CALIFORNIA STATE UNIVERSITY, NORTHRIDGE

Machine Learning using Brain Computer Interface (BCI) System

A thesis submitted in partial fulfillment of the requirements
For the degree of Master of Science in Mechanical Engineering

By
Kevin Motoyoshi Matsuno

December 2020

Copyright by Kevin Matsuno 2020

The thesis of Kevin Matsuno is approved:

Professor Aram G. Khachatourians	Date
Dr. Hamid Johari	Date
Dr. Vidya Nandikolla, Chair	Date

California State University, Northridge

Table of Contents

Copyright ii
Table of Contents iv
List of Figures vi
List of Tables xi
Abstract xiii
Chapter 1 – Introduction 1
Chapter 2 – Literature Review 3
2.1 BCI Controllers in the Developing Field 3
2.1.1 Synchronous BCI Controllers 3
2.1.2 Asynchronous BCI Controllers 4
2.2 The Six-Class BCI Controller Overview 5
2.2.1 Justification for a six-state BCI Mobile Arm Model 5
2.2.2 Schematic Representation of a six-state BCI Mobile Arm Model 6
Chapter 3 – Understanding the Human Brain 8
Chapter 4 – System Design and Modeling 12
4.1 High level block diagram for semi-autonomous BCI mobile robotic arm 12
4.2. BCI Controller Subsystem, Hardware 13
4.2.1 EEG headset electrode placement 14
Chapter 5 – BCI Subsystem Controller Design 17
5.1 Offline vs. Online Block Diagram 17
5.2 Software Integration 19
5.2.1 OpenViBE: Offline Training – Signal Acquisition 20
5.2.2 EEGLAB (MATLAB Plug-in): Offline Training –Signal Preprocessing 24
5.2.3 BCILAB (MATLAB Plug-in): Offline Training – Classification 36
5.2.4 BCILAB (MATLAB Plug-in): Online Testing – Streaming 52
5.2.5 Lab streaming layer (LSL): Online Testing – Stream Acquisition System 56
5.2.6 A Comparison to Past BCI Architecture 59
Chapter 6 – Testing and Simulation 61
6.1 Testing Objectives 61
6.1.1 Objective 1 – Active Electrode Locations Criteria 61
6.1.2 Objective 2 – BCI Controller Performance Criteria 62
6.1.3 Objective 3 – BCI Controller Implementation Criteria 62

6.2 Test setup 63
6.2.1 Test Subjects 63
6.2.2 Test Environment 63
6.3 Testing procedures 64
6.3.1 Objective 1 – Active Electrode Locations, Test Procedure 64
6.3.2 Objective 2 – BCI Controller Performance, Test Procedure 67
6.3.3 Objective 3 – BCI Controller Implementation, Test Procedure 70
Chapter 7 – Results and Analysis 72
7.1 Objective 1: Active Electrode Locations, Results 72
7.1.1 Single Electrode ERPs, Hand Motor Imagery 72
7.1.2 Single Electrode ERPs, Foot Motor Imagery 79
7.1.3 Single Electrode ERPs, Jaw vs Tongue 84
7.1.4 Overlay Electrode ERPs, Hand Motor Imagery 85
7.1.5 Overlay Electrode ERPs, Foot Motor Imagery 88
7.2 Objective 2: BCI Controller Performance, Results 90
7.2.1 2-Class BCI Controller Performance Results 90
7.2.2 3-Class BCI Controller Performance Results 94
7.2.3 5-Class BCI Controller Performance Results 97
7.2.4 6-Class BCI Controller Performance Results 98
7.3 Objective 3: BCI Controller Implementation, Results 99
7.3.1 BCI Parameters and Interface 99
7.3.2 Control of Robotic Base 101
7.3.3 Control of Robotic Arm, Guiding Cursor and Initiating Grab/Release Action 103
Chapter 8 - Conclusion and Discussion 105
8.1 Previous BCI Research at CSUN 106
8.2 Major Contributions Added to CSUN BCI Research 108
8.3 Future Work and Recommendations 109
8.4 Lessons Learned 111
References 115
APPENDIX A 121
APPENDIX B 122

List of Figures

Figure 1: A six state BCI Mobile Arm model showing the robot commands in relation to the user's brain signals. With the implementation of the jaw signal switching between controlling the robot base and arm, multiple commands can be assigned to an asynchronous motor imagery brain signal. 6

Figure 2: (Left) Three main parts of the brain [16]. (Right) Four main lobes of the forebrain [17]. 8

Figure 3: (Left) A neuron and its components [33]. (Right) White and Gray Matter, the composition of hte brain [34]. 10

Figure 4: Four types of EEG signals that reflect mental state [45]. 11

Figure 5: A high level block diagram of the BCI mobile robotic arm. The diagram can be broken into two sections, the BCI controller and the autonomous robot. 13

Figure 6: Signal Acquisition Hardware. (Left) Unaltered Emotiv Epoc+14 headset. (Right) Modified EasyCap headset with electrodes connect to Emotiv Epoch decoder on mannequin head. 14

Figure 7: (Left) Emotiv Epoc headset connectivity window; note, electrode placement is based off of original headset orientation, not modified.(Right)Electrode schematic used development of the BCI controller. 15

Figure 8: A comparison between the offline and online block diagram for a BCI controller. Notice that only the online testing is connected with the rest of the high-level block diagram. The software associated with each block is listed. 18

Figure 9: (Left) OpenViBE Acquisition Server main GUI window where the user can select the appropriate EEG headset. (Right) More detail about the provided can be seen in the driver properties and preferences. 20

Figure 10: (Left) Main GUI for OpenViBE Designer [51]; (Middle) An example of a connected box layout [51]; (Right) The acquisition client box show five outputs and two textboxes. 21

Figure 11: An OpenViBE example of five scenarios to record motor imagery signals, train a CSP filter, train a classifier, conduct and online test, and calculate the performance of the classifier. 22

Figure 12: (Left) Box layout for the stimulation script in the OpenViBE acquisition scenario. (Right) The window to modify the stimulation script. 23

Figure 13: (Left) The window running the stimulation script; notice the green cross and the red arrows pointing to either the left or right. (Right) A timeline used for the development of the BCI development controller. 24

Figure 14: A collection of sample graphs and data generated by EEGLAB 25

Figure 15: The windows and representation of the data at certain steps of the preprocessing procedure. Note, the numbers correspond to the steps in procedure table. 28

Figure 16: An illustration showing how one electrode and receive propagated signals from multiple brain sources and creating a mixed signal [60]..... 29
Figure 17: A graphical representation showing the difference between PCA and ICA analysis, which is used to identify and remove artifacts. The graph on the left shows the PCA method while the graph on the right shows ICA [61]. 30
Figure 18: A visual representation of ICA using signals and histogram bins [61]..... 31
Figure 19: Plotted EEG data going through the ICA training process [61]..... 31
Figure 20: The additional rules screen in the general infomax ICA run [61]..... 32
Figure 21:(Left) Navigation to find the ICA function in EEGLAB. (Right) The results are virtual channels that are ordered from most to least variance..... 33
Figure 22: A window showing the impact of removing a virtual electrode channel. The blue colored signals represent the original data while the red colored signals show the altered data after channel rejection. Note that the peaks in the original data, result of blinking, were removed. 34
Figure 23: The channel plots used when analyzing a subject's motor imagery signals when responding to a stimulus. The numerical labels correspond to the functions selected in the 'plot' drop-down. 35
Figure 24: A collection of sample menus and data generated by BCILAB [63]..... 36
Figure 25: (Top) Main BCILAB GUI with balloon callouts highlighting the offline training procedure. (Bottom) A table describing the step-by-step offline training procedure..... 37
Figure 26: Navigating the BCILAB GUI for offline testing. Numbers correspond to the steps outlined in the offline testing procedure. Note, step 2 has additional customization to further define the filter that can be accessed selecting the 'Review/edit approach…' function..... 37
Figure 27: (Left) An example of a visualization spatial filter, specifically a Common Spatial Pattern, used to identify sources found in the motor cortex region of the brain. (Right) If the linear weight matrix is inversed and multiplied by the detected sources in the brain, the electrodes affected by the source can be detected [64]..... 39
Figure 28: (Left) Raw samples of two electrodes C3 and C4. (Right) After spatial filtering is applied, the space is squished to form distinct variances. The red channel samples are affected horizontally and the blue channel is affect vertically [65]. 41
Figure 29: A graphical approach to applying a CSP filter to two electrode distributions of samples..... 43
Figure 30: A visual representation of Linear Discriminate Analysis (LDA) where the a plane is created to separate the two classes [66]..... 46
Figure 31: The parameters chosen on BCILAB to apply a CSP Filter and LDA machine learning method to train a model. 47
Figure 32: A comparison show a comparison between a support vector and relevance vector classifier using synthetic data. The decision boundary is represented as a dashed line and the encircled points show the vectors used to train the classifier. Note the error percentages as well [67]. 49

Figure 33: The parameters chosen on BCILAB to apply a CSP Filter and RVM machine learning method to train a model. 49
Figure 34: A comparison the different multi-class methods: one-vs-res (left)t and one-vs-one (right) [68]. 50
Figure 35: (Top) Main BCILAB GUI with balloon callouts highlighting the online training procedure. (Bottom) A Table describing the step-by-step online testing procedure. 53
Figure 36: (Left) Navigating the BCILAB for online testing. Numbers correspond to the steps outlined in the online testing procedure. 53
Figure 37: Sub-windows for the two selected methods for BCILAB to read incoming streamed EEG signals are shown. The red boxed highlight shows the incoming signal coming from Lab Streaming Layer. The blue boxed highlight shows the option for BCILAB to loop a dataset as a simulated EEG signal stream. 54
Figure 38: Sub-windows for the two selected methods for BCILAB to write outgoing command streams are shown. The red boxed highlight shows a streaming command matrix from BCILAB showing on python. The blue boxed highlight shows an option for BCILAB to generate a bar graph showing class '1' or '2'. 55
Figure 39: (Left) Core components that make up Lab Streaming Layer (LSL). Note the flexibility to communicate to multiple programs and languages. (Right) A high-level network view of different programs (i.e. stimulus presentation) and devices that use liblsl to send data out or pull data in from LSL. [69] 56
Figure 40: A flow diagram outlining the process of building Lab Streaming Layer (LSL). 57
Figure 41: A visual breakdown of the amount of testing each subject underwent. Three subjects were tested. Each subject had five sessions; each session is composed of four tests recording specific motor imagery signals. Each test contains 15 trials of both the left and right hand or foot. 66
Figure 42: (Left) Navigation to find the simple ERP study. A pop-up window appears asking for the number of conditions and subjects. (Right) A final pop-up window where the pre-processed recordings are uploaded. 66
Figure 43: A visual breakdown of the different class models and machine learning methods. 67
Figure 44: EEGLAB Navigation to rename marker labels and append recordings together. 69
Figure 45: Subject S01's most active left hemisphere electrodes from physical and imaginary hand motor imagery testing. 74
Figure 46: Subject S01's most active right hemisphere electrodes from physical and imaginary hand motor imagery testing. 75
Figure 47: Subject S02's most active left hemisphere electrodes from physical and imaginary hand motor imagery testing. 76
Figure 48: Subject S02's most active right hemisphere electrodes from physical and imaginary hand motor imagery testing. 77

Figure 49: Subject S03's most active left hemisphere electrodes from physical and imaginary hand motor imagery testing. 78
Figure 50: Subject S03's most active right hemisphere electrodes from physical and imaginary hand motor imagery testing. 78
Figure 51: Subject S01's most active left hemisphere electrodes from physical and imaginary foot motor imagery testing. 80
Figure 52: Subject S01's most active right hemisphere electrodes from physical and imaginary foot motor imagery testing. 80
Figure 53: Subject S02's most active left hemisphere electrodes from physical and imaginary foot motor imagery testing. 81
Figure 54: Subject S02's most active right hemisphere electrodes from physical and imaginary foot motor imagery testing. 82
Figure 55: Subject S03's most active left hemisphere electrodes from physical and imaginary foot motor imagery testing. 83
Figure 56: Subject S03's most active right hemisphere electrodes from physical and imaginary foot motor imagery testing. 83
Figure 57: Subject S03's most active left and right hemisphere electrodes from jaw and tongue press to the roof of the mouth. 84
Figure 58: Subject S01's imaginary left and right hand ERP plots with all 14 channels combined. Topographical maps of each time point connected to each maximum and minimum microvoltage values in the previous section are shown. 86
Figure 59: Subject S02's imaginary left and right hand ERP plots with all 14 channels combined. Topographical maps of each time point connected to each maximum and minimum microvoltage values in the previous section are shown. 86
Figure 60: Subject S03's imaginary left and right hand ERP plots with all 14 channels combined. Topographical maps of each time point connected to each maximum and minimum microvoltage values in the previous section are shown. 87
Figure 61: Subject S01 imaginary left and right foot ERP plots with all 14 channels combined. Topographical maps of each time point connected to each maximum and minimum microvoltage values in the previous section are shown. 88
Figure 62: Subject S02 imaginary left and right foot ERP plots with all 14 channels combined. Topographical maps of each time point connected to each maximum and minimum microvoltage values in the previous section are shown. 89
Figure 63: Subject S03 imaginary left and right foot ERP plots with all 14 channels combined. Topographical maps of each time point connected to each maximum and minimum microvoltage values in the previous section are shown. 89
Figure 64: A screenshot of the ROS program showing a successful link to the BCI command stream. 102
Figure 65: (Left) BCILAB plug-in outputting 1 (RF brainwave input). (Right) The robotic base initiating the move forward command. 102

Figure 66: (Left) BCILAB plug-in outputting 3 (RH brainwave input). (Right) The robotic arm and camera screen with the cursor moving right. The cursor position points are recorded. 104
Figure 67: (Left) BCILAB plug-in outputting 5 (JAW brainwave input). (Right) The robotic arm and camera screen with the cursor initiating a grab/release point. 104

List of Tables

Table 1: The procedure used in preprocessing the raw EEG recordings. Note, the values and location of these functions are in the main EEGLAB GUI. 26

Table 2: The list of numerical code markers used in OpenViBE raw signal recordings.. 27

Table 3: (Left) A list of loss/performance metrics used to evaluate the accuracy of a model after the training function. (Right) The mathematic representation for two possible metrics, Mean-Square Error, and Mis-Classification Rate. 52

Table 4: Subject S01, 2-Class (LH vs RH) classifier results using LDA. In bold is the best test recording out of all the sessions. 90

Table 5: Subject S01, 2-Class (LH vs RH) classifier results using RVM. In bold is the best test recording out of all the sessions. 90

Table 6: Subject S02, 2-Class (LH vs RH) classifier results using LDA. In bold is the best test recording out of all the sessions. 91

Table 7: Subject S02, 2-Class (LH vs RH) classifier results using RVM. In bold is the best test recording out of all the sessions. 91

Table 8: Subject S03, 2-Class (LH vs RH) classifier results using LDA. In bold is the best test recording out of all the sessions. 91

Table 9: Subject S03, 2-Class (LH vs RH) classifier results using RVM. In bold is the best test recording out of all the sessions. 91

Table 10: Subject S01, 2-Class (LF vs RF) classifier results using LDA. In bold is the best test recording out of all the sessions. 92

Table 11: Subject S01, 2-Class (LF vs RF) classifier results using RVM. In bold is the best test recording out of all the sessions. 92

Table 12: Subject S02, 2-Class (LF vs RF) classifier results using LDA. In bold is the best test recording out of all the sessions. 92

Table 13: Subject S02, 2-Class (LF vs RF) classifier results using RVM. In bold is the best test recording out of all the sessions. 92

Table 14: Subject S03, 2-Class (LF vs RF) classifier results using LDA. In bold is the best test recording out of all the sessions. 93

Table 15: Subject S03, 2-Class (LF vs RF) classifier results using RVM. In bold is the best test recording out of all the sessions. 93

Table 16: Subject S03, 2-Class (JAW vs ROOF) classifier results using LDA. In bold is the best test recording out of all the sessions. 93

Table 17: Subject S03, 2-Class (JAW vs ROOF) classifier results using RVM. In bold is the best test recording out of all the sessions. 94

Table 18: Subject S01, 3-Class: Hands vs Rest and Feet vs Rest using both LDA and RVM. In bold are the best test recording out of all the sessions. 95

Table 19: Subject S02, 3-Class: Hands vs Rest and Feet vs Rest using both LDA and RVM. In bold are the best test recording out of all the sessions. 95

Table 20: Subject S03, 3-Class: Hands vs Rest and Feet vs Rest using both LDA and RVM. In bold are the best test recording out of all the sessions. 96
Table 21: Subject S03, 3-Class: Jaw vs Roof vs Rest using both LDA and RVM. In bold are the best test recording out of all the sessions. 96
Table 22: Subject S01, 5-Class: Hands vs Feet vs Rest using both LDA and RVM. Note, the best test recordings were used. 97
Table 23: Subject S02, 5-Class: Hands vs Feet vs Rest using both LDA and RVM. Note, the best test recordings were used. 97
Table 24: Subject S03, 5-Class: Hands vs Feet vs Rest using both LDA and RVM. Note, the best test recordings were used. 98
Table 25: Subject S03, 6-Class: Hands vs Feet vs Jaw vs Rest using both LDA and RVM 98
Table 26: A summary of error rate percentages of both LDA and RVM for each subject 99
Table 27: 6-class controller reference table showing the input action and associated classifier output. 100

Abstract

Machine Learning using Brain Computer Interface (BCI) System

By

Kevin Matsuno

Master of Science in Mechanical Engineering

Engineers in the field of control systems have been recently drawn to the development of creating a hands-free and speech-free controller interface over computers and robotic devices. The primary individuals who would use this type of controller suffer from progressive nervous system diseases or other forms of paralysis that have severely restricted any movement of the limbs. Despite their physical limitations, these same individuals have an uncompromised brain full of cognitive and sensory functions. As a result, one solution to restore mobility and autonomy to the paralyzed is to create a controller that utilizes their brain signals. A brain computer interface (BCI) applies brain signals as input to a controller that will then drive a robot arm or transporter. By linking a specific mental task (i.e. imagine squeezing the right hand) to a command a robot (i.e. make a right turn), users have the ability to navigate an electrically powered wheel chair or robot-aid for themselves. While there is potential to create a wide range of controller commands, brainwaves come with their own set of challenges. These signals are non-stationary and non-linear; meaning, brainwaves constantly vary and are extremely difficult to model. In addition, noise from other involuntary functions (i.e. blinking and facial muscle activation)

may bury the unique signals associated to the mental task. To overcome these obstacles, control system engineers have implemented a signal preprocessing step and machine learning approach to these controllers. The combination of selecting the right preprocessor, machine learning algorithm, and training the user to conduct clear mental tasks creates an accurate and responsive BCI controller.

The main goal of this project is to design a six-class hybrid BCI controller for a semi-autonomous mobile robotic arm. The controller is designed to operate the robotic base and arm separately. To do this, a set of EEG motor imagery hand and feet signals serves two primary functions: they navigate the robot base in the environment and move a cursor on the robot's camera screen to highlight what object to grab. In addition, a jaw clench, which is an electromyogram (EMG) signal, is used to switch between commanding the base and the arm. Designing a controller with this capability for multiple users requires a compilation of hardware to record/stream brainwaves and software to preprocess and train a machine learning algorithm. A modified 14-channel commercial grade non-invasive electroencephalogram (EEG) headset from Emotiv Epoch was used to output the brain waves of three healthy males (ages 22 – 27) to the computer. Each subject recorded five sessions, each with four tests, of their responses to OpenViBE's stimulus presentation program. The recordings were then uploaded to EEGLAB, an open source MATLAB plug-in, where the signals were preprocessed with filters and the implementation of Independent Component Analysis (ICA). Additionally, EEGLAB was used to plot Event Related Potential (ERP) plots and topographical maps to observe each subject's brain activity. After reviewing all the plots, each subject shared the same behavior in electrodes C1, C3, C5, C2, C4, and C6. For comparison, two machine learning algorithms, linear discriminant analysis (LDA) and relevance vector machine (RVM) were chosen to process and classify the subjects' recordings. The performance for each classifier was recorded for a 2-class, 3-class, 5-class, and 6-class controller. RVM out performed LDA with multi-class controllers. For a 5-class controller, the error rate percentages were: 45% for subject S01, 30.8% for subject S02, and 29.2% for subject S03. With the proper electrodes and machine learning algorithms identified, the official 6-class controller was created with a common spatial pattern (CSP) filter and RVM classifier. It was observed that the accuracy of the controller decreased as the number of classes increased. The 6-class BCI controller was

integrated into a virtual model of the semi-autonomous robotic arm where it successfully demonstrated the ability to separately move the base, move the cursor on the robot's camera screen, and activate the action to pick up/drop off an object.

Chapter 1 – Introduction

Brain computer interface (BCI) systems are a trending topic in the system controls field because they are aimed at designing a hands and speech free approach to controlling a robotic device. While some occupations operate in extraordinary environments where this system would be beneficial, BCI research and development is primarily interested in restoring movement and capability to a wide range of the disabled. From quadriplegics to amyotrophic lateral sclerosis (ALS) and other forms of paralysis, the disabled persons' brain activity remains functional despite their limitations in speech and/or mobility. As a result, utilizing brainwave signals as a controller input for a robotic arm or transporter is desirable for the handicapped because they will be able to regain a sense of autonomy.

BCI systems are broken up into two subsystems: the controller and mechanical robotic device. While both subsystems are critical, this study will focus on developing a refined controller and assume the robotic device is already fabricated. Brain computer interface controllers typically use electrical potential waves that are generated from the complex network of brain synapses exchanging information and implementing decisions. This electrical activity can be observed on the human scalp and is measured via electrodes, also known as electroencephalography (EEG). EEG signals have been used extensively in the medical field to observe different parts of the brain as well as diagnosis a patient's sleeping disorders or impairment from a stroke. From the system controls engineering perspective, these signals can be harnessed to command a robotic device. Fortunately, commercially available hardware has made it possible to acquire EEG signals non-invasively and with an accurate time resolution.

While the myriad of unique brainwaves from different scalp locations show huge potential for a wide range of controller commands, EEG signals have some key challenges. First, brainwaves are non-stationary; to clarify, there is no long-term mean because the signal has a high variable variance. Second, every human brain is unique and processes information differently. For example, if several human beings were instructed to imagine squeezing their left hand, their EEG signals would differ from one another. And lastly, the desired brain signals are often buried with other signals generated by the user's five senses

(i.e. sight and hearing) and involuntary signals (i.e. signals commanding breathing and cardiac rhythms). These issues prevent the conventional controls approach of creating a universal controller because a human brain is too complex to be mathematically modeled.

As a result, to accommodate for all these challenges, engineers have designed BCI controllers to undergo machine learning and signal processing. Machine learning is a form of artificial intelligence where the BCI controller learns how a specific action and brain signals are associated from a set of examples. This specific action could be physical, like a muscle activation, or mental, like imagining hand movement. The controller is said to be accurate if the same action is presented and it correctly recognizes the action. In addition, signal processing also affects the controller's accuracy by applying filters to better improve the quality of the BCI signals. Selecting the right signal processing method and machine learning algorithm is critical to designing a reliable and responsive BCI controller.

The focus of this study is to show the development of a hybrid BCI controller that operates a semi-autonomous mobile robotic arm. This six-class controller recognizes a subject's imagined left/right hands, feet, and clenched jaw and sends the associated command to the robot. A compilation of hardware and software was used to design a controller for multiple users with this capability. In terms of hardware, a modified commercial grade EEG headset from Emotiv was used to record three healthy male subject's brainwaves. These recordings were preprocessed and then analyzed to look at each subject's brain activity through EEGLAB, an open-source software. Afterwards, a CSP filter and machine learning algorithms LDA and RVM were implemented through a software called BCILAB to investigate which approach has a better controller performance. Finally, the down selected controller was then implemented to the mobile robotic arm and a series of demonstrations showing how the subject's brain signals were used to move the robot's base and arm independently is presented. The contents of this study go more into depth and are broken up in the following sections. First, a review of BCI technology and neurophysics are discussed. Next, the overall BCI controller architecture is discussed. Afterwards, details of the test objectives and procedures are presented. Finally, the results and analysis are presented for the developmental work of the hybrid BCI semi-autonomous mobile robotic arm controller.

Chapter 2 – Literature Review

In this chapter, a literature review of BCI systems is discussed. This chapter is broken up into two main sections. The first section presents the different types of BCI systems seen in the research field. Specifically, the BCI controller design, EEG signals used as inputs, and accuracies for each system are discussed in detail. And finally, the last section shows the justification and overview of this study's BCI controller.

2.1 BCI Controllers in the Developing Field

The structure and behavior of a control model depends on what unique brain signals are being utilized. The types of brain waves seen in the field of neuropsychology and neuroscience are based on a human being's mental state (i.e. sleeping, relaxing, or awake). However, to utilize brainwaves as BCI controller inputs, these brainwaves need to be induced. These brainwaves are known as asynchronous and synchronous EEG signals and they both have tradeoffs in response, accuracy, and level of user training and concentration.

2.1.1 Synchronous BCI Controllers

Controllers that typically utilize synchronous EEG signals are keyboard spellers, virtual reality video games, and other applications that involve a screen in front of the user [1 – 6]. In the case of a speller controller, the work presented by Yin et al allowed users thirty-six possible letters and numbers to choose from. By creating a row and column grid of unique flashing frequencies, the controller can identify a particular letter the user chose by matching the combined frequencies from the grid to the electrode placed on the user's occipital lobe. The unique thirty-six possible choices, or stages, show the advantage to a synchronous BCI model in that it comfortably has multiple commands inputs for the user. Note, stages refer to the number of unique brain signals the controller can recognize and use.

However, there are multiple reasons why controllers using this approach are not being used on mobile robots. Driving a robot (i.e. a smart electrically powered wheel chair) requires the user's undivided attention of the surroundings. In a synchronous controller,

the user will be juggling his/her eyesight on the destination, surroundings, and on the screen to dictate how the robot should move next. Iturrate et al, demonstrated a smart wheel chair synchronous controller where the screen reimaged the surroundings and embedded flashing nodes to show the next destination [7]. While reimaging from the camera onto the screen attempts to make the user more aware of the surroundings, he/she is still unable to visually look around a quickly changing environment and react in a timely matter. As a result, asynchronous control models are the preferred approach because the user's eyesight is naturally intended for observation only.

2.1.2 Asynchronous BCI Controllers

Asynchronous BCI controllers rely on more complicated brain signals rather than mimicking a combined frequency on a screen. Compared to synchronous control models, these controllers are more responsive but are limited on the number of signals the controller can recognize and use. One solution to increase the number of brainwaves has been the incorporation of non-brain based signals and creating what is called a hybrid controller. Researchers advancing this field have created asynchronous BCI controllers for robotic arms and mobile robots [8-12]. Minati et al designed a hybrid controller of a vision-guided robot arm using electrooculogram, electroencephalogram, electromyogram biosignals, and head movement [13]. Four control models were created with unique signals to perform the following actions: rotate the base of the robot arm left/right, find/grasp an object, drop the object, and standby. A controller accuracy of 97 +/- 6% was accomplished with most models, but it should be noted that non-brain based signals were primarily used, not EEG asynchronous models.

Previous work at California State University, Northridge (CSUN) focused on creating BCI controllers while utilizing asynchronous for a robotic gripper control and smart wheel chair respectively [14] [15]. The robotic gripper controller focused on five commands: grab object, release object, move object to right/left, and neutral, while the smart wheelchair controller had four commands: start/stop, turn left, turn right, and rest. In both controller designs, the user's EEG brain signals were recorded, and a machine

learning algorithm was used. The next design iteration would be to incorporate both controller designs and create a mobile robotic arm that utilizes asynchronous brainwaves.

2.2 The Six-Class BCI Controller Overview

2.2.1 Justification for a six-state BCI Mobile Arm Model

A BCI mobile robotic arm is a solution to restore the capabilities of the physically impaired. Whether through natural causes like age, or paralysis from neurological disorders and spinal injuries, the disabled live in home and working environments that require the transportation of payloads. While hiring healthcare workers has been the traditional solution, it is a costly investment, especially depending on the level of care required (i.e. 24-hour care). For quadriplegics with severe physical and occupational limitations, 24-hour supervision and care is not a sustaining option. As a result, a BCI mobile robotic arm with semi-autonomous functions built-in can allow users to retrieve items from different rooms in a home safely through the power of their mind. Restoring this ability for human beings who are bedridden would dramatically improve their quality of life and gain some level of autonomy.

In addition to improving the disabled's domestic life, a BCI mobile robotic arm can work in environments too hazardous for humans. An example of this is if a payload is located at a site full of radiation and the operator's hands are pre-occupied. In this scenario, the user would be able to navigate the mobile robotic arm can while keeping his undivided attention on the task at hand. Whether the environment is an airtight controlled laboratory, or a sound-sensitive assembly area, a BCI mobile robotic arm will allow users to operate while behind safe walls.

2.2.2 Schematic Representation of a six-state BCI Mobile Arm Model

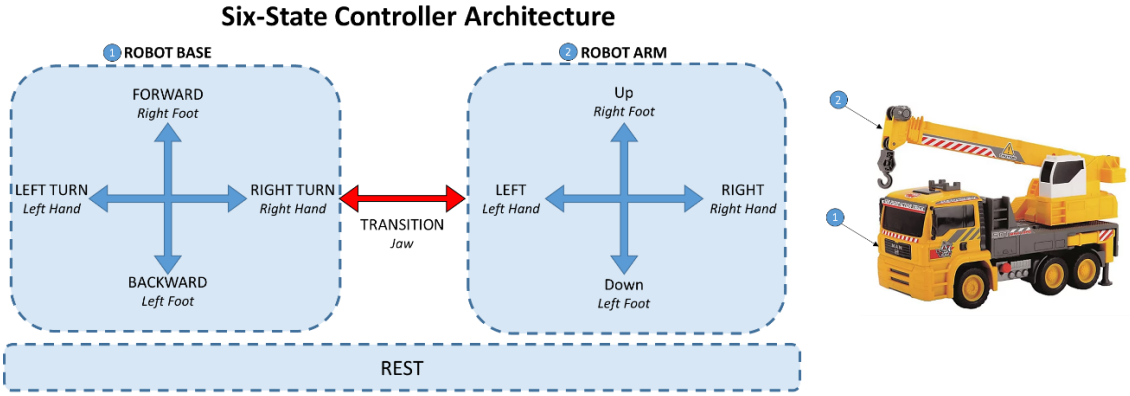


Figure 1: A six state BCI Mobile Arm model showing the robot commands in relation to the user's brain signals. With the implementation of the jaw signal switching between controlling the robot base and arm, multiple commands can be assigned to an asynchronous motor imagery brain signal.

A six-stage BCI control model is proposed to move and operate a mobile robotic arm (see Figure 1); the six stages refer to the number of unique tasks being used: left/right foot, left/right hand, jaw clench, and rest. This controller operates similarly to a truck crane where the user will first move the robot base to a desired location and then switch to operating the robot arm to pick up an object. To drop off the held object in another location, the end user then shifts back to moving the base. Once at the desired location, the arm side of the controller is activated one final time where the user moves the cursor to tell where to place the item.

Because these two operations are isolated, the controller can assign multiple actions to the same set of tasks without confusion. When moving the base, the user's asynchronous signals of the right/left foot and hand motor imagery commands the robot to advance forward, backward, and perform a right or left turns. Sensors on the base can detect obstacles or walls and avoid collisions while performing the user's movement commands. To transition to using the robot arm, the controller detects the user's electromyogram signal of clenching his/her jaw. Afterwards, the same asynchronous signals guide a cursor on a camera screen up, down, right, or left to a desired object. The jaw clench would confirm the object selected by the cursor and proceed with the picking up action before switching

to the base side controller. Built-in software will calculate the distance from the object and then activate the necessary motors in the right order to pick up the object. Likewise, the cursor would be utilized again to drop off the held item at a desired location. Having a smart robot arm frees the user from the complexity of controlling every mechanical joint like the fingers and wrist. Whether moving the robot or operating the arm, a rest command keeps the robot idle until a trained brain signal is detected.

This project's outcome is to develop a BCI controller that will operate the mobile robotic arm. In addition, this will help advance the field of biomedical engineering and robotic assisted technology for the physically disabled and operators working in hazardous environments. The next chapter will discuss the details of the human brain, and later chapters will discuss the overall system design, hardware and results.

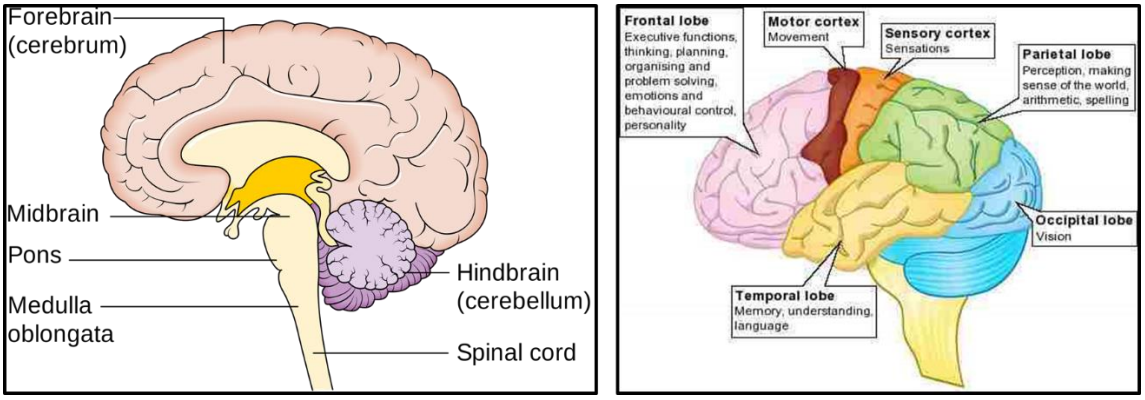


Figure 2: (Left) Three main parts of the brain [16]. (Right) Four main lobes of the forebrain [17].

The human brain is primarily composed of three main parts. Starting from the spinal cord, these are the hindbrain, midbrain, and forebrain [18] [19]. The hindbrain is mainly composed of the cerebellum, pons, medulla, and reticular motion. This region of the brain is responsible for functions that are fundamental for human being survival like respiratory rhythm, motor activity, and sleep [20]. Whether asleep or awake, a lot of the involuntary functions, such as breathing and heartrate, are monitored specifically by the medulla. Finally, the pons is the largest part of the brain stem that acts like a vital bridge between the cerebellum and cerebrum [21]. While important for living, this lower level of the brain would not be optimal as a controller input. Moving on, the midbrain is located above the hindbrain and below the cerebral cortex. This part of the brain is composed of the colliculi, tegmentum, and cerebral peduncles; in addition, two of the twelve cranial nerves that control eye and eyelid movement are a part of the midbrain [22]. Like the hindbrain, the midbrain is responsible for the involuntary relay of information between the visual and auditory systems; in addition, it is also responsible for suppressing pain and coordinating movements by communicating with the central nervous system. The final and largest region of the brain is the forebrain; it is formed of two divisions, the diencephalon and telencephalon [23]. Hidden from view, the diencephalon is located between the two cerebral hemispheres and is made up of the thalamus, subthalamus, hypothalamus, and epithalamus [24]. This portion of the brain connects the endocrine system to the nervous system. In addition, it also works with the limbic system to create emotions and memories.

Finally, the telencephalon, or cerebrum, makes up two-thirds of the brain's mass and encompasses most of the brain; this is the most iconic part of the brain because of its 'wrinkles' or grooves. This portion of the brain consists of two cerebral hemispheres that is connected by the corpus callosum [25]. The cerebrum is responsible for all high functions of intelligence, personality, thinking, interpretation, and motor functions. The outer surface of the cerebrum is the cerebral cortex and is approximately 1.5 to 5 millimeters thick; the surface is split into four lobes named after the cranial bones: frontal, parietal, occipital, and temporal [26]. The frontal lobes are where all the complex planning, language comprehension and organizing occur. While this is not a preferred area for BCI controller input, it is still necessary for the user to plan robot commands appropriately. The same can be said with the temporal lobe which contributes to a human being's ability to recall memories, create emotion, process hearing and language [27]. Both of these lobes are where a human being's cognitive functions primarily occur. The one exception is the motor cortex which is part of the frontal lobe. In contrast, the parietal and occipital lobes are where human being's movement and sensory functions like hearing, touch, and vision are located [28]. As a result, it is these lobes, and the motor cortex, that are attractive electrode locations for BCI signal input.

While the general regions of the brain and their functions are understood, the exact location of where a thought or action occurs is a developing field. Brain mapping is an emerging field where scientists and engineers are trying to plot the vast network of brain synapses. Many studies have tried to map a subject's motor imagery or simple command [29-32]. However, the biomedical field still needs to develop instruments with better resolution before these exact locations can be applied to BCI controllers. For now, the BCI research field is left to reading brain signals.

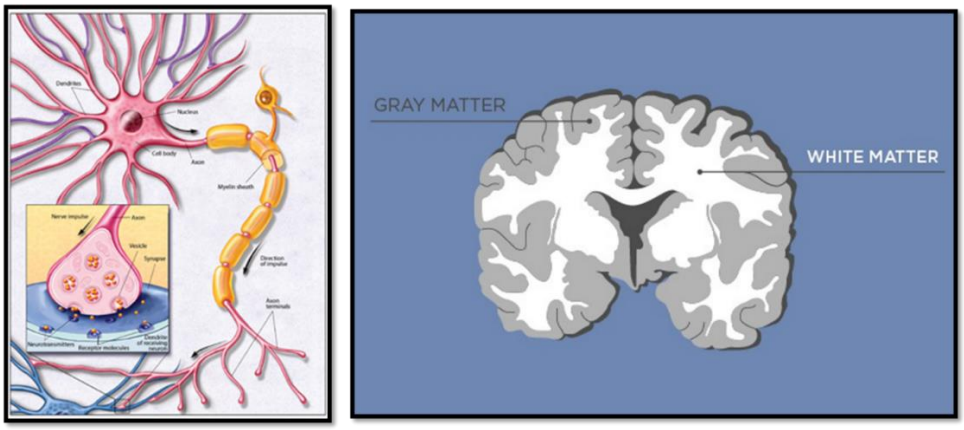


Figure 3: (Left) A neuron and its components [33]. (Right) White and Gray Matter, the composition of the brain [34].

Before understanding what kinds of electrical signals can be observed, it is important to understand how these signals are created. The human brain is made up of approximately 100 billion nerve cells called neurons [35]. Most neurons have a cell body, an axon, and dendrites. These dendrites connect to the axons of other neurons and are covered in synapses to communicate to one another. When neurons want to send or receive messages, they conduct an electrical impulse by using a chemical reaction called action potential [36]; at a cellular level, this is when an exchange of sodium and potassium ions through the neuron's membrane occurs and the cell depolarizes and repolarizes rapidly, roughly 2 msec. The series of connected synapses can have an electric impulse travel anywhere a fraction of an inch to about three feet [33]. A layer called myelin sheath can cover many neurons and accelerate the electrical impulses; myelin sheath is created by glial cells. Figure 3 shows an illustrated cross section of a human brain composed of gray matter and white matter. White matter gets its name from the myelin, a white fatty insulating protein, covering the neuron's axon tracts [37]. Likewise, gray matter gets its name due to color of the neuron's cell body and the presence of the glial cells [38]. White matter is found in the inner layer of the cerebrum and acts as a communication highway or channel for different regions of gray matter. Gray matter is where the cognitive and sensory processing occurs on the outer portion of the brain. As a side note, the ridges of the brain are called "gryi" and the valleys are called "sulci" help identify different lobes of the cerebrum [39]. With the brain constantly performing cognitive and sensory functions, a ton of rapid ion discharges happen consistently enough to form a current. As a result,

engineers and neuroscientists use electrodes to measure voltage changes that come from the ionic current traveling across the different lobes of the cerebrum; this is the definition of electroencephalography [40]. The locations of these electrodes with respect to the different brain lobes have been standardized and is called the international 10-20 system [41]. The type of electrode used to measure these constant voltage values, or signals, affects the resolution. Electrodes that are surgically placed in the user's gray matter can pick up clean signals in a localized region [42]; this type of electrode is gaining some popularity in developmental companies [43]. However, for the most part subjects are less willing to undergo surgery; in addition, the electrodes would only be able to access a small part of the brain. The alternative are non-invasive electrodes, which are placed on the scalp and read the signals with the assist of a water-based conductive gel. A headset of electrodes allows researchers to monitor all regions of the cerebrum. However, since EEG signals are propagated from the gray matter to the scalp, they do suffer additional noise created by the subject's skull and hair [44]; in addition, noninvasive electrodes have the possibility of picking up the same signal and creating a data overlap.

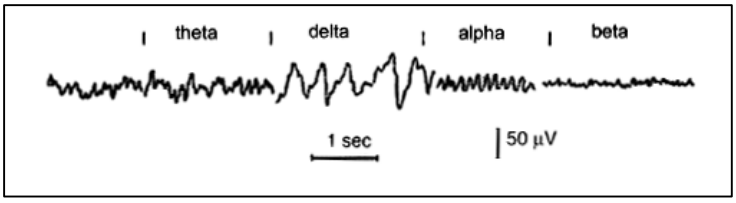


Figure 4: Four types of EEG signals that reflect mental state [45].

Through many EEG studies, four different types of human brain waves have been classified. Each of these brainwaves, theta, delta, alpha, and beta, reflect the user's electrical activity, which is correlated to their mental state; in addition, each wave is characterized by its frequency and amplitude, measured in microvolts. Delta brainwaves are the slowest and run between 1 and 3 Hz and highest amplitude. They occur when a human being is in stage 3 and 4 of sleep. Theta waves have a frequency of 3.5 to 7.5 Hz and are classified as 'slow' activity and are typically seen in sleep as well. Alpha waves have a frequency range between 7.5 and 13 Hz and are typically seen when subjects close their eyes and relaxing. And finally, beta waves are classified as 'fast' activity and have a frequency range of 14 Hz and greater; they are often the dominant wave when a subject is alert.

Chapter 4 – System Design and Modeling

In this chapter, the development of the BCI mobile robotic arm system design and hardware is discussed. The proposed BCI mobile robotic arm can be broken into two main sections, the BCI controller and the robot itself. Under the BCI controller section is the subject with the headset, the signal acquisition, and the BCI subsystem. Under the mobile robotic arm section, are two subsystems, base and arm subsystem, sensors and cameras to provide feedback from the surrounding environment, and motors to implement a command given from the subsystem. The subsystems in both sections contain programmed codes that handle multiple inputs while working across multiple software in order to send the proper output, and ultimately the desired action. Each subsystem uses hardware to receive information from the environment or subject and then sends commands to either the next subsystem or a group of motors. However, in the following sections, an overview of the high-level block diagram and the hardware of the controller subsystem is explained.

4.1 High level block diagram for semi-autonomous BCI mobile robotic arm

Figure 5 shows a high level block diagram that demonstrates how a subject's thoughts transform into a control voltage. The closed loop starts with the subject planning what action he/she wants the robot to perform. For instance, if the user wants the robot to travel forward, he/she would have to imagine tapping their right foot. The imagined action is picked up by the fourteen electrodes mounted on the user's EEG headset as measured voltage potential. This constant feed of voltage potentials, or raw EEG signals, are relayed to the linked computer and recorded on a signal acquisition program. The array of fourteen raw signals are inputs into a trained BCI controller that resamples, filters, and analyzes the signals based on machine learning algorithms. Based on offline data the user recorded earlier, the machine learning algorithm creates a single row command matrix out of the total number of possible actions for the robotic arm's subsystems to view.

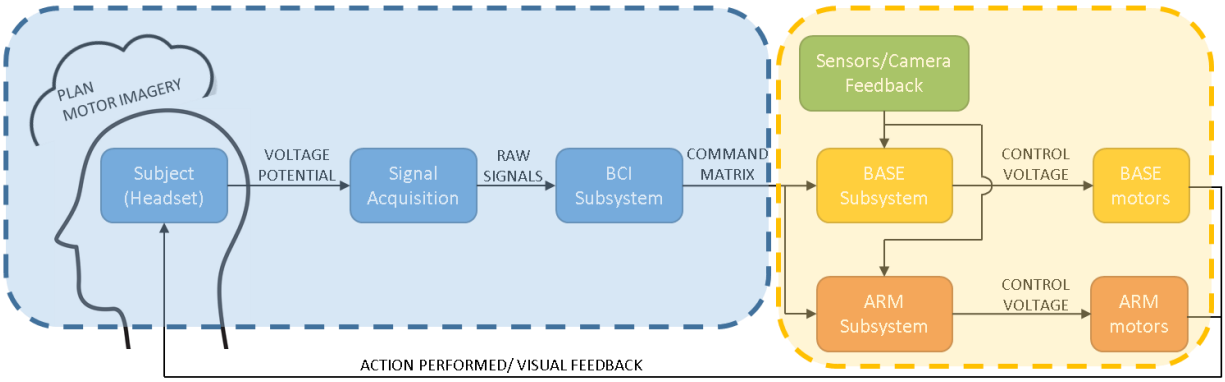


Figure 5: A high level block diagram of the BCI mobile robotic arm. The diagram can be broken into two sections, the BCI controller and the autonomous robot.

Continuing with the example of moving the robot forward, the column designated for right foot motor imagery has the highest numerical value. The robotic arm's command center recognizes this high value as an order to move forward. Before sending the control voltage to the three wheeled base motors, the command center will reference additional information from the sensors and camera to assess the forward path. If there is an obstacle blocking the projected path, the coding will calculate an alternate path around the object that is still forward pacing. Using the sensors as continuous feedback, the robotic arm will appropriately apply the control voltage to the motors to complete the desired action. After the action is performed, the new location of the mobile robotic arm is visual feedback for the subject to begin the block diagram again. The next action may be another movement command (i.e. forward, backward, or left/right turn) or a switch to the robotic arm which is initiated with a jaw clench.

4.2. BCI Controller Subsystem, Hardware

Under the BCI Controller Subsystem, signals are acquired from an Emotiv Epoc+ 14 Model 1.0 electrode headset. Like many commercial grade headsets, the Emotiv headset is readily available and relatively low cost. The electrode locations with the off-the-shelf headset are: AF3, F7, F3, FC5, T7, P7, O1, O2, P8, T8, FC6, F4, F8 and AF4, with CMS/DRL (Common Mode Sense/ Driven Right Leg) references in the P3/P4 location. To allow for greater electrode placement and orientation on the user's head, the Emotiv Epoc+ 14 headset was modified to a guide provided by another BCI and EEG

equipment company called EasyCap [46]. Provided in this modification was a neoprene head cap with 64 possible electrode locations that follow the 10-20 layout, Ag-AgCl sintered electrodes, housing to contain the Emotiv Epoc's original decoder, power supply, and Bluetooth transmitter (as seen in Figure 6). An identical neoprene cap set was purchased and assembled for a typical female's head ratio.

Figure 6: Signal Acquisition Hardware. (Left) Unaltered Emotiv Epoc+14 headset. (Right) Modified EasyCap headset with electrodes connect to Emotiv Epoch decoder on mannequin head.

The specifications for this modified headset are: 14 working electrodes supported by dual reference electrodes, CMS/DRL dual references, sampling rate of 128 Hz, resolution of 14 bits where 1 LSB = 0.51 µV, frequency detection between 0.2 – 43 Hz at an input voltage of 8400 µV peak to peak [47]. The brainwave signals transmit to the computer through a USB Bluetooth receiver dongle that the subject wears around his/her neck in close proximity to the Bluetooth transmitter.

In addition, a conductive, water soluble contact gel was applied to act as a medium between the scalp and electrodes to pick up the user's electrical potential. After a subject completed his/her training, leftover residue from the contact gel was removed from the electrodes and neoprene cap using a cotton ball and paper towels. Distilled water was used to rinse the set of electrodes often to avoid poor connectivity between the signal acquisition software and headset.

4.2.1 EEG headset electrode placement

Since a majority of the brain signals intended to command the BCI mobile robotic arm are asynchronous motor imagery signals, the electrodes need to be positioned

accordingly. Using the 64-channel extended international 10-20 system, twelve active electrodes were selected for motor imagery detection and were placed at FC3, FC4, FC5, FC6, C1, C2, C3, C4, C5, C6, CP3, and CP4. The Emotiv DRL and CMS were placed at location P3 and P4; in addition, a data reference and ground electrode were placed at Nz and Fpz respectively. The electrode layout chosen correlates with the motor imagery research conducted in other academic studies. In one study, three right-handed subjects had their magnetoencephalographic (MEG) and electroencephalographic (EEG) signals recorded with the electrodes in different positions across the scalp while bending their right fingers [48]; after collecting blocks of eighty trials of each subject, the results concluded signature brain wave patterns of hand movement occurred in the C3, C4, and CZ. While a CZ electrode position was not utilized in this project's design, the other electrodes mentioned are shown as active in the highlighted topographical maps on both tables. A more recent study also aimed at localizing brain signal movement with five right-handed individuals sitting in front of a monitor and push a 4-pad button with a respective finger while undergoing a function MRI. The fMRI results matched with the EEG recordings and proved that the motor imagery activation was in the sensorimotor strip [49].

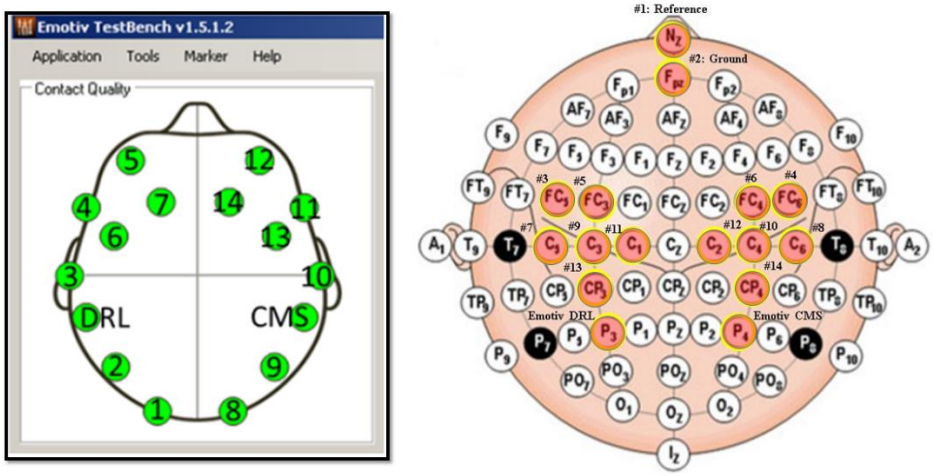


Figure 7: (Left) Emotiv Epoc headset connectivity window; note, electrode placement is based off of original headset orientation, not modified.(Right)Electrode schematic used development of the BCI controller.

Electrode connectivity is monitored using the Emotiv Epoc's development software which correlates signal strength to color orientation (green = good connection, red = very

weak connection). Note, the layout on the left in Figure 7, which is the Emotiv Epoc software is preprogramed to show the unmodified electrode layout. The electrodes are numerically labeled as a workaround so both schematics can be referenced for signal strength and location. With the electrode layout established, the voltage potential measurements, and locations on the human head, correlates between the subjects and provide meaningful insights for the BCI controller.

The high level block diagram and listed hardware shows the necessity to have clear communication between each subsystem of the mobile robotic arm. Looking more closely, each subsystem is layered with the complexity of governing equations, software, and multiple inputs. For instance, the robot base and arm subsystems have unique kinematic equations based on each of their physical design; in addition, they both have physical controllers, such as a PID, to tune each motor. Similarly, the BCI subsystem controller shares the complexity of having machine learning algorithms to classify incoming signals and weighted values to 'tune', or improve, its performance. While the robot subsystems can be discussed in greater detail, the BCI controller subsystem will be primary focus for the remainder of the discussion. One of the greatest challenges and fascinations with the BCI controller is the nature of EEG signals generated by the user. A successful BCI controller can identify key characteristics of a user's brainwaves and associate them to an action. Not only are brain signals nonlinear and nonstationary, they vary from subject to subject; even a subject's state of mind (i.e. how awake, tired, anxious, etc.) can lead to variation in these recorded brainwaves. As a result, unlike the two other subsystems, designing a BCI controller requires a different approach. Overcoming these challenges and details on the BCI subsystem controller will be discussed in the next chapter.

Chapter 5 – BCI Subsystem Controller Design

In this chapter, development of the BCI subsystem controller design is discussed. As mentioned in the previous chapter, brain signals like motor imagery varies between users. As a result, creating a universal controller is unrealistic given the amount of subjects and trials that would require. Instead, BCI controllers need to be customized to the user through training so the machine learning algorithms can recognize his/her brainwaves. To do this successfully, designing a controller has two phases: offline training, where the controller's classifier learns to recognize signature brain wave characteristics, and online testing, where the controller applies what it learned to classify new incoming brainwaves. The accuracy and performance of the controller is directly affected by the quality of the data recordings in the offline training. While the robustness of a machine learning algorithm can also affect a controller's performance, the main contribution to an accurate controller is the cleanliness of offline data. Unfortunately, no machine learning algorithm can make up for bad data. This chapter will first cover the differences between the offline training and online testing block diagrams, and where they occur in the BCI mobile robotic arm. Next, the software used in both runs will be discussed, as well as the configurations chosen for controller development. And finally, the comparison between past BCI architecture and an overview on the challenges of multi-class controller is demonstrated.

5.1 Offline vs. Online Block Diagram

As seen in Figure 8, developing a BCI controller is split up into two phases in the following order: offline training and online testing. The terms 'offline' and 'online' refer to the state of the brain signals. In offline training, the signals are recorded and saved as a data file; these files can be repeatedly viewed, modified, and analyzed. In this phase, the user's reactionary brainwaves are recorded when subjected to a program that presents specific stimuli (i.e. left/right arrow). In contrast, online testing signals are in real time, and show the natural tension between the subject's brain activity in response to the surrounding environment, and the command he/she wants to send to the robot. Unlike the previous phase, these brainwaves are based off of the user's desire, not a stimulus. Because of this key difference, offline training and online testing use different software to meet their

objectives. Note, while it looks like one software is share between both phases, the application is different and will be discussed later in this chapter. For now, the remainder of this section will walk through the individual phases and point out the sequential order of software used.

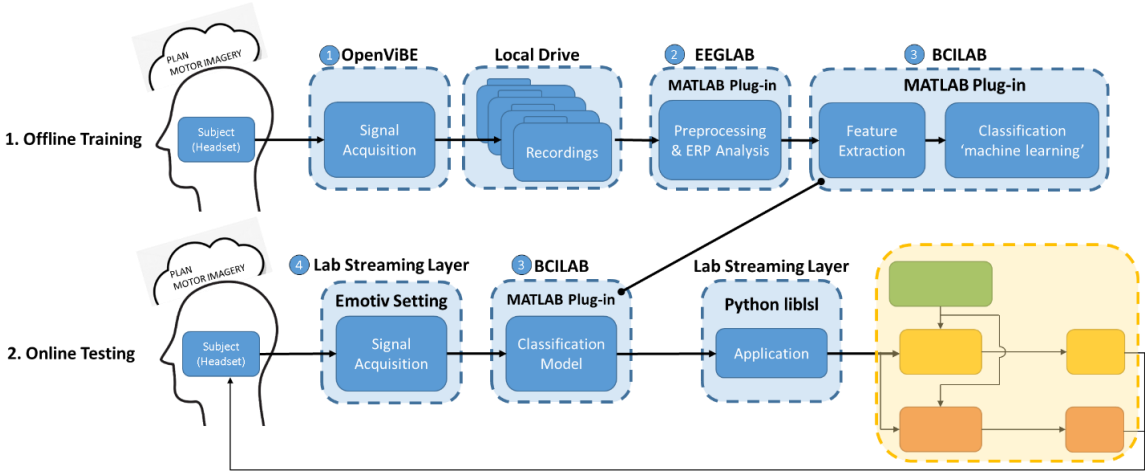


Figure 8: A comparison between the offline and online block diagram for a BCI controller. Notice that only the online testing is connected with the rest of the high-level block diagram. The software associated with each block is listed.

Offline training utilizes two main software, OpenViBE and MATLAB. After the subject is wearing the EEG headset that is connected to the computer, OpenViBE is launched. Under this software, the user's brainwaves are recorded and saved to a local drive as '.gdf' files, the equivalent to a '.pdf' file type for brainwaves. All the raw data is then uploaded to MATLAB where two plug-ins, EEGLAB and BCILAB, are used sequentially. EEGLAB functions primarily as a preprocessing step where the recorded brainwaves are filtered/cleaned and saved as a data set. This plug-in is also used extensively by researchers in the neurology and psychology field, where they identify electrodes that respond to stimuli through topographical maps and event related potential (ERP) time latencies. This application of the program is discussed in more detail later. After the clean data sets are saved on the local drive, MATLAB is relaunched with the BCILAB plug-in activated. BCILAB is the final program used in the offline training phase and performs the feature extraction and classification operations in signal processing. Here, the user selects a machine learning algorithm, runs it through the uploaded data set, and creates a classifier with an estimated performance diagnostic; this classifier can be

saved as a model on MATLAB and ultimately used in the online testing phase to process live data. In summary, the offline training block diagram starts with a user's raw brainwave responses to stimuli and ends with a classifier, a machine learning algorithm that went through a clean data set and converged.

All the work in the offline training phase is done in preparation for online testing; its final stage creates a classifier that will be used to identify desired signals in a real time stream of brainwaves. To stream a user's brainwaves, online training utilizes two main software: lab streaming layer (LSL) and MATLAB. Similar to OpenViBE's signal acquisition function, LSL can identify the incoming signals from the connected headset. The user then launches, BCILAB once again. While this plug-in was used initially to create a classifier in offline training, it also has built-in capability to link and read an LSL's incoming stream. Pairing the live brainwave stream and classifier, BCILAB will generate an output stream control matrix. The size of this row matrix is based on the number of brainwaves the controller needs to identify. Based on the controller's six-stage architecture in Figure 1, the output stream would be a size 1x6 matrix. BCILAB can send this output stream to other programs outside of MATLAB, like LSL, as a command stream. This highlights the impressive capability of LSL to act as a data pipeline that connects software together; acting like a multi-signal acquisition program, LSL can support the stream of both raw brainwaves from the headset and command matrices from BCILAB. Controllers and other software can link with LSL to receive the streams in their preferred coding language. For the mobile robotic arm, the control matrix is streamed in python because the arm and base subsystem's controllers use this language. In summary, the online testing block diagram starts with a user's raw stream brain signals, a mixture of noise and command motor imagery signals, and ends with a control matrix output stream sent to the mobile robotic arm.

5.2 Software Integration

With the software outlined in both the offline training and online testing block diagrams, this section elaborates on the specific software settings and configurations used

to develop the BCI mobile robotic arm controller. The number in the heading of each software heading is in reference to Figure 8 and will dictate the order.

5.2.1 OpenViBE: Offline Training – Signal Acquisition

OpenViBE is an open source software designed as an all-encompassing BCI system that will acquire, filter, process, and classify brain signals [50]. The version used for this project is v1.2.2. This software is split up into two applications: OpenViBE acquisition server and designer. The acquisition server can recognize and link to a number of commercial grade EEG headsets as long as the proper driver and software development kit (SDK) are installed; for the Emotiv Epoc headset, the heritage Emotiv Research SDK was installed, allowing connection between the OpenViBE and the electrodes on the modified headset. Note, using the Emotiv SDK comes with presets for the unmodified electrode labels and locations (but these can be modified in the OpenViBE designer application). For all offline training recordings, the following window opens when launching the acquisition server (see Figure 9).

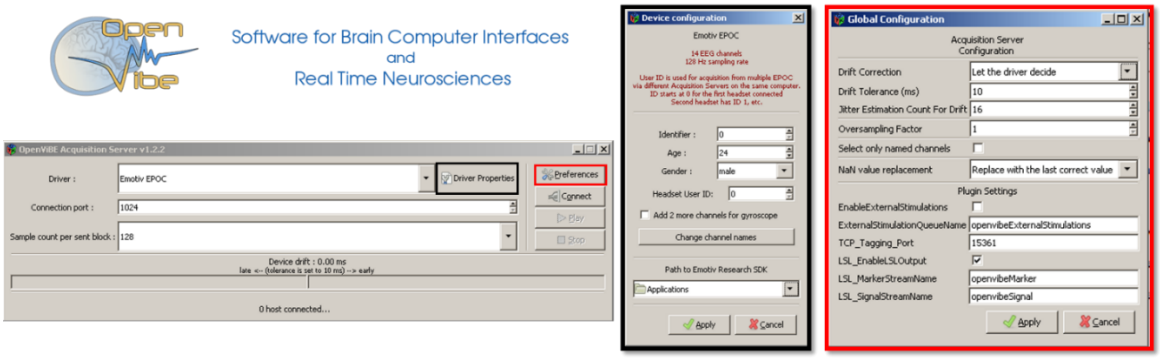


Figure 9: (Left) OpenViBE Acquisition Server main GUI window where the user can select the appropriate EEG headset. (Right) More detail about the provided can be seen in the driver properties and preferences.

Emotiv EPOC is selected as the appropriate EEG headset in the Driver drop-down field. Additional information about the headset can be viewed in the 'Driver Properties' button. Here, the acquisition server knows the number of EEG channels and sampling rate. Additional information about the subject can be filled, and the electrode locations can be changed using the 'Change channel names' button. When the 'preferences' button is

clicked, a ‘Global Configuration’ window is generated that allows the acquisition settings to be altered. In addition, the bottom half shows some cross compatibility with online testing application, lab streaming layer (LSL). However, for the offline training purposes, the user would select ‘Connect’ and then ‘Play’ to officially link the EEG headset to the computer; a blue horizontal bar of varying lengths will appear below the main window to confirm voltage potential readings are being picked up from the headset.

With OpenViBE acquisition server online, the OpenViBE designer application can be launched. Like LABVIEW, the designer application is a graphical programming software where the user adds pre-programed algorithm boxes to a window manager and links them to one another in a desired order (see Figure 10). One of the critical algorithm boxes that connects the OpenViBE acquisition server application is the ‘Acquisition Client’ box; this box has text fields where the acquisition server hostname and port can be inputted and five outputs: experiment information, signal stream, stimulations, channel localization, and channel units. Subsequent boxes connecting to signal stream and stimulation outputs can be computational where the signals are going through some mathematic operation (i.e. multiplication, square-rooted, etc.), and visual where the signals and stimulations are displayed on a graph. Other boxes like ‘GDF file writer’ and ‘GDF file reader’ allow recordings to be saved and played back for additional analysis. In some applications, the ‘GDF file reader’ may be the first box placed in the window manger instead of the ‘acquisition client’.

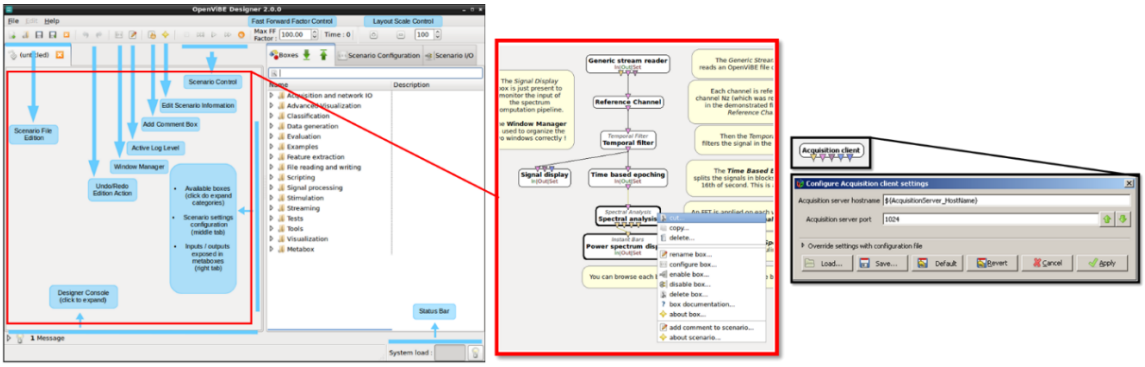
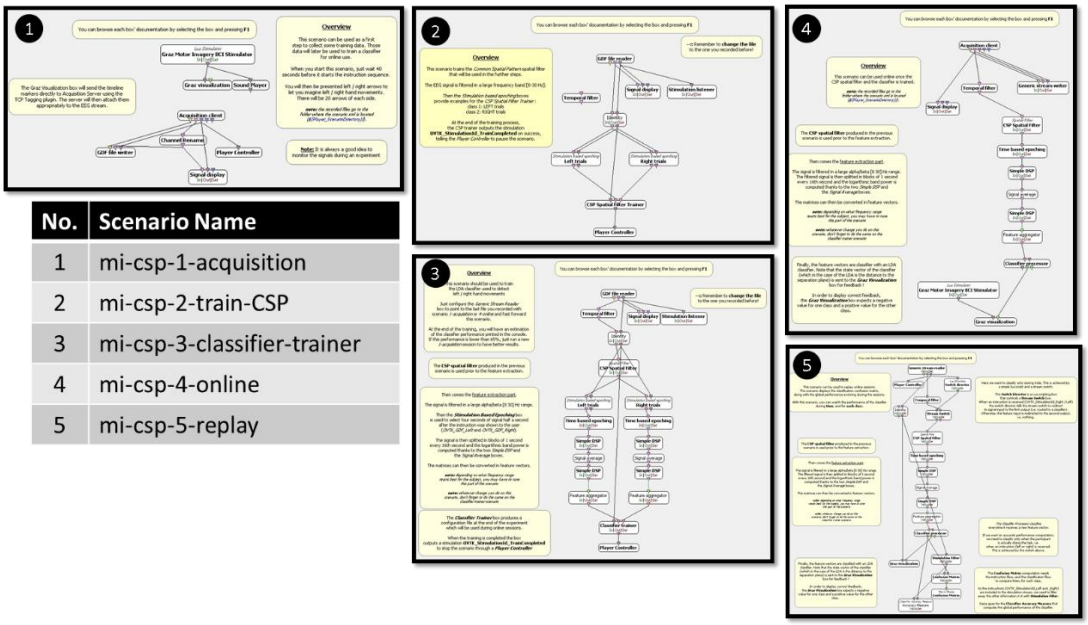


Figure 10: (Left) Main GUI for OpenViBE Designer [51]; (Middle) An example of a connected box layout [51]; (Right) The acquisition client box show five outputs and two textboxes.

A layout of connected boxes is known as a scenario and can be saved on a local server. Multiple scenarios can be used in a sequential order that encompasses both BCI controller phases: offline and online testing. An example of this is provided by OpenViBE where five scenarios are used to record motor imagery signals, train a common spatial pattern (CSP), train a classifier, perform an online test phase, and determine the accuracy of the classifier (see Figure 11).



No.	Scenario Name
1	mi-csp-1-acquisition
2	mi-csp-2-train-CSP
3	mi-csp-3-classifier-trainer
4	mi-csp-4-online
5	mi-csp-5-replay

Figure 11: An OpenViBE example of five scenarios to record motor imagery signals, train a CSP filter, train a classifier, conduct and online test, and calculate the performance of the classifier.

These five scenarios were the framework for the BCI controller semi-autonomous wheelchair design effort [15]. The previous controller development ran through these scenarios multiple times to create six filters and classifiers; each filter and classifier paired two of the four commands: left hand, right hand, jaw clench, and rest. A separate scenario was created where all of these filters and classifiers were linked to the 'Acquisition Client' and a command matrix was generated to MATLAB. While possible to distinguish the command signals, the computational demand of the design controller was high due to the amount of classifiers being used.

For the BCI mobile robotic arm controller, only the 'mi-csp-1-acquisition' design scenario was used to record a subject's brain signals responding to stimuli. This scenario

has a box called “Graz Motor Imagery BCI Stimulator” that generates a pop-up window that cycles through a script randomizing left and right red arrows (see Figure 12). The user is able to customize the stimulation script to increase or decrease the time duration of the images shown, and include an auditory cue to keep the subject focused.

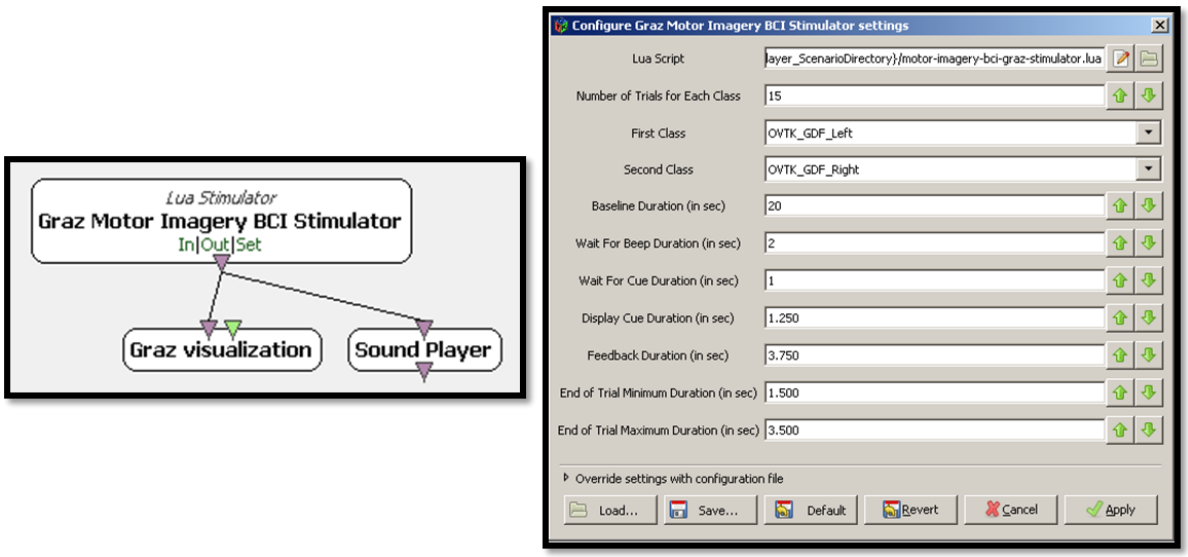


Figure 12: (Left) Box layout for the stimulation script in the OpenViBE acquisition scenario. (Right) The window to modify the stimulation script.

Per Figure 13, the settings of the stimulation script generate the timeline for one trial. At the start of the trial, t=0 sec, a green cross is displayed on the black screen. Two seconds after, the ‘sound player’ box is activated and an audible beep is to alert the subject that the stimulation is going to appear. One second after the beep, the left or right arrow is selected at random and displayed over the green cross for 1.25 sec. After the arrow disappears, the subject will perform a physical or imagined action unique to the arrow for the remainder of the time (approximately 6.5 sec). An example of this could be squeezing the left hand after the red left arrow appears. The trial timeline is repeated an additional twenty-nine times until both arrows have appeared fifteen times. This approximately lasts six minutes long and is called a test.

While the “Graz Motor imagery BCI Stimulator” box allows the user to customize the duration of each element, it is strictly limited to the responses to two stimuli, left/right arrows. In addition, after the stimuli script finished showing the total number of

stimulations, OpenViBE will automatically stop the recording and save the file. Both of these drawbacks make it hard to design a multi-class controller. Ideally, a multi-class BCI controller would have different six-minute tests, and breaks, recorded on one file; since brainwaves are non-stationary continuous signals, it is rational to keep recording the stream of voltage potentials to observe how the subject's mental state changes over time. The next software used in offline training is the proposed workaround to this problem in OpenViBE designer.

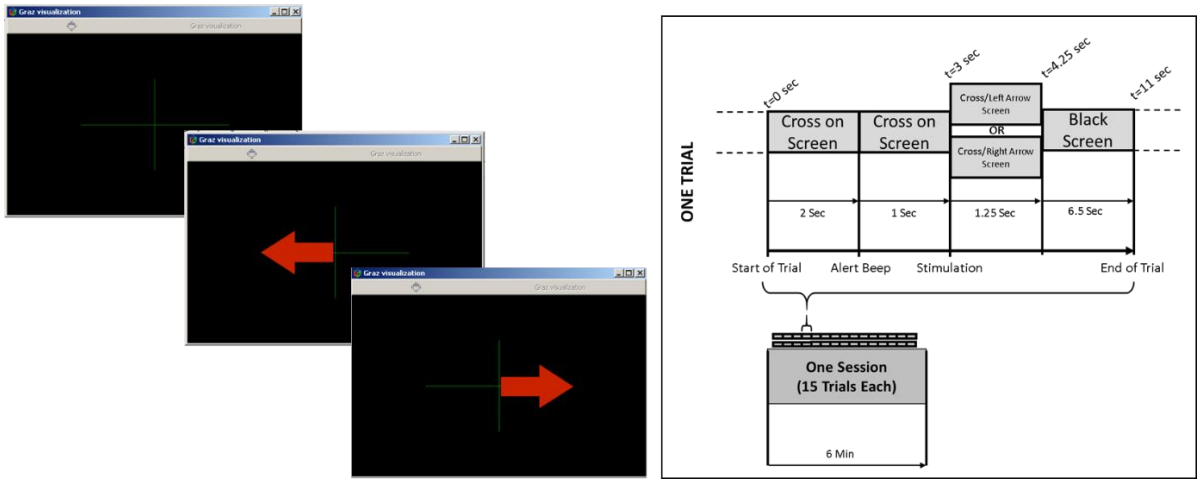


Figure 13: (Left) The window running the stimulation script; notice the green cross and the red arrows pointing to either the left or right. (Right) A timeline used for the development of the BCI development controller.

5.2.2 EEGLAB (MATLAB Plug-in): Offline Training –Signal Preprocessing

EEGLAB is an open source software developed by the Swartz Center for Computational Neuroscience (SCCN) at the University of California San Diego [52]. This MATLAB plug-in can process raw EEG recordings via applying filters, incorporating independent component analysis (ICA), time/frequency analysis, artifact rejection, and event-related statistics [53]. In addition, EEGLAB is capable of appending raw recordings together, renaming marker names, and deleting unwanted sections of data; this later capability is used extensively in creating a single file for training a classifier. The version used for the development of the BCI controller is version v2020.0.

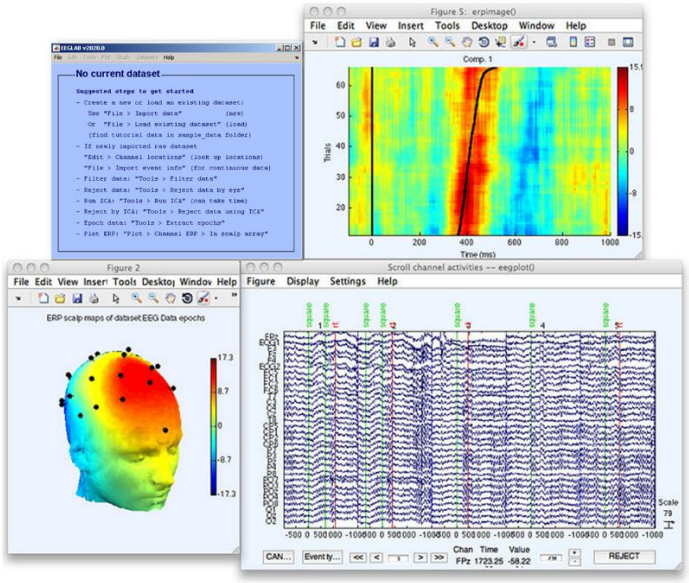


Figure 14: A collection of sample graphs and data generated by EEGLAB

5.2.2.1 Pre-processing the Raw Data

Before discussing in detail the kind of analysis performed on the raw recordings, it is important to discuss the preprocessing capabilities EEGLAB has and the configuration chosen for the development of the BCI controller. When cleaning the raw EEG data, the options under the 'Tools' and 'Edit' drop down menu are utilized. The procedure to pre-process the data was adapted from neuroscience professors and EEGLAB architects [54-56]. The procedure can be broken up into six steps: 1. uploading the raw recording, 2. applying channel locations, 3. removing epoch baseline, 4. applying a filter, 5. extracting epochs, and 6. scrolling the data to verify all changes have been made (see Table 1).

Table 1: The procedure used in preprocessing the raw EEG recordings. Note, the values and location of these functions are in the main EEGLAB GUI.

No.	Preprocessing Steps	Menu in EEGLAB
1	Upload Raw File, reference channel	**File**>Import data>From EDF/EDF+/GDF files (BIOSIG toolbox)
2	Channel Locations	**Edit**>Channel Locations> use MNI coordinate for BEM dipfit model.
3	Remove epoch baseline	**Tools**>Remove epoch baseline
4	Filter Raw Data *(high pass filter: 0.5 Hz) (Top notch filter: 50 Hz)*	**Tools**>Filter the data>Basic FIR filter
5	Extract epoch *(-1, 4 sec)*	**Tools**>Extract epochs> select event type and limits
6	Channel Data Scroll	**Plot**>Channel data(scroll)

When uploading the raw EEG recording, the EEGLAB software will want to specify what kind of file type it is and what is the reference node. Based on the electrode layout schematic in Figure 7, the electrode numerical '1' label in the Nz position would be selected. Next, for assigning channel locations coordinates, EEGLAB will reference both the channel labels OpenViBE assigned in the uploaded file, and the built-in 3-D coordinates for electrode location in the international 10-20 system; here the 'MNI coordinate file for BEM dipfit model' was used. Now that all the electrode locations have been verified, step 3 and onward start to focus on cleaning the actual signal. 'Removing the epoch baseline' in step 3 is a function that removes the 'DC offset' of all electrode channels and allows them to 'stack' on top of one another well. In Figure 15, the brain signals in step 1 and 3 show the outcome of removing the DC offset. Next, a filter is placed on the raw data. The step applies a high-pass filter at 0.5 Hz to minimize slow drifts in brain waves. In addition, a 50-Hz notch filter is applied to remove line-noise.

Extracting epochs in step 5 refers to isolating sections of interest in the EEG recording. This step will keep only the signal data surrounding the desired stimulation markers. EEGLAB will ask the epoch limit, how many milliseconds before and after the stimulation should be kept. Stimulation markers are vertical lines on the signal visualization window and are labeled as a string of numerical codes from OpenViBE that need to be referenced on their site (see Table 2) [57]. For the left and right red arrows in

the stimulation script from OpenViBE, the associated codes are 769 and 770. As a result, these codes are used with an epoch limit of -1, 4 sec to capture the signals before and after the arrow is shown. Note, deciding when to conduct the extract epoch step is a based how the brain signals are going to be analyzed. If the extraction is done during the offline training phase, the brain signals will be analyzed like computational neuroscience studies. In this case, a subject's event related potentials, power generation, and topographical maps will be observed; correlations on what electrodes react the most to doing a physical or imagined action among subjects can be analyzed. If the extraction step is done later in the Offline Training phase, i.e. training a classifier, the brain signals will be analyzed to find signature wave characteristics that a machine learning algorithm can utilize as a command. For the development of the BCI controller, both extractions are used to verify the correct electrode locations are being utilized and to find signature signal characteristics. The final step is to review all the preprocessing steps visually using the 'Channel Data Scroll' function found in the plot drop down. The voltage and time scale can be adjusted as well. Epochs that show malfunctioning electrodes or excess noise can be rejected from the set; simply left-mouse-button (LMB) on the epoch and select 'REJECT' on confirm the deletion.

Table 2: The list of numerical code markers used in OpenViBE raw signal recordings.

Order	Event Stimulation Codes	Stimulation declared
1	32769	OVTK_StimulationId_ExperimentStart
2	33282	OVTK_StimulationId_Beep
3	769	OVTK_GDF_Left
4	770	OVTK_GDF_Right
5	781	OVTK_GDF_Feedback_Continuous
6	897	OVTK_GDF_Correct
7	800	OVTK_GDF_End_Of_Trial
8	33281	OVTK_StimulationId_Train
9	32776	OVTK_StimulationId_BaselineStop
10	32775	OVTK_StimulationId_BaselineStart
11	1010	OVTK_GDF_End_Of_Session
12	898	OVTK_GDF_Incorrect
13	786	OVTK_GDF_Cross_On_Screen
14	768	OVTK_GDF_Start_Of_Trial
15	32770	OVTK_StimulationId_ExperimentStop

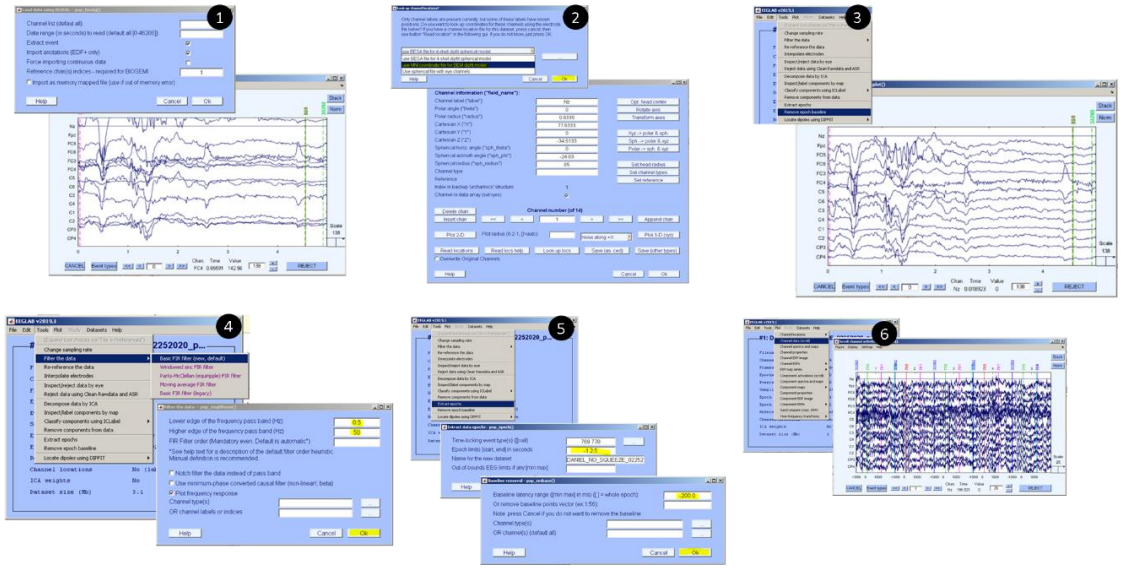


Figure 15: The windows and representation of the data at certain steps of the preprocessing procedure. Note, the numbers correspond to the steps in procedure table.

5.2.2.2 Independent Component Analysis (ICA)

While the brain signal recordings have been cleaned through the pre-process procedure, the data still has artifacts, like blinking, that is unavoidable. Fortunately, EEGLAB can remove these artifacts using a method called independent component analysis (ICA). This method separates independent sources linearly mixed in several sensors [58] [59]. While this can be applied to any kind of signal, like sounds waves, neurologists use it extensively in studying EEG recordings. If the electrodes on the scalp are described as microphones, and there are local areas in the brain that are firing, called local synchrony, the propagation of multiple synchrony can be picked up by one electrode, resulted in a 'mixed signal' (see Figure 16). The result is to unmix and locate the origins of these multiple local synchronies.

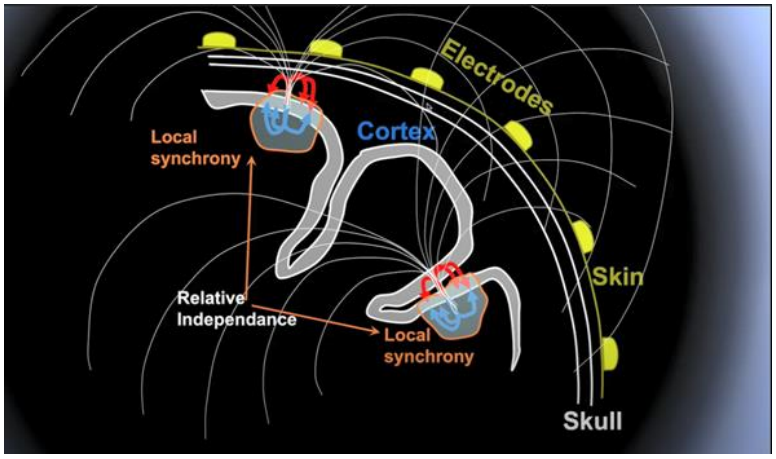


Figure 16: An illustration showing how one electrode and receive propagated signals from multiple brain sources and creating a mixed signal [60].

Mathematically, these can be represented using matrix multiplication.

$$U = WX \tag{1}$$

Where X is the recorded data (channels x time), U are the ICA source activities (component x time), and W is the ICA unmixing matrix (components x channels). After the ICA method is calculated, sources like an artifact and an actual brain source can be separated. These sources are statistically independent and constraint free from the raw data. Much like the fundamental understanding of switching between time and frequency domain in a controls class, the user is capable of switching between ICA components and time-based recorded data. In addition, the deletion of certain ICA components will remove these artifacts. There is a similar approach called principal component analysis, PCA, that was used in the past but is less useful because it destroys the link that allows the user to interchange between the two. Both methods can be illustrated graphically [61]. The graphs in Figure 17 show the application of PCA and ICA; variable 1 and 2 on the x- and y- axis are two electrodes of interest and the data points on the graph are time points during a recording.

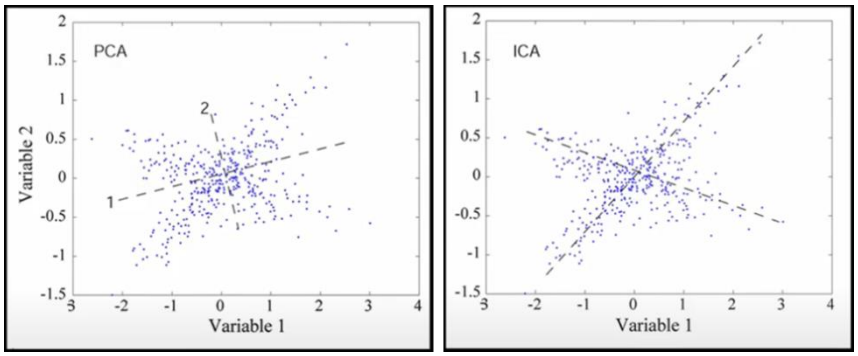


Figure 17: A graphical representation showing the difference between PCA and ICA analysis, which is used to identify and remove artifacts. The graph on the left shows the PCA method while the graph on the right shows ICA [61].

Here, PCA will find axes among the cluster of points where the projection of the data has maximum variance. In contrast, if the same electrodes and data points were used, ICA will find the projection of the data points that maximizes independence. In other words, the axes calculated will fit with the data points while highlighting two different trends. The graph on the right clearly shows two different processes between the desired electrodes; these processes indicate two sources and an unmixing matrix can be applied to find the sources. Another way to understand ICA is by taking brain signals and overlaying them with histograms. Transforming Figure 16 into a signal representation, two local synchrony correspond to brain source 'A' and 'B'. To quantify and compute these signals, the magnitudes of the signal can be sorted into different histogram bins, known as a probability density distribution(see Figure 18). The electrodes, or scalp channels, see a linear mixture of source 'A' and 'B' and consequently these histogram bins are a combination of the two. Now if there is an infinite amount of brain sources that are independent of each other, the mixture values taken from the scalp channel will be a Gaussian distribution; as a result, ICA will start from the scalp channels and then rotate axes in multi-dimensional space so its makes a projection as non-Gaussian, also known as the central limit theorem.

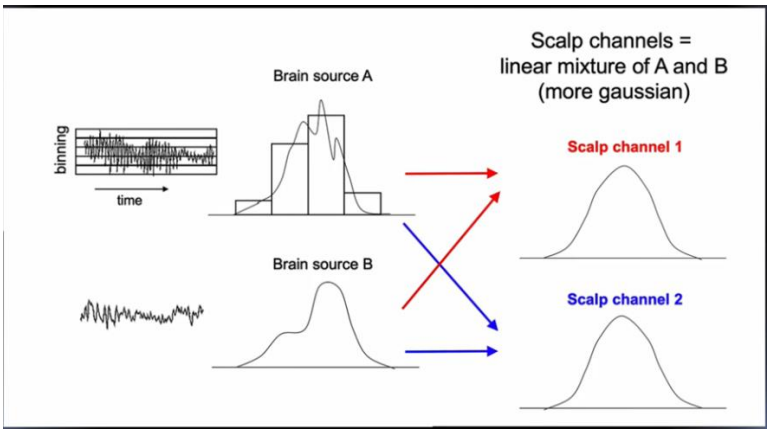


Figure 18: A visual representation of ICA using signals and histogram bins [61].

Combining these two examples illustrates the ICA training processes. A graph showing two variables from EEG data is shown with the time points plotted; the first step the computational process does is called whitening. Whitening is the process of normalizing each axis and is started by creating the projection of the data on the two axes. These two axes are then rotated in such a way so that the projections are independent of one another which visually flattens the Gaussian curve shape from the histogram into a somewhat equal distribution of bins.

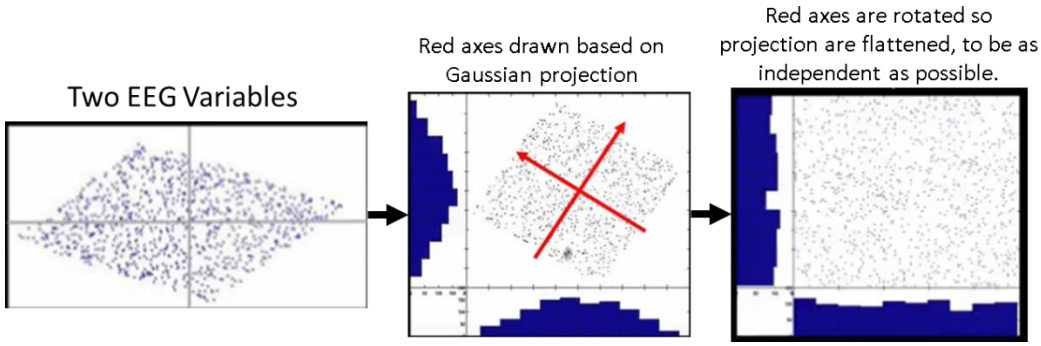


Figure 19: Plotted EEG data going through the ICA training process [61].

Rotating the axes to maximize independence of the variables is going to maximize the entropy. Entropy in this case does not refer to a thermodynamic variable, but is defined as the average level of "information", "surprise", or "uncertainty" in the variable's outcome. Claude Shannon formulates this information entropy in his 1948 paper "A Mathematical Theory on Communication," one of the foundations for information theory. The formula can be seen below:

$$H(x) = -\sum_{x \in X} p(x) log_b p(x) \quad (2)$$

Where p is the probability. In general, the more uniform distribution the rotated data gets, the more random and higher entropy there is in the possibility. Since Figure 19 shows a 2D representation of data with two variables, there is a combined, or joint entropy.

$$H(X,Y) = -\sum_{(x,y) \in XxY} p(x,y) log_b p(x,y) \quad (3)$$

Joint entropy also has the following relationship where the I-two-variable-function, $I(y_1, y_2)$, is known as mutual information. Here, the individual entropy of the first and second variable are combined and subtracted from the mutual information between the two. The joint entropy can be calculated from creating a contingency table showing all the possible outcomes; the probability from each cell are then calculated and then plugged Shannon's equation of entropy to get a value.

$$H(y_1, y_2) = H(y_1) + H(y_2) - I(y_1, y_2) \quad (4)$$

To increase the amount of entropy, the mutual information value needs to be minimized; as a result, the variables in the I-two-variable-function need to equal zero, which drives from numerical analysis. ICA infomax, a general version, incorporates a learning rule, which changes the weight matrix to rotate the axes and ultimately find the maximum entropy. There is one more term called the "natural gradient," or Amari, that is a rescaling factor that helps the weigh matrix to converge faster (see Figure 20).

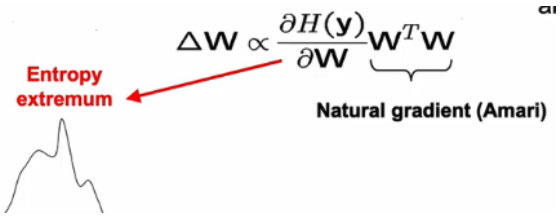


Figure 20: The additional rules screen in the general infomax ICA run [61].

There are different versions of ICA; the ones listed in EEGLAB are: runica, binica, jader, sobi, acsobiro, and picard; while some of them vary in computational runtime, the version that runs a fast but accurate time is known as 'picard'. Referencing Figure 21, these versions can be accessed through EEGLAB via the 'Decompose data by ICA' from the Tools drop down. Once, the ICA runs through all the data, the end result is a list of virtual channels numbered from most variance to least variance. Note, these virtual channels are isolating brain sources from the data; as a result, they do not represent a specific electrode location like the raw data. Each channel should be inspected to determine if the virtual channel holds an artifact. There is another built-in function known as ICLabel that classifies virtual channels as 'brain' or 'other' to help distinguish artifact channels.

Figure 21:(Left) Navigation to find the ICA function in EEGLAB. (Right) The results are virtual channels that are ordered from most to least variance.

Because this study is using a small number of active electrodes, twelve versus sixty-four, only one virtual channel was rejected. Once a virtual channel is rejected and removed from the data set, a window showing the impact to the actual electrode channels is shown (see Figure 22). As a result, cyclical signals, like peaks generated from blinking, can be identified and rejected in a virtual channel, and then removed as a consequence in the actual data set.

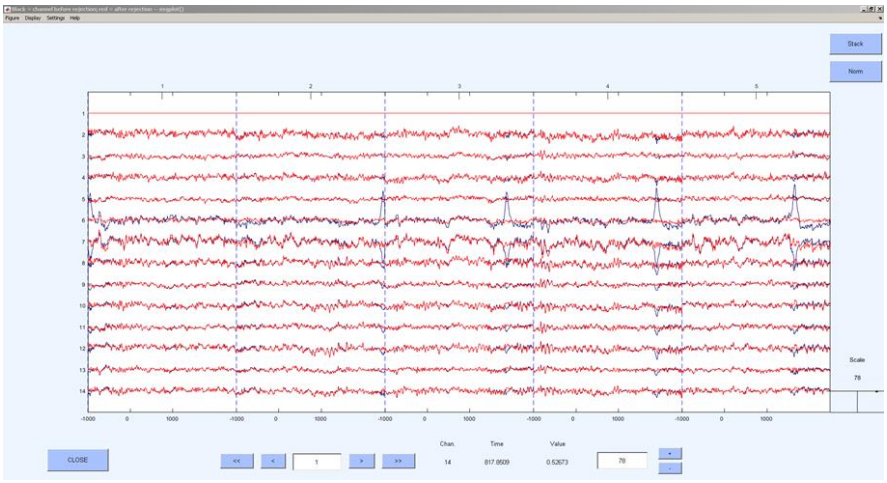


Figure 22: A window showing the impact of removing a virtual electrode channel. The blue colored signals represent the original data while the red colored signals show the altered data after channel rejection. Note that the peaks in the original data, result of blinking, were removed.

Now that the raw EEG signals have been preprocessed and artifacts removed, the final step is to generate plots to analyze the subject's response to stimulation. EEGLAB splits the analysis into two groups: real electrode recording and virtual electrode plots. These can be found in the under the plot drop down menu where options with the prefix 'channel' and 'component' refer to the real and virtual data. While some studies extensively use 'component' plots to analyze a subject's brain response, this paper will focus primarily on 'channel' plots.

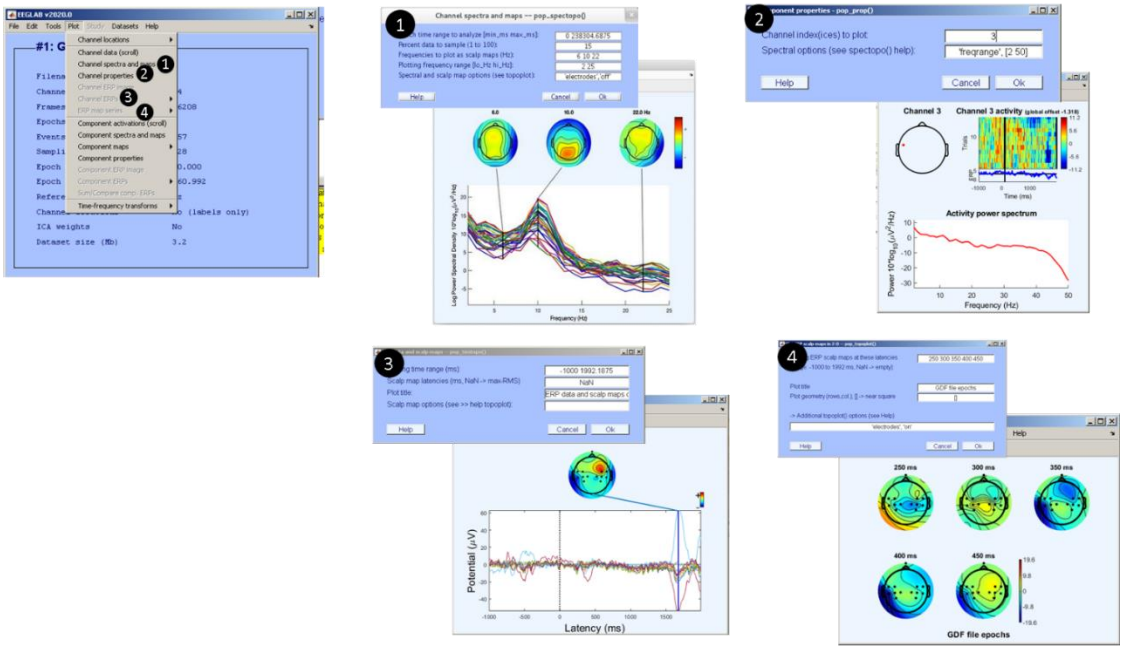


Figure 23: The channel plots used when analyzing a subject's motor imagery signals when responding to a stimulus. The numerical labels correspond to the functions selected in the 'plot' drop-down.

Referring to *Figure 23*, function 1 'Channel spectra and maps' plots a graph of Log Power Spectral Density vs Frequency; here all channels are plotted in different colors and show at what frequency is there a great distribution of power by the peaks (in this case 10 Hz). This function is used primarily as a sanity check to verify the appropriate frequencies are not being cut off when applying the high-pass and notch filter. Function 2 'Channel properties' will plot the ERP and power spectrum of a desired electrode. This function was used extensively in the beginning of the BCI controller development to understand the unique responses from all the electrodes. Function 3 'Channel ERPs' plots all the channel ERPs over the epoch limit and show the peaks. This function is used primarily to compare the subject's responses and will be shown in the results section. Finally, function 4 'ERP map series' will show the topographical map at specific time points and generate a scale showing the positive and negative voltage potential.

While EEGLAB is capable of performing analysis on single EEG raw recordings, it is also able to compare multiple recordings and utilize the same functions with clusters of data points. A simple ERP STUDY can be created where the number of conditions and

subjects are specified. The user can then upload each clean recording and then see the different ERP signals between the averaged conditions. This process will be described in more detail and the results will be shown later.

5.2.3 BCILAB (MATLAB Plug-in): Offline Training – Classification

BCILAB is an open source software developed by the SCCN as a compliment to EEGLAB [62]. This MATLAB plug-in will take clean continuous EEG signals and train a classifier using a machine learning algorithm that will be implemented into a BCI controller. Here, the user is able to design, prototype, test, experiment, and evaluate a broad range of machine learning algorithms. The version used for the development of the BCI controller is version 1.4-devel. Note, make sure the user restores the default file path in MATLAB and then route to BCILAB; BCILAB has an embedded version of EEGLAB that would conflict if EEGLAB used for preprocessing the recording.

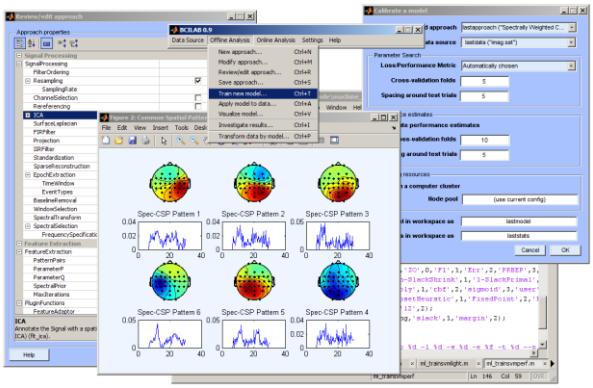
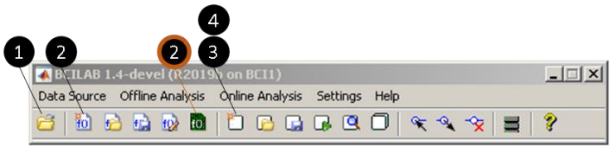


Figure 24: A collection of sample menus and data generated by BCILAB [63].

As mentioned before, BCILAB is the catalyst software that combines both the offline training and online testing. In this section, an overview of the GUI is provided, the procedure used to teach a classifier, and an in-depth of background on the selected machine learning algorithms is discussed. Specifically, steps discussing the filter and machine learning algorithm chosen will go into depth with a general background. As seen in Figure 25, the offline training can be broken down into five steps.



No.	Offline Training Steps	Menu in EEGLAB
1	Upload Preprocessed Recording	**Datasource**>Load recording(s)...
2	Define a new approach	**Offline Analysis**>New Approach...> Common Spatial Patterns (ParadigmCSP)
3	Define machine learning algorithm parameters.	**Offline Analysis**>Train new model...
4	Evaluate performance	**Offline Analysis**>Apply model to data...
5	Save Model	**Data Source**>Workspace>Save...

Figure 25: (Top) Main BCILAB GUI with balloon callouts highlighting the offline training procedure. (Bottom) A table describing the step-by-step offline training procedure.

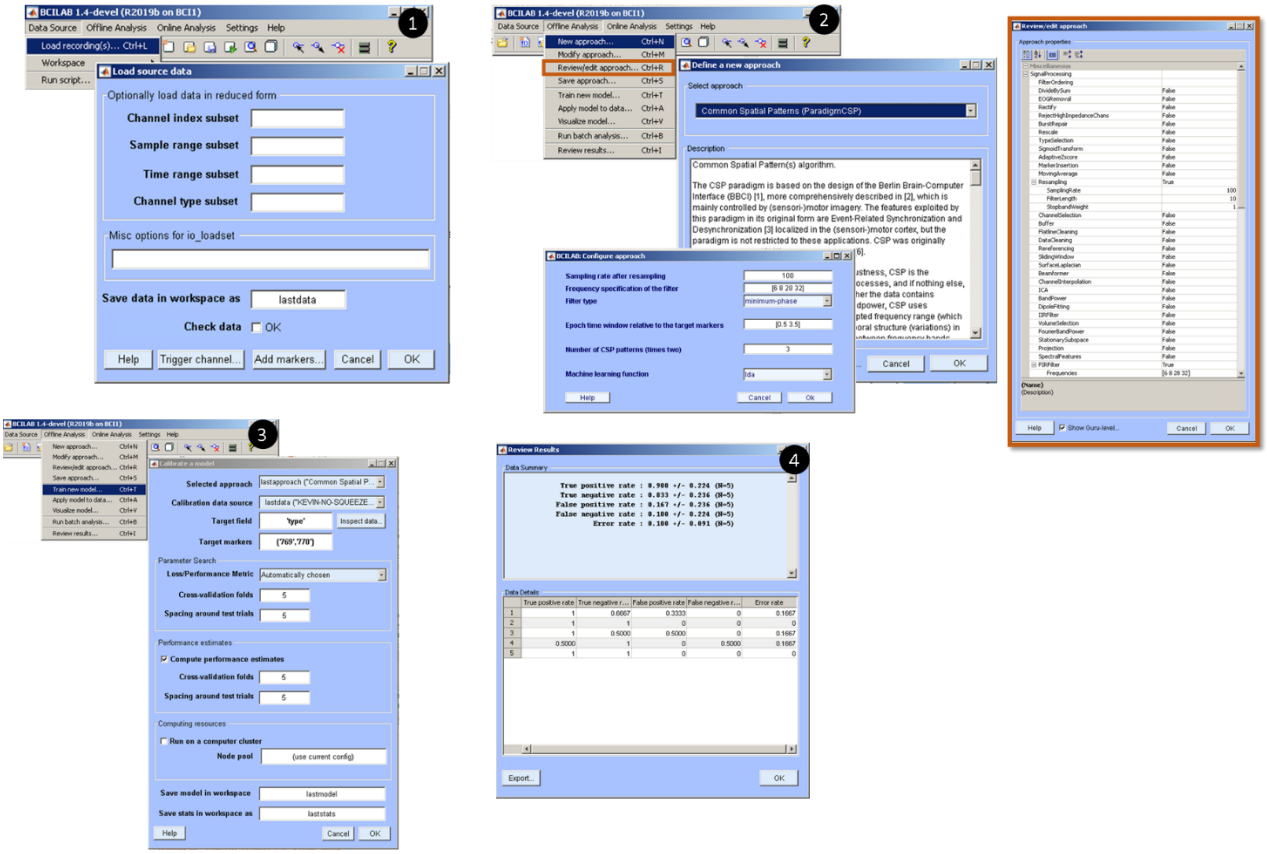


Figure 26: Navigating the BCILAB GUI for offline testing. Numbers correspond to the steps outlined in the offline testing procedure. Note, step 2 has additional customization to further define the filter that can be accessed selecting the 'Review/edit approach…' function.

5.2.3.1 Step 1 – Uploading Preprocessed Data

Referencing the balloon numbers with the table, the user first uploads the clean data from all the preprocessing done in EEGLAB by clicking the desired icon or the following the drop down description in the table.

5.2.3.2 Step 2 – Define a new approach

Next, the user will select a computational approach, or also known as a filter, that best matches the desired signals to the machine learning algorithm. In contrast to OpenViBE, BCILAB has an enormous selection to choose from; some options are experimental and harder computationally than the basic approach.

Step 2 – Define a new approach: background

Before discussing the filter/approach chosen for the development of this BCI controller, it is important to cover the general types of filters applied to brain signals. These filters can be broken up into the following groups: static filters, spatial filters, and temporal filters. One of the basic approaches for static filters is log-Bandpower, or log-BP. This filter is available in BCILAB and is typically used for recordings that involve oscillatory processes and detect the variance of a brain signal.

On the other hand, spatial filters are more complicated because they operate across space, but not across time; most spatial filters are linear in nature and this can be represented as the following:

$$Y(n) = \boldsymbol{M}X(n) \tag{5}$$

where Y(n) is the multi-channel output signal matrix, X(n) is multi-channel signal matrix, M is a constant matrix, and n input represents the sample. This is an important filter because the source to sensor relationship, i.e. the local synchrony and the electrodes in Figure 16, are linear in nature. Like the description of ICA in the previous section, this allows BCI developers the ability to trace the approximate source signals from the recording by inverting the volume conduction of the brain and remap the channel signals; this reliably and accurately allows the controller to rely on the source of the brain that is

creating these signals, not the signals themselves. Examples of spatial filters are re-referencing, surface Laplacian, and Common Spatial Patterns (CSP), which is the selected filter and will be discussed in more detail later. Re-referencing is where you subtract either a selected channel's signal from all channels, or the average of all channels' signal from each channel. Surface Laplacian is more locally applied where one channel is subtracted to all surrounding channels. Figure 27 shows how the spatial filter in Equation (5) can be visualized; here, the M matrix has six rows, each row is represented as one topographical plot, multiplied by the number of channels represented as black nodes, or X(n), on the topography map. Each topographical map is a spatial filter, but all together is a linear combination of channels that gets multiple to all the electrodes and produces an output sample. Here the colors represent positive or negative values that will be multiplied to the channel; note, green means a nearly zero value is multiplied by the channel. Applying this filter to an entire recording will create a new time series that help identify a particular source; in this case a CSP was used to pick up motor input and output sources in the motor cortex. This weighted linear matrix can be inverted and multiplied by the output and show which electrodes are affected by the source signals (see Equation (6)).

$$X(n) = \boldsymbol{M}^{-1}Y(n) \tag{6}$$

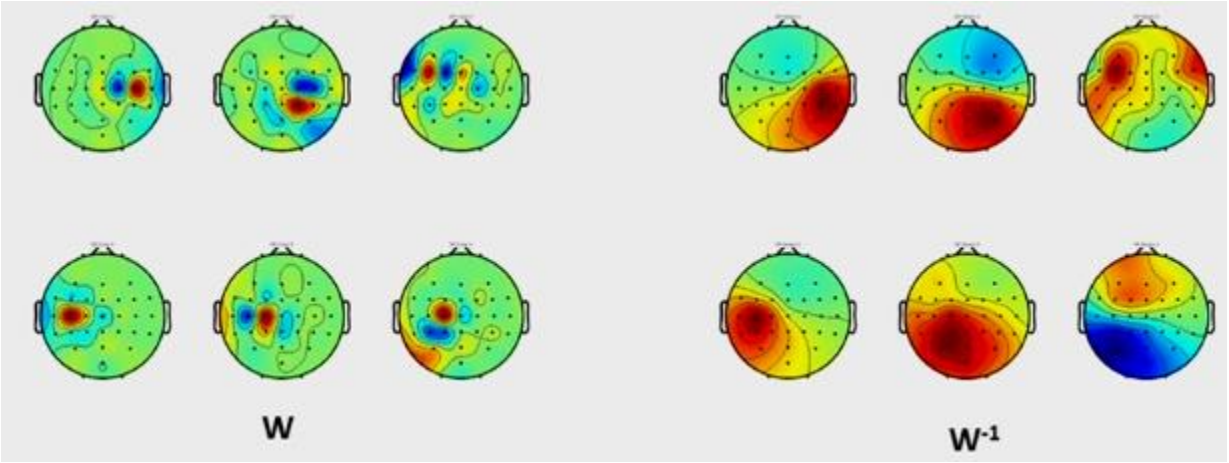


Figure 27: (Left) An example of a visualization spatial filter, specifically a Common Spatial Pattern, used to identify sources found in the motor cortex region of the brain. (Right) If the linear weight matrix is inversed and multiplied by the detected sources in the brain, the electrodes affected by the source can be detected [64].

In contrast to spatial filters, temporal filters do not go across channels or space, but transfer information across time. In creating a temporal filter, a time window or a wavelet transformation is used. A simple example of a temporal filter is a moving average. A special kind of temporal filters is a spectral filter where a sample input signal can be broken up into a summation of sinusoidal components (i.e. amplitudes and phases). These are primarily used when applying high-pass, low-pass, and band-pass filters; these were performed in the preprocessing stage in EEGLAB and can be formulaically shown below:

$$\tau = y_i(n) = \sum_{k=0}^{m} b_k x_i(n-t) \tag{7}$$

Known as a Finite Impulse Response (FIR) filter, this is another form of a linear filter that sums of some coefficients multiplied by some components of the signal, x_i. This is a bit more flexible than a moving average because there is a variable coefficient, b_k, known as a kernel. While temporal filters are used to identify specific brain signals, spatial filters, more importantly CSP, are used to identify specific signals used to command the BCI controller. The next section elaborates on Common Spatial Patterns before continuing with the BCILAB offline training section.

Step 2 – Define a new approach: Common Spatial Patterns (CSP)

Common Spatial Patterns are used extensively in most BCI controllers and it requires the following assumptions: the known frequency band and time window, the band-passed signal is Gaussian in the time window, and there is a definite difference between the two classes of signal. In other words, this filter will look at an epoch, a defined time range from selected stimulation markers, and try and find a distinction between the simulation markers (i.e. left/right hand or left/right foot). This can be seen on a scatter plot where two electrodes are represented as different colors; a linear transform will be applied to the data to better distinguish two electrodes.

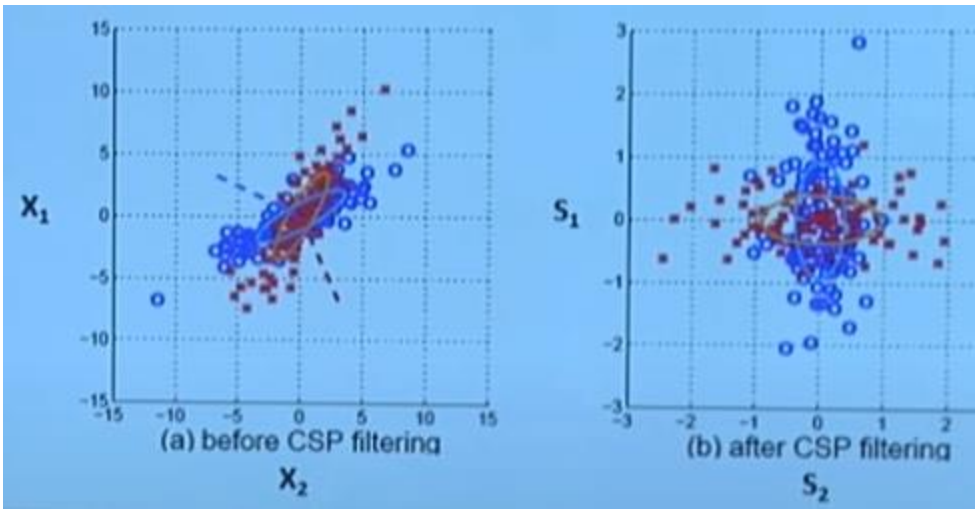


Figure 28: (Left) Raw samples of two electrodes C3 and C4. (Right) After spatial filtering is applied, the space is squished to form distinct variances. The red channel samples are affected horizontally and the blue channel is affect vertically [65].

There are three approaches to compute a CSP filter: as an optimization problem, a generalized eigenvalue problem, and a geometric approach.

The following information is provided when solving the CSP optimization problem: Equation (8) is the given set of t trial segments, Equation (9) is the per-trial covariance matrices, and Equation (10) is the per-class average covariance matrices.

$$X_t \in \mathbb{R}^{dxN} \tag{8}$$

$$\sum_t = X_t {X_t}^T \mathbb{R}^{dxd} \tag{9}$$

$$\sum^{(c)} = \langle \sum_t \rangle^c \tag{10}$$

The optimization approach is shown below:

$$W_c = \underset{W}{max}\ W^T \sum^{(c)} W\ s.t.\ W^T(\sum^{(-1)} + \sum^{(+1)})W = 1 \tag{11}$$

Because the signal is joint Guassian, the average covariance matrix of the signal in two conditions (-1) and (+1) can be calculated and completely characterized. Here, (-1) and (+1) can be related to a left or right hand motor imagery signal. In the equation above, this represents a one vs one class comparison, either left or right hand motor imagery signal.

Applying a transposed spatial filter before a covariance matrix and applying a spatial filter after it calculates the signal output after the spatial filter, and the variance of the signal. Ultimately, the W matrix will be optimized to create the maximum variance between the two signals. There is an extra constraint where the average covariance matrix, the sum of both classes, and apply a spatial filter, the answer will be 1. This means, both classes cannot exist; there can be a maximum of one class while having a minimum of the other.

Another way to solve the CSP filter is through a generalized eigenvalue problem. The following information is provided: the per-class average covariance matrices in Equation (10*)*, the simultaneous diagonalizer V of the average covariance of both classes in Equation (11*)* and (12*)*.

$$V^T \sum_{-1} V = D_{-1} \tag{12}$$

$$V^T \sum_{+1} V = D_{+1} \tag{13}$$

The general eigenvalue approach is shown below:

$$V^T \sum_{-1} V = D \wedge V^T (\sum_{-1} + \sum_{+1}) V = 1 \tag{14}$$

This is the most efficient way in which it jointly diagonalizes the covariance matrix and the average of the covariance matrices and outputs eigenvectors and eigenvalues, which end up being the spatial filters. Unlike the previous approach, this required some analysis during its proof and has some Lagrangian multipliers. Still, a built-in MATLAB function can calculate these values using one line of code.

One final way to implement a CSP filter is through a geometric approach. This approach can be broken down into three steps. Graphically, the distributions from both signals, and an average of the two, can be plotted see step 1 of Figure 29. Like the ICA method, a whitening transform matrix U is applied on the average covariance matrices of the two distribution. The principal components, matrix P, can be calculated in the red and green distributions create axes and become the blue vectors as well.

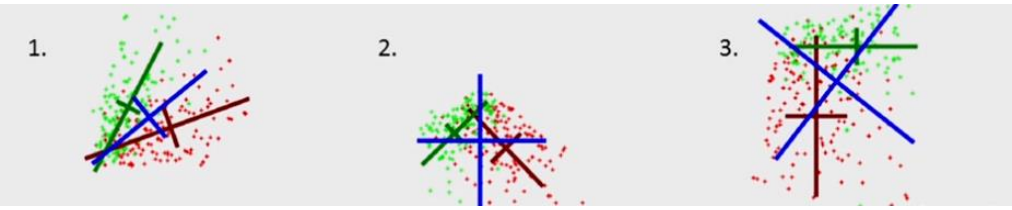


Figure 29: A graphical approach to applying a CSP filter to two electrode distributions of samples.

These blue vectors in step 2 become the new coordinate system; here, the data is rotated and scaled to show unit variance in all directions. In relation to the eigenvalue approach, this step is the visual form of diagonalizing the classes (see diagonal green and red distribution). By creating a scaled coordinate system based on the two classes, these blue axes cannot provide any quantitative information; in essence, it is more of a visual between the red or blue variance. The final step is to rotate the data once more with respect to the red or green coordinate system to directly quantify the values. As a result, these be summed up as whitening and rotation. Conducting the PCA allows the data to shift between distribution coordinate systems. This can be summed up in the following equations:

$$W = P^{-1}U \tag{15}$$

$$S = WX \tag{16}$$

where W is the spatial filter operation. It is obtained by first whitening using the U matrix and then transform by P^{-1}. Regardless of the which approach is taken, the result is seen in Figure 29, where multiple spatial filters are generated showing the maximum and minimum variances at different electrodes with different classes. The top and bottom halves show the maximum variances of the different classes. With the linear W matrix calculated, a classifier using a machine learning algorithm can now identify two or more unique tasks.

Step 2 – Define a new approach: BCILAB GUI interface

For this BCI controller, once the CSP filter is selected, a pop-window titled 'BCILAB Configure approach' is generated where the user specifies a resampling rate, frequency specifications of the filter, type of filter, the epoch time limit, number of CSP patterns, and type of machine learning algorithm. The 100 resampling rate and [6 8 28 32] frequency specifications are the default settings and have been shown to be the most effective in creating an offline training model. Under filter type, the user can select

minimum-phase, linear-phase, and zero-phase. These options affect the signal data in some way; for minimum phase, there is a slight delay, but the signal does suffer some distortion. For linear-phase, there is not signal distortion, but the signal is delayed. And finally, a zero-phase does not distort or delay the signal, but it cannot be used for online testing. Through trial and error, the minimum-phase has shown to be the optimal choice in creating a high performing model. The epoch time window is set so that the epochs are within timeline of the OpenViBE recording; per Figure 13, the trial examples BCILAB will collect have a time range where the red left/right arrow is briefly on the screen and then disappears. The number of CSP patterns chosen for the BCI controller is 3 and the machine learning function selected is both linear discriminant analysis and relevance vector machine. These algorithms were selected one at a time and each offline training model's performance was compared. The next step will cover machine learning in depth.

5.2.3.3 Step 3 – Define Machine Learning Algorithm Parameters

After the filter is chosen, the user will select a machine learning algorithm used to teach a classifier to identify the different classes in a brain recording. Similar to the number of filter options, BCILAB has an enormous selection to choose from. However, it should be noted that the selecting a filter confines to the number of algorithms that can be used.

Step 3 – Define Machine Learning Algorithm Parameters: Background

Recall that the difficulty in creating a universal BCI controller is because the brain signals vary from person to person. In addition, while the international 10-20 system helps approximate the ideal electrode location, the exact electrode placement will vary based on the headsize of the subject. And even if the locations were the same, it is not likely that the brain source creating these signals is in the same location. As a result, a BCI controller becomes extremely personalized to the user. While a BCI controller lacks flexibility once calibrated to a single user, one workaround is creating multiple profiles of filters and classifiers that can be uploaded for the appropriate user. Regardless of the user, selecting a machine learning algorithm is a key aspect in the controller's accuracy.

All machine learning algorithms have two functions in common: a training function and a prediction function. The training function first receives examples of signals tied to a specific marker. These are the cleaned data files obtained from EEGLAB in the preprocessing phase of offline training; in these files are repeated trials of both the imagined left/right hand or foot. Having multiples trials of a motor input trains the algorithm to look for similarities in the variation of identical labeled recordings, and recognize the difference with other inputs. Once the selected algorithm goes through the entire recording, a model is generated with parameters that will be used to classify online testing signals. The prediction function receives new data and uses the model from the training function to label incoming signals to a marker. Continuing with the example of the left hand motor imagery input, the predict function would recognize the signature waveform characteristics after referencing the model and output '769,'the OpenViBE numerical code for left-arrow. Another code will then be programmed to perform a routine of actions after seeing this text string. The data and labels/markers are matrices can be written like the following:

$$X \in \mathbb{R}^{NxF} \tag{17}$$

$$y \in \mathbb{R}^{NxD} \tag{18}$$

where N corresponds to the number of trials, F corresponds to the number of features, and D corresponds to the dimensionality of labels (typically one, a scalar). This form of trial-based machine learning falls under the category of supervised learning. Here, a prediction function is given a specific set of inputs and outputs (i.e. labels) and learns how to sort them. Other forms of machine learning are unsupervised learning and semi-supervised learning; under these categories, not all the examples provided will have a linked output. As a result, these algorithms will naturally require both more examples and longer computational times to create an accurate model.

The individual trials of selected training data may have too many points and/or not enough constraints for a machine learning algorithm; as a result, paired with these learning algorithms are feature extraction methods. This will lower the complexity of the design by highlighting key features in the data. The next sub-sections talk about the two machine learning methods chosen for the development of the BCI controller.

Step 3 – Define Machine Learning Algorithm Parameters: Linear Discriminate Analysis (LDA)

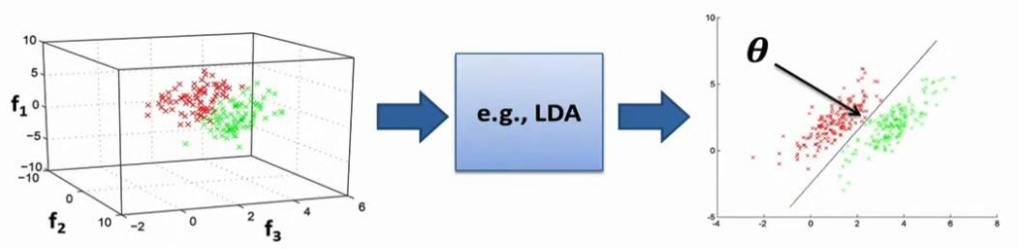


Figure 30: A visual representation of Linear Discriminate Analysis (LDA) where the a plane is created to separate the two classes [66].

Linear discriminant analysis (LDA) is one of the widest and frequently used machine learning algorithms for BCI controllers due to its simplicity. Graphically, this method is creating an imaginary plane, or hyperplane, that separates the classes of data. This hyperplane can be treated as a vector that defines its orientation in space and its size is determined by the dimensions of the space. This can be viewed formulaically (shown below):

$$\mu_i = \frac{1}{|c_i|}\sum_{k\epsilon c_i} x_k \ , \qquad \sum_i = \sum_{k\in c_i}(x_k - \mu_i)(x_k - \mu_i)^T \tag{19}$$

$$\theta = (\sum_1 + \sum_2)^{-1}(\mu_2 - \mu_1), \quad b = -\theta^T(\mu_1 + \mu_2)/2 \tag{20}$$

where $x_k$is in vector form and the data is split up according to the number of classes; in this example, because there are two classes, i=2. Equation (19) calculates the average of each class, μ_i; paired with Figure 30, it would be the average of the red and green points. Knowing the class average, the covariance matrix of each class can be calculated, which represents the spread of class data sets. The orthogonal plane θ is the mean difference of the two classes and is rescaled based on the distribution of each data cloud, which is shown as the inverse averaged covariance matrix. Variable b represents the shift of the hyperplane in reference to the origin and coordinate system. This is the midpoint between the two data clouds, hence the average of the class averages, and is projected onto the hyperplane θ. Note, this example is for the case of two classes with equal points of data; nevertheless, the general application can be extended to multiple classes.

$$y = \boldsymbol{\theta x} + b \tag{21}$$

$$y = sign(\boldsymbol{\theta x} + b) \tag{22}$$

When LDA goes through the training function, it creates a linear relationship with θ and b shown in Equation (21). This represents a statistical equation used to discriminate between two classes; here, the orthogonal plane matrix is multiplied by a trial and adjusted by an offset vector. The addition of the sign function outputs the classification as either (+1) or (-1).

Using LDA comes with some assumptions and understanding; as mentioned previously, the total data must be Gaussian distributed. As a result, this means that the data for each class must be Gaussian distributed as well. In addition, the size and shape of the data distribution must be identical of all classes; if one class has a larger distribution than the other, the plane will warp into a curve shape, losing the linear relationship. One final thing to note is the need for consistent class data. LDA is sensitive to outliers and if there are too few trials the covariance will be affected. The parameters chosen to run LDA machine learning is seen in Figure 31.

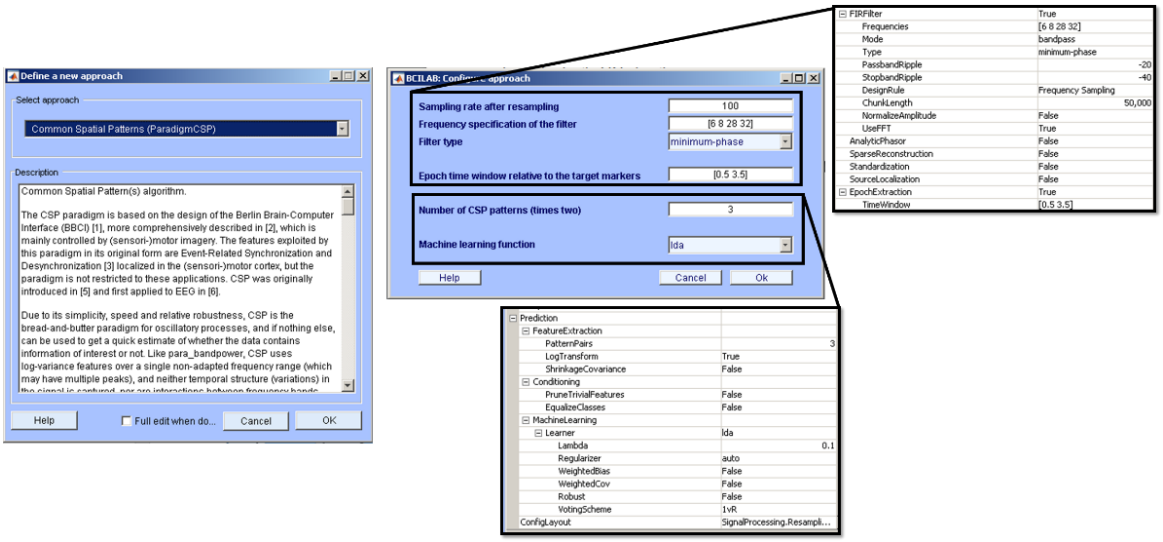


Figure 31: The parameters chosen on BCILAB to apply a CSP Filter and LDA machine learning method to train a model.

Step 3 – Define Machine Learning Algorithm Parameters: Relevance Vector Machine (RVM)

Compared to the linear discriminate analysis method, the relevance vector machine (RVM) accounts for outliers between the class data sets by creating a soft margin or threshold that is based off other data points, not just the data class average. Relevance vector machines are similar to support vector machines (SVM), which are also used extensively in EEG classification, but create predictions that are probabilistic. The math function describing an SVM is shown below:

$$y(x) = \sum_{n=1}^{N} w_n K(x, x_n) + w_0 \tag{23}$$

where w_n are the model weights and K(.,.) is a kernel function. A key feature in creating an accurate SVM classifier is the use of kernels, which increase the order of the data set. These kernels can stem from the most basic application, linear, to an empirical form, radial basis function, to help minimize the error and maximize the margin between two classes. But like LDA, the goal is to get the prediction function to output the probability of the classification, or p(t|**x**), as a way to recognize uncertainty in the method; this is something that the SVM method cannot provide and was realized during the experimental online testing phase. Other pitfalls of the SVM method are: although rare, SVMs will apply many kernels that can increase computational time, necessary to estimate the error/margin trade-off parameters defined as '*C*' and '$\in$' using a cross-validation procedure, and the kernel function must satisfy Mercer's condition. RVM is a probabilistic sparse kernel model similar to the SVM method. It does differ by adopting a Bayesian approach to learning where it establishes a set of hyperparameters associated to each element in the weight matrix based on the data. This creates what is known as sparsity in a model, which is good at preventing overfitting and lead to faster computations. Finally, the weights calculated by RVM are not associated with trials close to the decision boundary, but are generalized among the group of trials; in other words, the group of trials all share a similar trait that is 'relevant' to each other. RVM classification can be written as the following:

$$\sigma(y) = 1/(1 + e^{-y}) \quad , \tag{24}$$

$$P(t|w) = \prod_{n=1}^{N} \sigma\{y(x_n)\}^{t_n}[1 - \sigma\{y(x_n)\}]^{1-t_n} \tag{25}$$

This linear model is created by applying a logistic sigmoid function. While the weights cannot be solved through integration, an iterative procedure can be performed to obtain these values through MATLAB. The results comparing the SVM to RVM methods is shown in Figure 32. The parameters chosen to run RVM machine learning is seen in Figure 33.

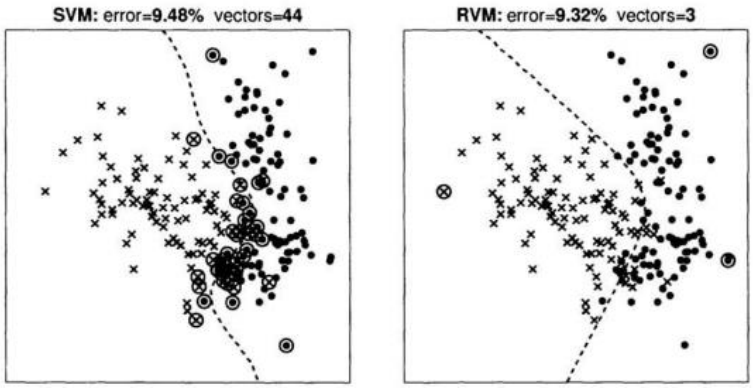


Figure 32: A comparison show a comparison between a support vector and relevance vector classifier using synthetic data. The decision boundary is represented as a dashed line and the encircled points show the vectors used to train the classifier. Note the error percentages as well [67].

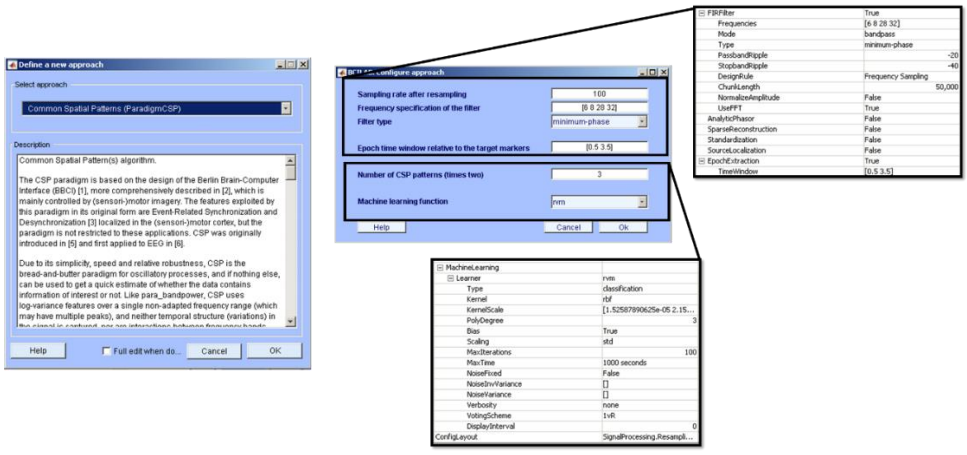


Figure 33: The parameters chosen on BCILAB to apply a CSP Filter and RVM machine learning method to train a model.

Step 3 – Define Machine Learning Algorithm Parameters: One-vs-One and One-vs-Rest

In the examples explaining the chosen machine learning algorithms for the BCI controller, two classes were used. The development of most machine learning methods are initially developed for binary classification and then later modified to include multi-class classification. Some have attempted to solve this problem applying multi-class data to a set of binary classifications; an example of this was the BCI controller used to control a smart wheel chair in previous CSUN work [15]. There are two methods that allow a multi-class classifier: one-vs-rest (OvR) and one-vs-one (OvO). The difference between the two methods are shown in Figure 34.

Figure 34: A comparison the different multi-class methods: one-vs-res (left)t and one-vs-one (right) [68].

The OvR compares the likelihood of one classifier to the rest of the classifiers. Following the example of three classifiers each represented by a different color, blue, red, and green, three classifiers are created. In the OvO method, a binary classifier is made for every possible combination of color. As a result, there would be six classifiers total compared to the three in the OvR method. This method would be used for the SVM method because of its rigid and binary structure. For this project, both machine learning methods, used the OvR method in the development of the BCI controller.

Step 3 – Define Machine Learning Algorithm Parameters: BCILAB GUI interface

Once a machine learning method is selected, a pop-up window titled ‘calibrate a model’ is generated where the user will confirm the algorithm approach and data recording file. Next, the user will specify which target markers to analyze the data. In this case, the OpenViBE code strings ‘769’ and ‘770’ will be inputted to create a binary classifier comparing the signals of the left and right hands or feet. When creating a multi-class classifier, this text box will have additional codes relating to the stimulation marker used

in the recording; note, the typed order of the codes corresponds to the class order in the command stream. The final two sections are parameter search and performance estimation. Here, the user can choose from a number of performance metrics and the number of cross-validations the training function will go through when generating the model. Both of these are explained in the next section.

5.2.3.4 Step 4 – Evaluate the Performance

After the appropriate filter and machine learning algorithm finish processing all the multi-class training examples and creates a model, BCILAB will output a performance diagnostic showing the overall accuracy of the model. If this is a binary classifier model, the performance results show the true positive rate, true negative rate, false positive rate, false negative rate, and error rate; note, all of these are read as percentage values. Some studies have shown these values visually as confusion matrices. If this is a multi-class classifier model, the performance diagnostics is reduced to just the error rate. Regardless of the number of classes, these diagnostics are tabulated using five cross-validation folds and five spacing around test trials as inputs. During cross validation, a loss/performance metric is chosen automatically from a list (see Table 3). The advantage to cross-validation is the ability to split a single recording up into segments where the machine learning algorithm will calculate initial values, and then those values are iterated and improved with the next segments until the entire recording is done; in the end, the best parameters are selected for the model. The performance estimates will be used to evaluate the performance between both machine learning algorithms. It is also important to note that since these are estimates, the true accuracy and performance requires another set of data, or in this case, real-time online testing to verify the model is classifying well.

Table 3: (Left) A list of loss/performance metrics used to evaluate the accuracy of a model after the training function. (Right) The mathematic representation for two possible metrics, Mean-Square Error, and Mis-Classification Rate.

No.	Loss/Performance Metric
1	Kullback-Leibler divergence
2	Negative log-likelihood (NLL)
3	Mis-classification rate (MCR)
4	Mean absolute error (MAE)
5	Mean square error (MSE)
6	Standardized mean square error (SMSE)
7	Maximum absolute error
8	Root mean square error (RMSE)
9	Directed mean bias
10	Median square error
11	Area under ROC (AUC)
12	Conditional Entropy
13	Cross-entropy
14	Negative F-Score

Mean-Square Error:

$$L_{MSE}(p,t) = \frac{1}{n}\sum_{k}(p_k - t_k)^2$$

Mis-Classification Rate:

$$L_{MCR}(p,t) = \frac{1}{n}\sum_{k}\begin{cases}1, p_k \neq t_k \\ 0, p_k = t_k\end{cases}$$

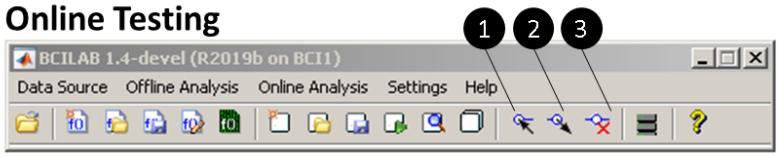
5.2.3.5 Step 5 – Save Model

To conclude the offline training phase of the BCI controller, the final step is to save the model. This is done by saving the workspace file that will be recognized by BCILAB. For this BCI controller, the final workspace contained binary classifying models for the hands and feet for each subject. In addition, another set of models were saved containing a five-class controller of the hands and feet for each subject. And finally, a single model was saved containing a six-class controller, the final version for the mobile robotic arm BCI controller.

5.2.4 BCILAB (MATLAB Plug-in): Online Testing – Streaming

Once the offline classifier model is trained, the BCI controller moves into the next phase, online training. As mentioned before, BCILAB is the binding software that links the two phases together. The second half of the BCILAB GUI is focused exclusively on online testing. Referencing Figure 35, the online testing can be broken down into three steps.

Online Testing



No.	Offline Training Steps	Menu in EEGLAB
1	Read input from (online)	**Online Analysis**>Read input from…
2	Write output to (online)	**Online Analysis**>Write output to…
3	Clear online process	**Offline Analysis**>Clear all online processing

Figure 35: (Top) Main BCILAB GUI with balloon callouts highlighting the online training procedure. (Bottom) A Table describing the step-by-step online testing procedure.

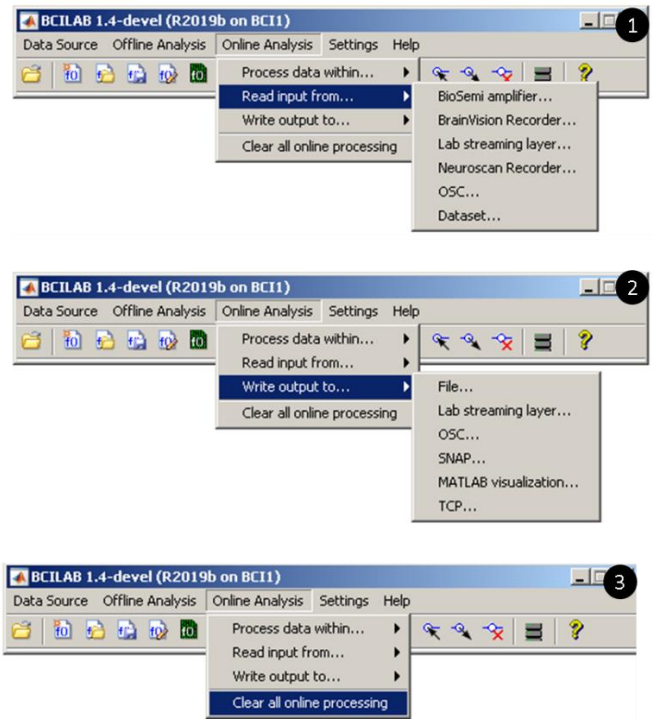


Figure 36: (Left) Navigating the BCILAB for online testing. Numbers correspond to the steps outlined in the online testing procedure.

5.2.4.1 Step 1 – Read input from online streaming source

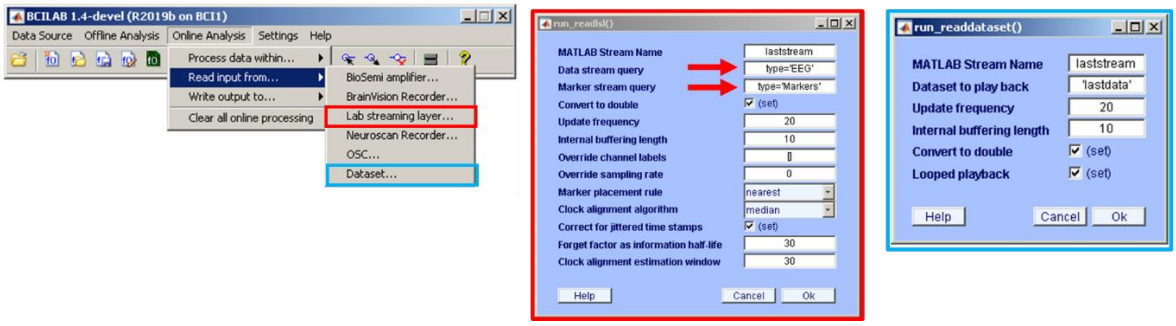


Figure 37: Sub-windows for the two selected methods for BCILAB to read incoming streamed EEG signals are shown. The red boxed highlight shows the incoming signal coming from lab streaming layer. The blue boxed highlight shows the option for BCILAB to loop a dataset as a simulated EEG signal stream.

BCILAB provides multiple options to read incoming EEG signal streams; most of these options correspond to different EEG acquisition software outside of MATLAB (i.e. Brainvision Recorder, Neuroscan Recorder, and lab streaming layer). Out of the few listed, lab streaming layer was chosen based on the software accessibility and compatibility with the Emotiv Epoc headset. The two red arrows in the middle box text boxes, data and marker stream query, help BCILAB locate the online EEG stream; BCILAB can either locate these by the type of signal (i.e. EEG and Markers), or by name. Recall in Figure 9 that the OpenViBE acquisition software has textboxes asking the LSL_MarkerStreamName and LSL_SignalStreamName. The names specified in this software would be typed into the BCILAB textboxes to confirm the link. In addition, there is another option for BCILAB to simulate an online incoming stream by looping one of the recorded datasets (see blue highlight in Figure 37). This alternate option was used extensively as a pathfinder to establish a connection between the controller and the virtual mobile robotic arm.

5.2.4.2 Step 2 – Write an output command stream

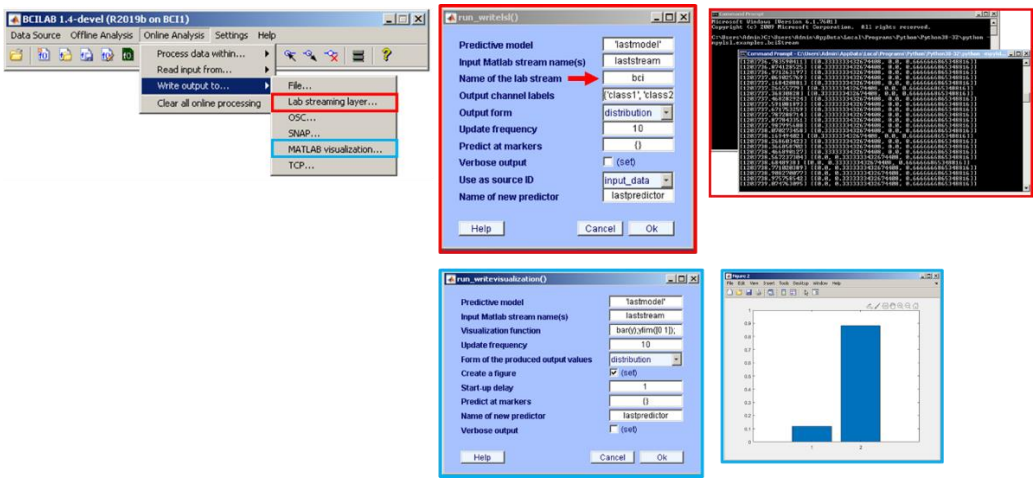


Figure 38: Sub-windows for the two selected methods for BCILAB to write outgoing command streams are shown. The red boxed highlight shows a streaming command matrix from BCILAB showing on python. The blue boxed highlight shows an option for BCILAB to generate a bar graph showing class '1' or '2'.

Once BCILAB is connected to an incoming streamed EEG signal, the user can then set up the outgoing command stream. Note the command stream is a single row matrix that is updated continually at a desired frequency (default is 10 hz). Similar to the previous step, the MATLAB plug-in offers multiple options to send the command stream to outside software (i.e. lab streaming layer, OSC, and SNAP). Continuing with software continuity and simplicity, LSL was chosen. Similar to the OpenViBE acquisition program, the name of the command stream will be used for other programs to detect (default is bci). There is an alternate option for BCILAB to generate a MATLAB visualization bar graph where the number of classes is generated on the x-axis and the y-axis is the probability percentage values generated from the classifier. Note, the numbers on the x-axis are associated to the order specified in the offline training phase (step 3, define machine learning algorithm). For example, continuing with the example of inputting '769' and '770' as target markers in step 3, the class '1' is associate with the user's imagined left hand movement, and class '2' is associated with the user's imagined right hand movement. The visualization option was used extensively to build multi-class classifiers and understand how probability is affected with more classes.

5.2.4.3 Step 3 – Clear online process

When the user wants to stop the online testing, the "clear all online processing" function is selected and BCILAB will properly terminate all established connections of the input EEG and command output streams. Connections can be reestablished once again if the user starts at the beginning of online testing procedure. In online testing, BCILAB heavily uses LSL to read the input EEG stream and to output the classifier's command stream. The next section will discuss the lab streaming layer program in more depth.

5.2.5 Lab streaming layer (LSL): Online Testing – Stream Acquisition System

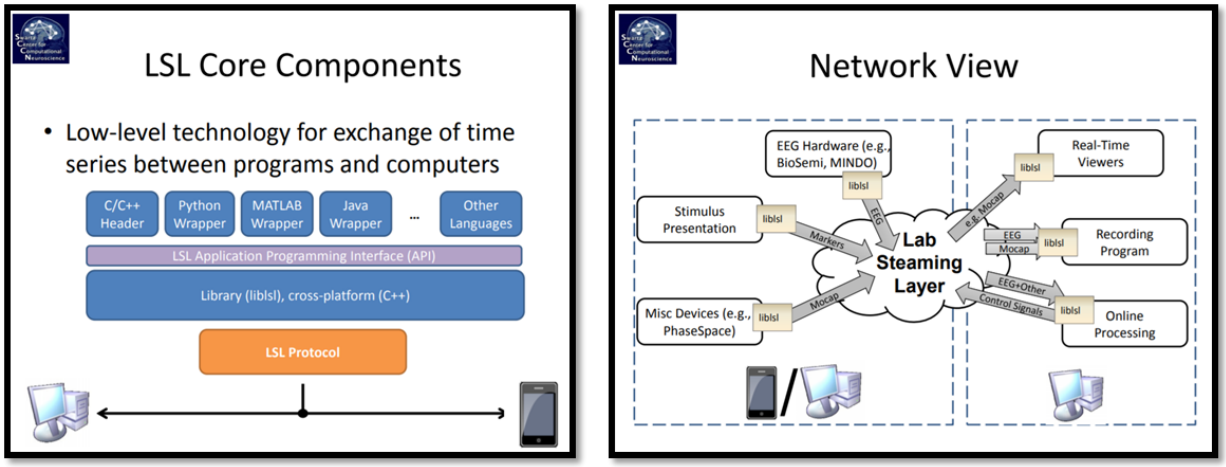


Figure 39: (Left) Core components that make up lab streaming layer (LSL). Note the flexibility to communicate to multiple programs and languages. (Right) A high-level network view of different programs (i.e. stimulus presentation) and devices that use liblsl to send data out or pull data in from LSL. [69].

Lab streaming layer is an open source software developed by the SCCN to function as a multi-signal acquisition system. This system's primary purpose is form a collection of unified measurement time series where compatible programs and computers can retrieve or add to the collection [70]. In the built-in code are libraries that do all the transactions between the different platforms and wrappers that allow communication between multiple languages (i.e. C/C++, MATLAB, java, and python). In addition to the core components shown in Figure *39*, LSL also has downloadable acquisition programs for EEG, eye tracking, human interfaces, motion capture, multimedia, generic viewers and recorder, and

example programs. The Emotiv headset is listed as one of the compatible pieces of hardware, and BCILAB as compatible software. Referencing the network view diagram, the Emotiv headset is categorized as 'EEG Hardware' and will only send data streams. Meanwhile BCILAB is classified as 'Online Processing' and will both receive and send data. And finally, a 'real-time viewer' in python will be used to receive the command stream from LSL, generated from BCILAB. There is no limit to how many devices can read streams of data; as a result, multiple computers controlling a smart wheel chair and mobile robot arm can all access the same command stream at once.

5.2.5.1 Lab streaming layer (LSL): Online Testing – Install Software

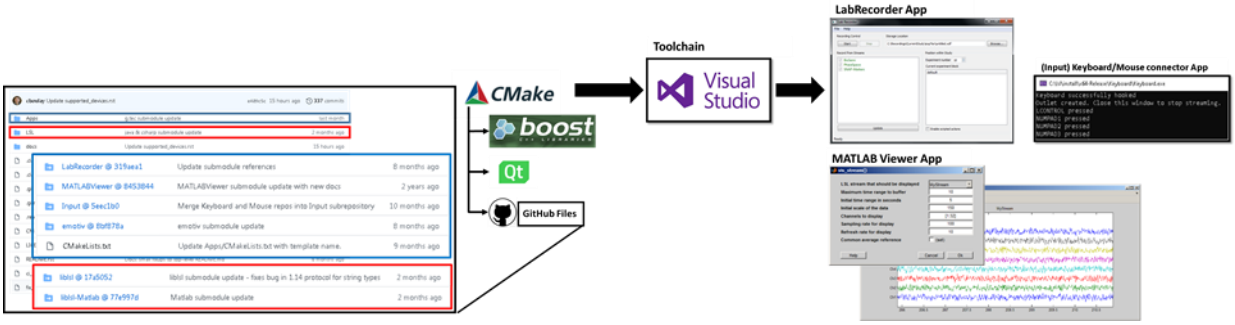


Figure 40: A flow diagram outlining the process of building lab streaming layer (LSL).

Unlike the previous software used for this BCI controller development, Lab streaming layer's core library, liblsl, cannot be downloaded as a whole; instead, it needs to be built using C++ language and boost libraries that are readily available online. Fortunately, there is an online manual that guides the building process for Windows, Mac OS X, and Linux platforms [71]. The directory hierarchy is available for download on the opensource software website, GitHub. There the user will download the folder structure, interested applications like LabRecorder, MATLAB Viewer, keyboard and mouse input markers, emotive headset, and liblsl wrapper languages like liblsl-Matlab. Each of these folders have additional subfolders and files that need to be downloaded separately and placed inside the main folder hierarchy.

Next, the user will download the appropriate toolchain. For windows platforms, the software is Microsoft Visual Studio (the most recent version will work). Toolchains like Microsoft Visual Studio physically builds the LSL program [72]. A custom

installation is used to ensure under the C++ programming language, the windows software development kit (SDK) is included. Note, these toolboxes need a code command program the order of building applications and programs. The LSL creators resorted to using CMake to take on this responsibility. CMake is an open-source software that will compile the LSL software using the files downloaded from GitHub [73]. Most of the applications and liblsl wrapper languages require additional C++ libraries, Boost and Qt [74] [75]. After these libraries are downloaded, the user launches CMake and sets two file paths specified in the manual. The user will then configure the location where the build program will be placed. Afterwards, the user selects which applications that will be included in the LSL build. CMake generates the build files and sends them to Visual Studio. From the CMake window, the user will launch Microsoft Visual Studio and select the LabStreamingLayer file for release and install. Once LSL is properly built, the user can open applications like LabRecorder and track streams of data (see Figure 40).

5.2.5.2 Lab Streaming Layer (LSL): Online Testing – Python: Receiving Data from LSL

With lab streaming layer wrappers, multiple languages can retrieve data streams from the multi-signal acquisition software. For the development of the BCI mobile robotic arm controller, python is the primary language used to program the robot base and arm. The liblsl-Python wrapper is the Python interface to LSL; to have Python read time series data from LSL, a direct installation of pylsl is required. Fortunately, the built-in Python add-on feature, pip, can search for pylsl online and begin the download [76]. Once installed, examples and scripts that want to be launched are placed in the subfolder "examples" under pylsl. The following command line can be used to run the script: "python –mpylsl.examples.XXXXX". The last part of this subsection goes over the script needed to read a data stream. Note, as with all python scripts, python must be launched via the platform's command line; then, the script is mapped through python and finally the command matrix is printed on the screen (see last red box highlight in Figure 38).

A script in python is needed to receive BCILAB's command matrix. First, python would need to discover the data stream either by type or name. Per BCILAB, the output data stream's name is 'bci' by default, and the 'type' is classified as 'MentalState'. Once

the stream is identified, an inlet variable is created and the function 'StreamInlet' is applied. Additional information like the timestamp can be included with the command matrix through extra lines of code. The script used to discover the data stream by the name or type can be referenced in Appendix A.

5.2.6 A Comparison to Past BCI Architecture

The BCI controller is split up into two phases: 'offline testing' and 'online training' where different software is applied to clean the data, create a classifying model, and retrieve/output streams of data. For the offline testing phase, OpenViBE, EEGLAB, and BCILAB were used sequentially. OpenViBE recorded a subject's brain response to visual stimulation. With EEGLAB, the user uploads the OpenViBE recording(s) to preprocess the data by applying filters and ICA. The user can also study the subject's brainwaves to observe which electrodes were most active during the experiment by epoching the data; however, to create a classifier model, the recording must remain continuous. Finally, BCILAB applies a filter and machine learning method to create a model that tells the difference between two or multiple classes. For the online testing phase, only LSL and BCILAB are used. While offline training primarily deals with recordings, online testing requires the headset on a subject to be connected to the computer via some signal acquisition software, and sends data to LSL as an input stream. BCILAB's built-in LSL compatibility allows MATLAB to read the EEG signal input stream and compare it to the generated offline training model. The command matrix stream generated by BCILAB is sent back to LSL. With the liblsl-Python interface, the command matrix stream can be retrieved by Python and used as instruction to command a BCI mobile robotic arm.

In designing the BCI mobile robotic arm controller, the intention was to improve on the previous controller design by creating a single multi-class classifier. Previous controller design had no way to thoroughly clean the signals and review recordings. In addition, OpenViBE was used as the main program to cover both BCI controller phases. While there is a convenience of having an all-in-one purpose program, OpenViBE has limitations that can reduce the accuracy and response of the BCI controller if it is solely used. Some of these limitations experienced are: 1) only basic preprocessing of low/high

pass band filters (no removal of artifacts and non-brain based activities), 2) inability to compile the user's brainwave responses to a specific stimulation and analyze which electrodes are active, and 3) limited signal classifiers algorithms (i.e linear discriminant analysis (LDA), preliminary multi-layer perceptrons, and preliminary support vector machine (SVM)). The proposed BCI controller suffers from none of the limitations listed for OpenViBE. Incorporating multiple software will improve performance and accuracy for the proposed BCI controller.

Chapter 6 – Testing and Simulation

In this chapter, the test objectives, procedures, and experiments conducted to develop the BCI mobile robotic arm is discussed. Creating a reliable BCI controller is a combination of two aspects: observing the human brain and designing the controller accordingly. While the hardware, software, and fundamental principles of machine learning techniques provides a solid foundation for controller design, it is important to remember that the human brain is a complex nerve center that cannot be predicted easily. For instance, a human being's mental, emotional, and physical state affects his or her brain signals. Understanding the biological impact of a BCI controller will gain better insight on the controller's variation in performance.

6.1 Testing Objectives

The testing objectives for the development of the BCI controller are, (a) observe the active electrode locations based on the subjects' motor imagery signals, (b) evaluate the BCI controller's performance when combining motor imagery recordings, and (c) finally, demonstrate the functionality of the BCI controller by applying it to the virtual mobile robotic arm. These objectives are done sequentially so that the results from one builds the foundation for the other. The first two objectives individually highlight the process of the BCI's offline training; and the final objective showcases the implementation of the controller and online testing. The next subsections discuss the criteria established for each objective.

6.1.1 Objective 1 – Active Electrode Locations Criteria

Recall the brainwaves being utilized for this controller are the left/right hands and feet motor imagery signals, and jaw clench. With the exception of the jaw clench, the user is supposed to imagine squeezing the hand, or tapping the foot. One way to accurately verify the active electrode locations is to see if the brain signals created from the physical and imagined action are the same. A set of experiments were designed to have the subject

first perform the physical action, followed by the imagine action; multiple subjects participated in these experiments to confirm this criterion is repeatable.

6.1.2 Objective 2 – BCI Controller Performance Criteria

A BCI controller's performance is based on a number of variables: the offline training data, the number of classes the controller needs to identify, and the machine learning method. As a result, the second objective is to understand the impact these variables contribute to the overall controller performance. Utilizing all the subjects' imagined action recordings from the previous objective, a two, three, five, and six-class models were generated. Each binary and multi-class model was constructed from LDA and RVM machine learning methods. BCILAB's performance evaluation of each model was recorded to observe if there is a difference between a subjects' trials, if more classes lower the overall controller accuracy, and which machine learning method is better suited for multi-class motor imagery controllers. In addition, the goal is to also down select the best motor imagery data in order to create the optimal multi-class controller for each subject.

6.1.3 Objective 3 – BCI Controller Implementation Criteria

To verify everything works as intended, the BCI Controller needs to be integrated to a robotic device and run through all possible commands. For this demonstration, the BCI controller was connected to the virtual mobile robotic arm. The user shall first operate the robotic base by using the imagined motor imagery signals to move forward, backward, left, and right. Next, the user will have to operate the robotic arm by jaw clenching to switch robot sub-assemblies. Then the user will use the same motor imagery signals to guide a screen cursor over a desired item to pick up. Ultimately, the user will have to navigate, pick up and drop off items to verify the controller works. The next section discusses the test setup and general background of the selected subjects.

6.2 Test setup

6.2.1 Test Subjects

Three healthy male subjects, ages 22 through 27, participated in the BCI controller development testing with consent. All subjects had normal or corrected-to-normal vision. One thing worth noting is the difference in dominant hand. Two of the three subjects, now labeled S01 and S02, are right hand dominant while the remaining subject, S03, is left hand dominant. All subjects had no prior experience with BCI systems. As a result, the purpose of the testing and procedures were explained in full transparency before preparing for the EEG recording.

6.2.2 Test Environment

The series of tests were performed in a normal office room setting. Subjects were seated in a comfortable chair in front of a computer monitor while wearing the modified Emotiv Epoch headset. A test coordinator was present at all times to monitor the headset's signal strength connections, the subject's brainwaves, and the subject's focus on the testing. If the signal connection weakened, the test coordinator followed a troubleshooting procedure. The first step was to apply more contact gel between the flagged electrode and the subject's scalp. Using a disposable cotton swab or applying a gentle pressure to the headset to work in the gel typically restored signal strength. If the additional application of gel did not solve the problem, the coordinator would then check the pin connection between the electrode and the Bluetooth transmitter box. If all the electrodes suddenly display poor connection, it typically meant that the CMS and DRL reference electrodes had poor connection. Note, signal strength is also heavily dependent on the user's hair and cleanliness; for example, a subject with thick greasy hair would require a lot of contact gel to establish a connection between the electrode and scalp. To ensure the subject stays engaged, the test coordinator would audibly call how many stimulation markers were left in a test set. Between tests, the test coordinator offered the subject a break. Once all the testing was completed and saved, the headset was removed from the subject and the test coordinator provided a paper towel for the subject to wipe off any excess contact gel in his

hair. The electrodes were then inspected individually to remove excess contact gel. If there was a buildup of dry contact gel, the electrodes were soaked in a distilled water bath. The next section describes the test procedures for all three objectives.

6.3 Testing procedures

6.3.1 Objective 1 – Active Electrode Locations, Test Procedure

The first objective testing requires both hardware, the Emotiv Epoch headset, and software, the OpenViBE acquisition and designer. Figure 41 shows the amount of sessions each subject underwent as well as the amount of trials collected for each unique motor imagery signal. Each subject was required to do five thirty minute sessions. Each session was broken up into four tests: two physical action and two imagined action tests. The testing order in Figure 41 was intentional so that the physical action test was always done prior to the imagined action test. While the subject has the freedom to choose either the hands or feet to start, he must complete both the set of physical and imagined action tests before moving on to the next pair. Each test was approximately six minutes long and the subject had a two-minute break in between each test. The six-minute test can be broken down into two sets of 11 second trials. The timeline of each trial can be seen visually on the computer monitor in front of the subject. A green cross appears on the screen for two seconds. Next, an audible beep is emitted from the computer warning the subject that the stimulation marker, a left or right red arrow, is about the appear. The stimulation marker is shown briefly and disappears, and the subject is supposed to perform the action until the next trial starts with the appearance of the green cross. Note, the appearance and disappearance of the symbols at the specified time interval also have OpenViBE stimulation codes.

For the physical hands test, the subjects were given two tennis balls to squeeze in each hand. When the randomized directional arrow appeared, the subject squeezed the corresponding hand; for example, if a left arrow flashed on the screen, the user would repeatedly squeeze the tennis ball in the left hand until the green cross reappeared on the monitor. While all of this was happening, the EEG signals responding to the unique

stimulation markers were recorded. After collecting fifteen trials of both the left and right physical hand squeezes, the test coordinator took the tennis balls away during the subject's break. When the subject was ready to continue, the test was repeated, but the subject now had to imagine squeezing his hands according to the directional arrow shown. Fifteen trials of each imagined hand squeeze were collected, completing half of the session. The same procedure was done with the subject's feet. For the physical action test, the subject gently tapped his left or right foot to the ground according to the directional arrow. It is important to note that the EEG headset is sensitive enough to record a physical response to a forceful foot tap, which will drown out the EEG signal. One work around was to ask subjects to take off their shoes before beginning the foot testing segment; a subject's socks dampened the physical response to foot tapping. The subject then repeated the test one final time and imagined tapping his left and right foot accordingly. All four tests were repeated five times to collect a total of 75 trials for each of the eight motor imagery signals (i.e. the physical and imagined, left and right, hand and foot EEG signals).

All the test recordings for each subject were uploaded onto EEGLAB and preprocessed as described earlier. With the sheer number of recordings to review, the MATLAB plug-in offers a simple group study function to view averaged ERPs, called Simple ERP Study (see Figure 42). Once selected, a pop-up window appears and asks for the number of conditions and subjects. While EEGLAB creators initially intended this function to cover a general study with higher number of subjects, it can be modified for this subject. The 'number of conditions' and 'number of subjects' text boxes were relabeled as the 'number of motor imagery signals' and the 'number of trials recorded'. Each subject's hands and feet were analyzed separately. As a result, the 'number of motor imagery signals' is four (i.e. left hand (LH) no squeeze, LH squeeze, right hand (RH) no squeeze, and RH squeeze); and the value for 'number of trials recorded' depends on the number of pre-processed recording not rejected. The final pop-up box visually shows the organization of the hands motor imagery example. As a result, the same approach was used for each subject's feet. The results for each subject's hands and feet channel ERPs are discussed in the Results and Analysis Chapter.

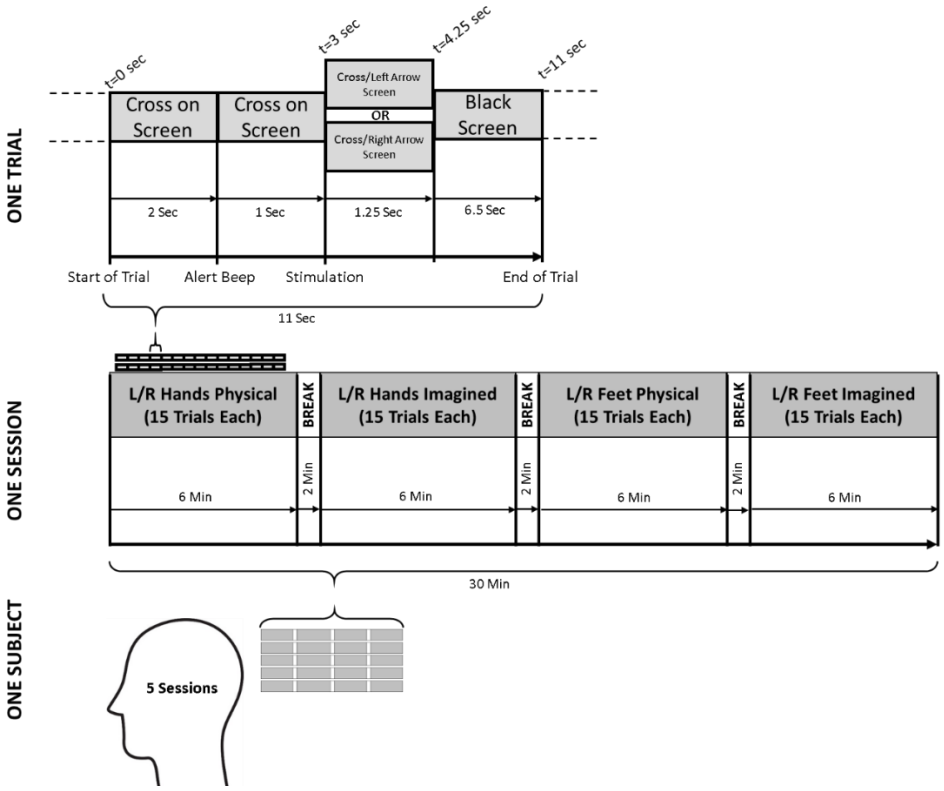


Figure 41: A visual breakdown of the amount of testing each subject underwent. Three subjects were tested. Each subject had five sessions; each session is composed of four tests recording specific motor imagery signals. Each test contains 15 trials of both the left and right hand or foot.

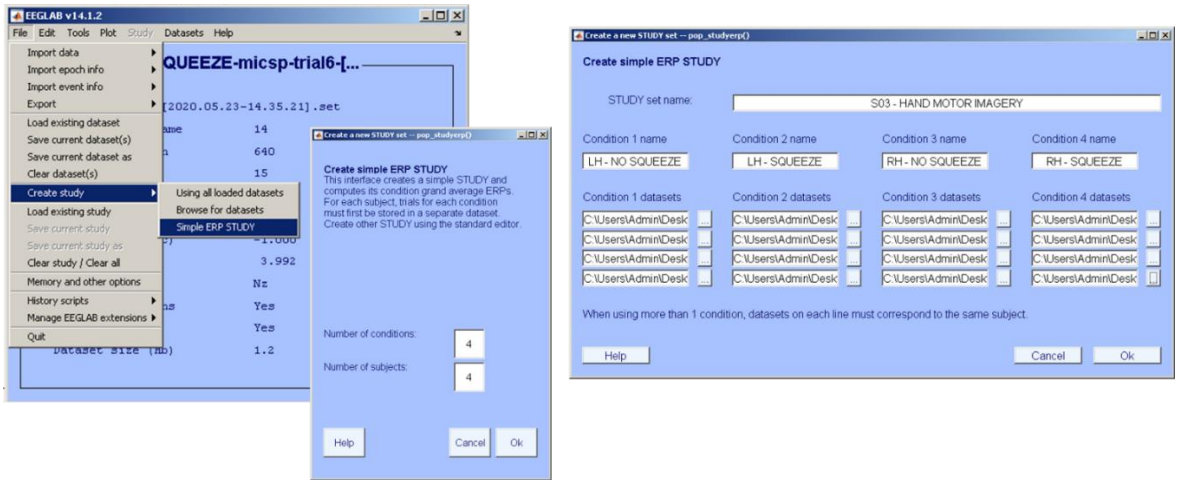


Figure 42: (Left) Navigation to find the simple ERP study. A pop-up window appears asking for the number of conditions and subjects. (Right) A final pop-up window where the pre-processed recordings are uploaded.

6.3.2 Objective 2 – BCI Controller Performance, Test Procedure

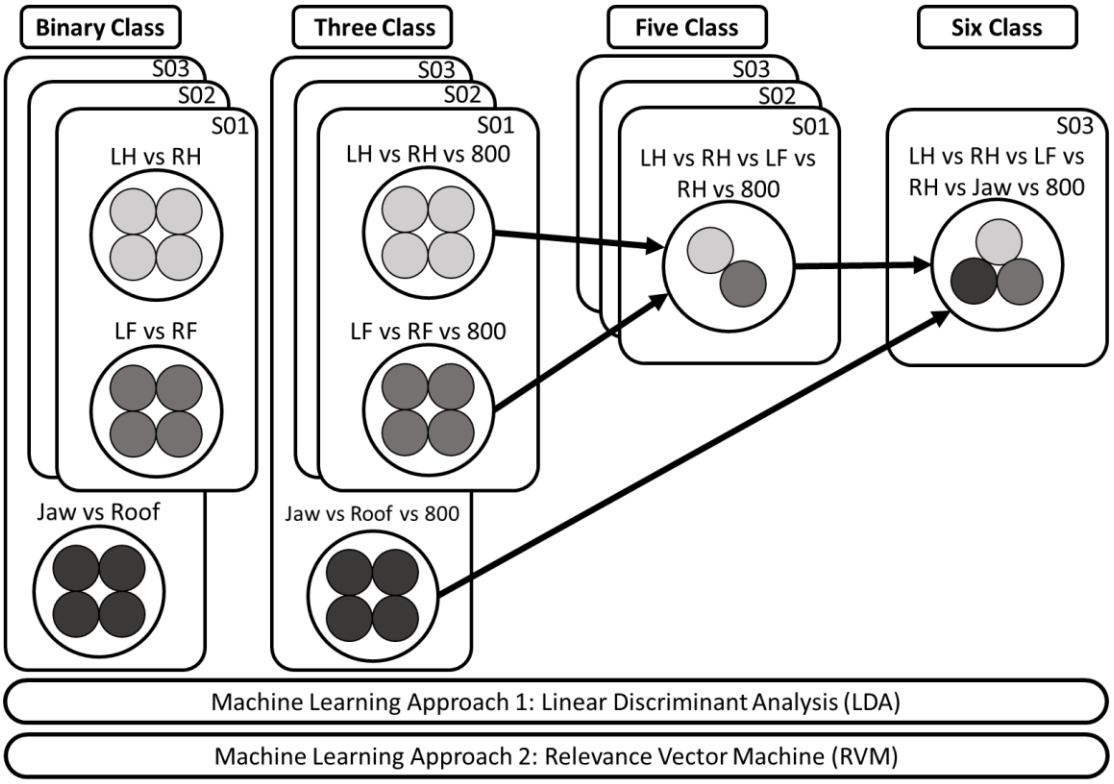


Figure 43: A visual breakdown of the different class models and machine learning methods.

With all imagined motor imagery signals recorded, the BCILAB software is launched to start analyzing the controller aspect of BCI development. Since all tests revolve around motor imagery signals, the common spatial pattern approach (CSP) was selected. Through trial and error, additional CSP parameters were adjusted to increase the classifier performance and accuracy. The first parameter used on the preprocessed EEG recordings was resampling; the sampling rate was reduced to 100 Hz, the filter length to 10 (default value), and stopband weight is 1 (default value). The filter length determines both the quality and the delay of the resampling and the stopband weight is the relative weight of the stop-band. The next parameter was the incorporation of ICA. BCILAB is capable of running ICA on raw EEG recordings with different variants (i.e. infomax, beamica, rica, etc.). However, unlike EEGLAB, BCILAB has strict data prerequisites to run the ICA successfully. Fortunately, since ICA was already applied to the data, BCILAB can carry over the calculated weights. Still the parameters chosen for BCILAB's ICA

during development were the variant infomax (similar to EEGLAB), with a max of 512 iterations, an extended-ICA sign estimation for 3 training blocks, applied sphering, applied 60 degrees for the anneal change, and performed a bias adjustment. Under the ICA application, there is also a data cleaning function where any flat lines, channels with uncorrelated signals, dropouts, bursts, and time windows with excessive signal power were removed. A FIR filter was applied to the data with the following parameters: a frequency specification low/high-pass filter with low and high transition start and end frequencies, [6 8 28 32], a minimum-phase filter type, -20 pass-band ripple, -40 stop-band ripple, a frequency sampling design rule (constructs the filter via the Fourier transform without tuning), and 50,000 data chunk length to break up the recordings into segments. An epoch extraction window of 0.5 to 3.5 seconds after the stimulation label was selected. The feature extraction parameters selected were 3 CSP pattern pairs and to perform a log-transform. The final parameters that were modified were the machine learning methods. Note, all the parameters mentioned up to this point are standardized for both LDA and RVM. When generating a classifier with a linear discriminate analysis method, the following parameters were selected: the optional regularization parameter, lambda, was set to 0.6, an analytical covariance shrinkage approach was emphasized, a weighted covariance was applied for unequal probabilities for different classes, and a 1vR (OvR) voting scheme was selected. When generating a classifier with a relevance support vector machine method, the following parameters were selected: a classification RVM approach was clarified, a linear kernel was applied with the default scale values, a 6 polydegree value was chosen, a bias term as well as a standard deviation scaling was applied, a maximum of 1000 iterations and 1000 seconds were defined, and finally a 1vR voting scheme was selected.

With both CSP LDA and CSP RVM approaches defined, the next step is to create multi-class classifiers and compare the controller's performance. In Figure 43, a binary class model was first generated for the subjects' recordings of each test, imagined left vs right hand and imagined left vs right foot. Recall, in the previous objective a total of five sessions, each with four tests, were recorded; after visual inspection through pre-processing step, four of the five motor imagery test recordings were retained (all of the physical action recordings were discarded). Five cross-validation folds and a spacing around test trials

value of five was applied to examine the performance of the two-class controller. The true positive, true negative, false positive, false negative, and error rate percentages for each model were recorded. As a result, the best left/right hand and left/right foot recordings were identified for each LDA and RVM classifier approach.

The next step was to create and evaluate the three-class model, which incorporates a third marker, '800.' This marker represents the subject's rest state; unfortunately, given the limitations of the OpenViBE stimulation presentation, the subjects were not visually told to rest, but had to anticipate the marker happened three seconds after the stimulus arrow disappeared from the monitor. Nevertheless, the same machine learning methods were applied and the performance of each test recording was recorded. When a multi-class controller is greater than two classes, the BCILAB performance metrics reduce to just the error rate percentage. Like the previous binary classifiers, each subject's best left/right hand vs rest and left/right foot vs rest recordings were identified for both the LDA and RVM approach.

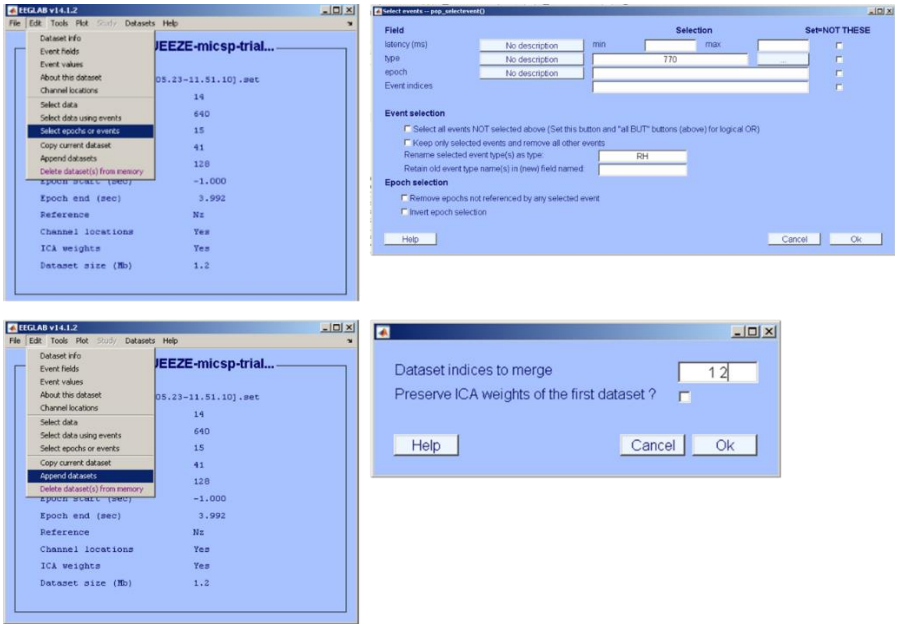


Figure 44: EEGLAB Navigation to rename marker labels and append recordings together.

A five-class model combines the best subject recordings identified in the three-class model. These down selected recordings were appended together using the EEGLAB software (see Figure 44). However, before this is done, the stimulation labels for each recording need to be renamed. The left/right hand and left/right foot recordings have the same left and right arrow stimulation code from OpenViBE, '769' and '770'. For the

subjects' hand recording, the codes were replaced with 'LH' and 'RH' labels; for the subjects' foot recordings, 'LF' and 'RF' labels were applied. Note, the '800' code was not renamed because the rest command is uniform across both recordings. After creating the combined files for each subject, LDA and RVM learning methods were applied to create a model trained to identify 'LH', 'RH', 'LF', 'RF', and '800' brain signals. The error rate percentages were recorded for each method and compared.

The final six-class model incorporates the best recording from the jaw vs roof tests to the down selected left/right hands and feet recordings. Due to social distancing policies and regulations, only subject S03 was able to complete this test. Nevertheless, the relabeling and appending procedure used in five-class modeling was applied. For subject S03's jaw and roof of mouth markers were relabeled 'JAW' and 'ROOF' before being combined. Similarly, marker '800' was not renamed because the rest command remains uniform across all recording. Finally, LDA and RVM learning algorithms were applied to train a classifier to recognize 'LH', 'RH', 'LF', 'RF', 'JAW', and '800' signals and the error rate percentages were recorded. The best performing model was chosen as the official BCI controller for the mobile robotic arm. The results of each binary and multi-class controller is discussed in the Results and Analysis Chapter.

6.3.3 Objective 3 – BCI Controller Implementation, Test Procedure

The third and final test objective implements the official six-class BCI controller to the mobile robotic arm. Unfortunately, due to social distancing policies, a physical demonstration was not possible; instead, a virtual version of the robot was designed and placed in a simulated single-floor house with typical objects (i.e. tables, chairs, etc.) and walls. Coded sensors and cameras were added to make the robot as realistic to the real design as possible; this also includes adding coefficients of friction between the wheels and the floor, and moments of inertia to the motors driving the arm, hand, and wheels. Similar to the robot, the actual presence of the BCI controller and subject S03 was replaced with a recording of the EEG brainwaves. As a result, multiple EEG datasets were created from the subject S03 recordings through EEGLAB. First, six individual datasets, each containing fifteen trials of a specific command, were created. For example, one dataset only contains

fifteen trails of ‘LH’, and another only has fifteen trials of ‘RH’. Next, a dataset combining all six individual sets sequentially was generated. Finally, a data set containing a single trial of every command was generated. As mentioned in the BCILAB software, datasets can be played in a loop and act like an EEG input stream. As a result, these newly created datasets can simulate subject S03’s motor imagery signals and create a command stream for the virtual robot.

The virtual demonstration of controlling the mobile robotic arm was performed in a modular approach. First, the BCI controller was used to move the robot base in the simulated house. Ideally, the final demonstration of controlling the robot base with the BCI headset would involve navigating it to a specified location in an enclosed area; unfortunately, with all the physical restrictions, this test is limited to testing a basic string of commands. The combined file of all the classes with multiple trials was used to move the robot forward, backward, left, and right. Next, the BCI controller was used to control the robot arm. Normally, the subject would guide the cursor on the robot’s camera screen over to an object that the robot can recognize and engage in. The jaw command would then be used to initiate the robot arm to pick up and put down an object. Note, the actual movement of the arm and hand will be pre-programed. Again, with physical limitations, this virtual demonstration was broken up into two tests. The first test utilized the same EEG combined dataset to move a cursor on the robot’s camera screen up, down, left and right. The second test had the cursor locked on an object and the jaw recordings were played to command the robot to pick up and put down the object. The summation of these individual tests demonstrate the BCI controller has control over all intended actions of the mobile robotic arm. The next chapter discusses the results from all three test objectives.

Chapter 7 – Results and Analysis

The results from testing are presented in this chapter. BCI controllers involve both a biological understanding of the human brain and an engineering controls background. The observations made from each objective integrate both fields cohesively and show the potential of creating a reliable and affordable BCI controller.

7.1 Objective 1: Active Electrode Locations, Results

The results for the active electrode locations with be split into two sections. First, the ERP plots for each subjects' hands and feet motor imagery tests will be shown. The more dynamic electrodes will be presented in this section, and the remainder can be referenced in Appendix B. Specific time points were recorded at the maximum and minimum potential voltages and used for the next set of plots. The final results for this objective include an overlay graph of all 14 channel ERPs; the specific time points captured in the individual electrode ERPs will be used to show the overall topographical plot of the subjects.

7.1.1 Single Electrode ERPs, Hand Motor Imagery

All three subjects' most active electrodes for physical and imagined hand motor imagery are presented. The series of plots were generated using the simple ERP study in Figure 41. For each subject, four conditions were plotted for each channel, left hand imaginary (LH-NO SQUEEZE), left hand physical (LH-SQUEEZE), right hand imaginary (RH – NO SQUEEZE), and right hand physical (RH-SQUEEZE). To do this, all of the pre-processed physical and imagined hand test recordings from each subject were split into two files, fifteen left and right hand trials with epoch limits of -1 to 4 sec. All of these specified hand files were then uploaded to the correct data column condition and are now referred to as "clusters". The grand average of each cluster's ERP was calculated and plotted for each electrode in four different colors and labeled according to the condition. The series of figures show the overall ERPs for each channel in the upper left hand corner; the four most active electrodes are highlighted in red and their ERPs plots are boxed. These

four ERPs are shown in the rest of the figure, and the maximum and minimum voltages at specific time points are identified.

For subject S01, the most active electrodes in both hemispheres were: FC3, C5, C3, C1, FC4, C2, C4, and CP4 (see Figure 45 and Figure 46). For the most part, the four averaged ERPs in every electrode plot display are synchronized with a series of negative and positive deflections. These deflections represent the brain's detection to specific stimuli and its response, and are studied extensively in the field of neurology to uncover various psychiatric conditions. When being presented with a stimulus, there are two notable deflections. The first deflection, or early waves, after a given stimulus (t=0 msec) can have a maximum or minimum peak between 100 and 200 msec [77]; these are known as 'sensory' or 'exogenous' and are dependent on the kind of stimulus presented. Later ERP wave deflections show the subject's evaluation of these stimuli and are called 'cognitive' or 'endogenous.'

Looking at subject S01's ERP plots, the first negative deflection has an amplitude ranging between -17.59 and -3.19 is seen between 258 and 328 msec; the most negative value was the C5 electrode. Reviewing the data, the sensory wave reflects an N200 or N300 waveform, which is a negative deflection peaking about 200 or 300 msec. For N200 waves, this behavior is frequently seen when the brain processes a noticeable change in a repetitive background. The N200 waveform can be broken down into three types: N2a, N2b, and N2c. N2a is the fastest detection and is seen in experiments where the auditory background music changes. N2b is a slightly later latency and is associated to a physical stimulus change. And finally, N2c is the slowest detection because it means the subject needs to classify the stimuli when shown. In contrast, the N300 waveform has been associated with a subject having an expectation or anticipation that a stimulus will appear. Based on these explanations, subject S01's reaction can be a mixture of both throughout the testing. Recall the trial timeline in Figure 13. The subject sees a green cross on the black screen and hears the auditory cue in preparation of the red left/right arrow. Then, the subject is presented with the sudden appearance of a red arrow with its direction chosen at random. Both the anticipation and classification of these stimuli are reflected well in the ERP plots. The other noticeable waveform seen in subject S01's plots is P300. This wave is a positive deflection that has a latency range between 250 and 400 msec. Similar to the other

waveforms, the time it takes for a P300 wave to emerge depends on how fast the subject can classify the stimulus. Other studies have shown that large P300 amplitudes also correlate to great attention. Subject S01's C1, C3, C2, and C4 ERP plots capture the N200 and P300 waves. This makes sense since the CZ electrode location is where the largest response can be recorded; this is known as the vertex potential and is approximately location of the somato-sensory cortex [78]. The other electrodes like C5, where the negative deflection occurred at 382 msec, have positive deflections later as well, 562 to 600 msec. Lastly, on all ERP plots is a voltage spike at around 1656 msec; this most likely is a blinking artifact that's been incorporated into the average; note, while ICA was applied, the blinking artifact was left because its removal would also take away other important signal information. As a result, excluding the voltage spike, the maximum and minimum voltage amplitudes were -10 and +10 microvolts. Examining the difference between the physical and imagined hand tests, the physical LH and RH were more out of sync with one another. Meanwhile the imagined left and right hand, blue and red signals, have some noticeable differences after 300 msec. For this reason, the BCILAB training model epoch limits were at 0.5 and 3.5 seconds. In some cases, like the FC4, the LH and RH were opposite charges.

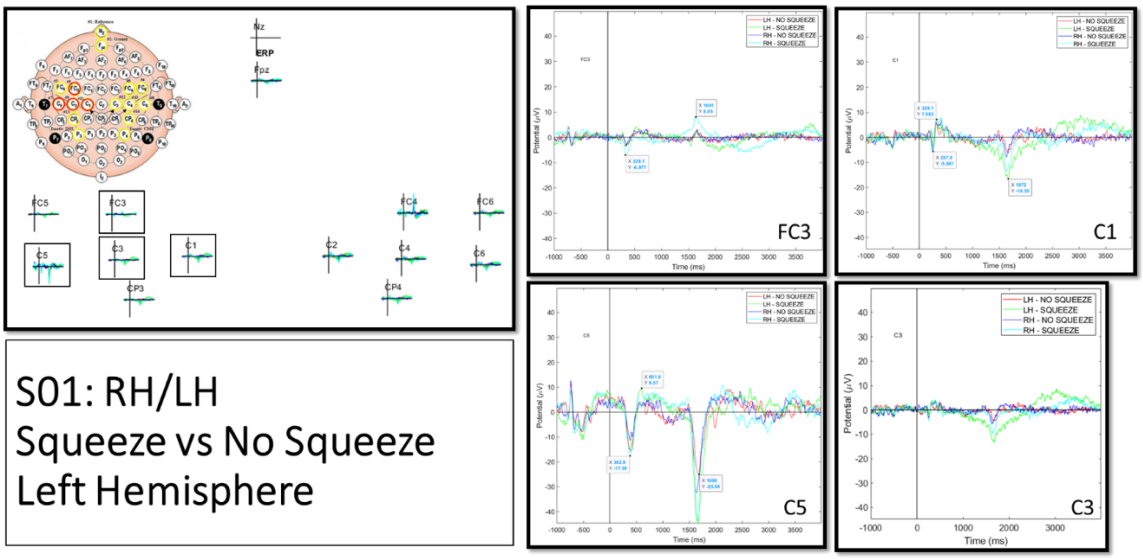


Figure 45: Subject S01's most active left hemisphere electrodes from physical and imaginary hand motor imagery testing.

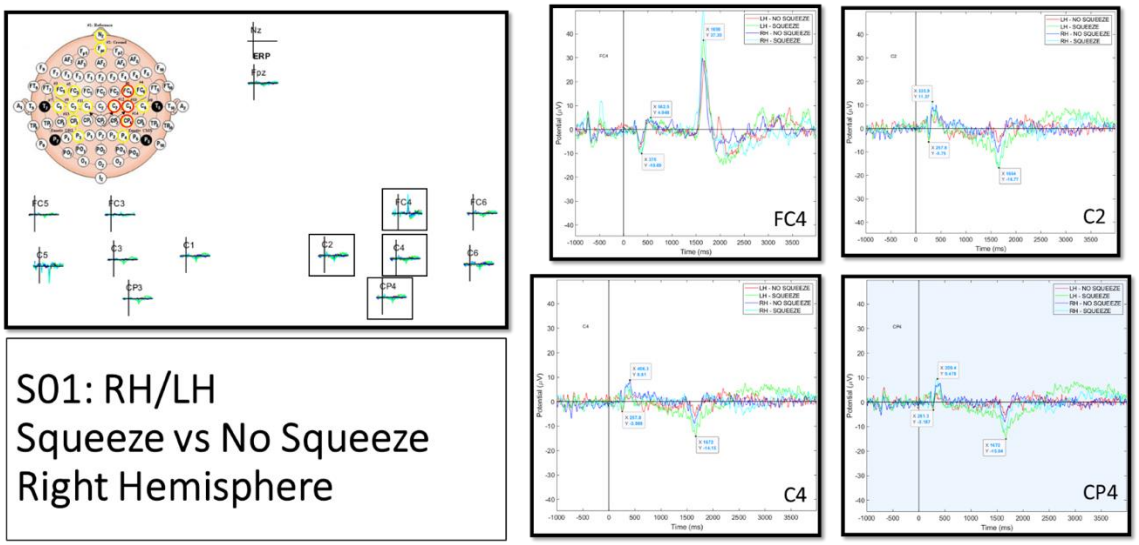


Figure 46: Subject S01's most active right hemisphere electrodes from physical and imaginary hand motor imagery testing.

Subject S02's hand motor imagery ERP plots for the four most active left and right hemispheres are shown in Figure 47 and Figure 48. The most active electrodes for both hemispheres were: FC5, C5, C3, C1, FC4, C2, C4, and C6. Like subject S01's results, the imaginary left and right hand ERP plots for each electrode were synchronized and demonstrate the N200 and P300 waves in C3, C1, C2, C4, and C6. The first negative deflection had an amplitude range of -11.93 and -4.66 microvolts and was seen between 257.8 and 359.4 msec. The subsequent positive deflection had an amplitude range of 2.92 and 6.14 microvolts was seen between 343.8 and 390.6 msec. Looking at both hemisphere figures, the RH SQUEEZE signal (light blue) had irregular behavior; one possible explanation was subject S02's hair impeding the connection between the electrode and scalp. When tucking his hair into the headset, subject S02 unknowingly distributed his hair unevenly underneath the cap. As a result, electrode location FC5, C3, C4, and C6 exhibit this sporadic behavior in RH SQUEEZE. When comparing to subject S01, subject S02 has a smaller general range, between -9.6 and 7.4 microvolts. Note, the peak voltage from the FC4 ERP plot at 1961 msec is also a blinking artifact. While electrode FC4 and C5 ERP plots may look more desynched than subject S01's plots, it is important to note the y-axis scale; subject S01's min and max y-axis value is -40 and 40 microvolts while subject S02's values are -15 and 25 microvolts. As a result, although more magnified, subject S02's

plots show the four conditions are more synched. When looking at the imaginary left and right hand signals, the right hand had smaller amplitudes and its positive and negative deflections occurred earlier than the left hand (see electrode FC5, C5, and FC4. Determining if the machine learning algorithm can better discriminate subject S02's brainwaves will be covered later in this chapter.

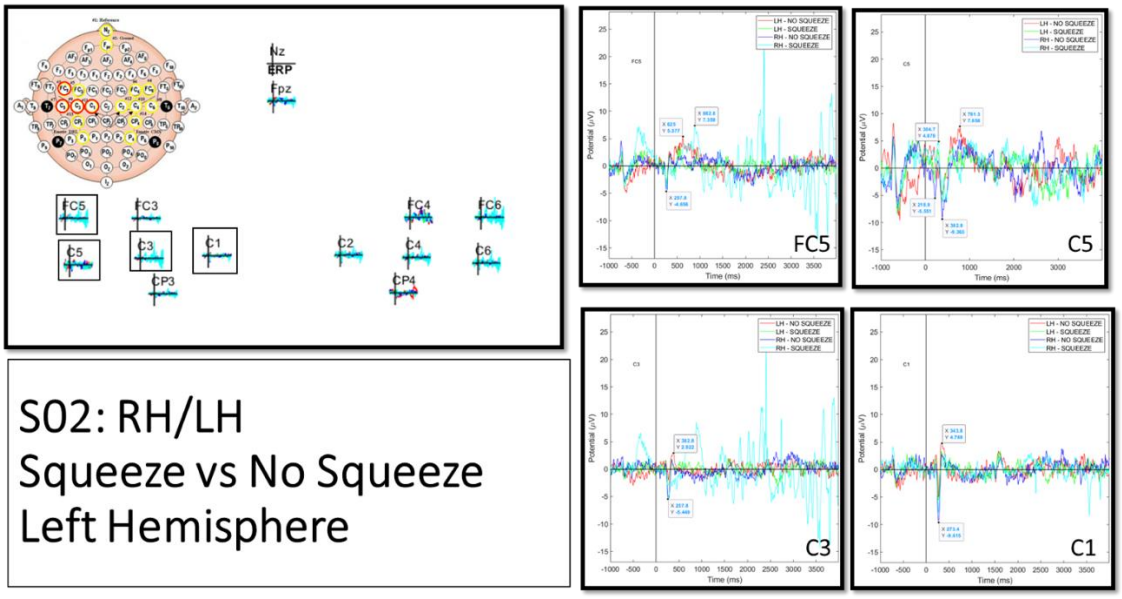


Figure 47: Subject S02's most active left hemisphere electrodes from physical and imaginary hand motor imagery testing.

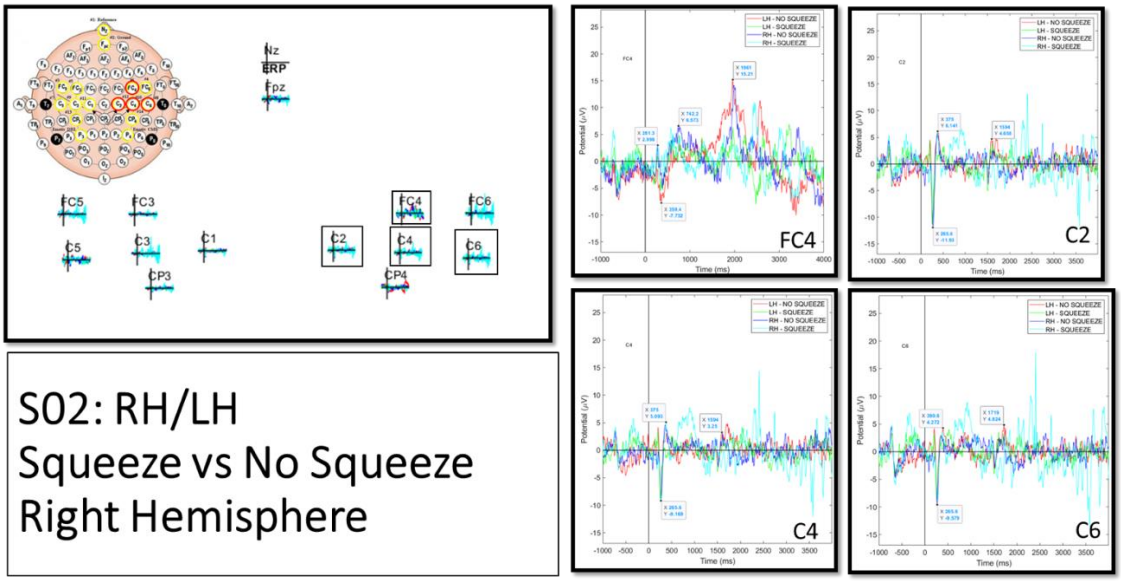


Figure 48: Subject S02's most active right hemisphere electrodes from physical and imaginary hand motor imagery testing.

Subject S03's hand motor imagery ERP plots for the four most active left and right hemispheres are shown in Figure 49 and Figure 50. The most active electrodes for both hemispheres were: FC5, C5, C3, C1, C2, C4, C6, and CP4. Unlike subject S01 and S02, subject S03's ERP plots show synchronization with all four conditions (with the exception of electrode C5). In addition, the y-axis range for these plots are -10 to 10 microvolts; this is even closer than subject S02's ERP plots. Nevertheless, N200 and P300 waves can be seen in FC5, C3, C1, C2, C4, C6, and CP4 plots. The first negative deflection had an amplitude range of -8.2 and -4.1 microvolts and was seen between 257.8 and 281.3 msec. The subsequent positive deflection had an amplitude range of 5.8 and 12.7 microvolts and was seen between 367.2 and 414.1 msec. One outstanding observation worth noting is the presence of both N400 and P400 in C4, CP4, and somewhat in C6 (right hemisphere). In all three of these electrode plots, the negative deflection occurred with the physical and imaginary left hand trials; and, the positive deflection occurred with the physical and imaginary right hand trials. There are two possible explanations for this. It could be possible that subject S03's left-handedness could cause this difference. Subject S01 and S02 are right handed and demonstrate similar wave patterns. Unfortunately, this will need to be confirmed by testing another subject who is left-handed. Another possible

explanation is that a blinking artifact could have interfered with subject S03's right hand test trials. Positive spikes have been noted in both subject S01 and S02's ERP Plots.

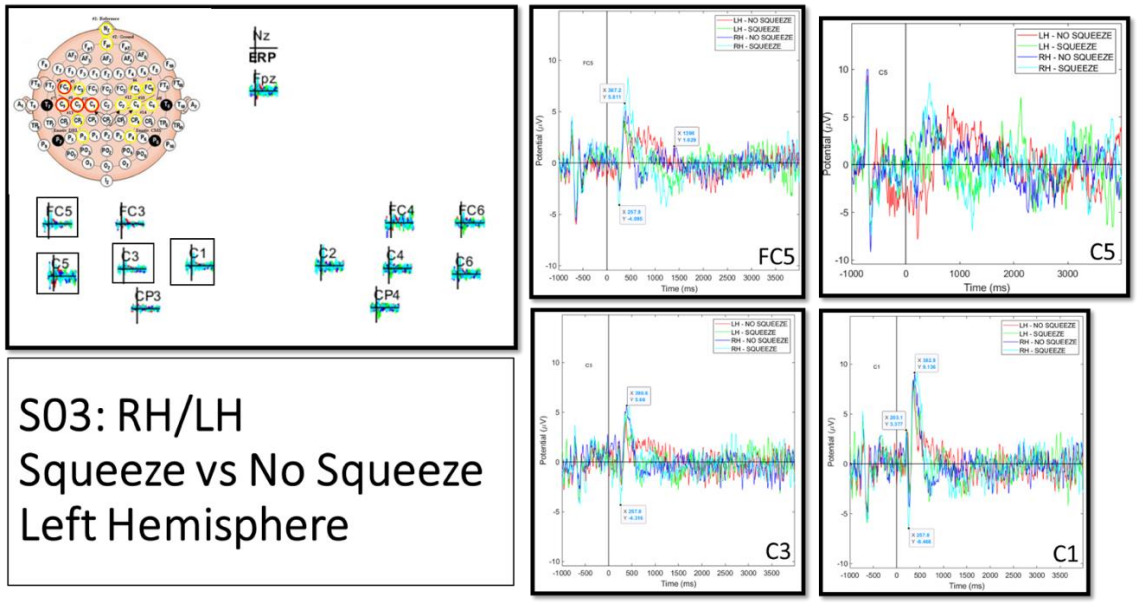


Figure 49: Subject S03's most active left hemisphere electrodes from physical and imaginary hand motor imagery testing.

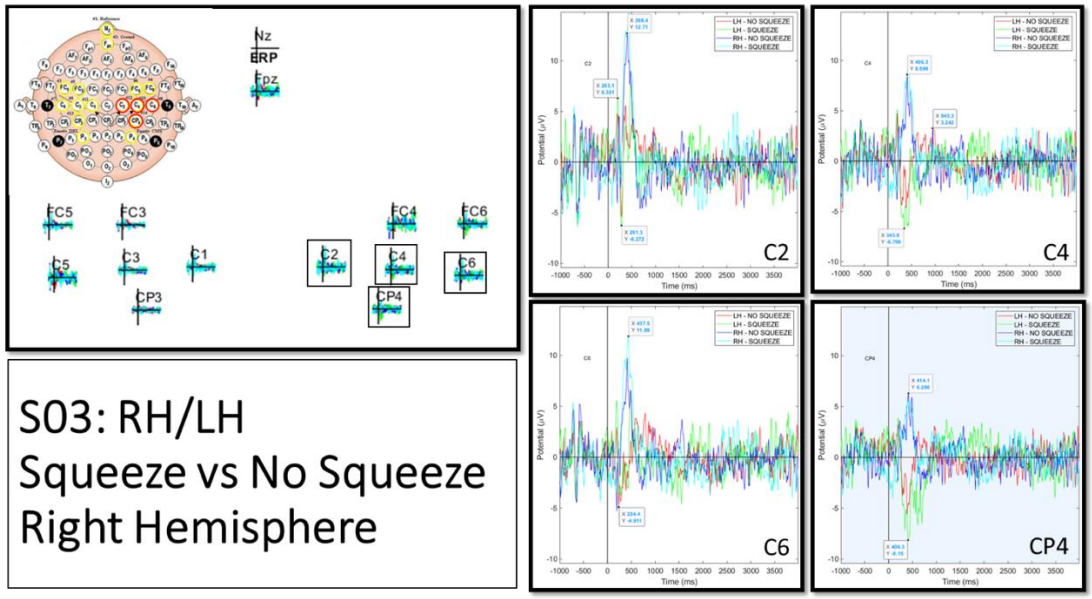


Figure 50: Subject S03's most active right hemisphere electrodes from physical and imaginary hand motor imagery testing.

7.1.2 Single Electrode ERPs, Foot Motor Imagery

All three subject's most active electrodes for physical and imagined foot motor imagery are presented. Similar to the previous section, simple ERP study plots were created for each subject's electrodes with four conditions, left foot imaginary (LF-NO TAP), left foot physical (LF-TAP), right foot imaginary (RF-NO TAP), and right foot physical (RF-TAP).

Subject S01's foot motor imagery ERP plots for the four most active left and right hemispheres are shown in Figure 51 and Figure 52. The most active electrodes for both hemispheres were: C5, C3, C1, CP3, C2, FC4, C4, and CP4. Similar to the hand test results, subject S01's physical and imaginary left and right hand results look synchronized; however, it is important to observe the min and max y-axis values of -25 and 25. These limits are due to the spikes seen in C5 and FC4 at 1672 msec. There are two possible reasons why subject S01's ERP values were extreme. Similar to the hand results, the blinking artifact is the most probable cause. Another possible explanation is Subject S01's physical foot tapping; it was observed that Subject S01 wore his shoes while doing the foot test. The electrodes are sensitive enough to pick up the subtle body movement from the impact of the shoes hitting the hard floor. Still, if the rest of the electrode ERPs are viewed (i.e. C3, C1, CP3, C2, C4, and CP4), the N200 and P300 waves are still evident. The first negative deflection post-stimulus had a range of -5.3 and -3.5 microvolts occurring between 265.6 and 281.3 msec. The subsequent positive deflection post-stimulus had a range of 4.2 and 7.9 microvolts occurring between 320.3 and 421.9 msec. The specific time points observed were later used to show subject S01's topographical maps throughout the epoch limit.

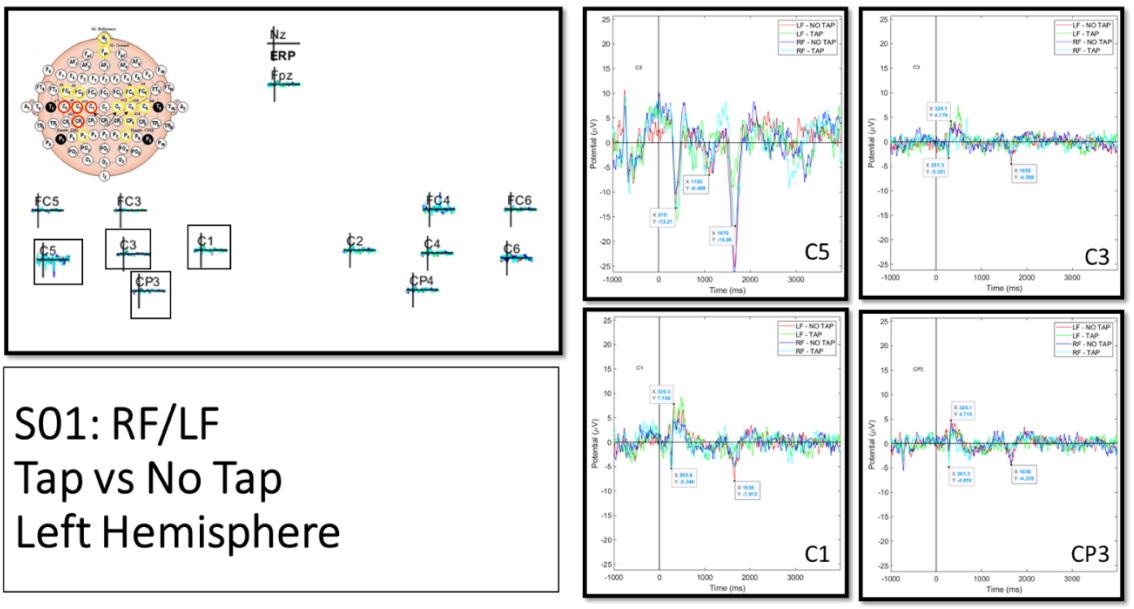


Figure 51: Subject S01's most active left hemisphere electrodes from physical and imaginary foot motor imagery testing.

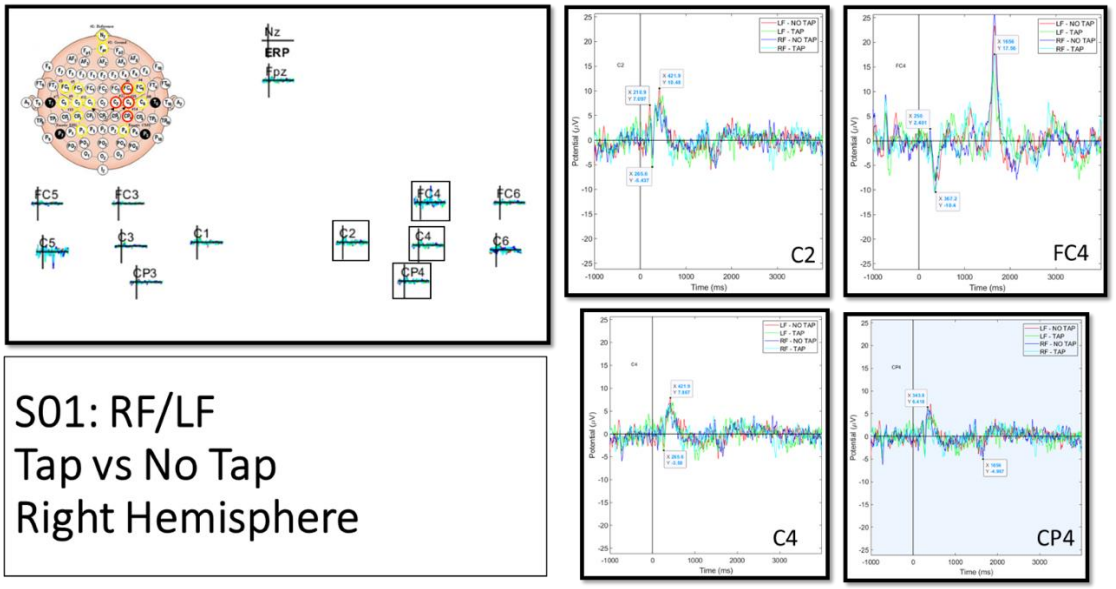


Figure 52: Subject S01's most active right hemisphere electrodes from physical and imaginary foot motor imagery testing.

Subject S02's foot motor imagery ERP plots for the four most active left and right hemispheres are shown in Figure 53 and Figure 54. The most active electrodes for both hemispheres were: C5, C3, C1, CP3, FC4, FC6, CP4, and C6. Unlike the hands motor

imagery test, the foot test results showed better synchronization with all four conditions. In addition, the y-axis scale limits for these figures are -10 and 10 microvolts, which is dramatically less the hand plots. With the exception of C5 and FC4, the microvoltage potentials stayed between -5 and 5 microvolts. These exceptions can be attributed to blinking artifacts once again; this is strongly supported given the same electrode position, FC4, demonstrates the same spike for both subject S01 and S02 in the hands and feet test. N200 and P300 waves can be viewed, especially in plots C3, C1, CP3, CP4, and C6. Note, for subject S02, the most active right hemisphere electrodes included FC6 and CP4; these differ from the subject's hand results.

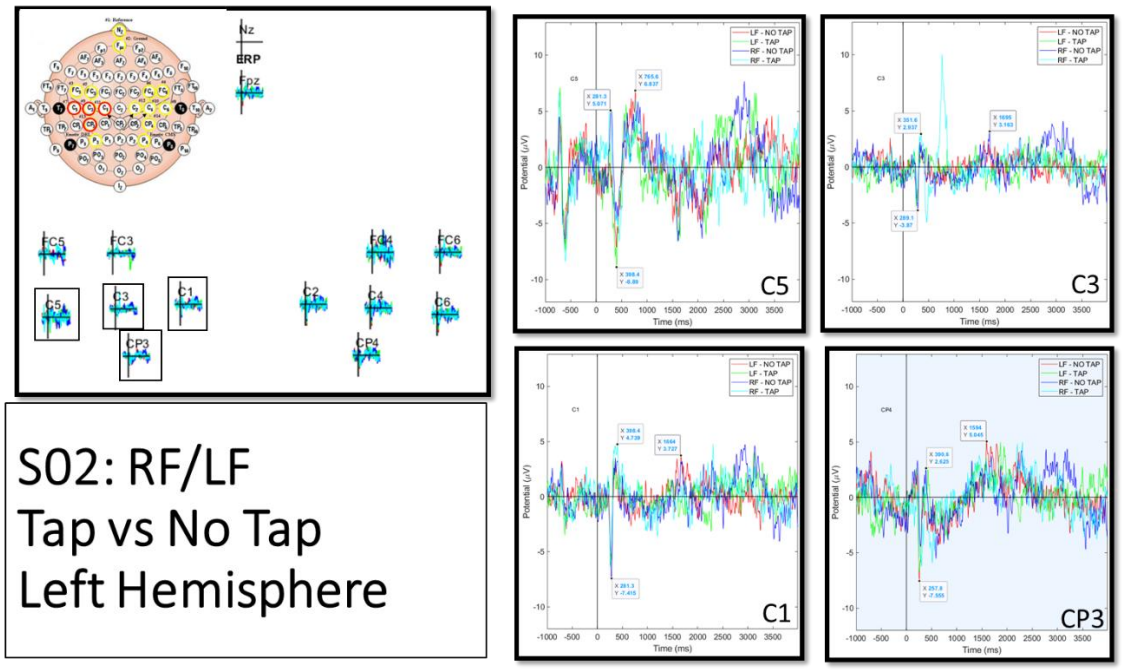


Figure 53: Subject S02's most active left hemisphere electrodes from physical and imaginary foot motor imagery testing.

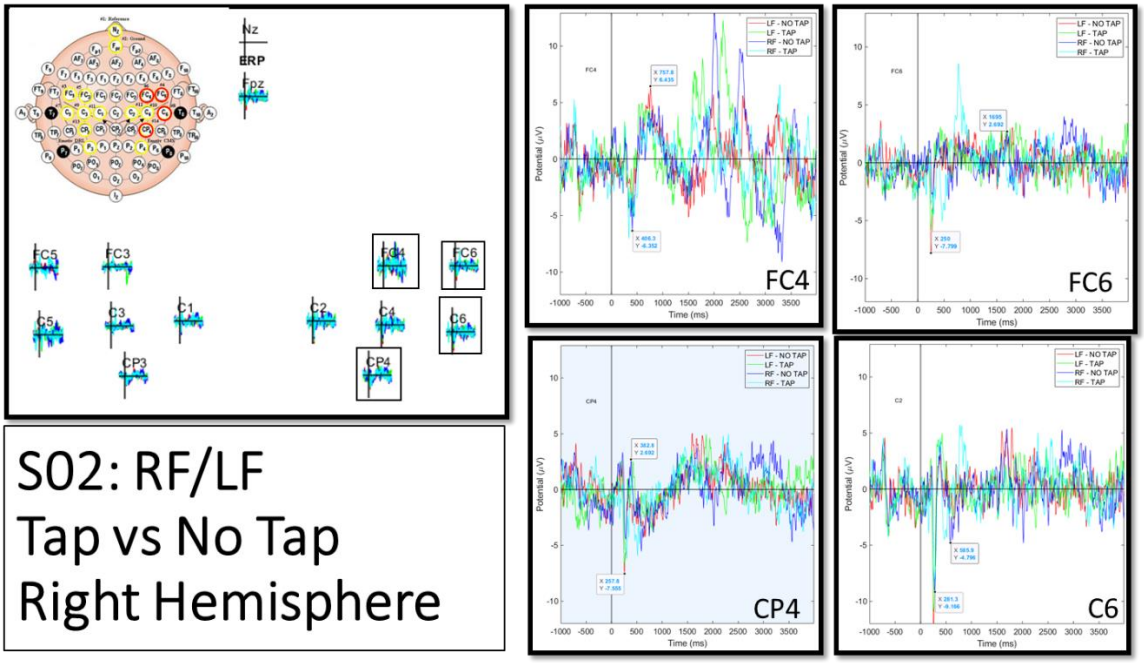


Figure 54: Subject S02's most active right hemisphere electrodes from physical and imaginary foot motor imagery testing.

Subject S03's foot motor imagery ERP plots for the four most active left and right hemispheres are shown in Figure 55 and Figure 56. The most active electrodes for both hemispheres were: FC5, C5, C3, C1, FC4, C2, C4, and C6. Both subject S03's hand and foot motor imagery ERP plots share the same scale and allows for better comparison; when looking at the eight electrodes, the foot ERPs do not show any distinct deflection in certain electrodes (see electrodes C4 and C6). Similar to subject S02, electrodes C4 and C6 were not active electrodes. This could mean that foot motor imagery is more clearly defined in the left hemisphere. Likewise, C5 and FC6 ERP plots show the four conditions not in synchrony. Still, based on FC5, C3, C1, and somewhat C2, N200 and P300 waves can be viewed. The lack of having eight active electrodes may indicate that foot tapping for subject S03 was not an effective approach for BCI. As a result, other alternatives like curling his toes may lead to more promising results.

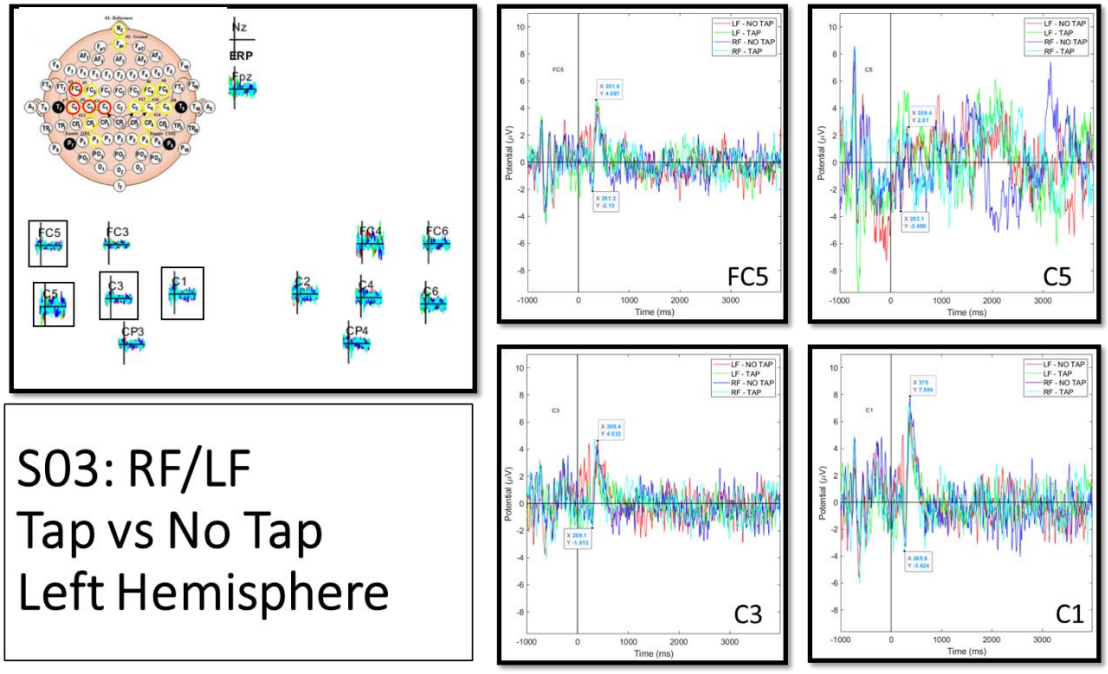


Figure 55: Subject S03's most active left hemisphere electrodes from physical and imaginary foot motor imagery testing.

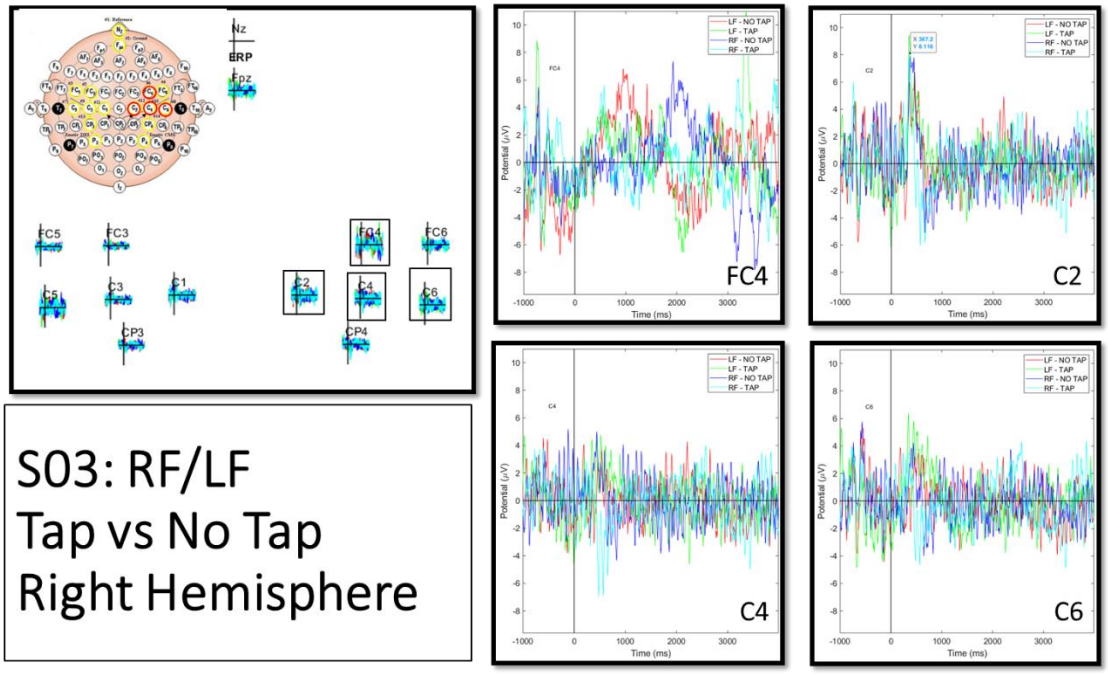


Figure 56: Subject S03's most active right hemisphere electrodes from physical and imaginary foot motor imagery testing.

7.1.3 Single Electrode ERPs, Jaw vs Tongue

Although not all subjects were able to do a jaw vs. tongue press test, it is still important to highlight subject S03's results. Unlike the hand and foot motor imagery signals, the jaw and tongue press are physical muscle movements that will be incorporated into the BCI controller. This simplifies the number of conditions to two instead of four for group ERP studies in EEGLAB. In addition, these two muscles movements are not similar to each other. This is clearly seen in Figure 57 with C5, C1, and FC4 ERP plots. These distinct waveforms will make it easier for the classifier and the results will be shown later in this chapter.

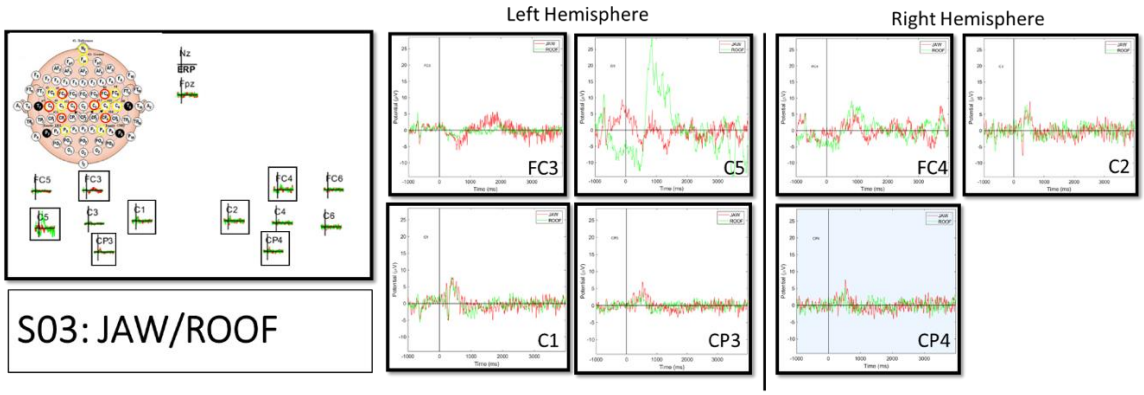


Figure 57: Subject S03's most active left and right hemisphere electrodes from jaw and tongue press to the roof of the mouth.

By analyzing each subject's hand and foot motor imagery results, relationships between OpenViBE's stimuli presentation and the brain's evaluation to the stimuli can be seen. It is important to note that unwanted artifacts, like blinking, can affect ERP plots. Still, with multiple examples of each imaginary left and right action, the classifier can better distinguish a special characteristic waveform. In nearly every test, electrodes in the 'C' row of the 10-20 layout have negative and positive deflections that relate to N200 and P300 signals seen in other studies. The time points at these minimum and maximum microvoltage values can be recorded and used to plot topographical maps. The next section will briefly show these maps for each subject.

7.1.4 Overlay Electrode ERPs, Hand Motor Imagery

With the observations made on which electrodes are most active and general ERP behavior, each subject's combined ERP plot can be analyzed. In this section, topographical plots will be generated at time points selected in the previous section. Note, these time points often correspond to a noticeable shift in signal behavior (i.e. N200, P300, and blinking artifacts). Unlike the previous section, topographical maps show how all the electrodes are responding at a particular moment in time. A gradient showing the range of positive and negative microvoltage potential values will show which regions of the subject's scalp are active. All of the subject's left/right hand and foot motor imagery plots are presented below.

Figure 58, Figure 59, and Figure 60 show the left and right hand ERP plots for subjects S01, S02, and S03. In Figure 58, over sixty trials of each imagined action was used to create an averaged signal for each electrode. Referencing the time points, the largest appearance of the P300 wave occurred at 330 msec. Between the two hand imagery stimuli, subject S01 had a larger region when presented with the right red arrow. Also, note that while the 'C' row of electrodes reacted positively, the rest of the electrodes show more negative regions, especially in the location where the FC4 electrode location. The ERP plot after the P300 wave shows subject S01's effort in imagining his left and right hands being squeezed; topographical maps at 400, 563, and 602 msec show how this imagining activity is propagated throughout the rest of the brain. At 602 msec, the red positive region is on the right side in the RH-NO SQUEEZE testing, and on the left side in the LH-NO SQUEEZE testing; visually, it can be determined which brain activity subject S01 performed. At 1672 msec, the topographical plot shows a local positive region in the FC4 region; this map, and the spike on the ERP plot, show that this is an artifact.

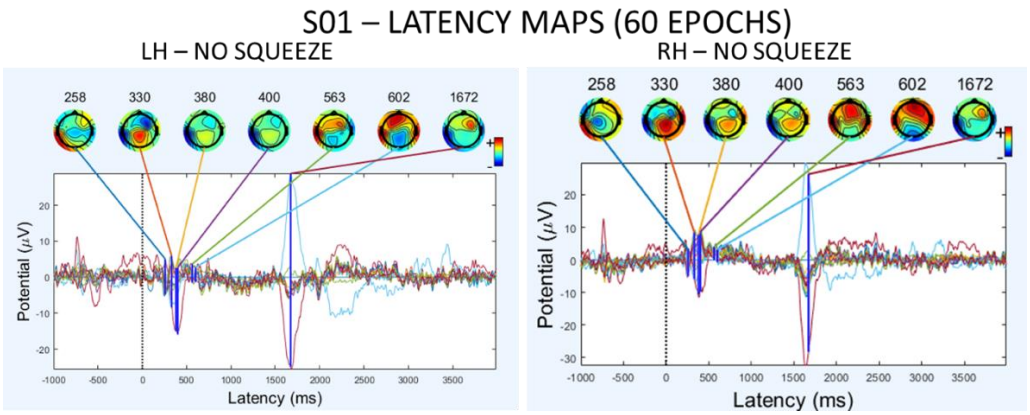


Figure 58: Subject S01's imaginary left and right hand ERP plots with all 14 channels combined. Topographical maps of each time point connected to each maximum and minimum microvoltage values in the previous section are shown.

Moving onto subject S02, the latency maps in Figure 59 show the time lapse of the imagined action of the left/right hand squeezes. Note, over seventy-five trials of each imagined action were used to average the ERP of each electrode. Like subject S01, the sensory waves are represented as a central red region at 355 msec. Subject S02 has two electrode signals do not follow the general trend; as stated in the previous section, this is most likely due to the additional impedance of denser hair. Nevertheless, at 625 msec, the same trend of positive microvoltage is seen in the topographical maps. In fact, for subject S02, the pattern seems to extend to even 781 msec.

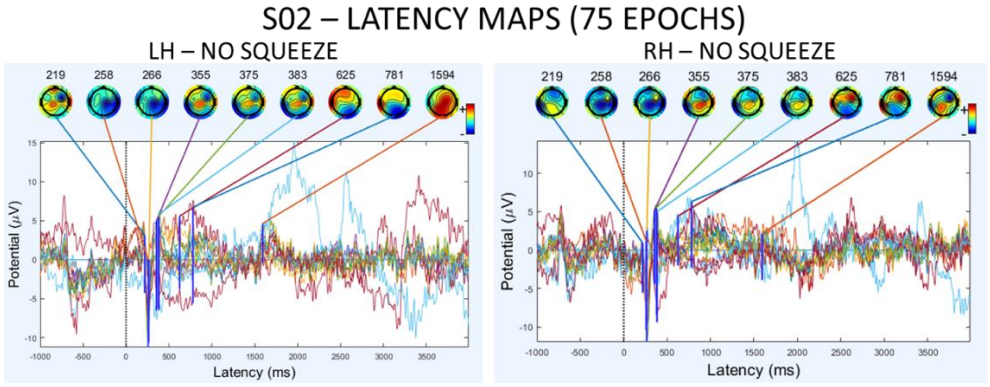


Figure 59: Subject S02's imaginary left and right hand ERP plots with all 14 channels combined. Topographical maps of each time point connected to each maximum and minimum microvoltage values in the previous section are shown.

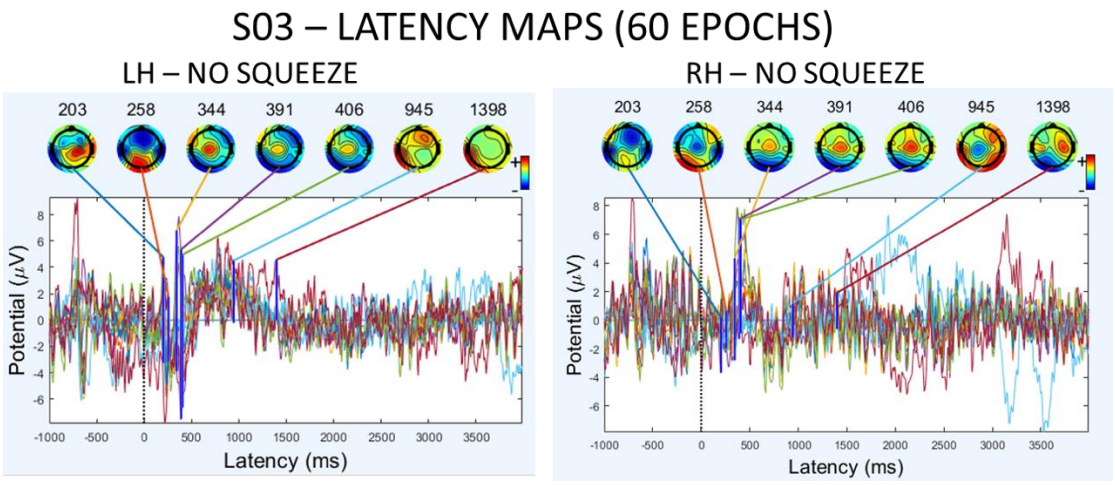


Figure 60: Subject S03's imaginary left and right hand ERP plots with all 14 channels combined. Topographical maps of each time point connected to each maximum and minimum microvoltage values in the previous section are shown.

Finally, subject S03's ERP plots and latency maps are presented in Figure 60. Unlike, subject S01 and S02, the center positive region representing sensory reaction stayed longer, between 344 and 406 msec. In addition, at 258 msec, subject S03's scalp was predominantly negative like subject S02's topographical map. Although not clearly seen in the maps, overall ERP plots for the LH-NO SQUEEZE and RH-NO SQUEEZE do show a difference after the P300 wave. For the LH-NO SQUEEZE ERP plot, the electrode signals are clearly above the baseline; in contrast, the RH-NO SQUEEZE ERP shows all fourteen electrodes under the baseline in the same time region of 500 to 1000 msec. Like the other subject's data, this time region indicates a clear distinction between the imagined left and right hand trials. Note, similar to subject S02's ERP plots, there are two channels that got out of synch during the right hand test trials. Overall, all three subjects' general ERP plots show how the brain responses to stimuli throughout the scalp. Specific time windows of sensory and cognitive processing can be identified through topographical maps. However, for BCI controllers that rely solely on motor imagery signals, the classifier needs to study the cognitive signals. The next section of results will show the results for foot motor imagery testing.

7.1.5 Overlay Electrode ERPs, Foot Motor Imagery

All three subjects' general ERP plots for the imagined left and right foot motor imagery are presented in Figure 61, Figure 62, and Figure 63. Subject S01's sixty epoch average ERP plots in Figure 61 show the same recognition of the stimulus as seen in the hand tests. However, one distinct topographical map is at 422 msec; a positive voltage region occupied in the C2, C6, and CP4 locations in both LF-NO TAP and RF-NO TAP plots. Also, as seen in the previous section, the artifact is noted at 1656 msec. For subject S02, the time selected time points did highlight the P300 wave in the RF-NO TAP plot, but not as well in the LF-NO TAP data (see Figure 62). At 760 and 1695 msec, the topographical maps show a positive voltage region in the upper and lower left hemisphere. In both time points, the stronger positive region is in the LF-NO TAP plot. For subject S03, the general ERP plots for both LF-NO TAP and RF-NO TAP are not as distinct as the hand ERP plots. The topographical maps at the selected time points do not show the brain's evaluation section. This data confirms the observations made in the previous section when looking at the electrode signals individually. Still, the time range between 1500 and 2500 msec show a general trend of the RF-NO TAP signals dipping while the LF-NO TAP are rising with respect to the baseline. While subject S03's foot recordings will be used for developing a BCI controller, the recommendation would be to have the subject train or find another motor imagery signal.

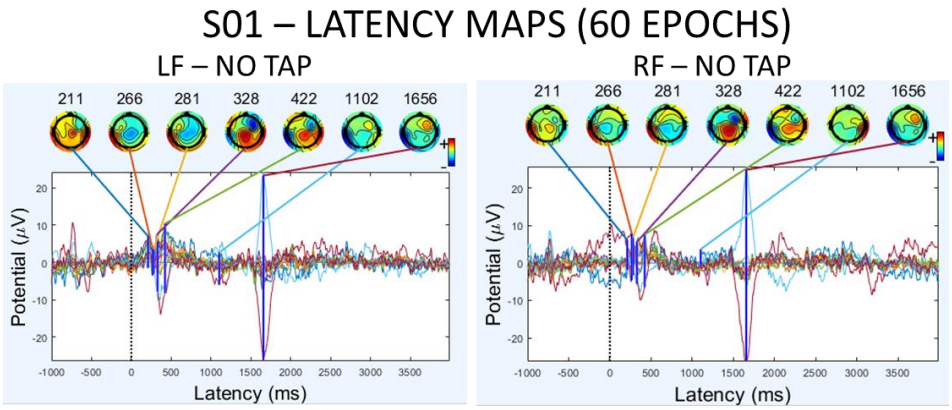


Figure 61: Subject S01 imaginary left and right foot ERP plots with all 14 channels combined. Topographical maps of each time point connected to each maximum and minimum microvoltage values in the previous section are shown.

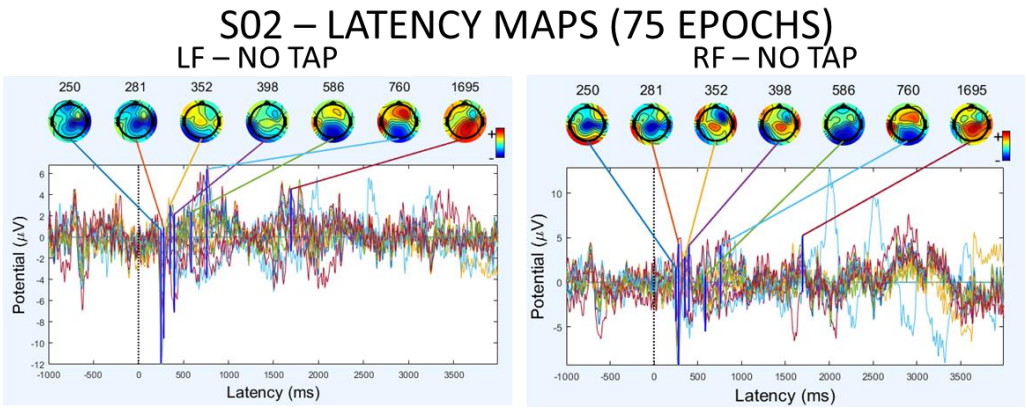


Figure 62: Subject S02 imaginary left and right foot ERP plots with all 14 channels combined. Topographical maps of each time point connected to each maximum and minimum microvoltage values in the previous section are shown.

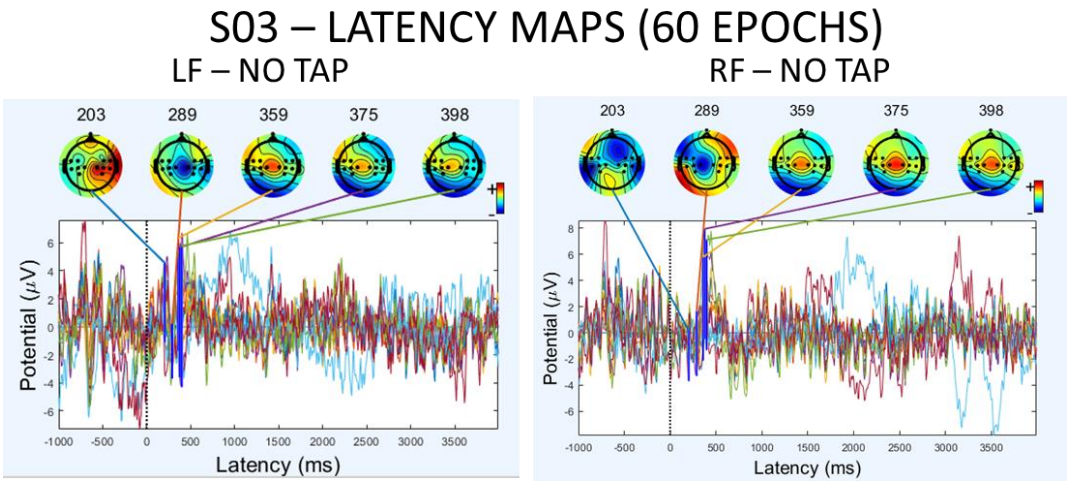


Figure 63: Subject S03 imaginary left and right foot ERP plots with all 14 channels combined. Topographical maps of each time point connected to each maximum and minimum microvoltage values in the previous section are shown.

Given enough trials, topographical maps can be used to visually to identify specific brain activity. As seen with the hands and feet ERP plots, a subject's performance varies with the imagined action; this may be due to familiarity with the action on a regular basis. In this case, the hand squeeze action probably done more frequently than foot tapping. Still, with machine learning, this data is better represented in a matrix form where finer

detail can be picked up. The next section will discuss the approach to identifying the most accurate classifier.

7.2 Objective 2: BCI Controller Performance, Results

The results for which machine learning algorithm, LDA or RVM, will create a classifier with the highest accuracy are presented in this section. Using all of the subjects' motor imagery tests, both methods' performance will be tabulated for two, three, five, and six-class classifiers. The best performing hands and feet tests will be paired for a five-class classifier. And finally, subject S03's best hands, feet, and jaw test recordings will be paired for a six-class classifier.

7.2.1 2-Class BCI Controller Performance Results

2-Class classifier, LH vs RH:

Table 4: Subject S01, 2-Class (LH vs RH) classifier results using LDA. In bold is the best test recording out of all the sessions.

TRIALS	True positive rate	+/-	true negative rate	+/-	false positive rate	+/-	false negative rate	+/-	error rate	+/-
2	0.787	0.197	0.747	0.256	0.253	0.256	0.213	0.197	0.300	0.183
3	**0.717**	**0.310**	**0.817**	**0.171**	**0.183**	**0.171**	**0.283**	**0.310**	**0.267**	**0.149**
4	0.800	0.274	0.300	0.274	0.700	0.274	0.200	0.274	0.467	0.139
5	0.433	0.365	0.583	0.190	0.517	0.190	0.567	0.365	0.567	0.091

Table 5: Subject S01, 2-Class (LH vs RH) classifier results using RVM. In bold is the best test recording out of all the sessions.

TRIALS	True positive rate	+/-	true negative rate	+/-	false positive rate	+/-	false negative rate	+/-	error rate	+/-
2	0.747	0.256	0.573	0.315	0.427	0.315	0.253	0.256	0.433	0.190
3	**0.833**	**0.236**	**0.667**	**0.204**	**0.333**	**0.204**	**0.167**	**0.236**	**0.267**	**0.190**
4	0.850	0.224	0.267	0.253	0.733	0.253	0.150	0.224	0.467	0.183
5	0.417	0.449	0.350	0.253	0.650	0.253	0.583	0.449	0.633	0.247

Table 6: Subject S02, 2-Class (LH vs RH) classifier results using LDA. In bold is the best test recording out of all the sessions.

TRIALS	True positive rate	+/-	true negative rate	+/-	false positive rate	+/-	false negative rate	+/-	error rate	+/-
1	0.55	0.298	0.727	0.3	0.273	0.3	0.45	0.298	0.4	0.224
2	0.633	0.247	0.683	0.41	0.317	0.41	0.367	0.247	0.367	0.183
3	0.267	0.327	0.4	0.253	0.6	0.253	0.733	0.327	0.633	0.139
4	**0.933**	**0.149**	**0.417**	**0.382**	**0.583**	**0.382**	**0.067**	**0.149**	**0.3**	**0.139**
5	0.39	0.385	0.3	0.411	0.7	0.411	0.61	0.385	0.733	0.19

Table 7: Subject S02, 2-Class (LH vs RH) classifier results using RVM. In bold is the best test recording out of all the sessions.

TRIALS	True positive rate	+/-	true negative rate	+/-	false positive rate	+/-	false negative rate	+/-	error rate	+/-
1	0.55	0.298	0.433	0.365	0.567	0.365	0.45	0.298	0.6	0.149
2	0.633	0.247	0.75	0.276	0.25	0.276	0.367	0.247	0.333	0.118
3	0.133	0.163	0.433	0.279	0.567	0.279	0.867	0.163	0.667	0.204
4	**0.7**	**0.183**	**0.367**	**0.342**	**0.633**	**0.342**	**0.6**	**0.183**	**0.433**	**0.149**
5	0.44	0.378	0.3	0.411	0.7	0.411	0.56	0.378	0.7	0.183

Table 8: Subject S03, 2-Class (LH vs RH) classifier results using LDA. In bold is the best test recording out of all the sessions.

TRIALS	True positive rate	+/-	true negative rate	+/-	false positive rate	+/-	false negative rate	+/-	error rate	+/-
3	**0.90**	**0.22**	**0.83**	**0.24**	**0.17**	**0.24**	**0.10**	**0.22**	**0.10**	**0.09**
4	0.533	0.075	0.75	0.25	0.25	0.25	0.467	0.075	0.333	0.167
5	0.733	0.435	0.833	0.236	0.167	0.236	0.267	0.435	0.167	0.118
6	0.65	0.418	0.733	0.253	0.267	0.253	0.35	0.418	0.267	0.253

Table 9: Subject S03, 2-Class (LH vs RH) classifier results using RVM. In bold is the best test recording out of all the sessions.

TRIALS	True positive rate	+/-	true negative rate	+/-	false positive rate	+/-	false negative rate	+/-	error rate	+/-
3	**1.00**	**0.00**	**0.78**	**0.22**	**0.22**	**0.22**	**0.00**	**0.00**	**0.10**	**0.09**
4	0.517	0.291	0.65	0.224	0.35	0.224	0.483	0.291	0.4	0.19
5	0.833	0.236	0.833	0.236	0.167	0.236	0.167	0.236	0.133	0.075
6	0.95	0.112	0.717	0.298	0.283	0.298	0.05	0.112	0.167	0.167

The 2-class hands performance tables are paired together to show a comparison between the LDA and RVM methods for each subject. When comparing two sets of brain waves, BCILAB will run five cross-folds and calculate the following metrics: true positive, true negative, false positive, false negative, and error rates. For each set of 2-class results, each subject's best test recording and method will be highlighted. Then, a comparison between the subjects will be made. For subject S01, the best LDA and RVM test was trial 3, or T3. The error rate was the same for both methods, 26.7 % nominal. For subject S02, the best LDA and RVM trial was trial 4, T4. The error rate percentage was 30% for LDA

and 43.3% for RVM. And lastly, subject S03's best LDA and RVM trial was 3, T3. The error rate percentage for both methods was 10% nominal. Among the three subjects, subject S03's classifier had a higher accuracy percentage, 90% nominal. Given that both approaches are exactly the same, the difference is a result of the recordings. When comparing error rates, both machine methods lead to the same relative accuracy.

2-Class classifier, LF vs RF:

Table 10: Subject S01, 2-Class (LF vs RF) classifier results using LDA. In bold is the best test recording out of all the sessions.

TRIALS	True positive rate	+/-	true negative rate	+/-	false positive rate	+/-	false negative rate	+/-	error rate	+/-
1	**0.833**	**0.236**	**0.733**	**0.365**	**0.267**	**0.365**	**0.167**	**0.236**	**0.3**	**0.209**
3	0.693	0.213	0.713	0.278	0.287	0.278	0.307	0.213	0.367	0.139
4	0.667	0.312	0.267	0.435	0.733	0.435	0.333	0.312	0.533	0.298
5	0.467	0.361	0.350	0.224	0.650	0.224	0.533	0.361	0.567	0.253

Table 11: Subject S01, 2-Class (LF vs RF) classifier results using RVM. In bold is the best test recording out of all the sessions.

TRIALS	True positive rate	+/-	true negative rate	+/-	false positive rate	+/-	false negative rate	+/-	error rate	+/-
1	**0.833**	**0.236**	**0.533**	**0.506**	**0.467**	**0.506**	**0.167**	**0.236**	**0.350**	**0.224**
3	0.593	0.379	0.463	0.385	0.537	0.385	0.407	0.379	0.467	0.217
4	0.667	0.312	0.333	0.471	0.667	0.471	0.333	0.312	0.5	0.289
5	0.483	0.384	0.500	0.395	0.500	0.395	0.517	0.384	0.500	0.204

Table 12: Subject S02, 2-Class (LF vs RF) classifier results using LDA. In bold is the best test recording out of all the sessions.

TRIALS	True positive rate	+/-	true negative rate	+/-	false positive rate	+/-	false negative rate	+/-	error rate	+/-
1	0.717	0.298	0.7	0.274	0.3	0.274	0.283	0.298	0.3	0.183
2	0.813	0.272	0.627	0.444	0.373	0.444	0.187	0.272	0.267	0.253
3	0.627	0.128	0.45	0.389	0.55	0.389	0.373	0.128	0.4	0.19
4	0.183	0.291	0.65	0.418	0.35	0.418	0.817	0.291	0.567	0.091
5	**0.717**	**0.31**	**0.817**	**0.171**	**0.183**	**0.171**	**0.283**	**0.31**	**0.267**	**0.149**

Table 13: Subject S02, 2-Class (LF vs RF) classifier results using RVM. In bold is the best test recording out of all the sessions.

TRIALS	True positive rate	+/-	true negative rate	+/-	false positive rate	+/-	false negative rate	+/-	error rate	+/-
1	0.717	0.298	0.6	0.224	0.4	0.224	0.283	0.298	0.333	0.118
2	0.853	0.202	0.52	0.375	0.48	0.375	0.147	0.202	0.3	0.139
3	0.587	0.084	0.5	0.373	0.5	0.373	0.413	0.084	0.4	0.149
4	0.467	0.38	0.55	0.512	0.45	0.512	0.533	0.38	0.533	0.274
5	**0.833**	**0.192**	**0.833**	**0.236**	**0.167**	**0.236**	**0.167**	**0.192**	**0.167**	**0.118**

Table 14: Subject S03, 2-Class (LF vs RF) classifier results using LDA. In bold is the best test recording out of all the sessions.

TRIALS	True positive rate	+/-	true negative rate	+/-	false positive rate	+/-	false negative rate	+/-	error rate	+/-
3	0.467	0.361	0.9	0.137	0.1	0.137	0.533	0.361	0.367	0.183
4	0.307	0.413	0.307	0.413	0.693	0.413	0.693	0.413	0.6	0.253
5	**0.667**	**0.204**	**0.85**	**0.224**	**0.15**	**0.224**	**0.333**	**0.204**	**0.233**	**0.19**
6	0.407	0.379	0.683	0.325	0.317	0.325	0.593	0.379	0.5	0.312

Table 15: Subject S03, 2-Class (LF vs RF) classifier results using RVM. In bold is the best test recording out of all the sessions.

TRIALS	True positive rate	+/-	true negative rate	+/-	false positive rate	+/-	false negative rate	+/-	error rate	+/-
3	0.617	0.261	0.8	0.236	0.2	0.326	0.383	0.261	0.333	0.204
4	0.333	0.471	0.357	0.38	0.643	0.38	0.667	0.471	0.667	0.236
5	**0.667**	**0.204**	**0.733**	**0.308**	**0.267**	**0.308**	**0.333**	**0.204**	**0.3**	**0.247**
6	0.447	0.362	0.85	0.224	0.15	0.224	0.553	0.362	0.367	0.247

Like the hands tables, the 2-class feet performance tables are paired together to show a comparison between the LDA and RVM methods for each subject. For subject S01, the best LDA and RVM trial was trial 1, T1. The nominal error rate for LDA and RVM was 30% and 35% respectively. For subject S02, the best LDA and RVM trial was trial 5, T5. The error rate percentage was 26.7% for LDA and 16.7% for RVM. Finally, the best LDA and RVM trial for subject S03 was trial 5, T5; the calculated error rate for the LDA and RVM were 23.3% and 30%. Between the three subjects, subject S02's T5 RVM performed the best. Unlike the hand test results, the feet performance data shows some variation between the methods. With the exception of subject S02, the RVM approach was not as accurate as the LDA approach.

2-Class classifier, JAW vs ROOF:

Table 16: Subject S03, 2-Class (JAW vs ROOF) classifier results using LDA. In bold is the best test recording out of all the sessions.

TRIALS	True positive rate	+/-	true negative rate	+/-	false positive rate	+/-	false negative rate	+/-	error rate	+/-
1	**1**	**0**	**0.95**	**0.112**	**0.05**	**0.112**	**0**	**0**	**0.033**	**0.075**
2	1	0	0.933	0.149	0.067	0.149	0	0	0.033	0.075
3	1	0	0.933	0.149	0.067	0.149	0	0	0.033	0.075
4	1	0	1	0	0	0	0	0	0	0

Table 17: Subject S03, 2-Class (JAW vs ROOF) classifier results using RVM. In bold is the best test recording out of all the sessions.

TRIALS	True positive rate	+/-	true negative rate	+/-	false positive rate	+/-	false negative rate	+/-	error rate	+/-
1	**1**	**0**	**0.95**	**0.112**	**0.05**	**0.112**	**0**	**0**	**0.033**	**0.075**
2	1	0	0.867	0.298	0.133	0.298	0	0	0.067	0.149
3	1	0	0.933	0.149	0.067	0.149	0	0	0.033	0.075
4	1	0	1	0	0	0	0	0	0	0

This last portion of the 2-class results shows how well the selected learning methods can classify signals resulting from a jaw clench and the tongue pressed against the roof of the mouth. Both tables show low error percentage rates as low as 0% nominal. This theoretically means the classifier was perfectly trained to sort between these two signals. While this is most likely untrue, this does reveal that both actions created signals that were not only distinct, but also consistently repetitive. This is in contrast to the motor imagery signals which are more complex due to imagined movement. Based on the 2-class results, the following signals are listed in the order of easy classification: jaw vs roof, left vs right hand, and left vs right foot.

7.2.2 3-Class BCI Controller Performance Results

This next set of performance tables will show the impact of adding an additional class, increasing the class controller to three. Unfortunately, BCILAB's five cross-fold evaluation will only populate the error rate percentage. Nevertheless, the best trials and method can be determined for each test. The discussion of the results will follow the same flow as the 2-Class section. For subject S01, the best LDA and RVM the hands classifier was trial 4 and trial 5 accordingly, T4 and T5. Note, both of these trials were not the best performing 2-class trials. In addition, the error rate percentage for LDA is 41.7% and RVM is 40.0%. This is roughly a 13% increase in uncertainty. For the feet, subject S01's best

LDA and RVM trial was trial 3, T3. The nominal error rate percentage was 43.3% for LDA and 40% for RVM.

Table 18: Subject S01, 3-Class: Hands vs Rest and Feet vs Rest using both LDA and RVM. In bold are the best test recording out of all the sessions.

LH VS RH VS 800	METHOD: LDA		METHOD: RVM	
TRIALS	error rate	+/-	error rate	+/-
2	0.517	0.07	0.467	0.075
3	0.467	0.095	0.433	0.07
4	**0.417**	**0.177**	0.433	0.124
5	0.517	0.070	**0.400**	**0.07**

LF VS RF VS 800	METHOD: LDA		METHOD: RVM	
TRIALS	error rate	+/-	error rate	+/-
1	0.525	0.256	0.55	0.227
3	**0.433**	**0.124**	**0.4**	**0.124**
4	0.55	0.225	0.5	0.204
5	0.60	0.181	0.6	0.181

Table 19: Subject S02, 3-Class: Hands vs Rest and Feet vs Rest using both LDA and RVM. In bold are the best test recording out of all the sessions.

LH VS RH VS 800	METHOD: LDA		METHOD: RVM	
TRIALS	error rate	+/-	error rate	+/-
1	0.417	0.212	0.433	0.091
2	**0.283**	**0.095**	**0.2**	**0.095**
3	0.55	0.112	0.467	0.151
4	0.583	0.156	0.517	0.137
5	0.567	0.16	0.5	0.204

LF VS RF VS 800	METHOD: LDA		METHOD: RVM	
TRIALS	error rate	+/-	error rate	+/-
1	0.517	0.07	0.483	0.109
2	0.383	0.192	0.417	0.144
3	**0.383**	**0.095**	**0.317**	**0.109**
4	0.617	0.075	0.483	0.07
5	0.383	0.162	0.383	0.162

For subject S02, the best LDA and RVM trial for hand motor imagery classifying was trial 2, T2. The error rate percentage for the LDA and RVM methods was 28.3% and 20%. Although this was not the same trial in the previous class, the error rate percentage actually decreased. Shifting focus to the right side of Table 19, the best trial for subject S02's foot classifiers was trial 3, T3. The error rate percentage for both LDA and RVM are 38.3% and 31.7%. With the exception of subject S02's hands, the addition of another class increases the error rate.

Table 20: Subject S03, 3-Class: Hands vs Rest and Feet vs Rest using both LDA and RVM. In bold are the best test recording out of all the sessions.

LH VS RH VS 800	METHOD: LDA		METHOD: RVM	
TRIALS	error rate	+/-	error rate	+/-
3	0.2	0.173	0.2	0.162
4	0.267	0.109	0.233	0.109
5	0.4	0.16	0.317	0.216
6	**0.167**	**0.083**	**0.2**	**0.139**

LF VS RF VS 800	METHOD: LDA		METHOD: RVM	
TRIALS	error rate	+/-	error rate	+/-
3	**0.233**	**0.091**	**0.25**	**0.102**
4	0.417	0.118	0.333	0.177
5	0.3	0.192	0.367	0.183
6	0.45	0.183	0.417	0.156

For subject S03, the best LDA and RVM trials for hand imagery data was trial 6, T6. The error percentages were 16.7% for LDA and 20% for RVM. Similar to subject S02, the hand trials for S03 improved with the inclusion of the third class. For the foot classifying, the best trial for LDA and RVM was trial 3, T3. The error percentages were 23.3% for LDA and 25% for RVM. This is roughly the same percentage as before with the 2-class performance. In general, the increase of another class has increased the uncertainty in the classifier. In multiple cases, the trial best trial changed. However, with the exception of subject S03's results, the RVM method outperformed the LDA approach.

Table 21: Subject S03, 3-Class: Jaw vs Roof vs Rest using both LDA and RVM. In bold are the best test recording out of all the sessions.

JAW VS ROOF VS 800	METHOD: LDA		METHOD: RVM	
TRIALS	error rate	+/-	error rate	+/-
1	**0.117**	**0.095**	0.15	0.07
2	0.183	0.149	**0.1**	**0.091**
3	0.117	0.095	**0.1**	**0.091**
4	0.133	0.112	0.167	0.102

This last portion of the 3-class results show how well the LDA and RVM method can classify the addition of the rest '800' class with the jaw and roof signals. For LDA, the best trial was trial 1, T1. For the RVM method, the best trial was both Trial 2 and 3, T2 and T3. While these error percentages did increase by about 6%, this test remains the most accurate. Based on the 3-class results, subject S02 and S03 have more accurate classifiers for the hand and feet. The order of signals in terms of easy classification has remained the same.

7.2.3 5-Class BCI Controller Performance Results

The 5-class performance tables are presented in this sub-section. Based on the accuracies obtained from the 3-class section, the best trials for both hands and feet testing were paired. Each table will show the trials selected as well as the error rate percentages for each method.

Table 22: Subject S01, 5-Class: Hands vs Feet vs Rest using both LDA and RVM. Note, the best test recordings were used.

LH VS RH VS LF VS RF VS 800			
TRIALS	**METHOD**	**error rate**	**+/-**
T4 HAND	LDA	0.633	0.16
T3 FOOT	RVM	0.492	0.095

LH VS RH VS LF VS RF VS 800			
TRIALS	**METHOD**	**error rate**	**+/-**
T5HAND	LDA	0.525	0.12
T3FOOT	RVM	0.450	0.068

For subject S01, the T4/T5 hand and T3 foot trials were combined. The 5-class performance error percentage for LDA and RVM was 63.3% and 49.2% for the T4 Hand and T3 Foot configuration. T5 Hand and T3 Foot configuration had an error percentage of 52.5% for LDA and 45.0% for RVM. While the uncertainty percentage increased, the RVM method shows higher accuracy. Between the two configurations, the T5 hand and T3 foot performed better.

Table 23: Subject S02, 5-Class: Hands vs Feet vs Rest using both LDA and RVM. Note, the best test recordings were used.

LH VS RH VS LF VS RF VS 800			
TRIALS	**METHOD**	**error rate**	**+/-**
T2HAND	LDA	0.358	0.091
T3FOOT	RVM	0.308	0.096

For subject S02, the T2 hand and T3 foot trials were used to form a 5-class controller. The LDA and RVM nominal error percentage was 35.8% and 30.8%. As seen with subject S01's 5-class classifier, the RVM method has better accuracy by 5%.

Table 24: Subject S03, 5-Class: Hands vs Feet vs Rest using both LDA and RVM. Note, the best test recordings were used.

LH VS RH VS LF VS RF VS 800			
TRIALS	**METHOD**	**error rate**	**+/-**
T6HAND	LDA	0.333	0.114
T3FOOT	RVM	0.292	0.083

For subject S03, the T6 hand and T3 foot trials were used to form a 5-class controller. The LDA and RVM nominal error percentage was 33.3% and 29.2%. Subject S03's RVM error rate percentage is by far the lowest. As seen before, increasing the number of classes, increases the uncertainty in the classifier. And, in all cases, the RVM method lower error rate percentages than LDA.

7.2.4 6-Class BCI Controller Performance Results

Table 25: Subject S03, 6-Class: Hands vs Feet vs Jaw vs Rest using both LDA and RVM

LH VS RH VS LF VS RF VS JAW VS 800			
TRIALS	**METHOD**	**error rate**	**+/-**
T6HAND T3FOOT T1JAW	LDA	0.345	0.046
	RVM	0.345	0.090

The 6-class classifier incorporates the best jaw clench trial, T1, to subject S03's 5-class controller. The LDA and RVM nominal error rates are both 34.5%. Note, looking at the LDA method, the inclusion of an additional class did not necessarily increase the uncertainty. This could be because the jaw signals are significantly different from the previous classes. One thing worth noting in getting the 6-class model to converge involved the presence of the '800' rest command. The '800' marker is in every trial. As a result, the rest class has more trials than any other class; when removing the rest command, the both methods would not solve. It is possible that BCILAB saw this class as a 'glue' to combine three different recordings together. Nevertheless, with the official BCI controller generated, the final objective is to implement it into the mobile robotic arm. Table 26 shows a summary of the error rate percentages for each method and class.

Table 26: A summary of error rate percentages of both LDA and RVM for each subject

		subject S01		subject S02		subject S03	
		LDA	RVM	LDA	RVM	LDA	RVM
Hands	2-Class	26.7	26.7	30	43.3	10	10
Feet	2-Class	30	35	26.7	16.7	23.3	30
jaw/roof	2-Class	-	-	-	-	3.3	33
Hands	3-Class	41.7	40	28.3	20	16.7	20
Feet	3-Class	43.3	40	38.3	31.7	23.3	25
jaw/roof	3-Class	-	-	-	-	11.7	10
	5-Class	52.5	45	35.8	30.8	33.3	29.2
	6-Class	-	-	-	-	34.5	34.5

7.3 Objective 3: BCI Controller Implementation, Results

7.3.1 BCI Parameters and Interface

In section 6.3.3, a '.gdf' recording of all six individual commands (RF, LF, RH, LH, JAW, and '800' Rest) in nearly sequential order was created using EEGLAB to operate the mobile robotic arm. This file shows fifteen consecutive epoched time segments of a specific action before moving onto the next one. As a result, the robotic device will systematically run through all the commands: forward, backward, right, and left for the robot base, and up, down, right, left, and initiate grab/drop command for the robot arm. Any discrepancy or deviation from this order can be ruled as an error. The only command action that is not followed sequentially is the '800' rest command. In every left/right hands, feet, and jaw/roof test, the '800' rest command was included. All the time segments in this recording have the '800' rest command embedded. As a result, the training data in test objective two, and the recording used to command the robot, have more rest examples and give the controller a bias to rest if it cannot properly classify a signal. This is strategic for a couple reasons. For machine learning algorithms such as LDA and SVM, having a universal command for each test establishes some commonality and achieves convergence; several runs were performed without the '800' rest command and the learning algorithms failed to solve. As for physically commanding the robot, the safety and reliability of the controller increases. If the classifier is unable to determine the user's brainwaves, it will more likely choose to wait until it can determine a brainwave.

As mentioned in the second test objective results, the error percentage after training the 6-class BCI controller is 34.5% for both LDA and RVM machine learning algorithms. When applying a recorded brainwave file to the classifier, BCILAB will calculate an error percentage as a metric to determine how well it was able to sort the brainwaves. After applying the combined brainwave file to the 6-class BCI RVM controller, the error percentage reduced to 18.5%. This increase in accuracy is most likely due to the reuse of brain recording segments used to train the 6-class controller, which is not entirely representative of online testing where the subject's brainwaves are being observed in real time. As a result, if this study was able to test in person with all subjects, the controller accuracy would most likely decrease. Nevertheless, the final objective is to demonstrate that the train classifier can use brainwave signals can command the mobile robotic arm.

Using BCILAB, the combined recording was looped to act like a simulated brainwave data stream; the entire recording runs approximately 6 minutes and 25 seconds. Following Figure 38, parameters to generate the command stream was 'mode' for 'Output form' and '10' for update frequency. Instead of sending a row matrix showing the distribution of class probability, the 'mode' setting has the controller output the class with the highest probability in regards to its position in the row matrix; as a result, if the controller determined the 'RF' brainwave characteristics are being observed, it would send output 1. Table 27, shows the rest of the BCILAB output integers that will stream to lab streaming layer and ultimately received by the mobile robotic arm. Finally, the '10' frequency output parameter corresponds to the rate at which the classifier will output integers 1 through 6. In this case, the classifier will send 10 integers per second. The next sections discuss the results from the robotic device receiving the output stream.

Table 27: 6-class controller reference table showing the input action and associated classifier output.

Brainwave Input Actions	Classifier Output Integers
RF	1
LF	2
RH	3
LH	4
JAW	5
REST	6

7.3.2 Control of Robotic Base

The robotic base python code initially used to perform goal seek commands was modified to read the command stream being generated by BCILAB. Having multiple configurations allow for the robotic base to vary in degrees of autonomy. In scenarios where the mobile robotic arm needs to travel in a different room, and the user is preoccupied with another task, the base can solely rely on its sensors to navigate successfully. However, when a more attention-demanding task is required of the robotic device, like grabbing a specific item in the desired room, switching over to the BCI configuration for a disabled user is required. In this configuration, the robotic base program is balancing the BCI command stream and physical sensor feedback. To prove that the robotic base is responsive to the BCI controller, the combined recordings was used. For brainwave input actions RF, LF, RH, and LH, the commands given to the base were to move 0.5 meters forward, backward, right, and left in a virtual environment built by the robot operating system (ROS) software. To increase the reliability and accuracy of the BCI controller configuration, the robotic base python code sampled 6 signals from the command stream before performing the desired action. As a result, if there is some case where the BCI classifier incorrectly identifies the brainwave input action, the robotic base program will take the majority of the 6 signals it saw. In this demonstration of pulling a string of movement commands, the robotic base was successfully able to move forward, backward, right, and left in the correct order. In Figure 64 and Figure 65, the robotic base program links with the BCI stream and initiates the forward move command. Note, while the JAW command is meant to transition BCI control to the robotic arm, this command does not need to be tested due to modular demonstrations of the mobile robotic arm. In addition, in the loop of string commands, the rest command was also implemented by the BCI controller and the robotic base remained on standby. With different samples of the BCI command stream and robotic base program being performed, it is difficult to accurately test the responsiveness of the BCI configuration. Other factors, like the computational power of the computer being used to run a simulated real time 14-channel brain signal stream, maintain a virtual environment, and calculate the next location of the robotic base prevent a meaningful time measurement. As a result, determining this aspect

of the BCI controller will be left to the physical demonstration that is separate from this study. Nevertheless, the BCI controller was successfully integrated to the robotic base.

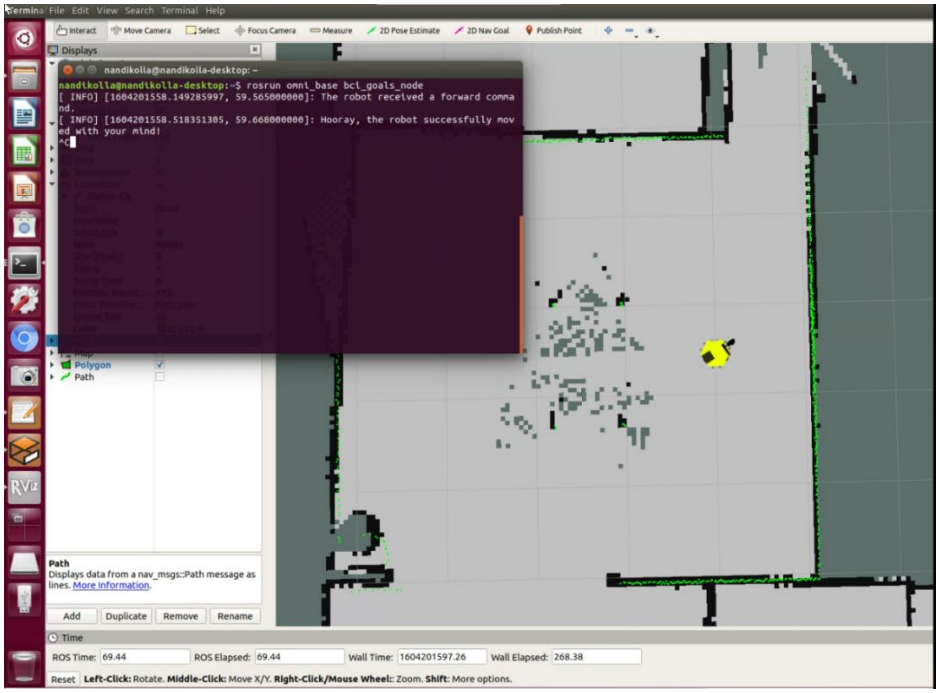


Figure 64: A screenshot of the ROS program showing a successful link to the BCI command stream.

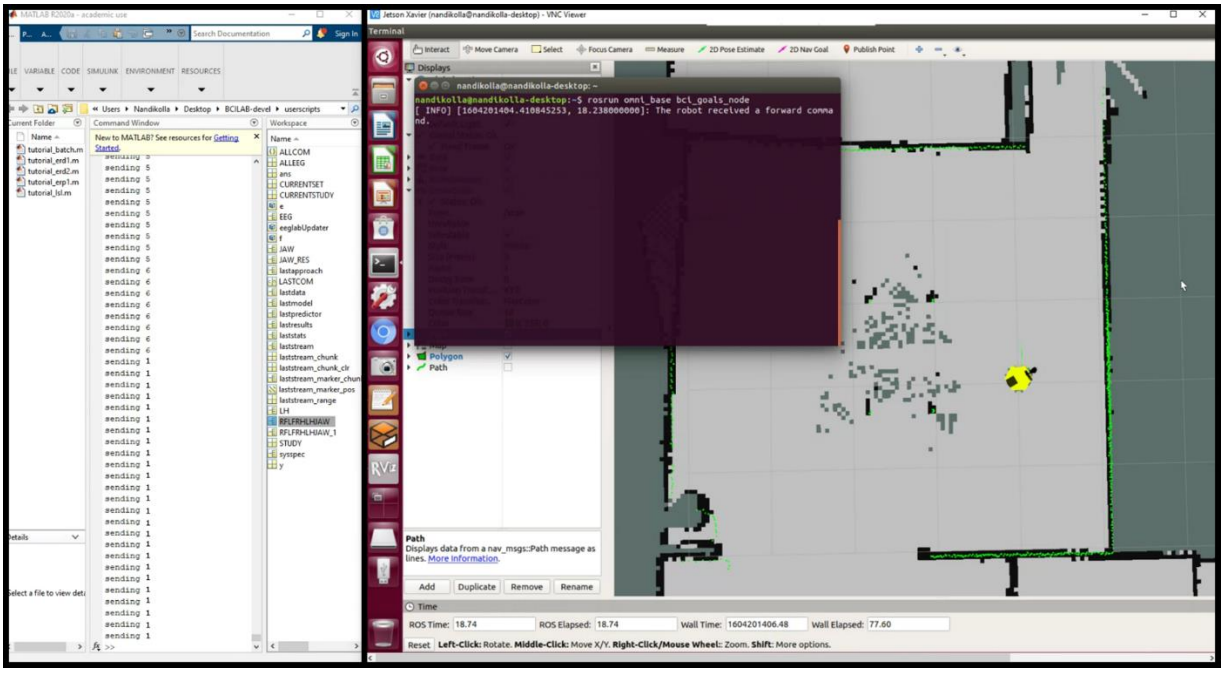


Figure 65: (Left) BCILAB plug-in outputting 1 (RF brainwave input). (Right) The robotic base initiating the move forward command.

7.3.3 Control of Robotic Arm, Guiding Cursor and Initiating Grab/Release Action

Similar to the base, the robotic arm python code initially used to perform goal seek commands was modified to read the command stream being generated by BCILAB. Per the intended design, the robot arm's individual motors that control the elbow, wrist, and fingers remain fully autonomous and controlled by the python code. Instead, the user will control the computer mouse's location on the robot's camera screen and initiate a grab/release point; as a result, this simplifies controlling the robotic arm, reduces user fatigue, and maintains the reliability of the arm program. In this configuration, the robotic arm program is using its kinematics and physical sensor feedback to grab or release an object after the BCI command is initiated. To prove that the robotic arm cursor is responsive to the BCI controller, the combined recordings was used. For brainwave input actions RF, LF, RH, and LH, the commands given to the arm cursor to move 10 pixels up, down, right, and left on the virtual camera screen by the ROS software. To increase the reliability and accuracy of the BCI controller configuration, the robotic base python code sampled 6 signals from the command stream before performing the desired action. In this demonstration of pulling a string of movement commands, the robotic arm cursor was successfully able to move up, down, right, and left in the correct order. In Figure 66, the robotic arm program initiates the mouse cursor to move right. Note, the JAW command was used to initiate the grab/release operation; this is visually shown by the appearance multiple vectors at the target location (see Figure 67). Note, the autonomous portion of the program would then take over and grab the object and switch back to the base robot afterwards. In addition, in the loop of string commands, the rest command was also implemented by the BCI controller and the robotic arm cursor remained on standby. Similar to the robotic base, the BCI configuration's responsiveness will have to be determined during a physical demonstration. Still, the BCI controller was successfully integrated to the robotic arm.

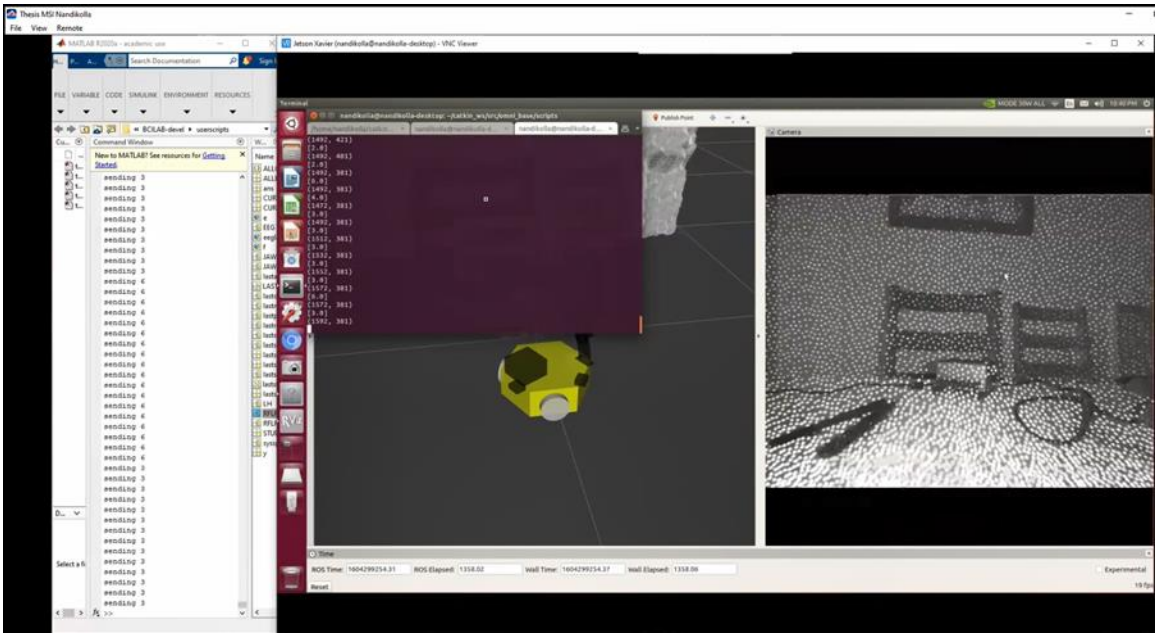


Figure 66: (Left) BCILAB plug-in outputting 3 (RH brainwave input). (Right) The robotic arm and camera screen with the cursor moving right. The cursor position points are recorded.

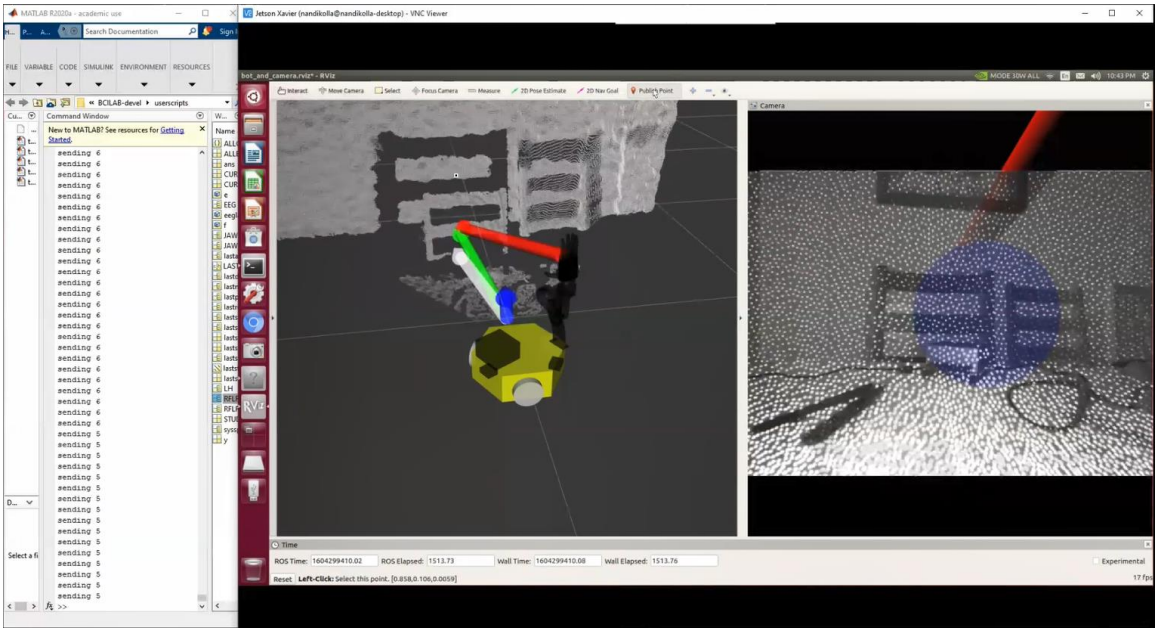


Figure 67: (Left) BCILAB plug-in outputting 5 (JAW brainwave input). (Right) The robotic arm and camera screen with the cursor initiating a grab/release point.

Chapter 8 - Conclusion and Discussion

Brain Computer Interface (BCI) controllers are gaining in popularity because they offer a hands and speech free solution for users who are physically impaired or work in extraordinary environments. Unlike traditional controllers, a BCI controller receives brainwaves as input commands from a noninvasive, or invasive, set of electrodes. The difficulty in using these signals is that they are nonstationary and nonlinear; as a result, studies in this field have resorted to machine learning algorithms that involve training a controller to recognize specific brainwave characteristics. While the parameters of these algorithms can be fine-tuned for better accuracy, a controller's accuracy and performance also depend on the user's ability to focus and conduct mental tasks consistently.

The focus of this study is to develop a hybrid BCI controller that operates a semi-autonomous mobile robotic arm. The motor imagery hand and foot electroencephalogram (EEG) signals and a jaw clench electromyogram (EMG) signal are used to operate the robot's base and arm separately. EEG signals serve two primary functions: they can navigate the robot base in an environment and move a cursor on the robot's camera screen to dictate what object to pick-up/drop-off. The user's jaw clench is used to switch command between the robot base and arm. To design a controller with this capability for multiple users, a compilation of hardware and software were used. A commercial grade EEG headset from Emotiv was used to record three healthy male subject's brainwaves responding to a stimulus presentation on OpenViBE, an open source acquisition software. Five sessions, each composed of four tests, were recorded to show the subjects' brainwaves to physical or imagine left/right hand and foot movement. EEGLAB, an open source pre-processing software used by many in the neurology field, was used to clean these recordings, apply independent component analysis to remove artifacts, and plot ERP graphs and topography maps to show brain activity. The cleaned recordings were then uploaded to BCILAB, a complimentary open source application that applies filters and classifiers to train a controller model. A CSP filter and machine learning algorithms LDA and RVM were chosen. The performance of each algorithm was measured and recorded for 2-class, 3-class, 5-class, and 6-class controller models. The best hands, feet, and jaw recordings for all three subjects, S01, S02, and S03, were down selected based on the 3-

class performances. Combining these files, a 5-class controller was created and the calculated error rate percentages after training with the RVM method were: 45% for subject S01, 30.8% for subject S02, and 29.2% per subject S03. These percentages show the difficult reality of creating a multi-class controller with accuracy. In addition, the performance results show that the RVM method outperformed LDA when the number of classes increased. Unfortunately, due to global and national pandemic policies, only subject S03 had the complete recordings to make the final 6-class controller for the mobile robotic arm. The final version of the 6-class controller had a 31.5% error rate after training. When applying recorded EEG signals to the controller model, the error percentage reduced to 18.9%. Note, these recordings were used in substitute of online testing since social distancing regulations were enforced. A final open source software, lab streaming layer, was used to output a command stream generated by the controller to Python, the primary language used to communicate to the mobile robotic arm. The final demonstration of the BCI controller was implemented on a virtual version of the semi-autonomous mobile robot arm. A modular approach was used to show the base and camera cursor can be moved using the controller and recorded brain wave data; in addition, a jaw clench recording was also used to show that the BCI controller can initiate a pick-up and drop-off command for the robot arm. These results show a 6-class hybrid BCI controller can be designed and implemented into a robotic design.

8.1 Previous BCI Research at CSUN

The BCI research done previously at CSUN has been influential in the development of the 6-class hybrid BCI controller. In 2018, Landavazo created a BCI controller to operate a 4-degree of freedom (DOF) robotic arm and 9-DOF hand system [14]. The robotic prototype included a hand-controlled mobile platform with a BCI-controlled robot mounted on top with a spring based support structure. An Xbox Kinect camera and sonar sensors were used to grab and manipulate a desired object. A subject participated in recording twenty-five sessions of each robot command. The controller's accuracy was 80% and 76% when performing four and five different cases. Although it is not clear what mental or physical tasks were used to create five unique beta and mu brainwaves, the following BCI controller did the following commands: (1) grab object, (2) release object,

(3) move object to the right, (4) move object to the left, and (5) neutral. As far as the hardware and software used to train the BCI controller, the standard off-the-shelf Emotiv Epoch+ Model 1.0 14 channel headset was worn on the subject and the brainwaves were recorded and trained on the Emotiv Testbench and MATLAB plug-in EEGLAB. This project did not pursue a real time testing due to the extra cost it would require to purchase software from Emotiv; instead, the recorded brainwave files were applied to MATLAB's machine learning toolbox where the algorithm that best fit the data was selected. When applying LDA, the probability for the controller to recognize the commands 1 through 4 were: 83.3%, 85.7%, 83.3%, and 66.7%. When applying the support vector machine (SVM) algorithm, the probability for the controller commands 1 through 5 were: 83.3%, 85.7%, 83.3%, 50%, and 80%.

In 2019, van Leeuwen created a hybrid BCI controller to operate a semi-autonomous electric wheel chair [15]. While the smart wheelchair was already built, the existing code was modified to process command streams from the controller. The controller relied on imagined hand motor imagery and a physical jaw clench for the following commands: turn left, turn right, start/stop, and rest. As far as the hardware and software used to train the BCI controller, the Emotiv Epoch+ Model 1.0 headset was modified using the EasyCap package and OpenViBE; OpenViBE was used to record, clean, and train the controller. MATLAB was used as an intermediary software to transmit commands to the smart wheelchair's code, LabVIEW. The electrode locations used for this project were: F7, F3, Fz, F4, F8, C3, Cz, C4, P7, P3, Pz, P4, P8, and Oz; the ground reference electrode locations were AFz and FCz for DRL and CMS accordingly. Because of OpenViBE's limitation in creating a two class LDA classifier, a fuzzy logic controller from the MATLAB toolbox was used to handle multiple classes. The complexity of the fuzzy controller lead to additional recordings comparing each action and a graphically coded logic tree. A male and female test subject, with an age range of 25 to 34 years old, were used to develop the controller. Ultimately, the controller's average positive rate for turning was 0.68 and 0.64 for Subject 1 and 2; the positive rate for executing the start/stop function was 0.5 and 0.38 for Subject 1 and 2. For both subjects, the rate of false positives was below 0.2; this project was able to demonstrate the electric wheelchair turning and stopping based on the recordings of the subjects' brainwaves.

8.2 Major Contributions Added to CSUN BCI Research

The development of the six-class RVM BCI controller builds off of the previous work done at CSUN by creating a single controller for multiple robot devices, analyzing brain signals in both individual channels and the entire scalp, investigating the impact of a user's hand preference, and overall software improvement. This project is designed to merge the previous robotic devices that is commanded by one controller. The preference of using the hand motor imagery and jaw clench tasks from van Leeuwan's controller were incorporated to foot motor imagery tasks. This allows the robot base to have an additional command of moving in reverse. Compared to Landavazo's controller of the robotic arm, the six-class RVM BCI controller is designed to have the user select a desired object to pick up or a location to drop off. The robotic system itself would then use the sensors and built-in code to activate the right motors for the desired action. This alternate approach was selected because other studies have shown that BCI arm controllers can create user fatigue if every motor was dependent on the brainwaves.

Another advancement in CSUN BCI research is the study of ERP plots and topographical maps. With the modified Emotiv Epoc headset, the electrode layout is based on the preferred tasks chosen for input signals. While van Leeuwan's work followed a suggested layout from EasyCap, there was not enough evidence in the project to determine if the locations were reactive. The six-class RVM BCI controller project used the MATLAB plug-in EEGLAB introduced in Landavazo's work and expanded its usage by incorporating the ICA, ERP plots, and topographical map functions. Now, a subject's recorded brainwaves can be studied and compared to other subjects who performed the same set of experiments. In addition, with added the need to study neuroscience and the brain's anatomy in more depth. This advancement was helpful in deducing that subject S03's foot motor imagery was not as effective as the other two subjects. In addition, the overlay of left and right, physical and imagined, tasks helped validate the reactive electrodes

In addition to the incorporation of ERP plots and topographical maps, this project also introduced subjects with different hand dominance. In van Leeuwan's work, the difference between a female and male's BCI controller performance was analyzed. This work selected subjects that were all one gender, male, but one subject was left-handed (and

the other two were right-handed). The ERP plots showed left-handed subject S03's hand motor imagery to be inversed when asked to squeeze, or imagine squeezing, the left hand. While this difference was observed, it requires additional left-handed subjects to be tested to see if the results are similar. In addition, the number of subjects used in this project was more than the previous works. The addition of one more subject significantly adds more work in analyzing the brain waves and developing a specific controller.

The final advancement in CSUN BCI research from this project is the incorporation of additional software to the development of BCI controllers. While OpenViBE can single handedly record, train, and develop a BCI controller on its own, the program only has basic functions. One of the desires in this paper is to clean recorded brainwaves and remove any unwanted artifacts. As a result, EEGLAB was introduced, but it was soon discovered that these clean data files cannot be implemented back into OpenViBE successfully. BCILAB, a software based on EEGLAB, was the solution to process clean data files and train more detailed machine learning algorithms. With these two programs being used, OpenViBE was only used for recording a user's training files. The addition of LSL allowed for a program to stream between different programs more easily instead of using MATLAB as the intermediary. It was discovered that LSL can talk to OpenViBE's acquisition server and has a built-in software call LabRecorder that can record a user's brainwaves; this can be used in conjunction with another stimulation program to create a training session that is not limited to two class recording in OpenViBE. It is LSL that is the possible link that can make online training possible without the need to purchase a license from Emotiv mentioned in Landavazo's work. In addition, it could be the solution to create a more streamlined offline training phase. Unfortunately, these pursuits were put on hold due to the social distancing policies. Nevertheless, the six-class RVM BCI controller, advanced CSUN BCI research in controller design, analyzing brainwaves, observing the impact of a subject's hand dominance, and forging a path for online testing.

8.3 Future Work and Recommendations

Due to pandemic regulations, the online testing and physical demonstration of the BCI controller and semi-autonomous mobile robot were placed on hold. As a result, the next step is to conduct live tests with a subject wearing the modified headset and ask

him/her to command the robot through the six-class controller in real time; this will solidify the virtual demonstrations in this paper. It was initially thought that the EEG headset needs to have the LSL Emotive headset application installed so the EEG streamed signals can be recognized; this requires a license from Emotiv to communicate with LSL which was released and available for purchase in August 2020. Mentioned earlier in this study, one workaround worth exploring is using OpenViBE's signal acquisition software, which is free and compatible with LSL. The OpenViBE Emotiv Epoch SDK used in van Leeuwan's work can link the headset and BCILAB through LSL. To make sure this link is possible, it is important to determine if incoming brainwaves are in a data type that can be read by BCILAB; if not, there may be some way in MATLAB or OpenViBE to change the data type. If this link is successful, then online testing phase can be testing via BCILAB's MATLAB visualization or virtual environment. With processing incoming data in real time, the computational power may need to be increased.

Another area of exploration is finding an alternate stimulus presentation software for recordings. OpenViBE's stimulation software was severely limited to showing two classes, a left and right arrow. As a result, to create a multi-class controller, multiple tests with different classes were recorded. In addition, after a test was finished, OpenViBE would stop the recording and save the file. For better continuity and performance, finding a stimulation presentation software and recorder that continues to record brainwaves between all these tests is ideal. One possible avenue to consider is LSL's LabRecorder app and SNAP, SCCN's open source stimulus presentation program [79]. The Simulation and Neuroscience Application Platform is a software environment in Python that can control simple experiment protocols. Here, the basic code can be adjusted to include multiple stimulation markers, faster tests with more trials, and conduct a series of tests while keeping the LabRecorder application running. The next step in offline training would be to replicate the experiments done in this project. If a single recording of each unique task, left/right, physical/imaginary, and hand/foot, can be saved for each subject, the data processing and analyzing will be more congruent and faster. Have the subjects' brainwaves recorded throughout the entire session, including breaks; this is better for all BCI software, including EEGLAB and BCILAB. The SNAP program can be edited to flash multiple letters on the screen; LH, RH, LF, and RF can be used and the order randomized if possible.

Having more trials of each test would lead to greater applications of machine learning algorithms.

Another area to explore is in understanding and training more machine learning algorithms. While LDA, SVM, and RVM have been used in CSUN BCI research, there are other variations or parameters that can be modified in these algorithms. Understanding which machine learning algorithms are better suited for multi-class controllers is important. BCILAB allows users to tweak parameters for these algorithms, but it requires knowledge in advance of how these parameters influencing the learning process. The next step in this case is to see if LDA and RVM algorithms can be further optimized and decrease the error rate percentage for a five or six class controller. Existing or new data can be used depending on the timeline for future work. For RVM, there are a list of hyper parameters that can increase in terms of polynomial order. BCILAB also includes some other more experimental versions of these machine learning algorithms that requires greater computational power and a greater number of electrodes. As a result, determining which algorithms are computationally available is worth investigating. All three of these recommendations would improve both the offline training and online testing phases and advance BCI research at CSUN.

8.4 Lessons Learned

A BCI controller's performance will be primarily dictated by the brain signals the developer wants to use. Simpler electromyogram signals like the jaw clench have shown incredible controller accuracy. Meanwhile, more complex signals like foot motor imagery can reach the same accuracy, but will often require more practice for users. To use motor imagery signals, it is preferred for developers to choose something that is intuitive so subjects can perform the action repeatedly. During initial research, subjects tapped their feet while wearing shoes and ended up shaking the headset itself during the recording; this problem was mitigated by asking the subjects to take off their shoes and tap with their socks. Similar to squeezing hands, another solution could be changing the task to curling the subjects' toes instead; both actions involve the subject's nerves that are extensively used for touching and feeling. Using an action that is tied to a sense will better help subjects

imagine. Still, because every human's brain activity is different, some selected signals may not be the best choice. As a result, achieving a higher performance controller may require observing and listening to the subject's feedback in order to identify a better signal.

For machine learning, obtaining more trials of each action is critical. The BCI controller developer must balance between collecting the maximum amount of data as possible, and minimizing subject fatigue. Prolonged tests with long trial timelines will rapidly deplete a subject's stamina. In fact, the recorded brainwaves will show the subject's loss of focus and directly impact the controller's accuracy. In the beginning stages of research, subjects were encouraged to perform two sessions in one sitting. However, the experiment settings were too long and left the subject tired. Subjects were then encouraged to perform one session a day; two sessions were possible if the subjects performed one in the morning and the other in the afternoon. To improve the subject's focus, it is better to create trials that last five to six seconds long instead of limiting the amount of sessions they can do; these trials are short enough to keep the subject's attention and nearly double the amount of data in a thirty-minute session. OpenViBE's visual stimulus presentation was too complex to create this desired trial time limit for this project; however, SNAP shows promising customization that will fit the desired trial time.

In the beginning stages of training the machine learning algorithms, there was an attempt to combine all the hand trials of one subject into one file; it was believed that this one file of trials would increase the accuracy of the controller. After several attempts, the BCILAB classifiers that learned from the combined file did not perform any better. One possible explanation for this surprising underperformance is that the combine data file is not an organic continuous set of brainwaves; in BCI controllers, machine learning algorithms have a preference to learn from a file that does not have breaks. As a result, creating a single data recording of a subject's uninterrupted brainwaves performing tests and taking breaks is the best scenario for machine learning algorithms.

One final lesson learned is the need for more electrodes on the scalp. While this project, and previous CSUN research studies, show a BCI controller can be created with the existing headset, having more electrodes increases the ability to detect other responses across the human brain. Topographical maps and ERP plots would show greater variation

in brain activity between electrodes in the frontal, temporal, parietal, occipital lobes, and sensory cortex. In addition, powerful computational analysis tools like ICA and other machine learning algorithms require a greater number of electrodes for better accuracy. For example, this project was not able to run variations of EEGLAB's ICA, like sobi and acsobiro. Having more options to clean and classify the data would help in developing more accurate controllers. One of the drawbacks to increasing the amount of electrodes is the increase of computational power required to analyze brainwaves and train machine learning algorithms. This project experienced intense computational demand when demonstrating control over the virtual mobile robot; at times, some of the software would either freeze or crash during the demonstration. As a result, purchasing a headset with more electrodes also requires actively monitoring how much these research computers can support. Another concern raised with increasing electrodes is the overlap of brain signals that creates additional noise in the recordings. While the threat mentioned is very real, the international 10-20 system established the minimum distance required to keep electrodes from interfering with one another. In addition, a sixty-four electrode layout is standardized and used extensively in neuroscience studies. As a result, as long as the number of electrodes does not exceed the international 10-20 system, the integrity of the recordings is maintained. Ultimately, the decision to purchase a headset with more electrodes must be weighed with subsequent costs like computer upgrades. In addition, if the agreed decision is to purchase another headset, it is important to see if existing software can link with the EEG electrode cap. The current headset company, Emotiv, corners its customers into purchasing licenses to fully communicate with the computer. Fortunately, this project and previous CSUN BCI research have been able to find workarounds.

There was an interest to see if brainwave recordings were more accurate electrodes were individually adhered to the human scalp instead of being pinned to a cap. This approach is used extensively in the medical field to monitor patients after brain surgery or during a sleep study. The trade-off when attaching individual electrodes to the scalp is the amount of time it takes to ensure the electrodes are in the right location. While the electrode-cap method generalizes the electrode location, it is quick to put on the subject. With the end-user being handicapped, applying the headset must be quick and easy enough

for the subject to do this on his/her own. The electrode headset may not be the final solution, but it is a step in the right direction in getting BCI controllers into the public.

Developing a BCI controller involves many technical variables, like software architecture and machine learning algorithms, and biological variables, like the condition of the subject's mental state. Understanding all of the variables at work, and their impact, gives greater appreciation to the human brain. Great patience and practice is required to develop a controller that will help humans who are prisoners to their own body. Ultimately, seeing these individuals regain their mobility and autonomy is worth the hardship.

References

[1] E. Yin, Z. Zhou, J. Jiang, Y. Yu and D. Hu, "A Dynamically Optimized SSVEP Brain Computer interface (BCI) Speller," *IEEE Transactions on Biomedical Engineering,* vol. 62, no. 6, pp. 1447-1456, 2015.

[2] F. Gembler, P. Stawicki, A. Saboor and I. Volosyak, "Dynamic time window mechanism for time synchronous VEP-based BCIs - Performance evaluation with a dictionary - supported BCI speller employing SSVEP and c-VEP," *PLoS ONE,* vol. 14, no. 6, pp. 1-10, 2019.

[3] A. Rakotomamonjy and V. Guigue, "BCI Competition III: Dataset II - Ensemble of SVMs for BCI P300 Speller," *IEEE Transactions on Biomedical Engineering,* vol. 55, no. 3, pp. 1147-1154, 2008.

[4] B. Kerous, F. Skola and F. Liarokapis, "EEG-based BCI and video games: a progress report," *Virtual Reality,* vol. 2018, pp. 119-135, 2018.

[5] R. Spataro, A. Chella, B. Allison, M. Giardina, R. Sorbello, S. Tramonte, C. Guger and V. La Bella, "Reaching and Grasping a Glass of Water by Locked-In ALS Patients through a BCI-Controlled Humanoid Robot," *Frontiers in Human Neuroscience,* vol. 11, no. 68, pp. 1-10, 2017.

[6] J. Tang, Y. Liu, D. Hu and Z. Zhou, "Towards BCI-actuated smart wheelchair system.," *BioMedical Engineering OnLine,* vol. 17, no. 1, 2018.

[7] I. Iturrate, M. Antelis, A. Kubler and J. Minguez, "A Noninvasive Brain-Actuated Wheelchair Based on a P300 Neurophysiological Protocol and Automated Navigation," *IEEE Transactions on Robotics,* vol. 25, no. 3, pp. 614-627, 2009.

[8] A. Kline and J. Desai, "Noninvasive Brain-Machine Interface to Control Both Mecha TE Robotic Hands Using Emotiv EEg Neuroheadset," *International Journal of Biomedical and Biological Engineering,* vol. 9, no. 4, pp. 323-327, 2015.

[9] D. Huang, K. Qian, D. Fei, W. Jia, X. Chen and O. Bai, ""Electroencephalography (EEG)-Based Brain- Computer Interface (BCI): A 2-D Virtual Wheelchair Control Based on Event-Related Desynchronization/Synchronization and Sate Control," *IEEE Transactions on Neural Systems and Rehabilitation Engineering,* vol. 20, no. 3, pp. 379-388, 2012.

[10] R. Leeb, D. Friedman, R. G. Muller-Putz, R. Scherer, M. Slater and G. Pfurtscheller, "Self-paced (Asynchronous) BCI Control of a Wheelchair in virtual Environments: A Case Study with a Tetraplegic," *Computational Intelligence and Neuroscience,* vol. 2007, 2007.

[11] C. Wang, B. Xia, J. Li, W. Yang, D. Xiao, A. C. Velez and H. Yang, "Motor Imagery BCI-based robot arm system," in *2011 Seventh International Conference on Natural Computation,* Shangai, 2011.

[12] R. Bousseta, I. El Ouakouak, M. Gharbi and F. Regragui, "EEG Based Brain Computer Interface for Controlling a Robot Arm Movement Through Thought," *IRBM,* vol. 39, no. 2, pp. 129-135, 2018.

[13] L. Minati, N. Yoshimura and Y. Koike, "Hybrid Control of a Vision-Guided Robot Arm by EOG, EMG, EEG Biosignals and Head Movement Acquired via a Consumer-Grade Wearable Device," *IEEE Access*, vol. 4, pp. 95828-9541, 2016.

[14] B. Landavazo, Brain-Computer Interface For Applications in Robotic Gripper Control, Northridge: California State University Northridge, 2018.

[15] T. K. van Leeuwen, Development of a Hybrid Brain-Computer Interface Controller for a Semi-Autonomous Wheelchair, Northridge: California State University Northridge, 2019.

[16] National Injury Prevention Foundation, "The Medulla," ThinkFirst, 2019. [Online]. Available: https://www.thinkfirst.org/youth-lesson3. [Accessed 7 October 2020].

[17] A. Ajibola, "Forebrain: Lobes of the Cerebrum and their functions," LEN Academy, 6 February 2020. [Online]. Available: https://www.len.com.ng/csblogdetail/481/Forebrain--Lobes-of-the-Cerebrum-and-their-Functions. [Accessed 7 October 2020].

[18] Northern Brain Injury Association, "Brain Structure and Function," NBIA, 2020. [Online]. Available: https://www.nbia.ca/brain-structure-function/. [Accessed 7 October 2020].

[19] E. R. Kandel, J. H. Schwartz and T. M. Jessell, "The Neural Basis of Cognition," in *Principles of Neural Science*, New York, McGraw-Hill, Health Professions Division, 200, pp. 324-325.

[20] The Editors of Encyclopaedia Britannica, "Hindbrain," Britannica, 18 February 2020. [Online]. Available: https://www.britannica.com/science/hindbrain. [Accessed 7 October 2020].

[21] P. Brodal, "Pons," SienceDirect, 2014. [Online]. Available: https://www.sciencedirect.com/topics/medicine-and-dentistry/pons. [Accessed 7 October 2020].

[22] Queensland Brain Institute, "The midbrain," The University of Queensland Australia, 4 July 2018. [Online]. Available: https://qbi.uq.edu.au/brain/brain-anatomy/midbrain. [Accessed 7 October 2020].

[23] R. Bailey, "Dvisions of the Brain: Forebrain, Midbrain, Hindbrain," ThoughtCo, 15 November 2019. [Online]. Available: https://www.thoughtco.com/divisions-of-the-brain-4032899. [Accessed 7 October 2020].

[24] The LibreTexts, "11.6A: Functions of the Diencephalon," Department of Eduation Open Textbook pilot Project, 13 August 2020. [Online]. Available: https://med.libretexts.org/Bookshelves/Anatomy_and_Physiology/Book%3A_Anatomy_and_Physiology_(Boundless)/11%3A_Central_Nervous_System/11.6%3AThe_Diencephalon/11.6A%3A_Functions_of_the_Diencephalon. [Accessed 7 October 2020].

[25] M. Irshad, "The Cerebrum," TeachMe Anatomy, 7 September 2018. [Online]. Available: https://teachmeanatomy.info/neuroanatomy/structures/cerebrum/. [Accessed 7 October 2020].

[26] Mayo Clinic, "Brain Lobes," Mayo Clinic, 2020. [Online]. Available: https://www.mayoclinic.org/brain-lobes/img-20008887. [Accessed 7 October 2020].

[27] J. Johnson and H. Moawad, "What to know about the temporal lobe," Medical News Today, 21 May 2020. [Online]. Available: https://www.medicalnewstoday.com/articles/temporal-lobe. [Accessed 7 October 2020].

[28] Centre for Neuro Skills, "Parietal Lobes," CNS, 2020. [Online]. Available: https://www.neuroskills.com/brain-injury/parietal-lobes/#:~:text=The%20parietal%20lobes%20can%20be,a%20single%20perception%20(cognition).. [Accessed 7 October 2020].

[29] M. Okamoto, H. Dan, K. Sakamoto, K. Takeo, K. Shimizu, S. Kohno, I. Oda, S. Isobe, T. Suzuki, K. Kohyama and I. Dan, "Three-dimensional probabilistic anatomical cranio-cerebral correlation via the international 10-20 system oriented for transcranial functional brain mapping," *NeuroImage,* vol. 21, no. 1, pp. 99-111, 2004.

[30] B. Edelman, B. Baxter and B. He, "Decoding and mapping of right hand motor imagery tasks using EEG source imaging," *2015 7th International IEEE/EMBS Conference on neural Engineering (NER),* pp. 194-197, 2015.

[31] S. Pilgramm, B. de Hass, H. Fabian, K. Zentgraf, R. Stark, J. Munzert and B. Kruger, "Motor Imagery of Hand Actions: Decoding the Content of Motor Imagery From Brain Activity in Frontal and Parietal Motor Areas," *Human Brain Mapping,* vol. 37, pp. 81-93, 2016.

[32] S. Borchers, M. Himmelbach, N. Logothetis and H.-O. Karnath, "Direct electrical stimulation of human cortex--the gold standard for mapping brian functions?," *Nature Review Neuroscience,* vol. 13, no. 1, pp. 63-70, 2012.

[33] Brain Facts, "The Neuron," Brain Facts, 1 April 2012. [Online]. Available: https://www.brainfacts.org/brain-anatomy-and-function/anatomy/2012/the-neuron. [Accessed 7 October 2020].

[34] R. Mackenzie, "Gray Matter vs White Matter," Technology Networks, 20 August 2019. [Online]. Available: https://www.technologynetworks.com/neuroscience/articles/gray-matter-vs-white-matter-322973. [Accessed 7 October 2020].

[35] C. Freudenrich and R. Boyd, "How Your Brain Works," HowStuffWorks, 6 June 2001. [Online]. Available: https://science.howstuffworks.com/life/inside-the-mind/human-brain/brain.htm. [Accessed 7 October 2020].

[36] Centre for Synaptic Plasticity, "Brain Basics: The fundamentals of neuroscience," University of Bristol, 27 September 2011. [Online]. Available: http://www.bris.ac.uk/synaptic/basics/basics-2.html. [Accessed 7 October 2020].

[37] My-Ms, "Brain Anatomy-Part 3," My-Ms, 2020. [Online]. Available: https://my-ms.org/anatomy_brain_part3.htm. [Accessed 7 October 2020].

[38] S. Pappas, "Why is Gray Matter Gray?," Live Science, 24 May 2010. [Online]. Available: https://www.livescience.com/32605-why-is-gray-matter-gray.html#:~:text=In%20contrast%2C%20gray%20matter%20is,nutrients%20and%20energy%20to%20neurons.&text=Because%20these%20cells%20are%20not,the%20neurons%20and%20glial%20cells.. [Accessed 7 October 2020].

[39] R. Bailey, "Anatomy of the Brain: Your Cerebrum," ThoughtCo, 5 March 2018. [Online]. Available: https://www.thoughtco.com/anatomy-of-the-brain-cerebrum-373218. [Accessed 7 October 2020].

[40] EMOTIV, "EEG Definition," EMOTIV, 2020. [Online]. Available: https://www.emotiv.com/eeg-guide/#:~:text=What%20is%20an%20EEG%3F,electrical%20activity%20in%20your%20brain.. [Accessed 7 October 2020].

[41] E. H. Chudler, "10-20 System of Electrode Placement," Neuroscience for Kids, 2012. [Online]. Available: https://faculty.washington.edu/chudler/1020.html. [Accessed 7 October 2020].

[42] Cleveland Clinic, "Invasive EEG Monitoring," Cleveland Clinic, 2020. [Online]. Available: https://my.clevelandclinic.org/health/diagnostics/17144-invasive-eeg-monitoring. [Accessed 7 October 2020].

[43] Neuralink, "Interfacing with the Brain," Neuralink, 2020. [Online]. Available: https://neuralink.com/approach/. [Accessed 7 October 2020].

[44] G. Wang and D. Ren, "Effect of Brain-to-Skull Conductivity Ratio on EEG Source Localization Accuracy," *BioMed Research International,* vol. 2013, pp. 1-10, 2013.

[45] M. Vollrath and M. Guevara, "Biomedical Signals Acquisition," McGill Physiology Virtual Lab, 2020. [Online]. Available: https://www.medicine.mcgill.ca/physio/vlab/biomed_signals/eeg_n.htm. [Accessed 7 October 2020].

[46] E. GmbH, "Brain Vision UK," [Online]. Available: http://brainvision.co.uk/products/products-by-manufacter/easycap-gmbh.

[47] Emotiv, "EMOTIV EPOC & TESTBENCH SPECIFICATIONS," 2014. [Online]. Available: www.emotiv.com/files/Emotiv-EPOC-Product-Sheet-2014.pdf. [Accessed June 2019].

[48] L. Deecke, H. Weinberg and P. Brickett, "Magnetic Fields of the Human Brain Accompanying Voluntary Movement: Bereitschaftsmagnetfeld," *Experimental Brain Research,* vol. 48, pp. 144-148, 1982.

[49] C. Neuper and G. Pfurtscheller, "Evidence for distinct beta resonance frequencies in human EEg related to specific sensorimotor cortical areas," *Clinical Neurophysiology,* vol. 112, pp. 2084-2097, 2001.

[50] Y. Renard, F. Lotte, M. Congedo, E. Maby, V. Delannoy, O. Bertrand and A. Lecuyer, "OpenViBE: An Open-Source Software Platform to Design, Test and Use Brain-Computer Interfaces in Real and Virtual Environments," *Presence: teleoperators and virtual environments,* vol. 19, no. 1, 2010.

[51] C. Riou, "Designer Overview," 30 October 2017. [Online]. Available: http://openvibe.inria.fr/designer/. [Accessed 19 September 2020].

[52] A. Delorme and S. Makeig, "EEGLAB: an open source toolbox for analysis of single-trial EEG dynamics," *Journal of Neuroscience Methods,* vol. 134, pp. 9-21, 2004.

[53] Swartz Center for Computational Neuroscience, "What is EEGLAB?," University of California, San Diego, [Online]. Available: https://sccn.ucsd.edu/eeglab/index.php. [Accessed 20 September 2020].

[54] M. X. Cohen, Analyzing neural Time Series Data: Theory and Practice, Cambridge: The MIT Press, 2014.

[55] S. Makeig, S. Debener, J. Onton and A. Delorme, "Mining event-related brain dynamics," *Tends in Cognitive Science,* vol. 8, pp. 204-210, 2004.

[56] M. Miyakoshi, "Makoto's preprocessing pipeline," 27 November 2019. [Online]. Available: https://sccn.ucsd.edu/wiki/Makoto's_preprocessing_pipeline. [Accessed 20 September 2020].

[57] Ibonnet, "Stimulation Codes," OpenViBE, 2 September 2011. [Online]. Available: http://openvibe.inria.fr/stimulation-codes/. [Accessed 20 September 2020].

[58] A. Delorme, "Infomax Independent Component Analysis for dummies," [Online]. Available: http://arnauddelorme.com/ica_for_dummies/. [Accessed 20 September 2020].

[59] A. Delorme, J. Palmer, R. Oostenveld, J. Onton and S. Makeig, "Independent EEG components are dipolar," *PLos One*, 2012.

[60] A. Delorme, *ICA applied to EEG part 1: What is ICA?,* San Diego: EEGLAB, 2020.

[61] A. Delorme, "ICA applied to EEG part 2: How does infomax ICA work?," EEGLAB, San Diego, 2020.

[62] C. A. Kothe and S. Makeig, "BCILAB: A platform for Brain-Computer interface development," *Journal of Neural Engineering,* vol. 10, 2013.

[63] SCCN, "BCILAB Wiki," MediaWiki, 20 April 2017. [Online]. Available: https://sccn.ucsd.edu/wiki/BCILAB#Introduction. [Accessed 24 September 2020].

[64] C. Kothe, "Lecture 3.2 Major Filter Classes," The Qualcomm Institute, 31 July 2013. [Online]. Available: https://www.youtube.com/watch?v=EvVEC7O6zlc&list=PLbbCsk7MUIGcO_lZMbyymWU2UezVHNaMq. [Accessed 26 September 2020].

[65] C. Kothe, "Lecture 7.3 Common Spatial Patterns," The Qualcomm Institute, 1 August 2013. [Online]. Available: https://www.youtube.com/watch?v=zsOULC16USU&list=PLbbCsk7MUIGcO_lZMbyymWU2UezVHNaMq. [Accessed 26 September 2020].

[66] C. Kothe, "Lecture 4.3 Concrete Case Study," The Qualcomm Institute, 5 August 2013. [Online]. Available: https://www.youtube.com/watch?v=khDj5gHfkAo&list=PLbbCsk7MUIGcO_lZMbyymWU2UezVHNaMq&index=24. [Accessed 26 September 2020].

[67] M. E. Tipping, "The Relevance Vector Machine," *Journal of Machine Learning Research,* vol. 1, pp. 211-244, 2001.

[68] J. Brownlee, "One-vs-Rest and One-vs-One for Multi-Class Classification," Machine Learning Mastery, 13 April 2020. [Online]. Available: https://machinelearningmastery.com/one-vs-rest-and-one-vs-one-for-multi-class-classification/. [Accessed 27 September 2020].

[69] C. A. Kothe, "Demo 1 The Lab Streaming Layer," The Qualcomm Institute, 31 July 2013. [Online]. Available: https://www.youtube.com/watch?v=Y1at7yrcFW0. [Accessed 27 September 2020].

[70] C. A. Kothe, D. Medine, C. Boulay, M. Grivich and T. Stenner, "LabStreamingLayer's Documentation," SCCN, 2019. [Online]. Available: https://labstreaminglayer.readthedocs.io/index.html. [Accessed 27 September 2020].

[71] C. Boulay, "Install," GitHub Inc., 19 October 2019. [Online]. Available: https://github.com/sccn/labstreaminglayer/wiki/INSTALL. [Accessed 28 September 2020].

[72] Microsoft, "Welcom to the Visual Studio IDE," Microsoft, 19 March 2019. [Online]. Available: https://docs.microsoft.com/en-us/visualstudio/get-started/visual-studio-ide?view=vs-2019. [Accessed 28 September 2020].

[73] CMake, "CMake About," CMake, 2020. [Online]. Available: https://cmake.org/. [Accessed 28 September 2020].

[74] B. Dawes, D. Abrahams and R. Rivera, "boost c++ libraries," Boost Software License, August 14 2020. [Online]. Available: https://www.boost.org/. [Accessed 28 September 2020].

[75] The QT Company, "Qt," The QT Company, 2020. [Online]. Available: https://www.qt.io/. [Accessed 28 September 2020].

[76] Python, "Project Description: pylsl," Python, 2020. [Online]. Available: https://pypi.org/project/pylsl/. [Accessed 29 September 2020].

[77] S. Sur and V. K. Sinha, "Event-related potential: An overview," *Industrial Psychiatry journal*, vol. 18, no. 1, pp. 70-73, 2009.

[78] M. Moayedi, M. Liang, A. L. Sim, L. Hu, P. Haggard and G. D. Iannetti, "Laser-Evoked Vertex Potentials Predict Defensive Motor Actions," *Cerebral Cortex*, vol. 25, no. 12, pp. 4789-4798, 2015.

[79] C. Kothe, "Simulation and Neuroscience Application Platform," SCCN, 31 January 2013. [Online]. Available: https://github.com/sccn/SNAP. [Accessed 6 October 2020].

APPENDIX A

A1. "bciStream.py" script for receiving command data stream via stream type.

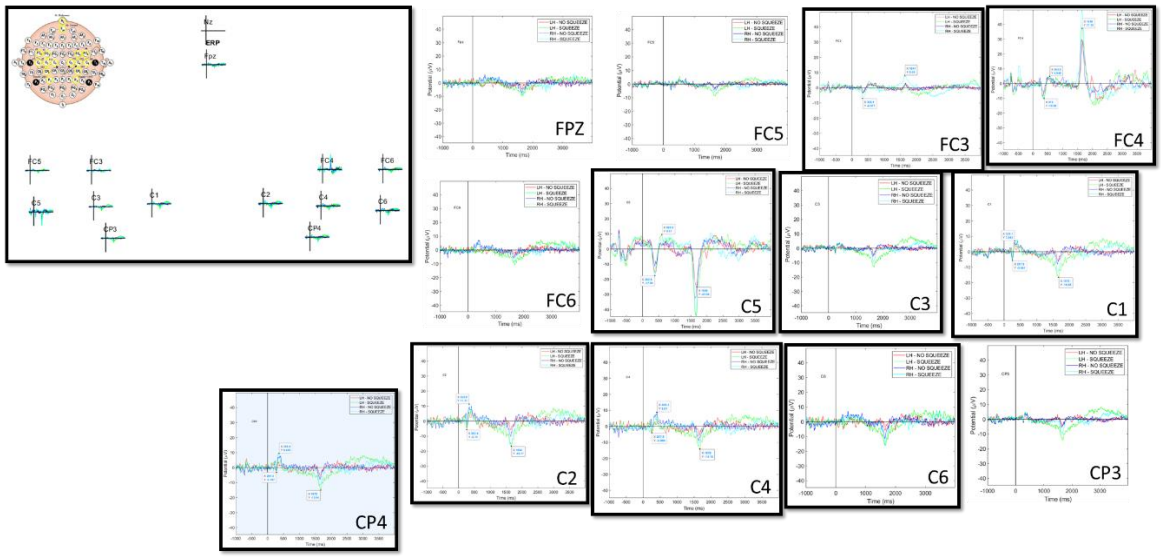
```
"""Example program that shows how to attach meta-data to a stream, and how to
later on retrieve the meta-data again at the receiver side."""

import time

from pylsl import StreamInfo, StreamOutlet, StreamInlet, resolve_stream

# === the following could run on another computer ===

# first we resolve a stream whose name is MetaTester (note that there are
# other ways to query a stream, too - for instance by content-type)
stream_info  = resolve_stream('type', 'MentalState')

# open an inlet so we can read the stream's data (and meta-data)
my_inlet = StreamInlet(stream_info[0])

info=my_inlet.info()
# print("The stream's XML meta-data is: ")
print(info.as_xml())

while True:
    chunk, timestamps = my_inlet.pull_chunk()
    if timestamps:
        print(timestamps, chunk)

time.sleep(3)
```

A2. "bciQuery.py" script for receiving command data stream via stream name.

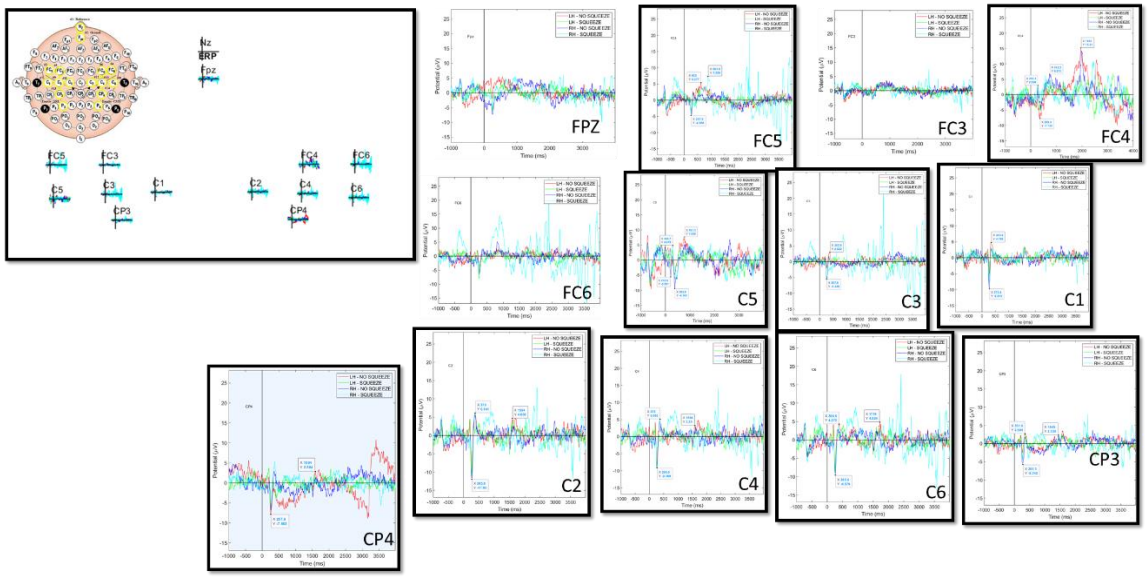
```
"""Example program that shows how to attach meta-data to a stream, and how to
later on retrieve the meta-data again at the receiver side."""

import time

from pylsl import StreamInfo, StreamOutlet, StreamInlet, resolve_stream

# === the following could run on another computer ===

# first we resolve a stream whose name is MetaTester (note that there are
# other ways to query a stream, too - for instance by content-type)
results = resolve_stream("name", "bci")

# open an inlet so we can read the stream's data (and meta-data)
inlet = StreamInlet(results[0])

# get the full stream info (including custom meta-data) and dissect it
info = inlet.info()
# print("The stream's XML meta-data is: ")
print(info.as_xml())
# print("The manufacturer is: %s" % info.desc().child_value("manufacturer"))
# print("Cap circumference is: %s" % info.desc().child("cap").child_value("size"))
# print("The channel labels are as follows:")
# ch = info.desc().child("channels").child("channel")
# for k in range(info.channel_count()):
#     print("  " + ch.child_value("label"))
#     ch = ch.next_sibling()

time.sleep(3)
```

APPENDIX B

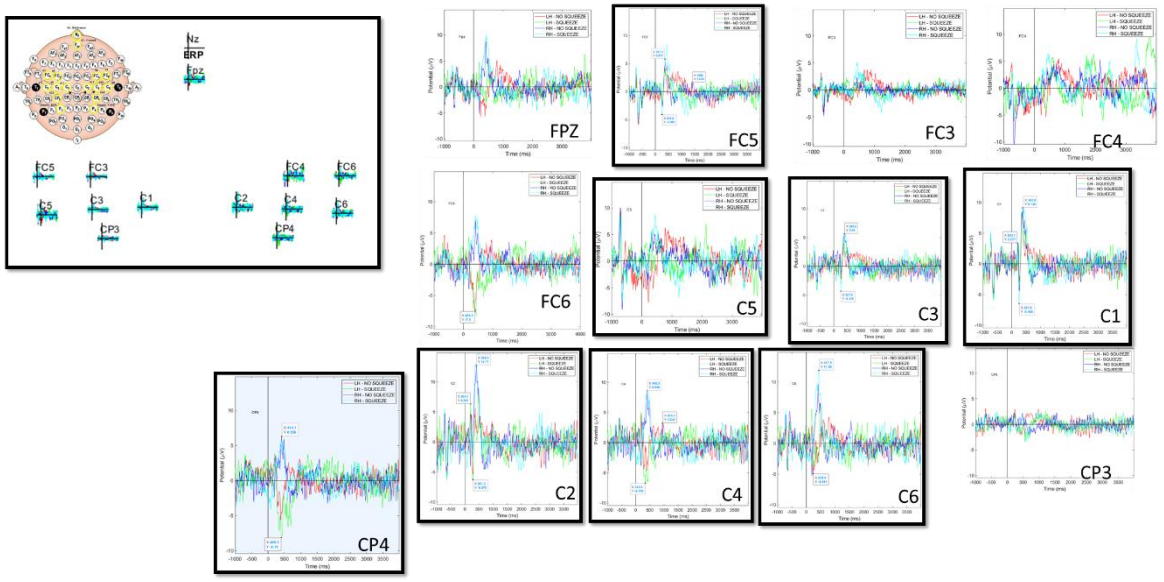
B1. Subject S01 LH/RH: Physical vs Imaginary.



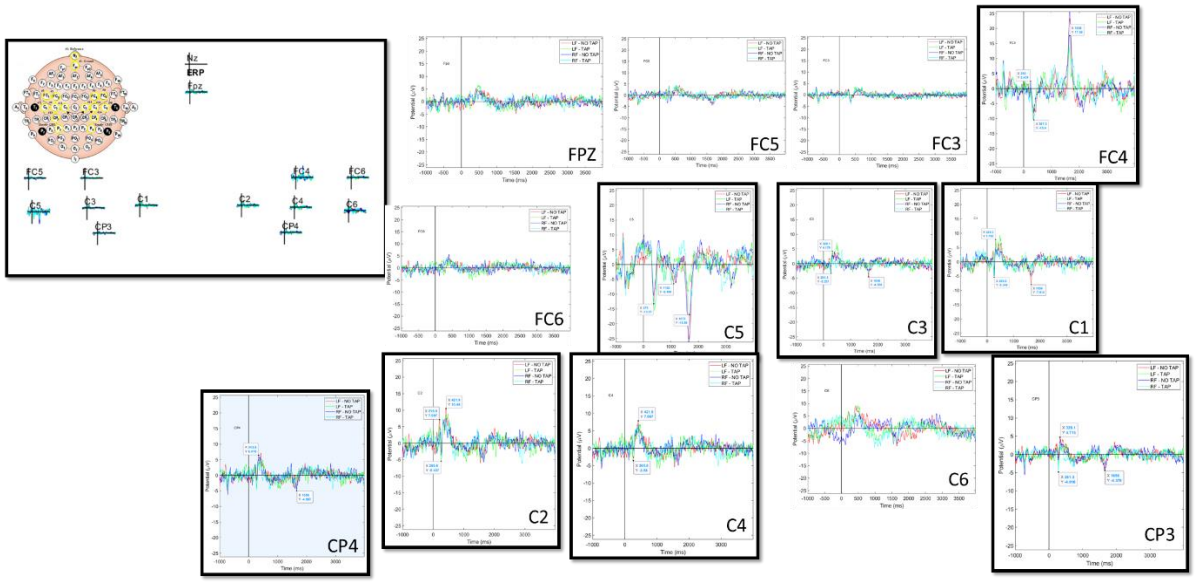
B2. Subject S02 LH/RH: Physical vs Imaginary.



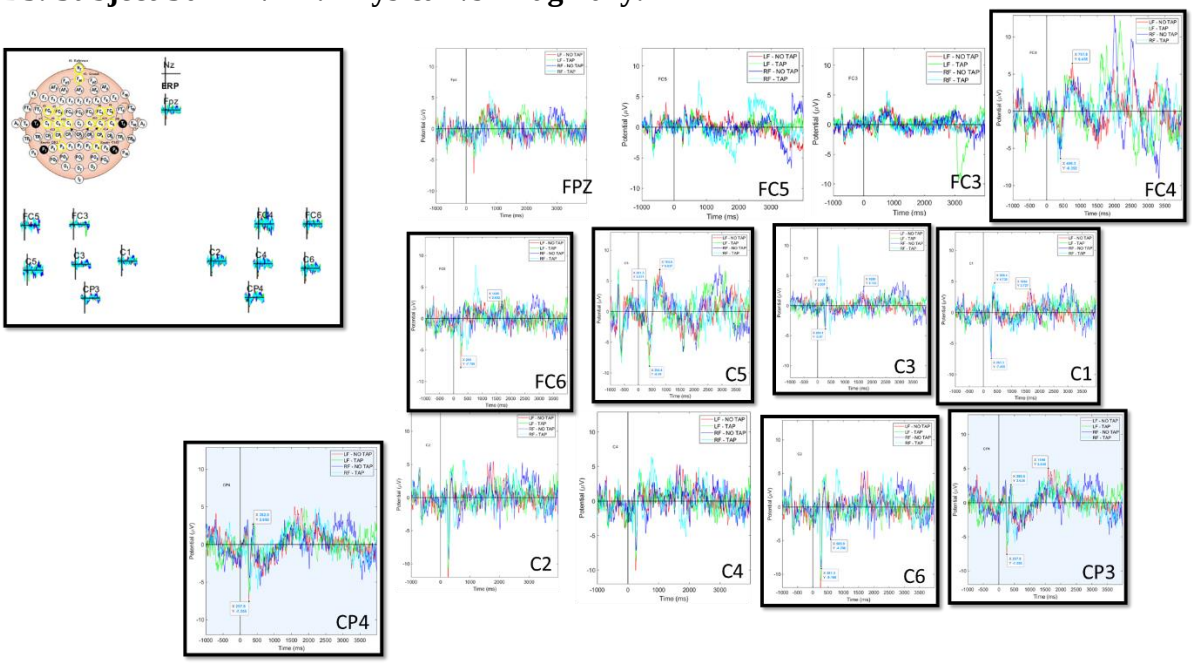
B3. Subject S03 LH/RH: Physical vs Imaginary.



B4. Subject S01 LF/RF: Physical vs Imaginary.



B5. Subject S02 LF/RF: Physical vs Imaginary.



B6. Subject S03 LF/RF: Physical vs Imaginary.