Electromyography Application Control

Enhancing Multidisciplinary Knowledge Appropriation

Amilton da Costa Lamas(ORCID ID 0000-0003-1664-4559)
Polytechnic School
Faculdade de Engenharia Elétrica
Pontifícia Universidade Católica de Campinas
Campinas Brasil
amilton@puc-campinas.edu.br

***Abstract*—Biomedical Engineering courses provide opportunities for students to develop technical and non-technical skills due to their interdisciplinary characteristics. A well-designed course should promote the appropriation of both engineering and biomedical contents in a practical fashion. In this work we report on the use of electromyography signals to control applications developed in three different experiments. The experiments were implemented with an OpenBCI platform using Python scripts. Three experiments are described: Music Control, Facial LED Control, and Virtual Piano Playing. The proper performance of the experiments was achieved after corrections were made in the platform.**

Keywords— application control; Biomedical Engineering; Electromyography; integrated laboratory

I. INTRODUCTION

Biomedical Engineering courses combine the best of Electrical Engineering and Biomedicine worlds. This blend results on framework that offer tremendous opportunities for student development if the course's setup takes the advantage of the interdisciplinary aspects of these fields. A curriculum program can be arranged so that it properly mixes the core concepts of both pathways. A suitably trained electrical engineering student can significantly contribute to the development of solutions for biomedical challenges. On the other hand, if the student has some ability or sensitivity towards Biomedicine questions, he/she can rapidly identify opportunities to develop competencies in electrical engineering field. Either way one always gets positive results.

The Biomedical Engineering course staff at PUC-Campinas is aware of the relevance these issues so much that there are multiple project classes suitable to promote activities of such nature. By the time the student reaches the fourth semester they have acquired or are in the process of acquiring knowledge on Python programing; human anatomy and neuro anatomy; histology, physiology, and biophysics applied do engineering; electrical circuits; kinesiology, and biomechanics. This moment comprises excellent opportunity to implement a project involving these expertise's.

Projects were conducted in an Integrated Laboratory discipline. The discipline challenge to the students was to propose and develop laboratory experiments/demonstrations where they would practice all the accumulated concepts. Furthermore, the project's outcome must be useful to promote the mental abilities, cognitive process evolution and muscular training of handicapped people, especially children with cerebral palsy. After a few discussions the students proposed the development of three different experiments based on electromyography (EMG). There were three different platforms available for experiment/demonstration development: a) Backyard Brains [1], b) Myoware [2], and c) OpenBCI [3]. The students selected the last one mostly because it was the all-around most flexible for experiment creation.

II. BACKGROUNG

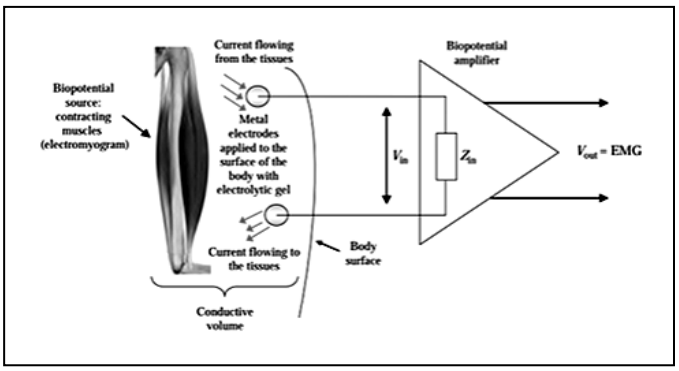
A. Electromyography

Nerve and muscle cells generate bioelectric signals that are the result of electrochemical changes within and between cells. If a nerve or muscle cell is stimulated by a stimulus that is strong enough to reach a necessary threshold, the cell will generate an action potential. The action potential, which represents a brief flow of ions across the cell membrane, can be measured with intracellular or extracellular electrodes. Action potentials generated by an excited cell can be transmitted from one cell to adjacent cells via its axon. When many cells become activated, an electric field is generated that propagates through the biological tissue. These changes in extracellular potential can be measured on the surface of the tissue or organism by using surface electrodes [4].

Electromyography is a medical technology that maps, in time and space, the electrical activity of the electric potentials from muscles. When a resting nerve or muscle cell, with a membrane potential of about 200 mV with respect to the external medium, is stimulated, a wave of depolarization spreads over the surface of the cell. The resting cell has no dipole moment, but while the cell is undergoing depolarization it can be electrically represented by a time-varying electric dipole moment that goes to zero after the resting membrane potential is restored in a process of repolarization. Such changes will lead to local variations in electric potential, which gives rise to variable bioelectric currents which irradiates energy in the form of electromagnetic waves. These waves are then detected by surface electrodes, in the present case, and the electric signal is sent to a device for proper electrical treatment and process. Recordings of muscle activity can check for normal functioning of nerve stimulation of muscle.

Measurements at a few distances along a muscle can determine conduction velocities along the stimulating nerve. The most common surface electrode used for EMG analysis is like the disc Ag–AgCl electrode. Eventually the coupling between muscle and electrode can be improved by using electrode gel between the electrode and the tissue. Nowadays self-steaking adhesive electrodes do not need the use of such gel. Figure 1 shows an electrical schematic representation of the tissue-electrode arrangement.

Fig. 1. Tissue – electrode coupling representation [5].



The EMG signal emerges from the electrical currents generated in muscles during contraction and expansion (relaxation) due to neurologically activation. This signal was used as input in the OpenBCI platform.

B. OpenBCI Platform

OpenBCI is an open-source brain–computer interface platform, created by Joel Murphy and Conor Russomanno, after a successful Kickstarter campaign in late 2013. It's known for providing research grade brain computer interfaces at affordable prices. OpenBCI boards can be used to measure and record electrical activity produced by the brain (EEG), muscles (EMG), and heart (EKG), and is compatible with standard EEG electrodes. The OpenBCI boards can be used with an open-source graphics user interface (GUI) or can be integrated with other open-source EEG signal processing tools. The OpenBCI 32-bit board uses the ADS1299, an IC developed by Texas Instruments for biopotential measurements. The board, named Cyton, uses a microcontroller for on-board processing and the 16bit version uses an Arduino-compatible ATmega328P IC. It uses a PIC microcontroller and write the EEG data to an SD card or transmit it to software on a computer over a Bluetooth link. For this work the platform consisted of a) the 32bit Cyton board; b) an OpenBCI Dongle, which allows data communication through Bluetooth between the board and the PC, and a c) Graphics User Interface (GUI).

The GUI for EMG measurements consists of 3 widgets; 1) Time Series; 2) EMG, and 3) Network, which was set to LSL data transmission. The Time Series is the main widget for displaying biosensing data. It processes and displays the electrophysiological signal in real-time, with each line graph representing the voltage detected at one point in time by an electrode. It measures the absolute amplitude of the signal voltage in units of µVrms (microvolts, root mean squared). Figure 2 shows the Time Series widget with channels 1 and 2 active, where channel 2 shows an EMG signal.

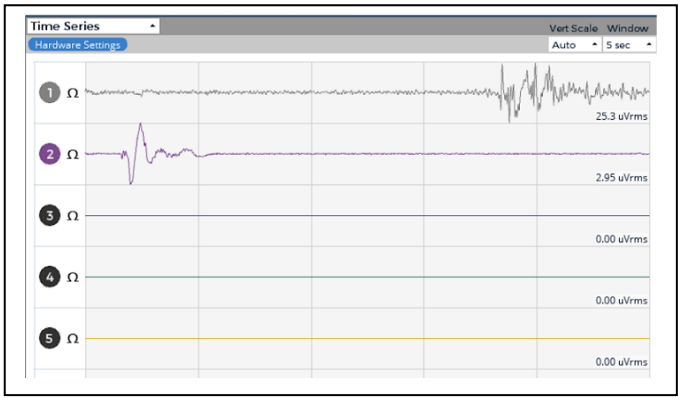
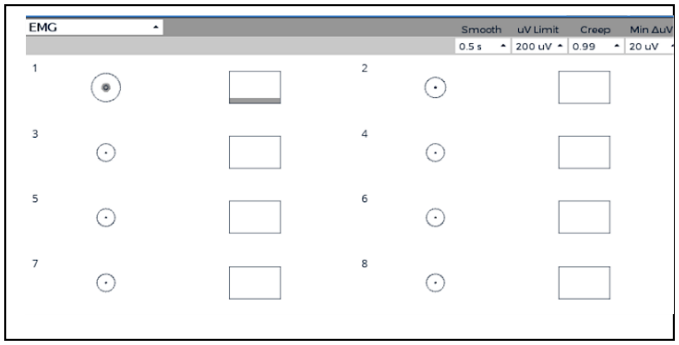


Fig. 2. Time series widget [3].

The EMG widget defines the parameters for EMG signal visualization. The EMG strength is a collection of raw voltage values averaged (or smoothed) over a known window of time. The bigger the window the signal is averaging over, the smoother the data. There is an upper threshold (the outer dark blue circle of the circular visualizer) and a lower threshold (the inner dark blue circle) for the constantly updating "comfortable EMG range." mapping the current EMG (the filled circle that matches the color of the channel) value between the upper and lower thresholds. This pseudo-analog mapped value is represented more clearly in the bar graph off to the right of each channel's circular visualizer. The upper threshold is constantly creeping downwards, and the lower threshold is constantly creeping upwards until they get the Min ΔµV away from one another. This ensures that the overall system never creates an upper/lower flex range that is too big to influence



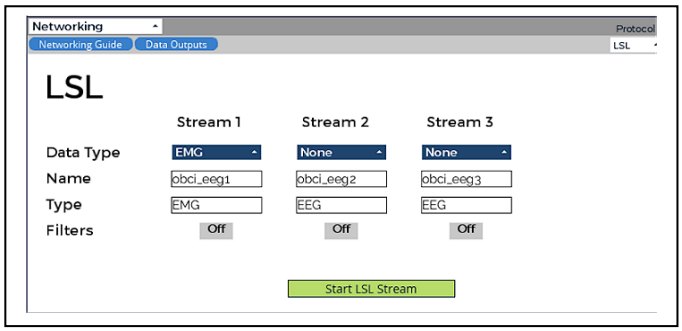
with a muscle flex. Figure 3 shows the EMG widget.

Fig. 3. EMG widget [3].

The last widget in the GUI é the Network Widget, in this case set for LSL communication as shown in Figure 4. The networking widget allows the streaming of data to other apps. Lab Streaming Layer is a system protocol for synchronizing streaming data for live analysis or recording. LSL is a good way to send OpenBCI stream to applications that can record or manipulate the data, such as Matlab. In this widget one can set the data type, stream name and type. The correct parameter definition procedure is critical for proper application work.

The OpenBCI GUI also provides a convenient button on the top right of the interface (Console Log) that opens a pop-up

window showing all the streamed data between the Cyton board and the computer. This is very useful in the case of



troubleshooting, as will be detailed later.

Fig. 4. Network widget [3].

C. Surface Electrodes

In this work a set of Kendall 530 Foam Electrodes (Conductive Adhesive Hydrogel) were acquired from Cardinal Health [6]. Whenever needed a set of 60" goldcup electrodes, acquired from Brainmaster [7], were also employed for good grounding of the experimental set up. The surface and goldcup electrodes can be seeing in Figure 5 (a) and (b), respectively. All electric connections were made using touchproof cables for noise reduction.

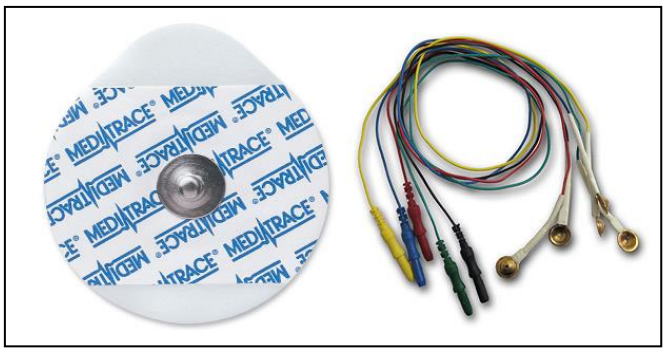


Fig. 5. (a) Surface electrodes [6], (b) goldcup electrodes [7].

D. Python scripting/programming

OpenBCI provides a Python script for communication testing such as lslStreamTest.py. The scrip allows to inspect data streaming and identify, in real time, the values of every single variable used during testing This is very useful since it provides the basis of every script used in the lab experiment that follows and can be used do identify error sources. Scripts for the experiments were also provided but they had to be rewritten due to unclear documentation and/or logic mistakes. Python 3.1.1 was used in script development and correction.

III. Methodology

An Open BCI Cyton board with 8 channels was used to data acquisition and control for all three experiments presented below. The GUI was set up as describe in the OpenBCI Platform subsection. Each group of students, upon deciding which experiment to implement, wrote a scientific essay describing the electrical engineering and biomedical concepts needed to understand each experiment. Next the students wrote a feasibility study to demonstrate that the experiments would be successful. Finally, each experiment was implemented, demonstrated, and had a final report written in the form of scientific paper. For each experiment the electrical connections between the surface electrodes and the Cyton board were slightly different and will be discussed in each subsection that follows.

All experiments require a previous installation of the OpenBCI GUI, Python 3.1.1 and the packages *pylsl* and *pyautogui* in the computer. A sample of a Python script for each experiment can be found on the OpenBCI GitHub section and a very short tutorial is provided by the manufacturer. Nevertheless, some changes had to be made otherwise the scripts would not read the data stream. These changes will be discussed in the Results and Discussion section.

A. Experiment 1 – Music Control.

The purpose of this experiment is to pause and unpause a music without pressing any keys on a laptop keyboard. To do that, the system reads the peaks in EMG signals from arm muscles produced when they're flexed and use it as a trigger for pausing the music played via Youtube through the computer. In this case a Python script will search for an EMG data stream. Once it finds the stream it will read it and detect any spikes that correspond to muscle flexing. If a flex is detected and 2 seconds have passed since the last one, it will press the space bar, which will make the music stop. The threshold for the time between flexes can be modified as needed to avoid debouncing.

Every time the arm is flexed the music will pause or start playing if paused already. By modifying the *time_thres* and *flex_thres* parameters in the Python code the time to wait between flexes and the flex strength can be adjusted.

B. Experiment 2 – Facial LED Control.

This experiment shows how to change the color of an LED depending on the facial expression. Once again, a Python script reads the peaks in EMG signals the face muscles produce when there are flexed and use it to change the color of an LED. The yellow color will indicate smiling, the red color frowning, and the blue color a neutral expression. An Arduino with a small pull up transistor circuit is needed to activate the LEDs. This experiment requires five gold cup electrodes. The first electrode is connected to the bottom AGND pin of the Cyton board, and the other four electrode cables to the top and bottom pins of Channel 1 (N1P) and Channel 2 (N2P). The five electrodes must be placed on the face. The two electrodes on top of the eyebrow go to top and bottom N2P pins on the Cyton, the two electrodes closest to the mouth go to top and bottom N1P pins, and the electrode closest to the ear goes to bottom AGND. A small provide Arduino code must be programed in the CPU. By opening the Serial Monitor in the Arduino IDE and sending the commands 'Y', 'G', and 'B' to the Arduino, the LED color should change to Red, Green, and Blue respectively.

C. *Experiment 3 – Piano Playing.*

This experiment shows how to play a virtual piano using just the EMG signals from the fingers. EMG data is read from the muscles joining the fingers to the arm and find the peaks which correspond to pressing a key on a piano, using them as a trigger to play a virtual keyboard. In this experiment, 4 channels are used: Channel 1 = right pinky, Channel 2 = right pointer, Channel 3 = left pointer, Channel 4 = left pinky. Only one ground is needed. Again, a Python script searches for an EMG data stream. Once it finds the stream it reads it and detects any spikes that correspond to finger flex. If a flex is detected and 1 second has passed since the last flex, it presses a certain key, which plays a certain note on a virtual keyboard.

IV. Results and Discussion

At first none of the experiments worked, this was because not only because of unclear instructions, but mostly because of not robust coding. The first trials results were very inconsistent. Eventually one or another time the data streaming readings would be correct, and the experiments performed as expected. In most of other moments there would be no data streaming detected whatsoever. Furthermore, the script lslStreamTest.py always worked perfectly. It is important to mention that such script was designed for EEG experiments. This was very uncomfortable because there was no indication of this happening in other published articles.

Eventually was found that the problem lied in the procedure to set up the parameters in the GUI. Depending on the sequence on parameters input the *StringList* variable content would read ["EMG", "obci_eeg1", "\u0000EMG", "8", "false”] and not ["EMG", "obci_eeg1", "EMG", "8", "false"], as expected. Please note the difference in the third value. The Python scripts then were unable to identify the data streaming as from an EMG experiment, not starting the data acquisition. Two possible solutions exist to repair this error. Solution 1: a) in the LSL widget replace EEG por EMG (typing box); b) then select EMG in the drop-down menu. The Console Log will show that the *StringList* variable content is now ["EMG", "obci_eeg1", "EEG", "8", "false"]. Solution 2: a) select EMG on the drop-down menu; b) replace EEG by EMG (typing box); c) open the console log and look for the *StringList* content. It may be: ["EMG", "obci_eeg1", "\u0000EMG", "8", "false”] (for example); d) Open the experiment script and replace EMG by \u0000EMG in the *resolve_stream* line from: streams = resolve_stream('type', 'EMG') to: streams = resolve_stream('type', '\u0000EMG') . Both solutions will make the data reading work.

Once this problem was overcome, another issue showed up with the Facial LED control experiment. The Arduino would not read the EMG signal. To surpass this problem the following procedure should be executed: a) open de Arduino IDE and start the script to ensure that a communication port is secured. b) set up the GUI interface and start streaming data (LSL inclusive). c) check the COM port and the *StringList* variable content in the console log. One may need to edit the Python script to avoid errors. d) make sure the COM ports (Arduino and OPENBCI) are different. Example: COM7 and COM5. e) Edit the Python script if needed. f) On Step 3 of the EMG Controlled LED instructions, in the "Important Note" paragraph, it is crucial to close the Arduino IDE as a whole and not only the Serial Monitor. After this, the example runs just fine.

After these corrections were implemented all three experiments worked as expected and the students were able to demonstrate how to use EMG signals to control applications.

Image 1 shows the surface electrodes position during the demonstration of the Music Control experiment. In the image it can be observed the touchproof cable connections and the OpenBCI GUI.

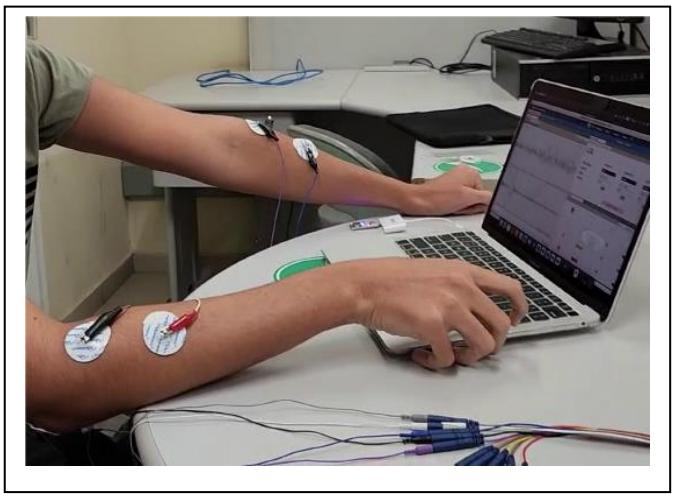


Image 1. Music Control demonstration.

Experiment 2, Facial LED Control, demonstration is a bit more complex since the experiment involves an Arduino and a protoboard where the LEDs are attached to. Image 2 shows the experiment demonstration with the Green LEDs on.

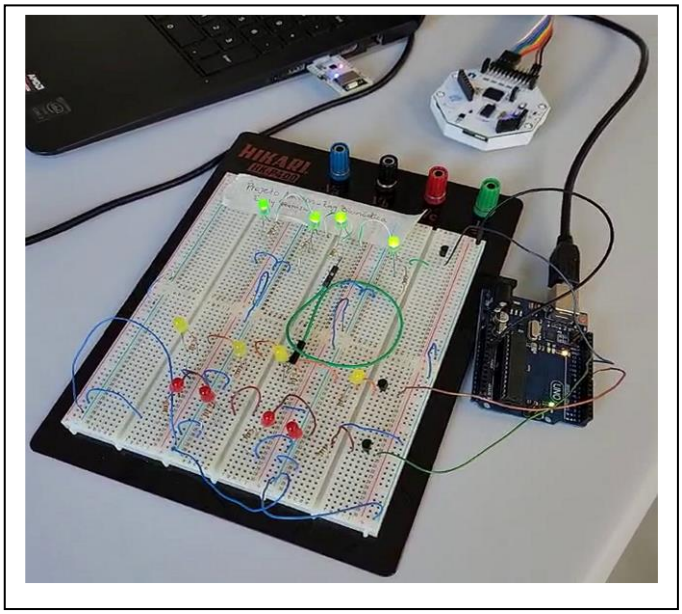


Image 2. Facial LED Control demonstration.

The image 2 shows the Cyton board, the Dongle attached to an USB computer port, the Arduino Uno board, and the protoboard with the LEDs and the pull up circuitry.

A Virtual Piano application available in the WEB [8] was employed during the demonstration of the Piano Playing experiment. Image 3 shows the experiment demonstration, where the Cyton board, the surface electrodes placements, and the virtual piano can be seeing.

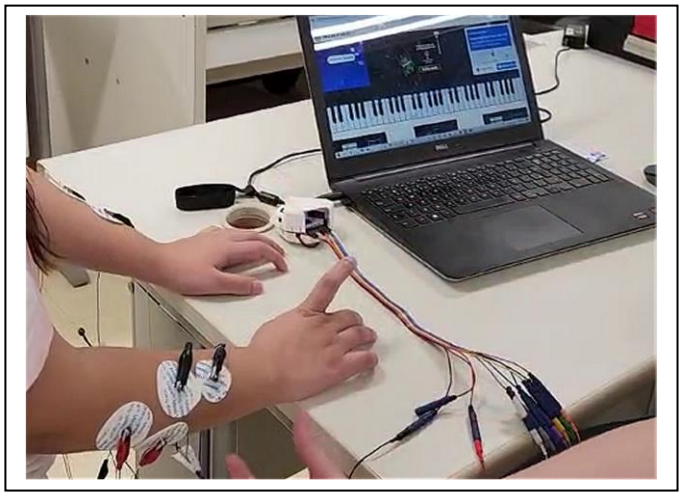


Image 3. Piano playing demonstration.

V. Conclusions and Final Remarks

An OpenBCI platform was successfully employed in an integrate laboratory of the Biomedical Engineering at PUC-Campinas to implement experiments based on EMG. After a few adjustments were made in the Python coding and some instructions rewritten The Music Control, Facial LED Control, and the Piano Playing experiments were well-off demonstrated. The student participation in the discipline promoted the knowledge appropriation on Python programing; human anatomy and neuro anatomy; histology, physiology, and biophysics applied do engineering; electrical circuits; kinesiology, and biomechanics, needed to write the final essays. The application of these results are being explored in the construction of prototype devices aimed at the physical rehabilitation of children with cerebral palsy.

Acknowledgment

The author would like to thank the Pró Reitoria de Pesquisa Pós-Graduação e Extensão for the financial support and the students for participating in the projects.

References

[1] Backyard Brains, https://backyardbrains.com, 308 1/2 S. State, Suite 35, Ann Arbor, MI 48104

[2] Myoware, https://myoware.com

[3] Open BCI, https://openbci.com

[4] Joseph D Bronzino and Donald R Peterson, Biomedical Engineering Fundamentals, CRC Press; 2nd ed., 2014

[5] V.L.S.N Button, “Principles of Measurement and Transduction of Biomedical Variables”, ISBN: 978-0-12-800774-7, Academic Press (2015)

[6] https://www.cardinalhealth.com/en/product-solutions/medical/patient-monitoring/electrocardiography/monitoring-ecg-electrodes.html

[7] https://brainmaster.com/product/60-gold-cup-electrodes/

[8] https://virtualpiano.net/