Overview:

The insect experiment testing platform provides a flexible platform to allow a wide range of insect neuromuscular experiments to be performed. The platform consists of a wireless, battery-powered backpack capable of performing four channel biphasic voltage stimulation along with four channel EMG recording at 2kHz per channel. This wireless backpack can be coupled with an LED arena designed to optically stimulate a tethered insect to yaw rotation. When coupled with the LED arena, the backpack placed on the insect can be used to study the muscle movements during flight as well as study the effects of voltage stimulation on the insect muscles.

The backpack and LED arena are controlled from a user interface that allows the LED arena rotation speed and direction to be set along with wirelessly controlling the backpack and displaying and saving the EMG data.

User Interface:

The User Interface (GUI) allows the operator to stream and save EMG data wirelessly from the backpack, control the stimulation pattern, and set the LED arena rotation parameters. The GUI logs the entire experiment to allow the operator to explore the relationship between stimulation, EMG performance, and light stimulation. The GUI creates two folders “log” and “EMG Data” and saves the button presses into a text file in the “log” folder and all the streamed EMG data as a csv file in the “EMG Data” folder.

When the GUI is opened, the operator inputs the com ports for the USB host device and opens that port. That allows the backpack to stream EMG information or stimulate. The operator can also input the com port for the LED arena and start the arena.

The user interface (GUI) is written in python 3.5 with tkinter. Figure 1 shows the layout of the program. I used pyinstaller version 3.2 to create an exe file for the GUI. Use the following command “pyinstaller --onefile gui.py” and the exe will be added in a “dist” folder.

NOTE: The arduino IDE must be installed to have the FTDI USB drivers for the serial communication.

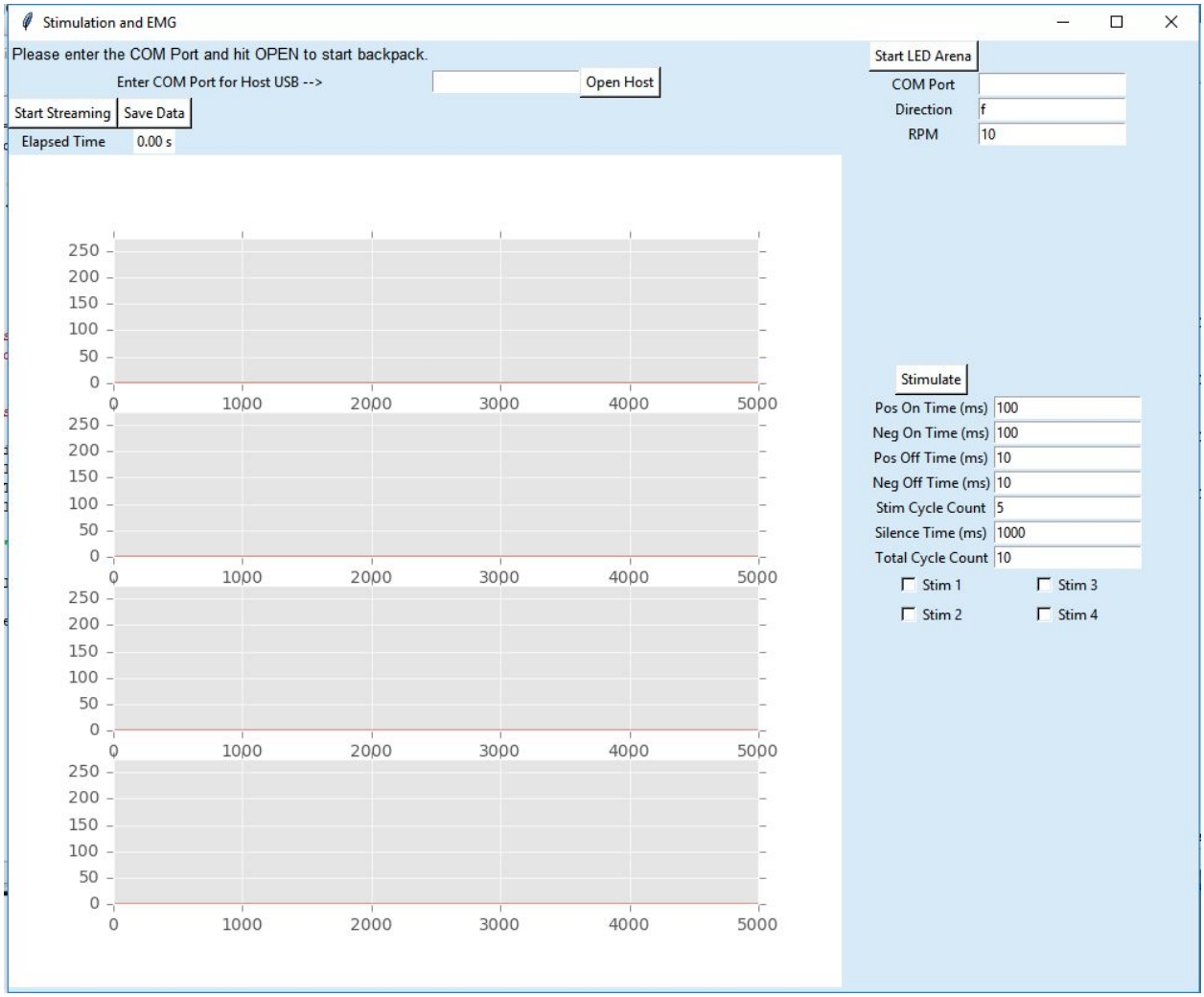

Figure 1: GUI to control LED arena and wireless backpack

Embedded Software

The embedded software consists of three parts - the Host, Device, and LED Arena. The software was written for the arduino platform using the arduino language and some C. The Host runs on the USB Simblee dongle and acts as the intermediate between the python program and the wireless device through a serial port. The Device is the software that runs on the backpack PCB. The wireless network uses the nordic gazelle protocol to allow higher throughput than is possible using standard bluetooth.

Figure 2 shows the data flow between the four software components (GUI, HOST, DEVICE, LED).

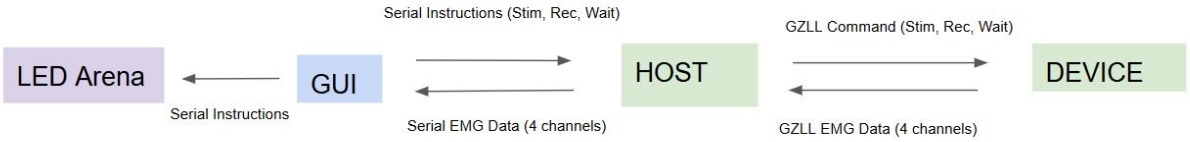

Figure 2: Data flow between software elements

Host Software:
The Host software runs on the USB dongle that bridges the wireless backpack with the python program by sending and receiving data over a serial port.

The host receives a command over the serial port and sends the message to the device. If this is a stimulation command the host does nothing further until another serial command is received. Figure 3 shows the Host command structure.

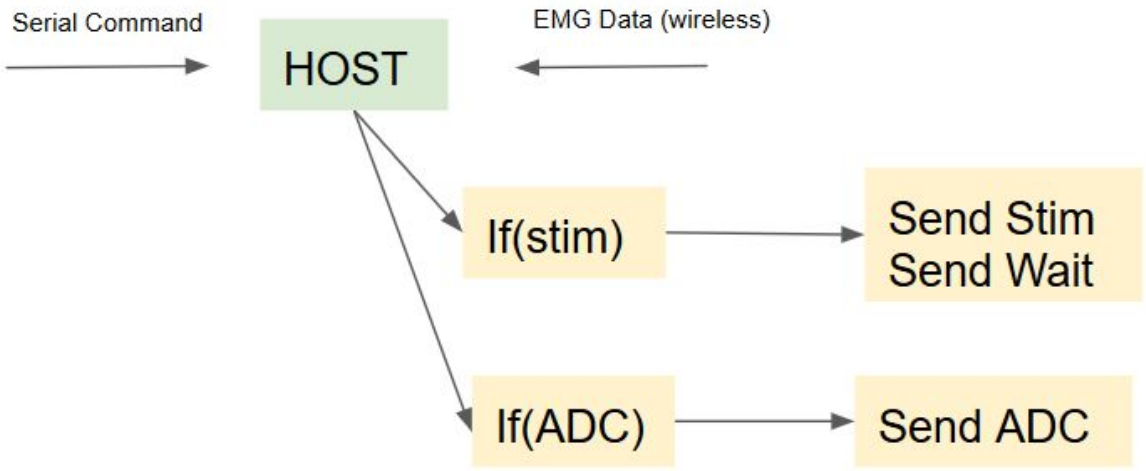

Figure 3: Host Data Flow

If the serial command is for the Device to stream data, the Host receives the packet containing the ADC data and adds it to the head of a ring buffer as shown in Figure 4. A timer on the Host every 2ms indicates when a serial packet should be sent from the Host to the computer containing the wireless data. This serial packet is created by pulling 16 bytes from the tail of the ring buffer, calculating a checksum on those 16 bytes, and then sending a serial packet to the computer in the following format.

|'A'|'A'| 16 bytes of data | checksum |'\n'|

The two initial AA values are part of the SLIP (Serial Line Internet Protocol) packet setup designed to minimize data corruption. The python GUI program is set to watch for two 'A'

characters and save the next 17 bytes and stop at the new line character (‘\n’). It then calculates a checksum on the 16 data bytes and compares it to the 17th character. If the calculated checksum matches the checksum sent in the SLIP packet, the data is added to the plotting array.

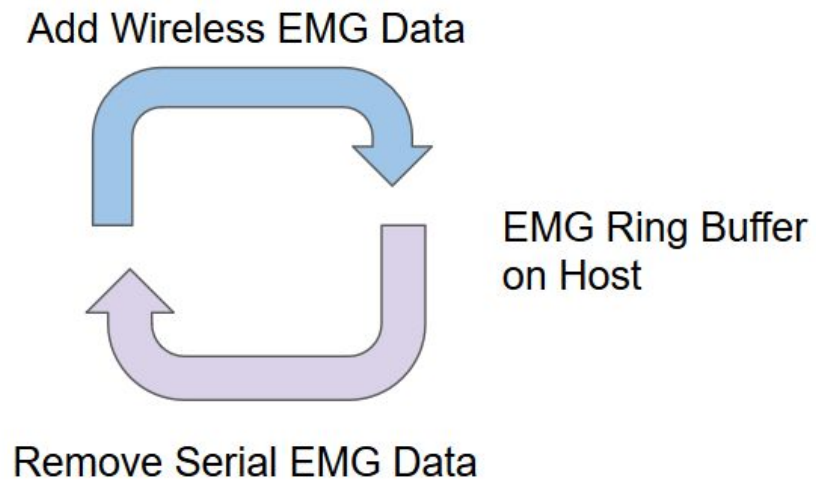

Figure 4: Host EMG ring buffer. Wireless adds and serial removes

Device Software:
The Device has a state machine that controls the operation of the backpack. It has three possible states; WAIT, ADC, and STIM. Upon turning on the Device, it starts in the WAIT state until a wireless command from the Host is received. Upon receiving a command to either start the ADC or Stimulate, it enters the respective state in the state machine.

While in the ADC state an ADC interrupt fires every 125uS and adds the results to the head of the ring buffer. After adding the data to the ring buffer it reconfigures the ADC for the next desired channel. Every 4th channel it returns to the original channel allowing four ADC channels to be sampled every 500uS (4*125uS) for a total sampling frequency of two samples per millisecond for each of the four channels.

While the ADC interrupt is filling the ring buffer with the four EMG channels, a timer interrupt is firing every 2mS to indicate to the Device when to send a wireless packet. The packet is prepared by pulling 16 bytes from the tail of the ring buffer and transmitting to the host as shown in Figure 5. This transmission is performed every 2mS while the device is in the ADC state. This transmission stops when the device receives the WAIT command.

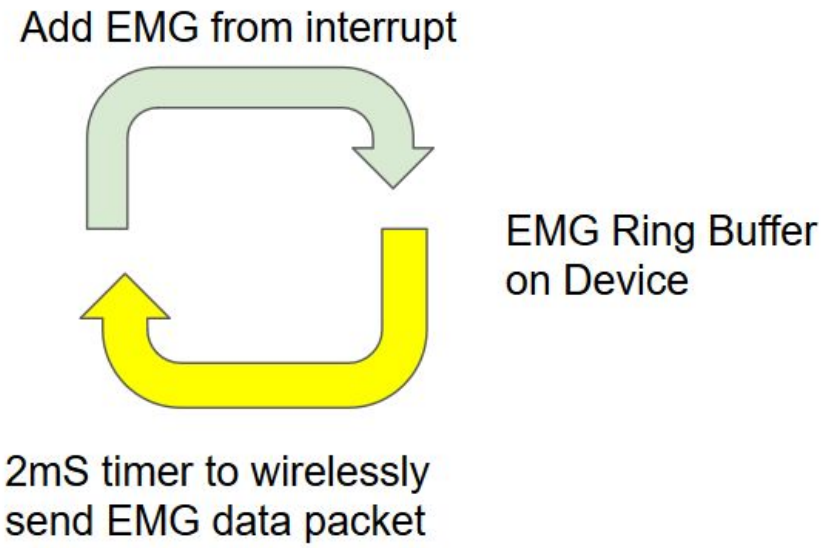

Figure 5: Device Ring buffer to add data and wirelessly send

When the device enters the STIM state, a packet is received from the host with the stimulation parameters. A struct is created with the received parameters and a stimulate function is called which sets GPIO ports in various configurations to allow the hardware to create the desired stimulation pattern. The hardware section below details the stimulation hardware design.

The embedded software is a mixture of standard C and arduino language. Due to the timing requirements to achieve a 2kHz sample rate on four different ADC channels an ADC interrupt was setup using the nordic documentation to bypass the standard arduino analogRead function.

LED Arena Software:
This software simply reads a character, either ‘f’ or ‘b’, along with an integer to set the RPM for the arena. This can be done over any COM port interface. The software uses a library provided by adafruit to set the individual LEDs.

Hardware:
Our system consists of a two USB COM devices connected to a python program. The first is a USB COM dongle (Host) that wirelessly interfaces with a custom backpack Device. The second is an LED Arena that accepts command over a serial USB interface. Figure 6 shows a schematic of the system.

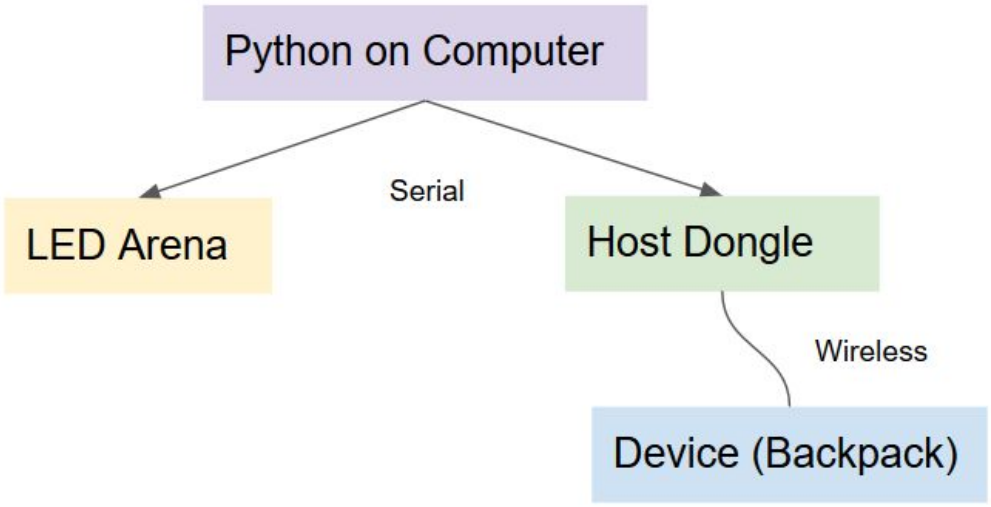

Figure 6: Communication paths between system components

Host USB dongle: The USB dongle consists of a Simblee development board along with USB adapter module to interface the serial port to the microcontroller. This board is part of the Simblee development kit.

This dongle provides a gateway between the wireless backpack and the python program. Using serial communication a command is sent from the python program to the USB dongle which then sends the message wirelessly to the backpack.

SimbleeBoard: https://www.sparkfun.com/products/13768
USB adapter: https://www.sparkfun.com/products/13209

Backpack: The backpack is a custom PCB consisting of a microcontroller (Simblee), power supplies, EMG circuits, and stimulation circuits.

The backpack receives a wireless command from the USB dongle and performs the requested command. The commands are either record and transmit EMG or perform stimulation. The commands are performed until a wait signal is sent.

Stimulation Hardware:
The stimulation is performed using hardware switches triggered through GPIO ports. Figure 7 shows how a biphasic voltage output can be shaped using two switches with the outputs tied together. The input to switch 1 is the positive voltage, and the input switch 2 is the negative voltage. When switch 1 is active

through signal S0 the output signal is positive and when switch 2 is active through signal S1 the output signal is negative. The code ensures that both are not active at the same time. By alternating S0 and S1 control signals by using software timers, a precise biphasic voltage stimulation pulse can be created.

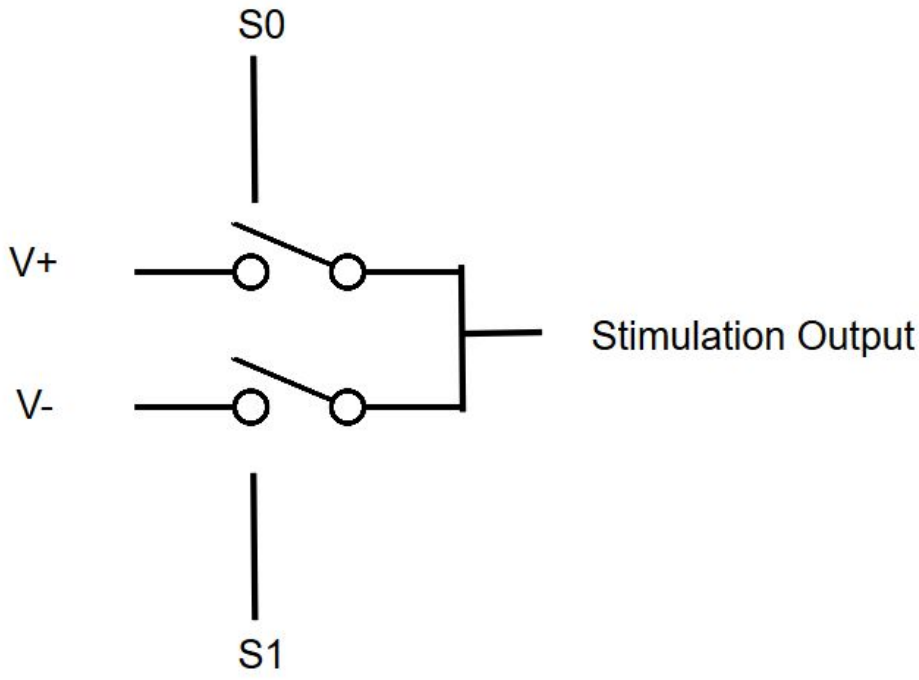

Figure 7: Hardware to create voltage stimulation output

Stimulation Parameters:

Various parameters of the Stimulation Output can be controlled through software. The GUI has input parameters for various settings and these can be changed to change the stimulation pattern. The stimulation is a set of alternating positive and negative stimulation with pauses between each pulse. After each set of pulses, a longer wait is set and the set repeats. This combined pulse set + wait is repeated for the desired number of repeats. Figure 8 shows the two pulse sets consisting of 4 pulses with a delay between each pulse set.

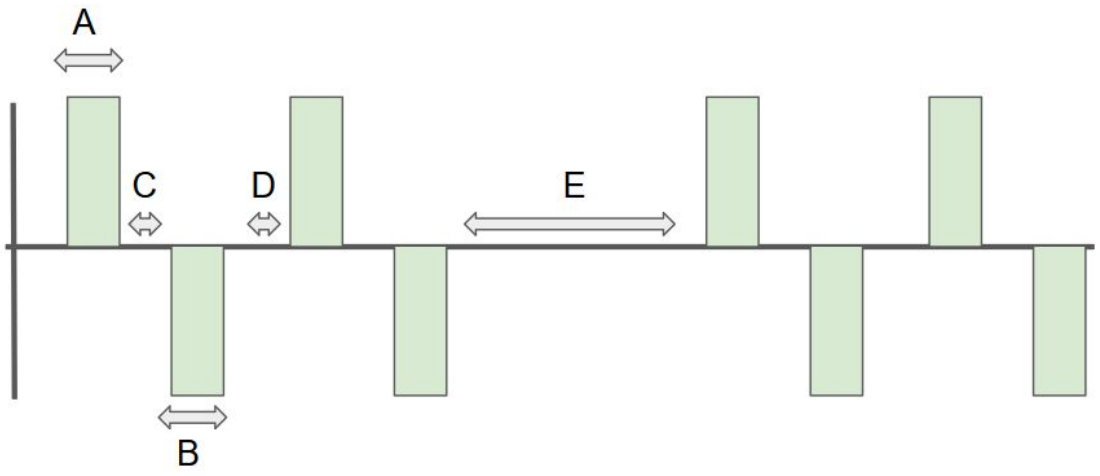

Figure 8: Example voltage stimulation with parameters shown

In addition to inputting the above parameters into the GUI. The GUI also has 4 check buttons (Stim 1 … Stim 4) that can be used to select which electrodes are used for stimulation. Based on the table below, the GUI selects the pin combo [A-P] which is sent to the Device. The backpack has four stimulation channels. These channels can be used in any desired combination based on the table below and the backpack Device decodes the letter into into the proper combination.

Pin Combo	Stim 4	Stim 3	Stim 2	Stim 1
A	None	None	None	None
B	None	None	None	Yes
C	None	None	Yes	None
D	None	None	Yes	Yes
E	None	Yes	None	None
F	None	Yes	None	Yes

G	None	Yes	Yes	None
H	None	Yes	Yes	Yes
I	Yes	None	None	None
J	Yes	None	None	Yes
K	Yes	None	Yes	None
L	Yes	None	Yes	Yes
M	Yes	Yes	None	None
N	Yes	Yes	None	Yes
O	Yes	Yes	Yes	None
P	Yes	Yes	Yes	Yes

EMG:

The EMG signal is created by a differential amplifier (AD8222). A single ended EMG was created by tying one input to ground and inserting the other into the muscle. This creates a simpler EMG system. The output of the EMG differential amplifier is AC coupled using a 0.1uF capacitor and then a DC offset is created using a 1M and 3.3M voltage divider. This DC offset allows the signal to be read via the ADC on the Simblee microcontroller.

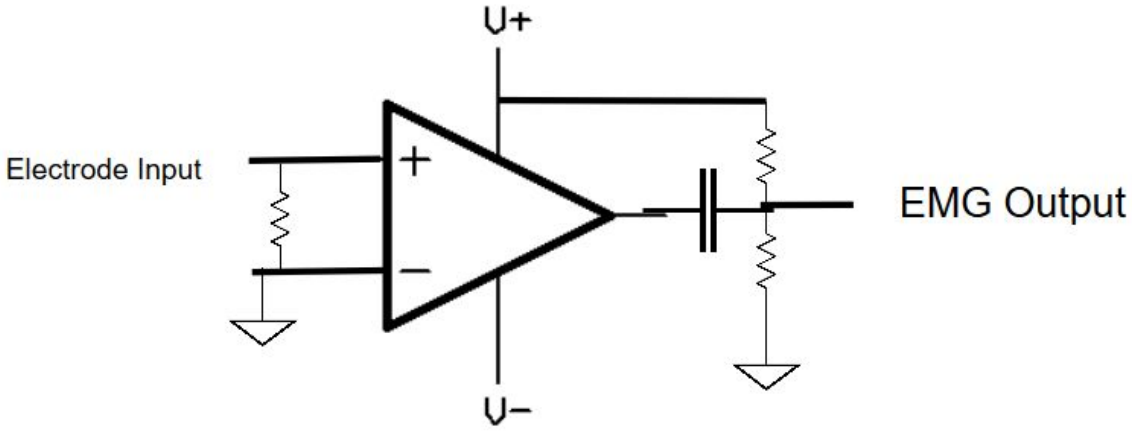

Figure 9: EMG circuitry with AC coupling and DC offset

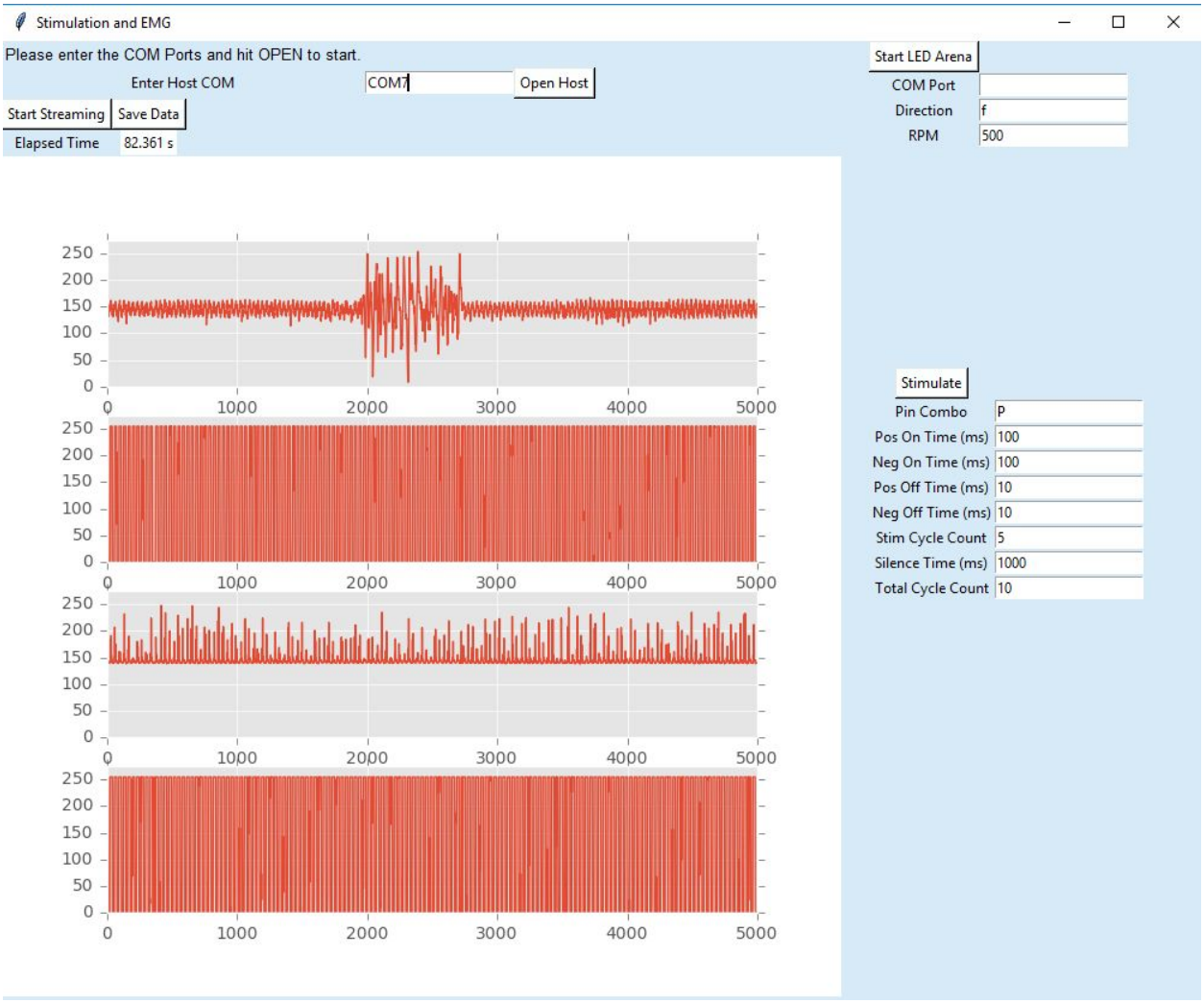

Figure 10: Received EMG data on the GUI. The noise on the other channels is due to dangling wires leading to random signals.

Interface:

The interface port is a 10 channel FFC connector. The outermost channels are ground and starting from the top of the port, the stimulation and recording alternate across the ports as shown in the table.

Ground
EMG1 In
Stim1
EMG2 In

Stim 2
EMG4 In
Stim4
EMG3 In
Stim 3
GND

LED Arena: The LED arena is built from six 32x32 pixel LED panels set in a hexagonal array. The arena is set to rotate vertical stripes of white light around the arena based on a command sent over a serial port from the python program. The rotation speed and direction of the vertical stripes can be set using the python program. An Arduino is set as the interface between the LED arena and the python program.

Software and packages required:
The software for this project was produced using the following libraries and programming environments.

LED Arena: The LED arena is programmed using the Adafruit GFX library and Adafruit RGB Matrix library available here:
GFX: https://github.com/adafruit/Adafruit-GFX-Library
Matrix: https://github.com/adafruit/RGB-matrix-Panel

Both libraries are also available as included zip files in the software zip file. The arduino IDE was used for the software development.

A tutorial on the panels (including wiring diagrams) are available here:
https://learn.adafruit.com/32x16-32x32-rgb-led-matrix/

Simblee: The simblee arduino code requires the arduino IDE v 1.6.5 and was built using the Simblee Boards 1.0.3 library version provided by RFdigital. This library can be installed using the Arduino IDE board manager.

Python:
Python 3.5.1 was used to develop the software
Pyinstaller 3.2 was used to create the GU exe file: http://www.pyinstaller.org/

Module versions (from pip, not all directly used)

cycler==0.10.0
future==0.15.2
matplotlib==1.5.1
numpy==1.11.1
pandas==0.18.1
pefile==2016.3.28
PyInstaller==3.2
pyparsing==2.1.5
pypiwin32==219
pyserial==3.1.1
python-dateutil==2.5.3
pytz==2016.4
six==1.10.0