Lab #5 part 2 – sEMG on multiple muscles

Overview:

This lab is a "small" continuation from last week. The only difference – it's bigger and better! Last week we collected an sEMG from one muscle and (perhaps) used it to control one LED. Now we'll put sets of electrodes on two or three different muscles, giving us lots of options about what to control.

Like last week, all calculations will be on the PyBoard, not the host.

In principle, the PyBoard can handle at least three muscles. However, we may or may not have enough AD8232 boards for everyone, and your ideas for what to do with the muscle signals may work better with just two muscles, so you may wind up using just two. The rest of this document will mostly just refer to three boards for simplicity, but bear in mind that you may only have two.

Legal reminder:

This lab involves capturing your sEMG. Legally, this signal does represent medical data. As such, federal law says that you have the right to keep it private, and we will of course respect that law. Please refer to our separate legal sheet (which you should have read and signed).

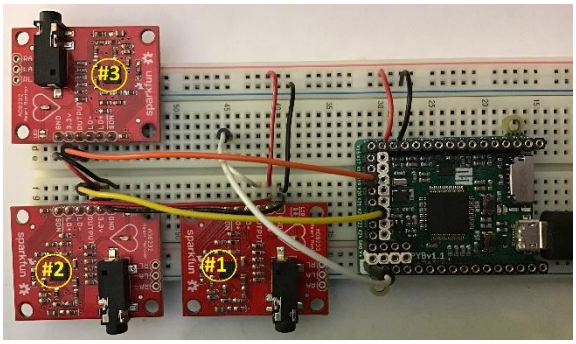
Hardware changes

The main difference between the hardware this week vs. last week is simple – this week we have more of it! So:

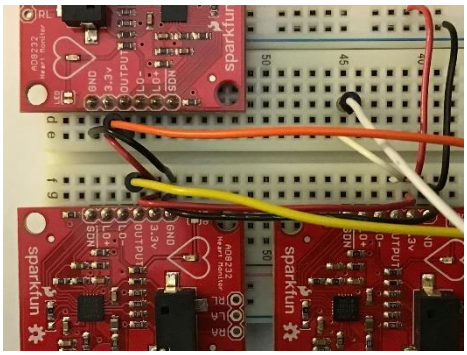
- Still just one host computer
- Still just one PyBoard (though more wires driving it)
- Three AD8232 boards rather than just one
- Three sets of electrodes rather than just one

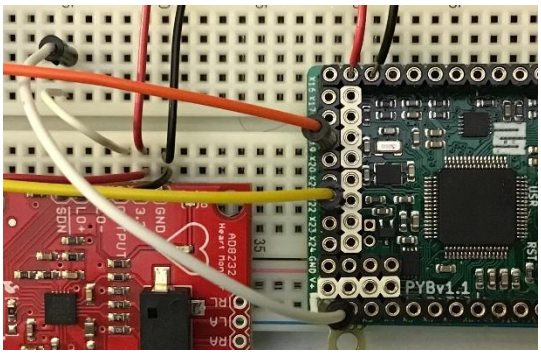
The three AD8232 boards, of course, now each drive an output. So where we previously had a wire from our one lonely AD8232 output to the PyBoard pin X1, we now have each of the three AD8232 boards driving outputs. The three AD8232 boards drive PyBoard pins X1, X19 and X22 respectively.

With three AD8232 boards, we now of course have to supply each one with power. They still all get their power from the PyBoard 3.3V power output. The final hookup looks like this:



Then here are some closeups:





You'll note that the original AD8232 board is unchanged. We've added the two new AD8232 boards in about the only places we could fit them! Their outputs drive the PyBoard with the new yellow and black wires (still reserving black and red for ground and power, and using new colors so as to not confuse them with the white wire). The boards are numbered #1, #2 and #3 (which will correspond to parameters `musc1_on, musc2_on, musc3_on` in *`muscle_fun`*`()` later).

Power and ground are wired in what is often called a "daisy chain;" the main power/ground rows on top drive our original AD8232, which drives one of the new ones and then the next new one in a long chain. The black ground wire connecting the bottom two AD8232 boards is hard to see in the picture, but it's there.

I've connected the outputs of the two new AD8232 boards directly to the PyBoard, which will make it a bit difficult to check them out with an oscilloscope – that was perhaps not the best idea!

Software changes

We'll use a very-slightly-different Python program this time; you can find it in *5b_emg_process_pyboard_3chan.py*. Pretty much the only difference is that we now use 3 ADC channels rather than 1. This implies a few small changes to the code.

First, the algorithm now calls *`muscle_fun`*`(musc1_on, musc2_on, musc3_on, changed)`. The new parameters `musc2_on` and `musc3_on` tell you the state of muscles #2 and #3, just as `musc1_on` did for the original muscle. The parameter `changed` now tells you if *any* of the three muscles has changed. As noted above, the 1, 2 and 3 correspond to the board numbers in the picture above.

Second, the code in multiple locations works with three ADC channels rather than one. Each of these has its own set of filter parameters. I wrote the file with all three channels having the same parameters. However, you may find it better to give each channel different parameters for Schmidt0 and Schmidt1 (e.g., if different muscles on your body produce different signal strengths, or some electrodes are making better skin contact than others).

If you only have two, then you should go to roughly line #169, where you'll see the following code:

```
ch1_On,changed0=Schmidt(ch1_On,ch1_run_sumsqr,ch1_Schmidt0,ch1_Schmidt1)
ch2_On,changed1=Schmidt(ch2_On,ch2_run_sumsqr,ch2_Schmidt0,ch2_Schmidt1)
```

```
ch3_On,changed2=Schmidt(ch3_On,ch3_run_sumsqr,ch3_Schmidt0,ch3_Schmidt1)
muscle_fun (ch1_On, ch2_On, ch3_On, changed1 or changed2 or changed3)
```

With the third channel unconnected, it will be especially noisy and may cause `changed2` to bounce unpredictably. You can fix this (and make the code slightly faster) by changing the code as follows:

```
ch1_On,changed0=Schmidt(ch1_On,ch1_run_sumsqr,ch1_Schmidt0,ch1_Schmidt1)
ch2_On,changed1=Schmidt(ch2_On,ch2_run_sumsqr,ch2_Schmidt0,ch2_Schmidt1)
~~ch3_On,changed2=Schmidt(ch3_On,ch3_run_sumsqr,ch3_Schmidt0,ch3_Schmidt1)~~
muscle_fun (ch1_On, ch2_On, ch3_On, changed1 or changed2 ~~or changed3~~)
```

That's all that's different for this week. Of course, the ability to use three muscles gives you a lot more flexibility to do creative things inside of *`muscle_fun`*`()`!

Suggestions for `muscle_fun()`

You can program whatever interesting things you like inside of *`muscle_fun`*`()`. Here are a few thoughts to spur your creativity:

- Each of the three muscles could turn one LED on/off
- Two muscles could work together, with their four combinations picking one of the four LEDs.
- You can use sound – three muscles could choose one of eight notes. Or two groups could work together to play a duet. (If you want to try working with sound, you should read the hints from last week about how to do it).
- Just as you may have used global state to program sequences of events last week, you can now program even more complex sequences.

As numerous amputees have discovered the hard way, controlling multiple muscles independently is not as easy as it sounds. Try it – can you control your biceps and triceps independently? An alternate solution is to use two biceps (i.e., your left and right arms) instead.

What to turn in:

A lab report that contains your code for *`muscle_fun`*`()` and describes what it does. There are no questions to answer, and the only checkout is to show us your creation!

You need only turn in one report for each group.

Debugging (repeated/modified from last week)

Hopefully all of your code will work smoothly and correctly 😊. But what if it doesn't? How can you figure out what part is going wrong? Debugging can be an art as much as a science – and an art that you will get *lots* of practice at in the next few years!

The debugging suggestions from last week are hopefully still useful: using an oscilloscope to check signals, adding *`print`*`()` statements to *`muscle_fun`*`()`, making the Schmidt-trigger thresholds higher or lower, and using *adc_samples.py* to examine your signal. See last week's lab for a refresher on these if you like.

In addition, one extra debugging suggestion for this week is that you can easily swap electrodes by unplugging their headphone plugs from the AD8232 boards and plugging them into different boards. That may help you determine if the issue is the electrodes or the boards.

Bonus versions (repeated/modified from last week)

As noted above, having multiple muscles available gives you *lots* of good options for exercising your creativity.

Another, harder, option is that we talked in class about *mode switching* for a prosthetic that has multiple degrees of freedom. Can you implement mode switching here? It's probably easier to use LEDs as your output rather than trying to drive our prosthetic arm for now, and mode switching is probably more suited to a final project than a one-period effort – but feel free to get started on it if you like.

Disconnecting from the host (repeated from last week)

So far, we've used Thonny to run programs on the PyBoard. Thonny is convenient because it lets us use a keyboard and a display, allowing us to edit programs and easily see output. But Thonny requires the host USB connection, and it's often nice to run "untethered."

The instructions for running untethered were in last week's lab; please refer to that lab for full details.