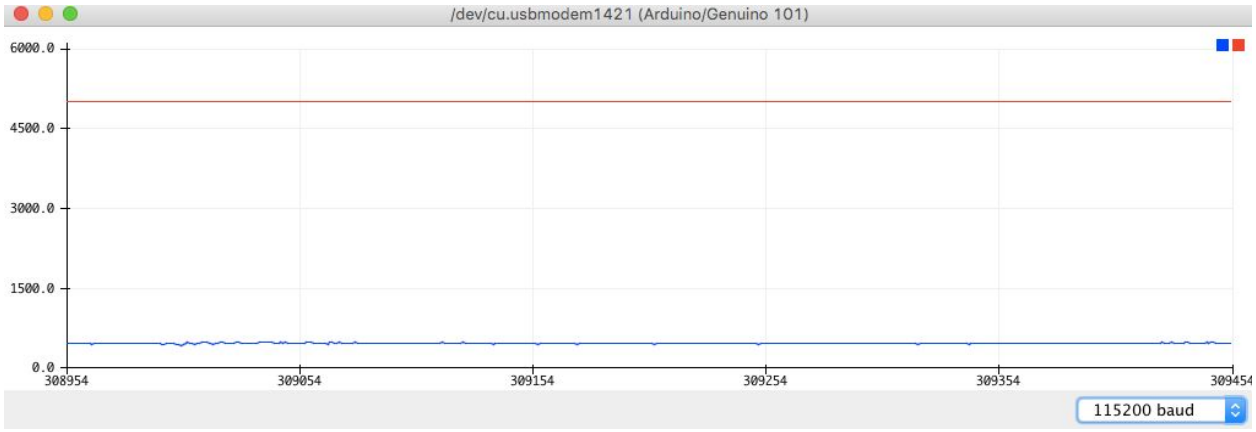


Figure 1: Before flexing

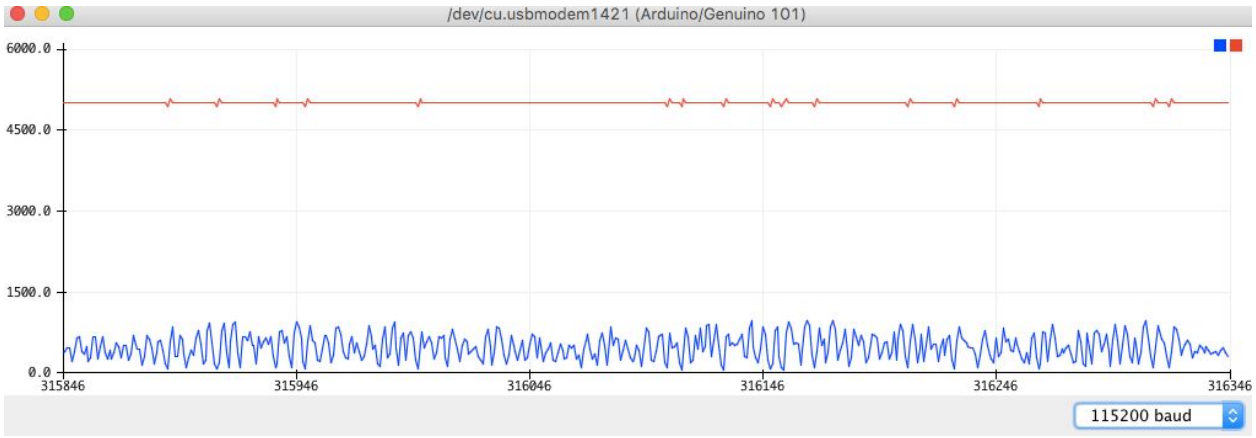


Figure 2: Flexing

After getting this information, I scaled the raw ADC data to voltage. This was done by using the formula: EMG_voltage= Scaled_term*(ADC value-Offset) and saved both the raw emg values and scaled values in a .txt file using python.

ReadEMGRaw

```
#include "CurieTimerOne.h"
#include "CurieIMU.h"
int Start_time=0; //variable used to set timer
int end_time;
char my_input;
int val0=0;
int timeISR=5000; //sampling at 200Hz
void printData()   //callback funstion for interrupt


{
  //Get raw EMG value
  val0=analogRead(A0);
  Serial.print(val0);
  Serial.print("\t");
  end_time=micros();
  Serial.println(micros()-Start_time);
  Start_time=end_time;
  Serial.println();


}

void setup() {
  // put your setup code here, to run once:
  Serial.begin(115200);
  while(!Serial);
    Start_time=0;
    CurieTimerOne.start(timeISR,&printData);
}

void loop() {}
```

Figure 3: Raw EMG on Arduino

RadEMGScaled §

```
#include "CurieTimerOne.h"
#include "CurieIMU.h"
int Start_time=0; //variable used to set timer
int end_time;
float scale;
float bias;
int val0=0;
int i=1;
int timeISR=5000; //sampling at 200Hz
void read_emg()   //callback funstion for interrupt
{

   val0=analogRead(A0);
   scale= ((val0*3.3)/1023); //
   Serial.print(scale);
   Serial.print("\t");
   bias=(scale-1.5);
   Serial.print("\t");
   Serial.print(bias/3.6); //scale down gain
   Serial.print("\t");
   end_time=micros();
   Serial.println(micros()-Start_time);
   Start_time=end_time;
   }

void setup() {
  // put your setup code here, to run once:
  Serial.begin(115200);
  while(!Serial);
  Start_time=0;
  CurieTimerOne.start(timeISR,&read_emg);
}



void loop() {}
```

Figure 4: Scaled EMG

data_collection(Obj1).py* | untitled11.py

```
1 import serial #import serial library
2 import time
3 ser= serial.Serial('/dev/cu.usbmodem1421', baudrate=115200, timeout=1) #open serial port
4 i=1;
5
6 f1 = open('EMG_scaled_data.txt', 'w')  #create file
7
8 time_start = time.time()
9 time_end =time.time()
10
11 while (time_end-time_start < 5):     #read 5 seconds of data
12      time_end=time.time()
13
14      arduinoData=ser.readline()       #read arduino data
15      print(arduinoData.decode())      #print to console
16      f1.write(arduinoData.decode())
17      if not arduinoData:
18          break
19
20 f1.close()
```

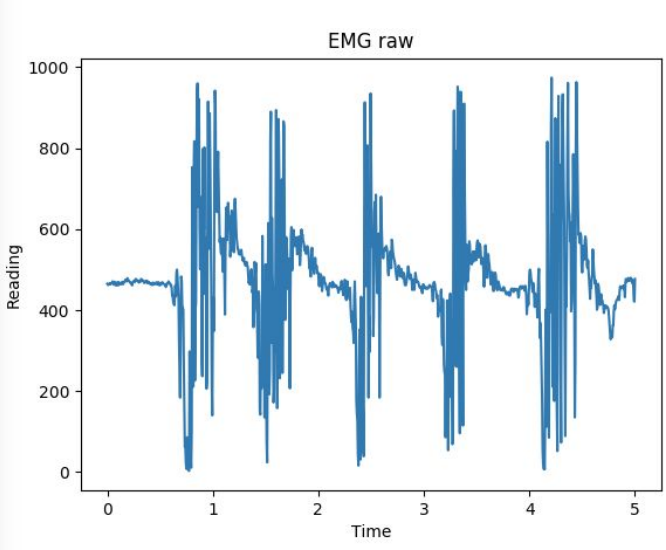
Figure 5: Saving 5 seconds of data to .txt file

Objective 2: Plotting

In this objective we got introduced to matplotlib. We firstly used this library to plot both text files from the previous objective. As you can see from the pictures below, both of the graphs are the same except for the scaling.

```
1 import numpy as np
2 import matplotlib.pyplot as plt
3
4 data= np.loadtxt("EMG_raw_data.txt")  #load the file
5 p=np.linspace(0,5,num = data.shape[0]) #0 to 5 seconds then take the first column from data
6 plt.figure(1)
7 plt.plot(p, data[:,0])  #plot data
8 plt.xlabel('Time')      #title axis
9 plt.ylabel('Reading')
10 plt.title('EMG raw')
11 plt.show()
12
13 data=np.loadtxt("EMG_scaled_data.txt")  #load file
14 p=np.linspace(0,5,num = data.shape[0])
15 plt.figure(2)
16 plt.plot(p, data[:,0])
17 plt.xlabel('Time')
18 plt.ylabel('Reading')
19 plt.title('EMG scaled')
20 plt.show()
21
22
```

Figure 6: Python plotting



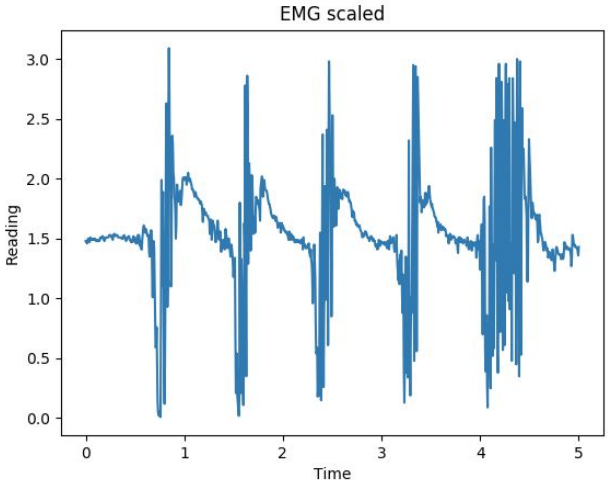


Figure 7: Raw and Scaled EMG reading

Then, I wrote a second python script where I plotted .txt files from the previous lab regarding the X.Y, and Z positions of the gyroscope and accelerometer.

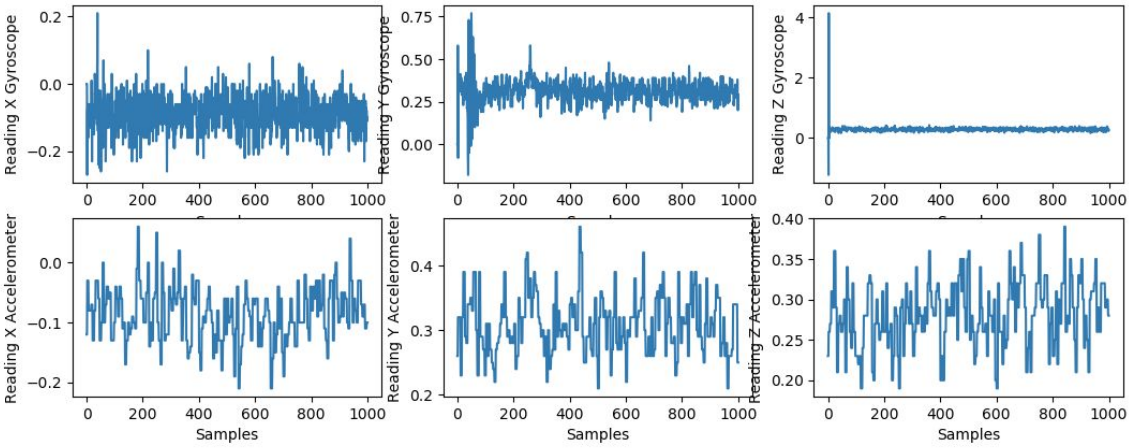


Figure 8: Accelerometer and Gyroscope (X,Y,Z) plots

Objective 3: Filtering Data

As you can see from the graphs above, there is a lot of noise in the data. One way of cleaning it up is by using filters by using SciPy. I wrote a Python script that created a high pass filter to reduce the effect of the bias, a low pass filter to reduce the high frequency noise, a bandpass, and a box car filter for power estimation. I then plotted seven plots including the original unfiltered scaled EMG signal, EMG after using low pass, EMG after using high pass, EMG after using bandpass, EMG signal rectified, smoothed signal, and the power spectral density using matplotlib.

Specifically in my code, I defined a lowpass, highpass, and bandpass filter to be called once they needed to be plotted then took the absolute value of the bandpass filter to get the rectified signal. After that I used a 100-point boxcar window to smooth the signal then lastly used the “sig.welch” function with a frequency of 200 Hz to get the Power Spectral Density.

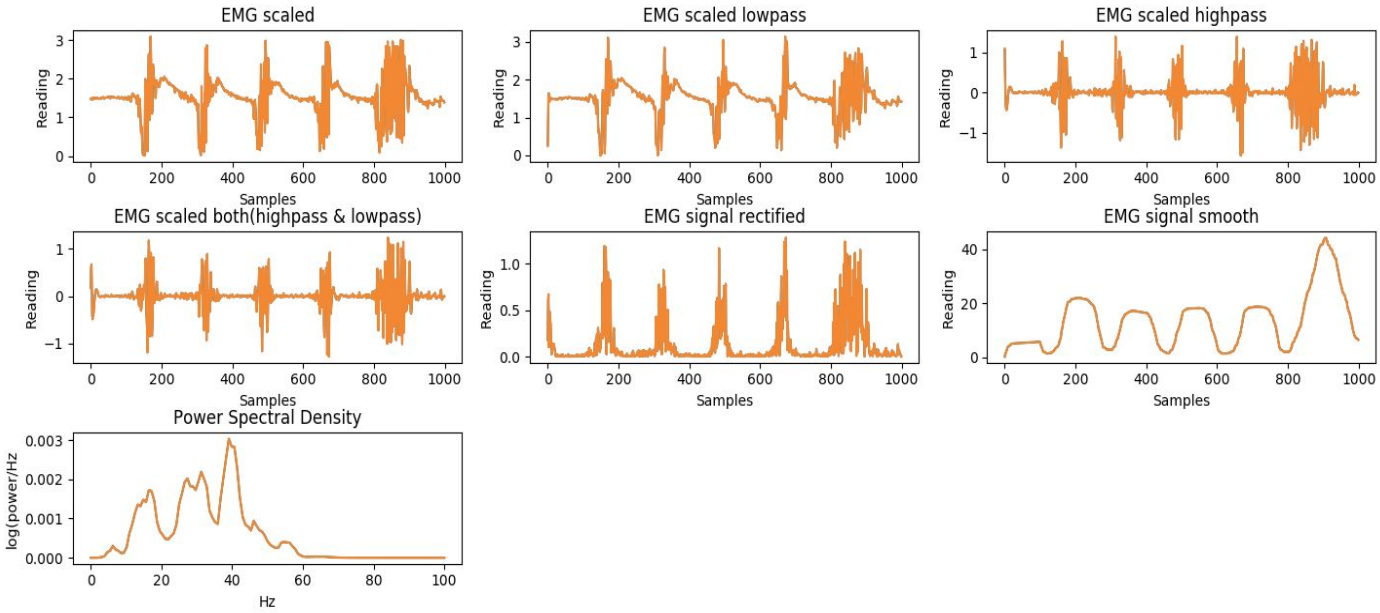


Figure 9: Filtering Data plots

Objective 4: Creating Filters

The main point of this objective was to understand how filters fully work in order for complications to not arise later. Therefore, I wrote a Python script that performed a third order low-pass and high-pass filtering on the same signal from the previous objective, without using lfilter. Comparing the graphs shown below to the graphs from the objective before, they are very similar.

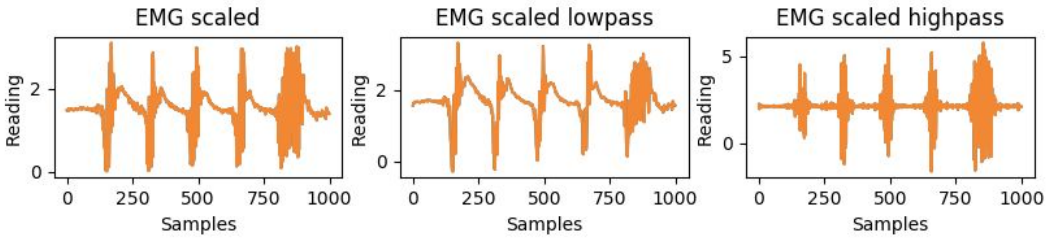


Figure 10: Manually creating the filters

Objective 5: Streaming/Real-Time

The purpose for this objective is to write a Python script that streams real time data coming in from serial and plots the original EMG signal, high pass and low pass, rectified signal, smoothed signal, and the power spectral density. Although I did not get this fully, I had a good understanding on how to start it. Please read my comments on my code to get my thinking process.

Conclusion

Throughout this lab we saw how we went from raw to scaled readings on the EMG sensor, plotted them using matplotlib in Python, filtered them, and real live streamed the signal. We also went in depth in learning how an IIR filter works and coded our version of the built in function: “lfilter.” Each step added upon the previous one and I really got to see how it all built up at the end.