Lab 1 Report: Data Acquisition and Analysis of EMG Signals using Myo Band (Makeup)

Course: EEEE 536.60L1 and EEEE 636.60L1 – Biorobotics/Cybernetics

Group Members: Youssief Khalifa(yak9700)

Date: 18-4-2025

---

1) Manifesto

The EMG signal acquisition process with Myo Armband was our first introduction in this laboratory session. Through Python we collected muscle activity data to analyze its correlation with voluntary movements while investigating how EMG signals represent such movements. Technical problems regarding the Myo device connection and vacuum tube blockage prevented laboratory success yet we learned about signal acquisition techniques used in current robot and prosthetic technology.

---

2) Introduction

The method of electromyography (EMG) serves to detect electrical signals that emerge during skeletal muscle contractions. The laboratory team examined the principal steps of gathering and analyzing EMG signals through the Myo Armband. Undergraduate students needed to configure Myo hardware followed by muscle signal data acquisition before using Python and Jupyter notebooks for analysis. The primary purpose of this instruction was to demonstrate practical signal applications for prosthetics devices and rehabilitation equipment as well as gesture recognition systems.

---

3) Theory

Electrical signals known as biopotentials originate in human body tissue from the activity of nerve and muscle cells. The technological framework of EMG monitors skeletal muscles since those control voluntary movement. Biopotentials become measurable through surface electrodes when muscles contract because of action potentials.

EMG signals exist between 2–500 Hz in frequency range and produce amplitudes spanning from 50 μV to 5 mV. The signals become ready for processing through rectification methods or RMS value calculations.

EMG serves several applications that include neuromuscular condition diagnosis and prosthetic control in addition to sport athlete fatigue measurement and gesture-based device operation through the Myo Armband.

---

4) Methodology

Myo Setup:

- Installed MyoConnect.
- Connected the armband via Bluetooth dongle.
- Verified connection using the 'Ping' feature.

Python Environment:

- Used Anaconda with Python 3.8.
- Followed GitHub setup: https://github.com/abhilasha2614/myo_ecn_airc.
- Activated conda environment and ran streaming.py and acquisition.py.

Data Acquisition:

- Ran python acquisition.py 10 testfile to capture 10 seconds of EMG data.
- Data saved as .csv with 8 muscle channels.

Data Analysis:

- Used Lab_1_EMG_Analysis.ipynb to analyze data.
- Loaded .csv into the notebook and processed EMG signals using visualization and RMS calculations.

Issues Encountered:

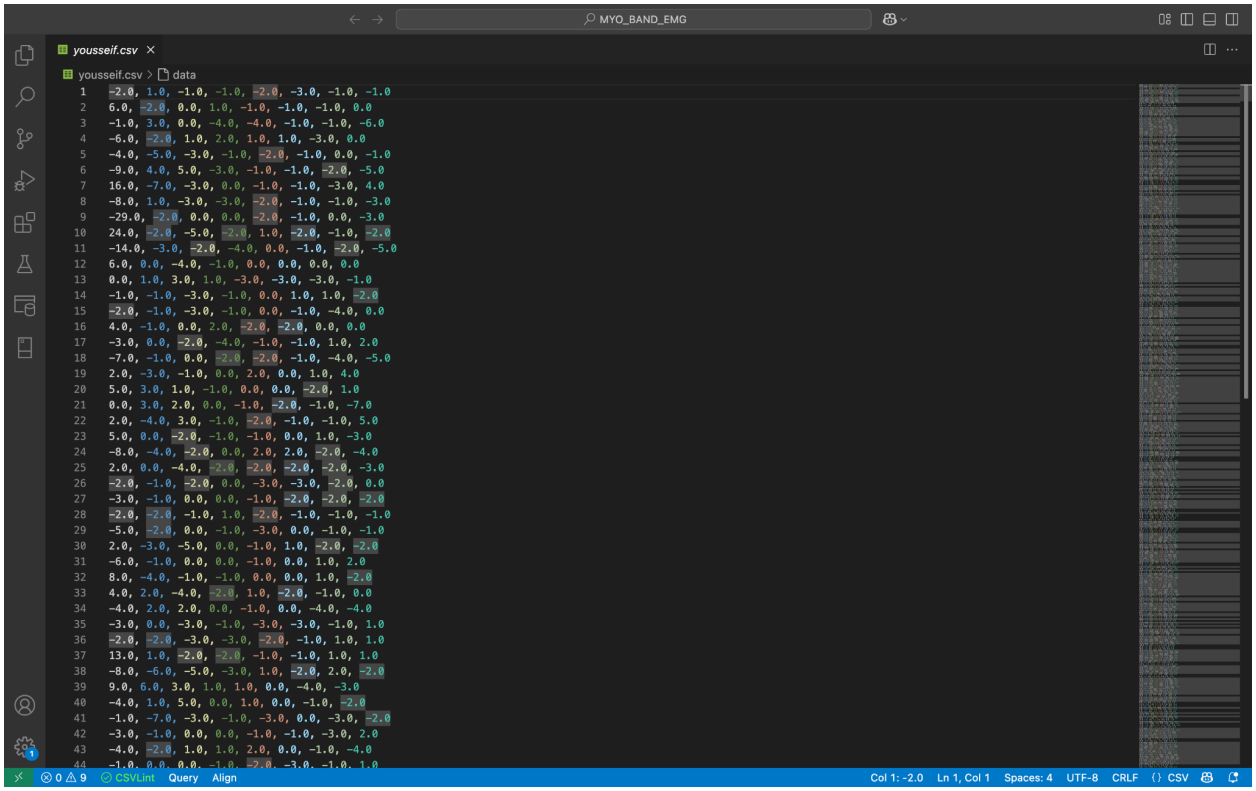
- Myo Armband failed to connect once due to Bluetooth error.

---

5) Results

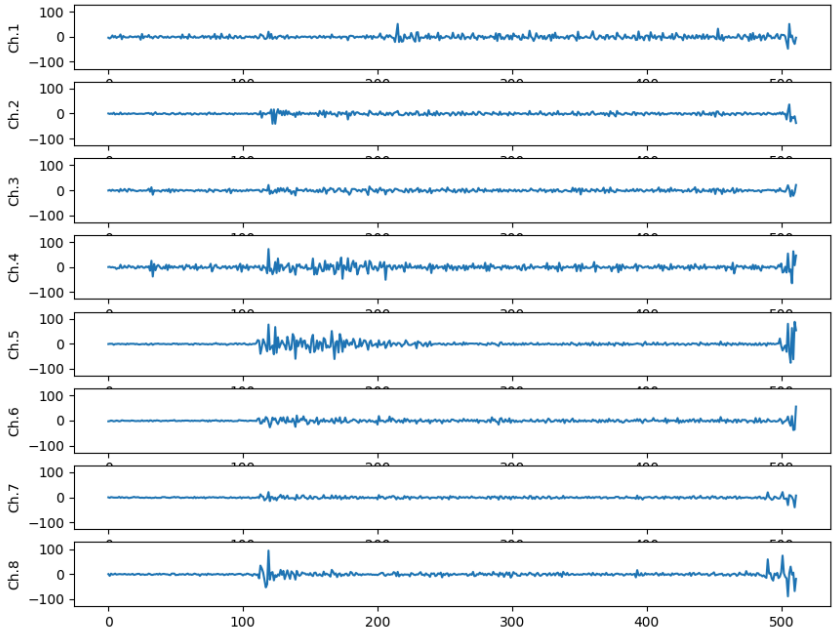
Data Acquisition:

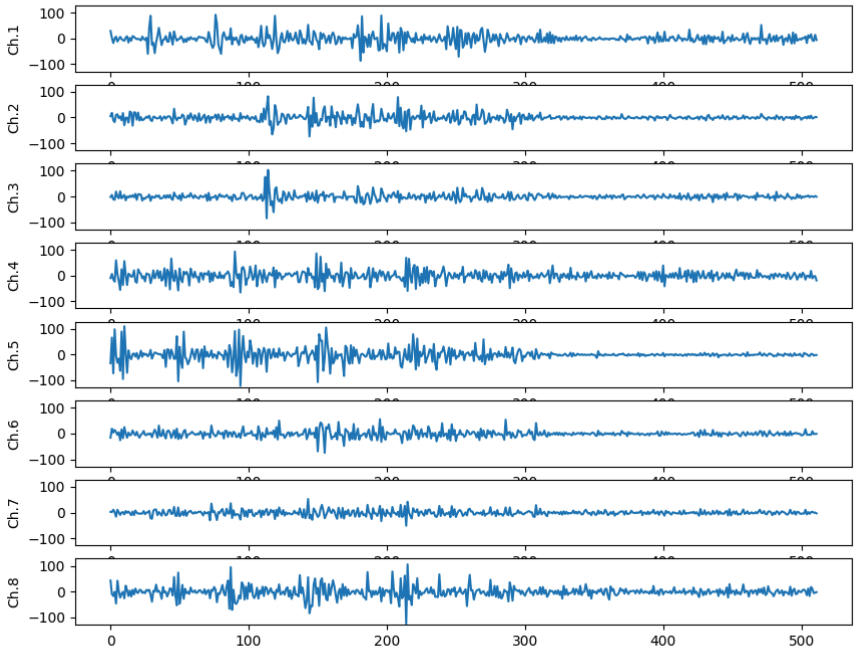
(Obtaining the full CSV File after successfully connecting to the Myo Band, **yousseif.csv**)



Data Streaming:

The figures demonstrate EMG signal information from the Myo Armband which distributes across eight channels. Real-time monitoring of muscle activity produces the signals detected by the device.





Data Analysis:

The processed EMG signals obtained through Myo Armband show signal differences across the whole spectrum of eight channels in the presented visuals. The waveforms in each display show the level of muscle activation duration followed by distinct peaks during movements. Analysis of raw signals occurred through Python and Jupyter Notebook before RMS (Square Mean Root) calculations measured the muscle activity. The processing method allows identification of strength level together with signal consistency to separate active muscle periods from rest intervals. The evaluation shows how EMG signal features provide actionable information to decode hand movements under practical operative conditions.

Lab 1: EMG Signal Analysis

Basics of signal analysis with python

Libraries

The following libraries are the "bread and butter" of data analysis in python

[7]:
```
import pandas as pd # Tabular data
import numpy as np # Working with Arrays
import scipy # Scientific computation
import scipy.signal
import matplotlib.pyplot as plt # Plotting Data
```

[6]:
```
pip install matplotlib
```

```
Collecting matplotlib
```

Working with files using `os` library

the `os` library, "operating system", allows for interaction with the operating system such as reading and writing to files.

[11]:
```
import os
directory = '/Users/yousef/Desktop/MYO_BAND_EMG/'
file_name = 'p18_emg.csv' # Add file name here
emg_df = pd.read_csv(directory + file_name) # Read the CSV file into a pandas dataframe
emg_df.columns = emg_df.columns.str.replace(' ','') # Replace the spaces in the column titles for easier handling
```

[12]:
```
# Display the data frame preview
display(emg_df)
```

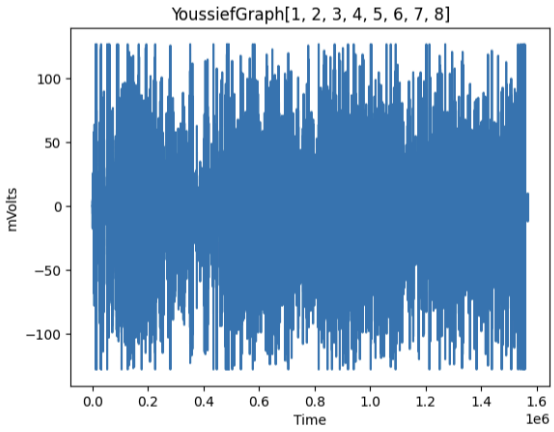
	DeviceID	Warm?	Sync	Arm	Timestamp	Orientation_W	Orientation_X	Orientation_Y	Orientation_Z	Acc_X	...	EMG_4	EMG_5	EMG_
0	2291479052128	warm	True	left	2019-02-14 14:46:53 744402	-0.514465	-0.485779	0.287659	-0.645508	1.326660	...	-2	1	
1	2291479052128	warm	True	left	2019-02-14 14:46:53 744402	-0.514465	-0.485779	0.287659	-0.645508	1.326660	...	-3	-1	
2	2291479052128	warm	True	left	2019-02-14 14:46:53 752382	-0.514465	-0.485779	0.287659	-0.645508	1.326660	...	0	-1	
3	2291479052128	warm	True	left	2019-02-14 14:46:53 752382	-0.514465	-0.485779	0.287659	-0.645508	1.326660	...	0	0	
4	2291479052128	warm	True	left	2019-02-14 14:46:53 756350	-0.519470	-0.484497	0.291138	-0.640930	1.186035	...	0	2	
...	...	...	...	...	...	...	...	...	...	...	...	...	...	
1569135	2291479052352	True	False	unknown	2019-02-14 16:14:04 919846	0.176941	0.678650	-0.692627	0.168640	0.981934	...	2	2	
1569136	2291479052352	True	False	unknown	2019-02-14 16:14:04 926902	0.176941	0.678650	-0.692627	0.168640	0.981934	...	0	1	
1569137	2291479052352	True	False	unknown	2019-02-14 16:14:04 926902	0.176941	0.678650	-0.692627	0.168640	0.981934	...	4	5	
1569138	2291479052352	True	False	unknown	2019-02-14 16:14:04 941786	0.174622	0.678345	-0.693115	0.170166	0.985352	...	-4	-7	
1569139	2291479052352	True	False	unknown	2019-02-14 16:14:04 941786	0.174622	0.678345	-0.693115	0.170166	0.985352	...	3	0	

1569140 rows × 29 columns

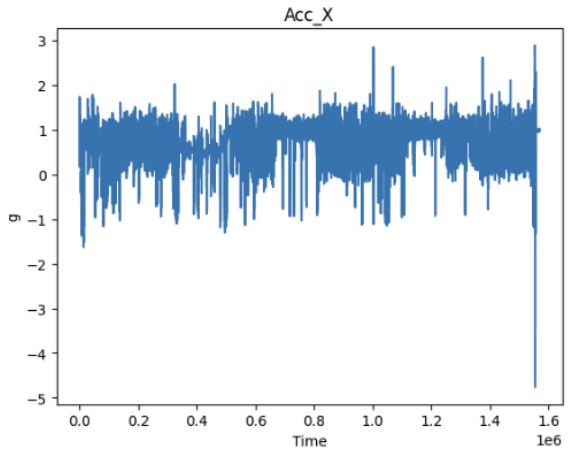
Plotting data

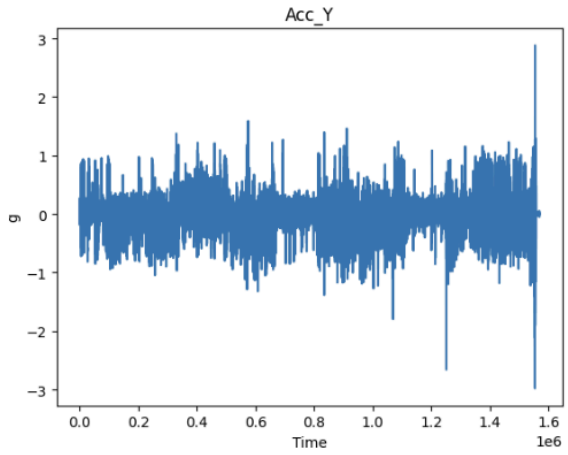
Previewing tabular data is nice, but does not give insight into the data itself. Large datasets can have thousands of lines and real data measurements will be very intensive (this data is 1,405,884 rows) which is unreasonable to review in a tabular method. To view the data, `matplotlib` library will be used to plot mVolts measurement vs Time.

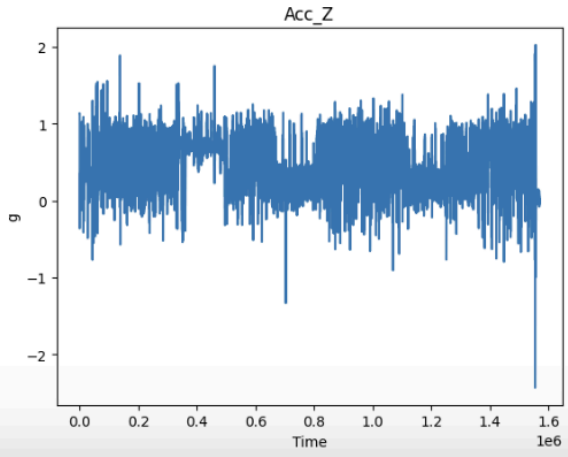
[14]:
```
channel = [x for x in range(1,9)] # Create channel variables to be used

plt.figure()
ax = emg_df['EMG_'+str(channel[0])].plot()
plt.title('YoussiefGraph' + str(channel))
plt.ylabel('mVolts')
plt.xlabel('Time')
plt.show()
```



[15]:
```
# Plotting multiple channels at once

for channel in ['X','Y','Z']:
    plt.figure()
    emg_df['Acc_'+channel].plot()
    plt.title('Acc_'+str(channel))
    plt.ylabel('g')
    plt.xlabel('Time')
    plt.show()
```

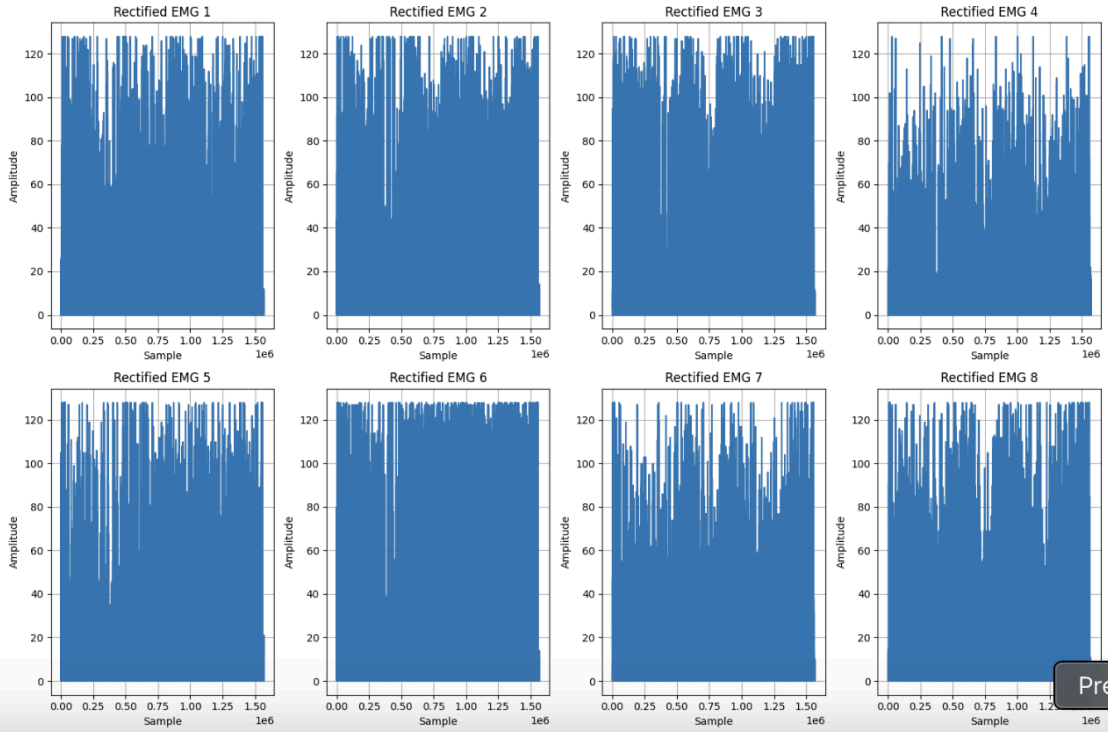






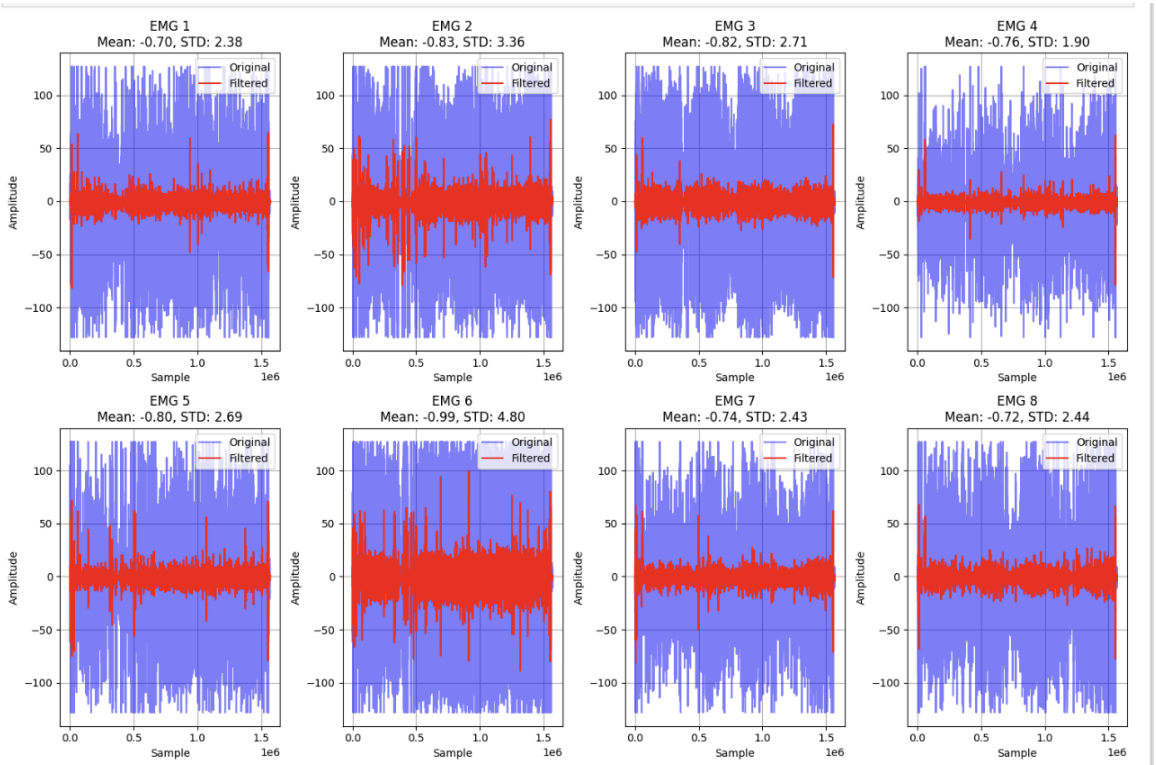
Task 1: Plot the rectified signals

[17]:
```python
# Create a figure with appropriate size
plt.figure(figsize=(15, 10))

# Create 8 subplots (2 rows, 4 columns)
for i in range(8):
    plt.subplot(2, 4, i+1)

    # Get the EMG signal and rectify it (take absolute value)
    emg_signal = np.abs(emg_df[f'EMG_{i+1}'])

    # Plot the rectified signal
    plt.plot(emg_signal, label=f'EMG {i+1}')

    # Add labels and title
    plt.title(f'Rectified EMG {i+1}')
    plt.xlabel('Sample')
    plt.ylabel('Amplitude')
    plt.grid(True)

# Adjust the layout to prevent overlap
plt.tight_layout()

# Show the plot
plt.show()
```



Task 2: Plot the filtered signal and obtain its mean and Standard deviation

[36]:
```python
# Define filter parameters
fs = 200  # Sampling frequency (Hz) - typical for EMG
cutoff = 10  # Cutoff frequency (Hz)
order = 4  # Filter order

# Create Butterworth low-pass filter
nyquist = fs/2
normalized_cutoff = cutoff / nyquist
b, a = scipy.signal.butter(order, normalized_cutoff, btype='low')

# Create a figure with appropriate size
plt.figure(figsize=(15, 10))

# Create 8 subplots (2 rows, 4 columns) and store mean and std
means = []
stds = []

for i in range(8):
    plt.subplot(2, 4, i+1)

    # Get the EMG signal
    emg_signal = emg_df[f'EMG_{i+1}']

    # Apply the filter
    filtered_signal = scipy.signal.filtfilt(b, a, emg_signal)

    # Calculate mean and standard deviation
    mean_val = np.mean(filtered_signal)
    std_val = np.std(filtered_signal)
    means.append(mean_val)
    stds.append(std_val)

    # Plot both original and filtered signals
    plt.plot(emg_signal, 'b-', alpha=0.5, label='Original')
    plt.plot(filtered_signal, 'r-', label='Filtered')

    # Add labels and title
    plt.title(f'EMG {i+1}\nMean: {mean_val:.2f}, STD: {std_val:.2f}')
    plt.xlabel('Sample')
    plt.ylabel('Amplitude')
    plt.grid(True)
    # Specify legend location explicitly
    plt.legend(loc='upper right')

# Adjust the layout to prevent overlap
plt.tight_layout()

# Show the plot
plt.show()

# Print summary of means and standard deviations
print("\nSummary of all channels:")
for i in range(8):
    print(f"EMG {i+1}: Mean = {means[i]:.2f}, Standard Deviation = {stds[i]:.2f}")
```




```
Summary of all channels:
EMG 1: Mean = -0.70, Standard Deviation = 2.38
EMG 2: Mean = -0.83, Standard Deviation = 3.36
EMG 3: Mean = -0.82, Standard Deviation = 2.71
EMG 4: Mean = -0.76, Standard Deviation = 1.90
EMG 5: Mean = -0.80, Standard Deviation = 2.69
EMG 6: Mean = -0.99, Standard Deviation = 4.80
EMG 7: Mean = -0.74, Standard Deviation = 2.43
EMG 8: Mean = -0.72, Standard Deviation = 2.44
```

B)

We decided to include all the screenshots as we felt like that all figures were necessary to include.

---

6) Picture of the Myo Band Attached to my arm

---

7) Discussion

The modern world increasingly depends on EMG signals as a vital bioengineering instrument. The Myo Armband enables affordable portable non-invasive EMG signal collection which users can operate easily. This technology finds use in:

- Healthcare: Diagnosing motor neuron diseases like ALS.
- The use of EMG signals enables the powering of prosthetic devices and their training for patients who underwent limb amputations.
- The interface of gaming and robotics uses gesture control as an enabling technology.
- Sports Science: Tracking muscle fatigue and recovery.

The lab session delivered important information concerning the complete EMG signal workflow scope from equipment acquisition to analysis methods while demonstrating crucial roles of clean acquisition methods and signal processing techniques.

---

8) References

MyoConnect Software:
https://myo-connect.software.informer.com/download/

Myo Python SDK:

https://github.com/abhilasha2614/myo_ecn_airc

GitHub Lab Files:
https://github.com/BioRobotics-Spring-2020/Lab_1

CSV Dataset: **yousseif.csv**, **p18_emg.csv**

EMG Analysis Notebook: **Lab_1_EMG_Analysis.ipynb**
