Lab 4. EMG Device Lab - Part 1

Introduction to Design Challenge:

You are tasked with creating a prosthetic system which wirelessly receives EMG data from a user, displays the data on an LCD screen, and then does a task with it. The task can be anything, from replicating the motion of the arm, to being used to play a game, etc.

In this part of the lab you will be reading the EMG signals. Dealing with EMG signals is difficult for several reasons. The signal is extremely small and is thus very susceptible to noise. Second, the signal deviates from the baseline incredibly easily. The “0” constantly moves. Third, the signal has both positive and negative components which cannot be read directly by common microcontroller packages like the Arduino.

As with all good design work, we will be building and testing the circuit in stages. This will help us to easily troubleshoot and make sure our circuit is working properly. Even experienced electronics designers do simple designs in stages.

Important Safety Note
Connecting your body through a circuit to high power sources such as an electric outlet or your computer can be fatal.

Tips

- Make sure to try and color code your wires. Size your wires appropriate to the distance they need to cover without looping them. You may wish to cut your own wires.
- **Unplug your batteries when you are done with a build & test session**. Regularly check your batteries’ voltage to make sure they are sufficient. Make sure to have spare batteries.
- You may wish to draw out your circuit’s breadboard layout and appropriately label things so you can easily go back and determine where you are. You can also add labels to your breadboard.

Overview of Sessions:
Session 1:

- You will design and build a differential amplifier to convert a three-lead EMG signal into a 2-lead signal which can be easily manipulated. You will design and build a buffer to isolate the signal from further downstream components.

Session 2:

- You will design and build a Band Pass filter to isolate the dominant frequencies of the EMG signal. You will design and build a Full Wave Rectifier and Non-Inverting Amplifier to condition the signal to be read more easily by a microcontroller.

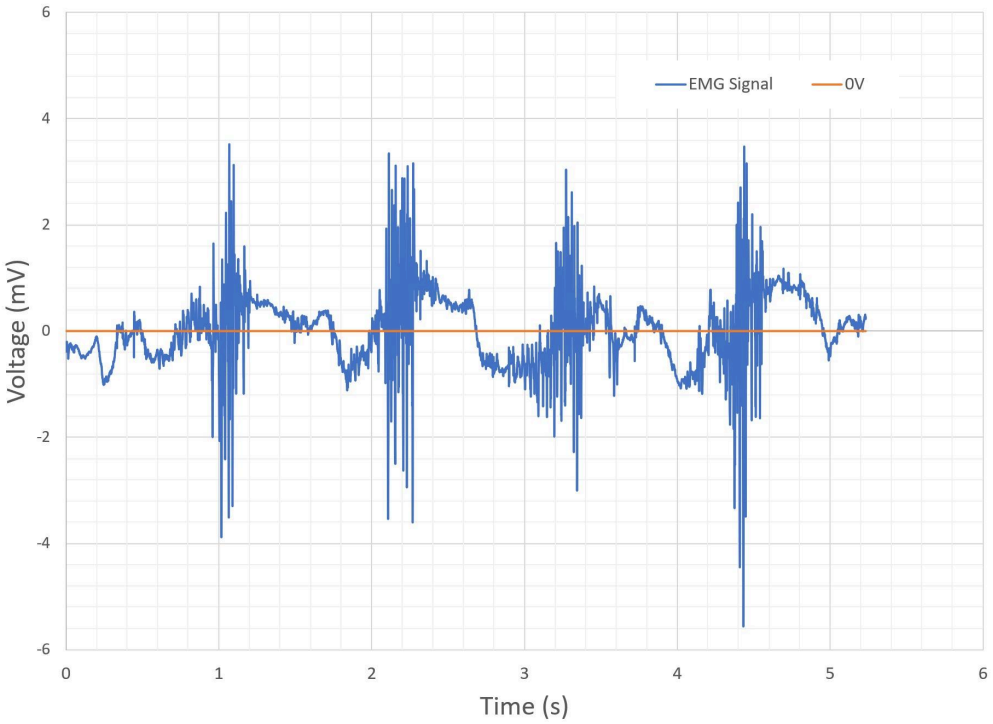
Session 3:

- You will be creating a Bluetooth Low Energy (BLE) connection between your computer and microcontroller to wirelessly transmit the conditioned EMG signal.

Session 1

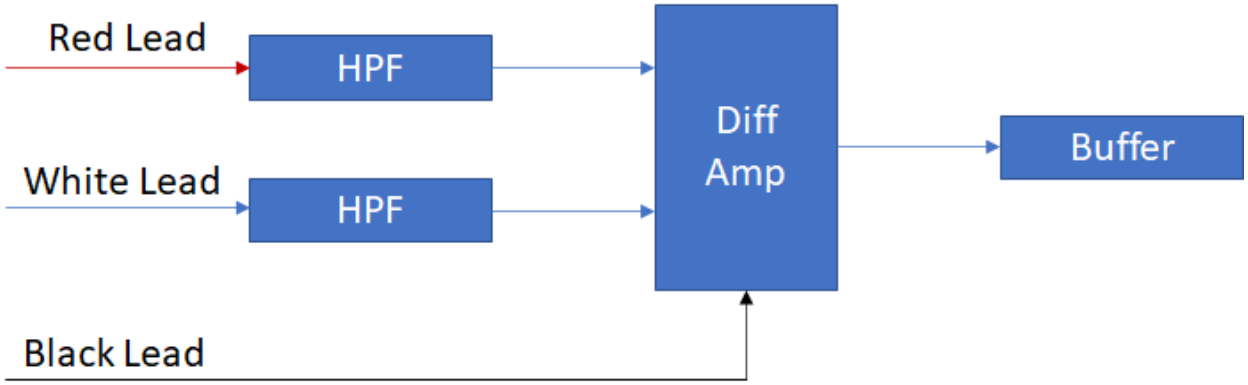
BACKGROUND

As mentioned earlier, EMG signals are difficult to deal with because the signal's baseline tends to drift over time. This drift is the "DC component" of the signal which has a very low frequency, close to zero. We remove this by creating a high pass filter of ~0.16Hz.



Another difficulty with EMG signals is that the three-leads cannot be read into devices like an Arduino. To solve this problem, we use a differential amplifier to convert the three-lead input to a two-lead output. Read more about why differential signals are used here: Link.

The figure below shows a block diagram of what you will be designing. "HPF" is the high pass filter and "Diff Amp" is a differential amplifier.



METHODS

As you complete today's work, please take note of each deliverable requested in the Deliverables section. These deliverables will be shown in your final submission for Part 1. Also, please take videos and photos of your process.

When working with your circuits be sure to disconnect from power until you are ready to test or run your circuit

M1. Install the WaveForms software onto your computer (Link).

Building the filter:

M2. Design a one-poll passive high pass filter with a cutoff frequency of ~0.16Hz.

M3. Verify your circuit is blocking low frequencies by inputting a DC signal. Verify it is allowing high frequencies to pass by using the Scope and Wavegen features of the Analog Discovery.

Setting up the differential amplifier:

M4. Review the datasheet for the INA126 Differential amplifier available on the lab's website.

M5. Set up two voltage sources, to generate the positive and negative voltages. You will use both the 5 V and -5 V regulators (note that their pinouts are different and require different input voltages!). Make sure that they are working properly before moving to the next step. You may wish to label the rails on your breadboard with tape to help you remember which is which over the course of these modules.

M6. Power your chip using the +/-15 V source from the myDAQ or use two power supplies to get +/-12 V.

NOTE: The +15/12 V from the power supply is V_{IN} to the 5 V regulator and -15/-12 V from the power supply is V_{IN} to the -5 V regulator. ***IF YOU SWITCH THESE INPUTS YOU WILL CAUSE YOUR REGULATORS TO GET EXTREMELY HOT***.

M7. Design and set up the differential amplifier with a gain of 5 (View the **typical application** for details). Make sure to draw out your circuit. Make sure to use the 0.1 uF capacitors (more details on what these capacitors are for can be found here).

M8. Test your system by inputting a .25 V signal (DC) in the Vin+ and a 0.1 V (DC) signal in the Vin- pins from the Wavegen feature of the Analog Discovery. Be sure to have a common ground between the Analog Discovery and your circuit.

Interface filters with your differential amplifier:

M9. The EMG signal has a V+ and V- referenced to ground. Both of these signals will each need to go through their own high pass filter before being input into the differential amplifier (see the block diagram in the background). Design and build this full system.

M10. Test the system by using an input of a 100 Hz sine wave with 0.25 V Amplitude and no Offset from the Wavegen feature of the Analog Discovery. This signal will be called the **Test Signal 1** throughout this document. You will also use a second input of a 100 Hz sine wave with a 0.1 V Amplitude and no Offset. This signal will be called **Test Signal 2** throughout this document
   a. **Test Signal 1** will go through a high pass filter (with the same cutoff as above) then be connected to the Vin+ of your differential amplifier.
   b. **Test Signal 2** will go through another high pass filter (with the same cutoff as above) then be connected to the Vin- of your differential amplifier.
   c. Make sure to have a common ground between your circuit and test signals
   d. Make sure to set the no synchronization setting to **Synchronized**. Think about what would happen if the two sign waves were not synchronized or if they were phase shifted.
   e. Measure the output of the system using the Scope feature of the Analog Discovery to verify that your differential amplifier is working correctly.

Add a buffer stage:

M11. To prevent the signal output of the differential amplifier from being affected by the later circuit, build a buffer stage (see the unity gain buffer reading). Think about which op-amp you should use (**HINT:** Read datasheets) and check with TAs.
M12. Verify that the buffer stage works separately from your differential amplifier.
M13. Interface the buffer stage with the rest of your circuit and test with **Test Signal 1** and **Test Signal 2**.

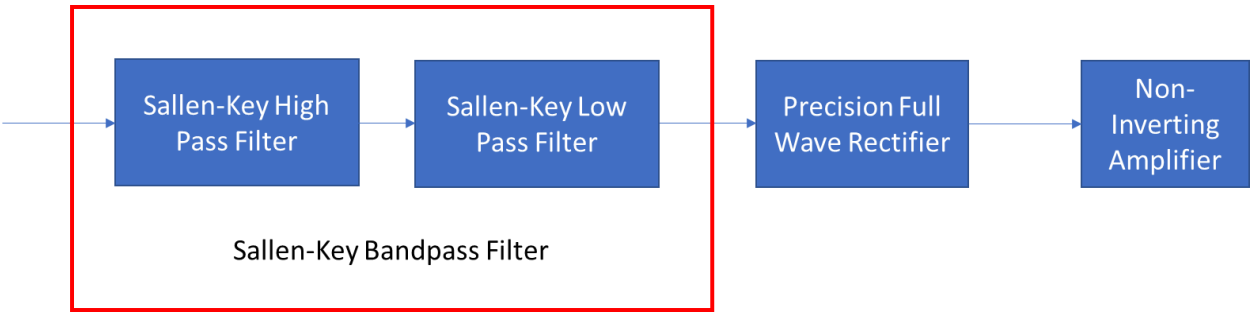
DELIVERABLES

D1. Submit circuit diagrams of your filter circuit and differential amplifier circuit (with a gain of 5), along with evidence that they work properly (these should be screen shots of the waveforms application and **NOT** photos taken on your phone).
D2. Submit a circuit diagram of the filters, differential amplifier, and buffer stages together. Include evidence that the system works (these should be screen shots of the waveforms application and **NOT** photos taken on your phone).

Session 2

BACKGROUND

In this session, you will complete the system you started as per the block diagram below:



Recall your understanding of the Nyquist theorem and sampling related from your Signals and systems class. For this EMG signal, since ultimately, it will be converted to digital, the signal will need to be cut to its most usable frequencies, which is from 50-160 Hz. Consequently, you will design a Sallen-Key bandpass filter for this range, by combining a Sallen-Key High Pass with a Sallen-Key Low Pass filter. Learn more about it here:
https://www.electronics-tutorials.ws/filter/sallen-key-filter.html

Since Arduinos cannot accept signals which are less than zero, you will need to make all of the negative signals positive. You will use a full wave rectifier for this. This is akin to using the **abs** function in Matlab. Learn more about it here:
https://www.electronics-tutorial.net/analog-integrated-circuits/precision-rectifier/precision-full-wave-rectifier/

Finally, you will need to amplify the signal to maximize the 3.3 V range of the Arduino MKR 1010. Rather than an inverting amplifier which you've designed before, you will use a non-inverting amplifier to keep your signal positive. Learn more about it here:
https://www.electronics-tutorials.ws/opamp/opamp_3.html

METHODS

As you complete today's work, please take note of each deliverable requested in the Deliverables section. These deliverables will be shown in your final submission for Part 1. Also, please take videos and photos of your process.

Design and build each module as described in the block diagram. Make sure to test each section as you go along.

Building the Sallen-Key Bandpass Filter Stage:

M1. You will first start with building a High Pass Filter with a ~50 Hz cutoff frequency. Follow the Sallen-Key High Pass filter designer from http://sim.okawa-denshi.jp/en/Fkeisan.htm. Think about which op-amp(s) you should use (**HINT:** Read datasheets) and check with TAs.

M2. Verify your circuit is working (be sure to test both a rejected and passed frequency)

M3. Next build a Low Pass filter with a ~160 Hz cutoff frequency. Follow the Sallen-Key Low Pass filter designer from http://sim.okawa-denshi.jp/en/Fkeisan.htm.

M4. Verify your circuit is working (be sure to test both a rejected and passed frequency)
M5. Now combine the two filters together. Connect the output of the high pass filter to the input of the low pass filter.
M6. Verify your circuit is working (be sure to test both a rejected and passed frequency).

Building the Precision Full Wave Rectifier Stage:
M7. Build the Precision Full Wave Rectifier from the background material. Think about which op-amp(s) you should use (**HINT:** Read datasheets) and check with TAs.
M8. Verify that your circuit is working.

Building the Non-Inverting Amplifier Stage:
M9. Build a Non-Inverting Amplifier with a gain of ~3. Think about which op-amp(s) you should use (**HINT:** Read datasheets) and check with TAs. This gain may change depending on each person, but 3 should be a good starting point.
M10. Verify that your circuit is working

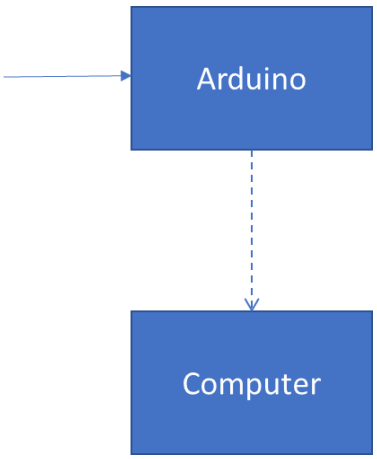
Interfacing your stages with your previous circuit:
Now you will combine all of these stages with your previous circuit from **Session 1**
M11. Using the block diagrams as a guide, connect the Buffer Stage to the input of the Sallen-Key Bandpass Filter Stage. Then connect the output of the Sallen-Key Bandpass Filter Stage to the input of the Precision Full Wave Rectifier Stage. Then connect the output of the Precision Full Wave Rectifier Stage to the input of the Non-Inverting Amplifier Stage.
M12. Verify that your circuit is working by using **Test Signal 1** and **Test Signal 2** as inputs to your system. Measure the output of your circuit with the Analog Discovery.

DELIVERABLES
D1. Submit circuit diagrams of each stage built in **Session 2**, along with evidence that they work properly (these should be screen shots of the waveforms application and **NOT** photos taken on your phone).
D2. Submit a circuit diagram of all of the stages combined together (from **Session 1** and **Session 2**). Include evidence that the system works (these should be screen shots of the waveforms application and **NOT** photos taken on your phone).

Session 3

BACKGROUND

In this session, you will integrate your system and interface it with an Arduino MKR 1010. You will use the device's Bluetooth capabilities to transfer data to your PC. You will start with a test signal before measuring your own EMG signals



METHODS

As you complete today's work, please take note of each deliverable requested in the Deliverables section. These deliverables will be shown in your final submission for Part 1. Also, please take videos and photos of your process.

Setting Up Your Arduino MKR WiFi 1010:

M1. Connect the Arduino using a USB hub. **It is important to always use this to prevent accidentally damaging your computer.**

M2. Go through the following getting started guide for the MKR WiFi 1010 Getting started with the MKR WiFi 1010

M3. Make sure to update the firmware for the MKR WiFi 1010 Check and update the firmware for WiFiNINA and WiFi101 – Arduino Help Center

M4. Make sure to download the **ArduinoBLE** library onto your version of the Arduino IDE
   a. Follow along with Installing Libraries | Arduino Documentation

M5. Download BLE_Arduino_EMG_Sender.ino from Canvas and load onto the Arduino IDE software.

M6. Change the variable defined as "**pennKey**" in the script. This variable will have to be different for each student as it is used to create identification variables for the Bluetooth Low Energy library (BLE).

M7. It is used to adjust the DeviceName and LocalName of the BLE Peripheral
NOTE: Do NOT edit anything else in BLE_Arduino_EMG_Sender.ino
NOTE: The sampling rate of the device is 1000 Hz
   a. Disconnect the arduino from all circuitry and upload the BLE_Arduino_EMG_Sender.ino file and then connect the Arduino to your circuit.
      i. **NOTE:** If you are having trouble uploading the script please follow the steps outlined here Troubleshooting the MKR WiFi 1010

Setting up your computer to receive the BLE Signal:

In order to receive the BLE signal, we will be using python (so that it runs on lab computers, your personal computers and eventually a raspberry pi).

M8. Install the Anaconda distribution if needed.

M9. **Create and activate** a virtual environment for this project with python 3.11 from either the environment file posted to canvas: emg_environment_mac.yml or emg_environment_windows.yml (depending on your operating system)

a. Open your terminal/anaconda prompt (macOS/windows)

b. Navigate to where you downloaded the environment file with

```
cd path_to_file
```

c. Run the following command to create your environment, where environment.yml is the environment file for your OS.

```
conda env create -f environment.yml
```

NOTE: If you do not want to use conda to create your environment, you can use native python and pip (instructions) to install the bleak numpy asyncqt qasync pyqt5 pyqtgraph packages. However you should not mix pip and conda, as you can break your python installation. Again, you should not mix conda and pip unless you know how each package will interact!

M10. Download the starter files from canvas (**Final_Starter_Code_Part_1.py**, **bluetooth_discover.py**, and **simple_GUI.ui**)

M11. The easiest way to run python files is through an integrated development environment or IDE for short. We are recommending that you use Visual Studio Code as it is a lightweight application that can easily be customized. After installing Visual Studio Code follow along with Python in Visual Studio Code to setup Visual Studio Code to run your Python Code. Make sure you know how to select the newly created python environment.

a. **If you are using Spyder or any other IPython System, you must run the code in an "external system terminal"**

M12. With your MKR WiFi 1010 powered on, run the **bluetooth_discover.py** script to find its address. You will have to scroll through to find your arduino. You are looking for the device label "**EMG_Sender_pennKey**" or "**MKR1010_pennKey**", where pennKey is your penn key. macOS will sometimes show EMG_Sender and other times will show "MKR1010"

a. On Windows and raspberry pi it will look like this

```
6C:80:67:11:28:E1: None
79:1C:72:8D:1F:D6: None
24:0A:C4:AD:94:36: EMG_Sender_pennKey
7D:DC:B6:FA:27:45: None
28:33:04:F2:24:E9: None
```

The address would be 24:0A:C4:AD:94:36

b. On macOS it will look like this

```
2D2BF05E-2565-F51C-0C95-526B141C90FD: AP-310344001FC3
7204FA9B-6833-F52D-3708-9816CC937D94: None
E52DA0E0-4701-025E-34C2-4196E6BA1A91: None
7769E0A3-DC86-353A-13C8-C6C9850ECEE2: MKR1010_pennKey
16B663B4-187A-FB21-8494-47D287CE665A: None
7B8C2D2F-4D0A-D528-22CD-6D4D44A35330: None
```

The address would be 7769E0A3-DC86-353A-13C8-C6C9850ECEE2

M13. Edit the **ADDRESS** variable in **Final_Starter_Code_Part_1.py** with the address of your device **DO NOT EDIT ANYTHING ELSE in the script**

M14. Your computer is now ready to receive the BLE data

BLE Connection Testing:

Test the system by using an input of a 10 Hz sine wave with 0.5 V Amplitude and 1V Offset from the Wavegen feature of the Analog Discovery. This Signal will be called **Test Signal 3**. The offset voltage is to ensure that no negative voltage is sent to the arduino.

M15. Connect the GND of the Analog Discovery to the Arduino's GND and the output of Wavegen to the A3 pin of the Arduino.
   a. **Make sure these are the only connections to the arduino.**

M16. Ensure that the Arduino is powered on and that the built-in LED is on and not blinking (*this means that the device properly went through the BLE setup function*)

M17. Make sure that your computer's bluetooth is on.

M18. Run the **Final_Starter_Code_Part_1.py** script
   a. **Make sure that your python editor has access to all the necessary support files. HINT: You may need to open the folder containing all your files**
   b. This will open up a simple GUI
   c. select the "Connect BLE" button on the window, wait until the message on the window says "Connected"
   d. Start the sine wave in wavegen
   e. Select the "Stream" button on the GUI
   f. **NOTE: Axis Information on the generated plot**
      i. X-Axis is index point and is **NOT** time. You can calculate time by using the sample rate from the Arduino code
      ii. Y-Axis is the bit value (out of 255) from the Arduino, **NOT** voltage.

M19. You should be able to see the sine wave from Wavegen, in which case, your BLE connection is working properly. If not follow the steps below to troubleshoot:
   a. If your python script throw one of the following errors: 'NoneType' object has no attribute 'lower', The object has been closed, or anything from the "bleak" package
      i. Close the GUI window and reset the MKR 1010 with the reset button. Wait until the built-in LED is on and not blinking. Then try again. If the script does not re-run, then would should close the terminal window and try again
      ii. You might need to repeat this step 2 times to get it working. If after multiple attempts you are still getting the error message, contact a member of the teaching staff
   b. If your python script throws an error that says something like "package not installed"

i. you will need to use pip to install the package and try again. However you should reset the MKR1010 being pressing the reset button

Integrate your circuit to the Arduino:

M20. Connect the output of your circuit (from the non-inverting amplifier) to pin 3 of the Arduino, and the ground of your circuit to the arduino ground.

M21. Disconnect the Arduino from your computer. In order to power your Arduino you will need a 9V battery and a separate 5V regulator (**NOTE: You will have 2, 5V regulators**). You will need to connect the positive terminal to Vin, the negative terminal to Ground and then connect the Vout pin (of the 5V regulator) to the Vin pin on the Arduino

M22. Using **Test Signal 1** and **Test Signal 2** verify that your circuit is working and that you can read data in python. You may need to pause data collection and zoom in to the graph to verify.

EMG signal:

M23. You will finally test your system on your own body. Remember, for safety reasons, your Arduino and circuit cannot be powered by anything other than 9 V batteries. There should be no physical connection between the circuit and arduino and the rest of the world.

M24. Adjust the gain of the differential amplifier to approximately 1707 (refer to the data sheet). This gain may change depending on each person

a. NOTE: the Arduino MKR WiFi 1010 cannot read values over 3.3 V, so be mindful of how much you gain

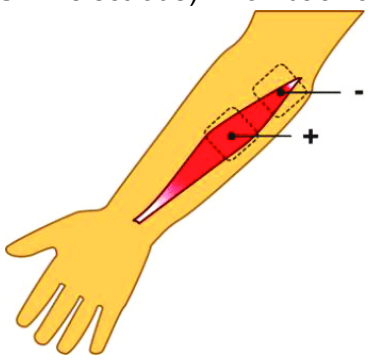
M25. Disconnect all signals from your circuit (including batteries)

a. It is ok to be connected to the arduino as long as it is not powered

b. STOP AND DOUBLE CHECK YOUR CIRCUIT IS DISCONNECT FROM ALL POWER SOURCES

M26. Connect the surface electrodes as follows and then use the EMG (SS2LB) cables and DB9 to breadboard adapter to connect the electrodes to your circuit. You will be measuring the EMG signals from your forearm (specifically the flexor carpi radialis).

a. Red (positive electrode) → right forearm [further from elbow]
b. White (negative electrode) → right forearm [closer to elbow]
c. Black (GND electrode) → on back of left hand

i. 

M27. **Make sure that your Arduino is disconnected from the computer and powered from the separate 9V battery through the separate 5V regulator.** If you are unsure how to power the system **ASK**. Your system will now require 3 different 9V batteries.

M28. Connect the batteries to your circuit. And wait until the built-in LED on the Arduino is on and not blinking.

M29. Now you can start collecting your EMG signal. In order to see the EMG signal you

will need to flex your forearm. Do this by bending your wrist such that the fingers on your right hand point to the left.

DELIVERABLES

D1. Submit a complete circuit diagram including both parts 1 & 2, **and use the appropriate gain for your differential amplifier**.

D2. Submit a video of the live plot from measuring your EMG signal. We should see the electrodes coming from you, the circuit, and your screen.

D3. Submit an image of 2-3 flexes on the plot from your EMG signal.