**FES/Exoskeleton Experiment Protocol**

1. **FES Setup:** Connect the stim board to the host computer according to the instructions in Mike Brodnik’s “UECU Operation and Setup”
   1. Electrode Placement: Wash and dry the participant’s upper arm, then place a pair of electrodes on the skin to span the bicep muscle mass. Test this placement using the slider GUI. Try different electrode placements until the resultant motion is predominantly elbow flexion that begins at a reasonably low pulse width.
   2. Determine Amplitude: Use the test60, test70, and test80 executables to investigate the effect of different amplitudes on the range of pulse width values that produce motion. Choose the amplitude that requires low pulse width values to initiate motion and produces various degrees of flexion over a wide range of pulse width values.
   3. Record Relationship Between Pulse Width and Observed Motion: Use the executable corresponding to the optimal amplitude found in the previous step to vary the pulse width. Record the lowest pulse width value that produces motion, the maximum value that the participant can comfortably handle, and values that correspond to angles along the desired trajectory (including the pulse width that achieves the max angle of the trajectory).
   4. Generate Trajectory and Open Loop Data Files: Update the following variables in traj\_data.m as needed:
      1. spec
      2. theta1
      3. theta2
      4. t
      5. tstep
      6. stim
      7. angle
      8. minstim

Run the code, then transfer the generated files (traj\_data.txt and open\_stim.txt) to the appropriate folder or computer as needed.

* 1. Update Execution Code Parameters: Update the following values in UDP\_feedback.cpp:
     1. GAIN
     2. PULSEMAX
     3. total\_t

Recompile using “-lm” in order to link to the math header file.

g++ UDP\_feedback.cpp –lm –o ExecutableName

1. **Exoskeleton Setup:** TBD
2. **Experiments:** Random Order
   1. FES Only (3)
   2. Exoskeleton Only (3)
   3. FES & Exoskeleton (3)

**Important Files & Variables**

* test60.exe
  + Use: Test 60 mA amplitude by changing the pulse width in 1 µsec increments starting at zero.
  + Inputs: Press 1+ENTER to increase the pulse width, 2+ENTER to decrease the pulse width, or press 0+ENTER to end stimulation.
  + Outputs: Current pulse width is printed to the terminal.
  + Source: test\_amp60.cpp
* test70.exe
  + Use: Test 70 mA amplitude by changing the pulse width in 1 µsec increments starting at zero.
  + Inputs: Press 1+ENTER to increase the pulse width or press 0+ENTER to end stimulation.
  + Outputs: Current pulse width is printed to the terminal.
  + Source: test\_amp70.cpp
* test80.exe
  + Use: Test 80 mA amplitude by changing the pulse width in 1 µsec increments starting at zero.
  + Inputs: Press 1+ENTER to increase the pulse width or press 0+ENTER to end stimulation.
  + Outputs: Current pulse width is printed to the terminal.
  + Source: test\_amp80.cpp
* traj\_poly.m
  + Use: Generate trajectory and open loop pulse width values and save in text files.
  + Inputs: Change variables listed in section 1d above as needed.
  + Outputs: traj\_data.txt, open\_stim.txt
* UDP\_feedback.cpp
  + Use: Stimulate using the previously determined open loop commands with added positional feedback. The gain can be set to zero to eliminate feedback.
  + Inputs: Change variables listed in 1e above as needed. The output files from traj\_poly.m should be in the current folder when compiling and running.
  + Outputs: logged\_data.txt, pulse\_log.txt
* stim\_playback.cpp
  + Use: Stimulate using the same pulse width commands previously used in UDP\_feedback.cpp.
  + Inputs: pulse\_log.txt
  + Outputs: playback\_data.txt