DEVELOPMENT AND EVALUATION OF A BRAIN-COMPUTER INTERFACE FOR HUMAN-ROBOT INTERACTION IN SIMULATION AND HARDWARE ENVIRONMENTS

by

SHANE WHITAKER

Presented to the Faculty of the Graduate School of
The University of Texas at Arlington in Partial Fulfillment
of the Requirements
for the Degree of

MASTER OF SCIENCE

THE UNIVERSITY OF TEXAS AT ARLINGTON

December 2020

To my amazing parents, Dennis and Guyette, who have provided everything I ever needed to get to where I am today. To my beautiful wife, Teresa, for her love and support.

ACKNOWLEDGEMENTS

I would like to thank my supervising professor Dr. Panos Shiakolas for his invaluable guidance, ideas, and motivation throughout the course of my research. I would like to thank Dr. Seiichi Nomura, Dr. Kent Lawrence, and Dr. Ioannis Schizas for their interest in my research and for taking time to serve on my thesis committee. I would like to thank the members of the Manufacturing Automation and Robotic Systems (MARS) lab for their insight and feedback on my research. I would like to thank Shubham Gunjal for his assistance with Webots.

I would like to thank my parents for their interest in my education throughout my life. Finally, I would like to thank my wife and parents for their encouragement during this process.

December 4, 2020

ABSTRACT

DEVELOPMENT AND EVALUATION OF A BRAIN-COMPUTER INTERFACE FOR HUMAN-ROBOT INTERACTION IN SIMULATION AND HARDWARE ENVIRONMENTS

Shane Whitaker, MS

The University of Texas at Arlington, 2020

Supervising Professor: Panos Shiakolas

The aim of Brain-Computer Interface (BCI) research is to create a communication system that identifies human intent by processing brain signals with the objective to develop a control signal for an external device, in this case a robotic arm. In this research, a framework to acquire, process, evaluate, and map BCI signals to a specific process is developed and tested in software and hardware. The BCI used is the Emotiv EPOC+, a non-invasive 14-electrode electroencephalogram (EEG) headset, which is also equipped with additional sensors to detect facial expressions and head movement. The development and testing of the interface is primarily performed in Webots, a robot simulation environment. The simulation environment provides a platform to analyze the reproducibility of the EEG or other signals, for a particular action. A pick and place process utilizing mental commands, facial expressions, and head movement was successfully demonstrated in a Webots simulation and seamlessly transferred to the proof of concept robotic hardware. The success of the multiple ex-

periments validates the developed BCI framework and provides a solid foundation for further research into Human-Robot Interaction (HRI).

The ultimate goal of researching this BCI is to further enhance the field of HRI, particularly in assistive robotics. BCI devices could provide the means to help those who rely on others for seemingly simple and routine daily tasks, such as picking up a bottle of water or manipulating other objects in their environment.

TABLE OF CONTENTS

LIST OF ILLUSTRATIONS

LIST OF TABLES

CHAPTER 1

Introduction

The aim of this research is to expand upon the development of Human-Robot Interaction (HRI) through an Electroencephalography (EEG) Brain-Computer Interface (BCI), particularly for assistive applications. As robots become more integrated into our lives and widely available in the future, it is important to consider how humans interact with them to generate a desired behavior. Ease of use is important as most people are not robot programmers, therefore other methods of interaction are needed. Which is why the research into these different HRI modalities is critical for widespread use of robots, and more importantly assistive type robots.

The high-level framework developed for this research is shown in figure 1.1. This flowchart outlines the process of extracting EEG signals from a user generated command and converting it into a robotic process. Once the signals are extracted from the EEG device, they are then processed and the command is identified. The identified command is passed to the robotic controller, where it is mapped to a particular process to be performed by the robot. This robotic process can be tested in simulation, and once satisfactory performance is achieved, then the process can be transferred to the hardware.

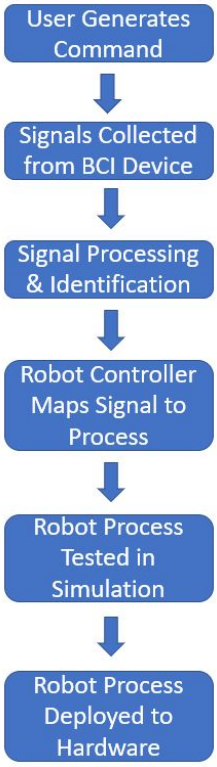

Figure 1.1. BCI Framework Flowchart.

1.1 Motivation

Assistive robotics is extremely important to improve quality of life for a number of people who have lost the ability to control their muscles due to a number of reasons such as strokes, spinal cord injuries, or muscular dystrophy. While these people suffering from these disorders lose motor control, a majority of them retain the ability to produce the motor function related brain activity similar to a healthy individual [2]. In addition to these neuromuscular disorders, amputees also fall into this category of people who could greatly benefit from an assistive robot. The number of people in the United States living with some form of paralysis is approximately 5,400,000 [9], and an estimated 185,000 people undergo an amputation every year [10]. In the case of paralysis, an end user may benefit from being able to control a robotic

arm to manipulate objects in their environment. For the case of upper limb amputees, they could benefit from being able to control a prosthetic robotic hand though a BCI. The numerous control signals that could be generated from a BCI would increase the level of dexterity for a robotic hand therefore enabling self dependence.

1.1.1 Future of BCI

While this research focuses on the non-invasive EEG BCI, I am inspired by the future state of BCIs particularly the work being done by the company Neuralink. The focus of Nerualink's research is to develop a minorly invasive device capable of establishing a high bandwidth communication with the brain [11]. While still in its early stages, once this breakthrough happens, it will make the control of robots from mental commands effortless. This technology will play a major role in the advancement of assistive and telerobotics. In assistive robotics it will allow for the precise control of robotic arms and hands. Telerobotics is the act of controlling a robot from a distance to perform various tasks. A high bandwidth BCI would allow a human to essentially transport their senses into a robot and remotely perform a complex task as if they were there. Telerobotics has several significant applications including space exploration, disaster relief, and medical procedures. However, until this type of invasive method is proven to be safe and inexpensive, the non-invasive EEG is likely to be a favorable alternative.

1.2 Electroencephalography

Electroencephalography (EEG) is an electrophysiological process that records the electrical activity of the brain [12]. This electrical activity occurs when the billions of neurons in our brain communicate with each other producing changes in voltage, to form non-linear patterns called brainwaves. Measuring the brain's electrical activity

through a non-invasive process involves placing EEG electrodes on the scalp. The EEG signals are amplified and then sent to a computer for data processing. This method measures the electrical activity in the outer layer of the brain, known as the cerebral cortex. The cerebral cortex is made up of three types of areas: sensory, motor and association areas. These areas of the brain are where most information processing occurs, and they account for a majority of behavior and human cognition. The human brain is constantly processing and absorbing information even during sleep, which allows the EEG sensors to detect changes in the brainwaves even in the absence of visual behavior responses, such as facial expressions or movement. EEG has the advantage of having a high temporal resolution which allows it to pick up on the rapid reactions in the brain that can be at the speed of milliseconds. For performing research on BCIs, EEG headsets are practical due to their portable capability and are relatively inexpensive compared to other non-invasive methods, such as functional Magnetic Resonance Imaging (fMRI) or magnetoencephalography (MEG).

1.3 Literature Survey

The research into EEG based BCIs has grown over the past few years as more EEG devices have become available at a reasonable price. The Emotiv EPOC has shown to be one of the most used EEG devices in academic research, probably due to its relatively low cost. There has been a number of studies that used an Emotiv EEG headset for some form of robotic control [13–20]. Several of these have utilized a legacy Emotiv Software Development Kit (SDK), which allowed access to the raw EEG data [13–19]. However, the SDK has been discontinued and is no longer supported. The raw EEG data is now only accessible through a subscription with Emotiv.

One study attempted to use four mental commands to have a simple mobile robot move forward, back, left, and right [13]. However, in this case several of the

subjects had trouble triggering all of the commands and were only able to get one with an average success rate over 80%. The study was performed with three groups of people: age 14-20, age 21-30, and disabled. The disabled group had the highest success rate for each of the four mental commands, demonstrating higher mental strength.

Research performed at the University of Dayton (UD) attempted to map four mental commands to the lift, lower, rotate left, and rotate right movement of a robotic end effector [19]. A majority of subjects were only able to manage one to two mental commands with any accuracy, and had difficulty triggering more. Additional research performed at UD, showed that mental commands can be trained easier when associated with hand gestures such as holding the index finger and thumb as close together as possible without touching [14]. When performing this action with the left and right hands, it possible to achieve high quality training with two mental commands.

Two of the studies that utilized Emotiv EEG headsets relied on eye winks, facial expressions, and data from the gyroscope to control a simple robot arm [17, 20]. In both cases subjects were able complete various tasks with an accuracy of over 80%. In one of these cases, a LabVIEW Emotiv Toolkit allowed for extraction of several data streams [17]. This LabVIEW Emotiv Toolkit was investigated early in this research, however it did not work as well since the SDK was being phased out.

Research at Carnegie Mellon University (CMU) has shown that complicated 2D tracking of a robotic end effector control can be accomplished with a dense EEG headcap [1]. The EEG headcap used in these experiments had 128-channels to provide a dense map of the electrical activity. In these experiments, the subject's goal was to follow a virtual target that was presented on a screen in the horizontal and vertical position. The first part of the experiment was to have the subject control a

mouse on the screen to track the target. Once this was successfully verified, a 7-axis JACO robotic arm was introduced to demonstrate that control of the virtual mouse translated to accurate control of the robot end effector.

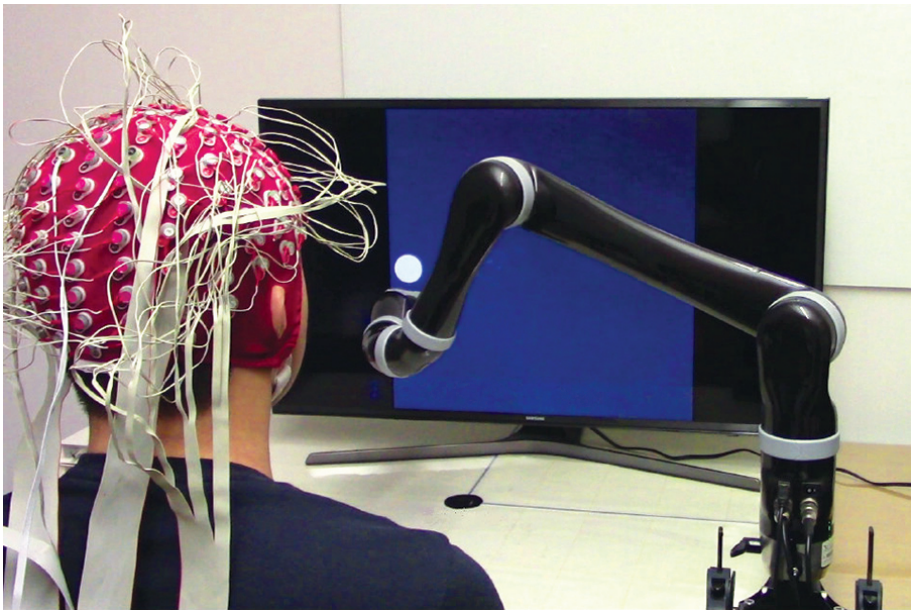

Figure 1.2. CMU Experiment Setup with 128 Electrodes [1].

A study performed at the University of Minnesota (UMN) demonstrated that 13 subjects could perform robotic manipulation of different colored foam blocks by utilizing a 64-channel EEG cap [2]. During these experiments, the foam blocks were placed in fixed locations for the robot to pick up. To execute motion of the robot, a virtual cursor was displayed on a computer screen that was visible to the subject. Each subject was instructed during training to imagine movement of their left hand, right hand, both hands, or relaxation of both hands to control the left, right, up and down movement of the cursor and robotic arm, respectively. The subjects were able to instruct the robot to grab the foam blocks roughly 74% of the time. A setup of this experiment is shown in figure 1.3.

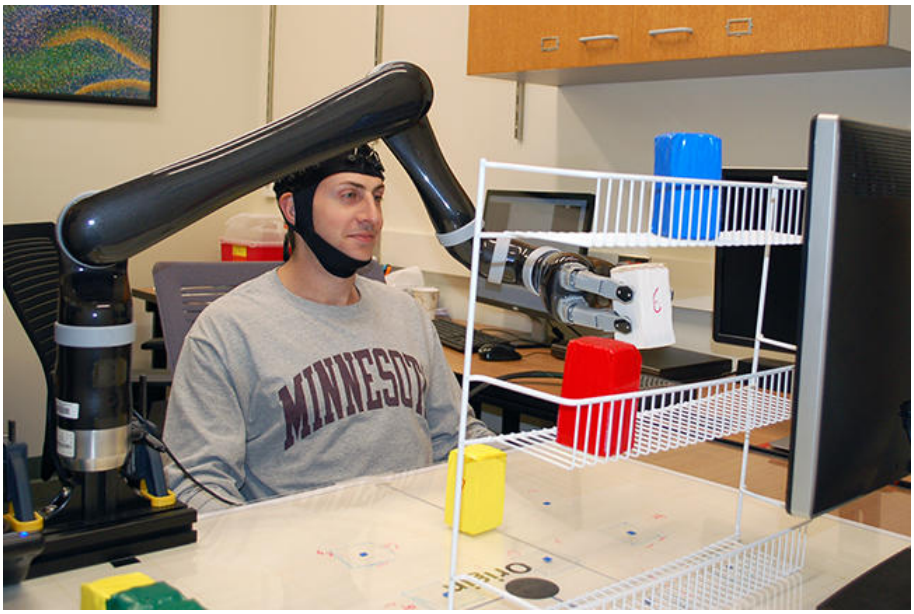

Figure 1.3. UMN Experiment Setup with 64 Electrodes [2].

1.4 Thesis Outline

Chapter 2 provides an overview and discusses the hardware and software tools used for this research. Additionally, it covers the kinematic analysis of the robot used during the hardware experiments. Chapter 3 focuses on how the BCI is trained for mental commands and facial expressions. Also, this chapter introduces the software architecture and algorithms developed to turn the commands from the EEG headset into robotic actions. Chapter 4 discusses the experiments performed using the EEG headset to control the simulated and physical robots. Chapter 5 provides conclusions and recommendations for future research.

CHAPTER 2

Hardware and Software Tools

This chapter introduces the hardware and software tools utilized to develop the BCI framework. The BCI device is described along with the Node-RED software application used to extract the EEG data. A description of the Braccio robot and controller hardware along with the necessary control software is presented. Finally, a description of the chosen robotic simulators and their functionality is discussed.

2.1 Brain Computer Interface Hardware

The hardware chosen for the BCI was an EPOC+, an EEG research device made by the company Emotiv [3]. The EPOC+ headset, shown in figure 2.1, has 14 sensors that collect EEG signals, along with two references, and a set of semi-rigid plastic arms that allows the end user to repeatably place the sensors in the same location. The EPOC+ follows the 10-20 system, which is an internationally recognized standard on how to apply electrodes to the scalp [21]. The sensors are placed at the following locations: AF3, AF4, F3, F4, FC5, FC6, F7, F8, T7, T8, P7, P8, O1, O2, as shown in figure 2.2. Eight of these sensors (AF3, AF4, F7, F3, F4, F8, FC5, and FC6) are located around the frontal and prefrontal lobes, which allow the EPOC+ headset to pick up signals from facial muscles and the eyes, in addition to the EEG signals [22]. From this data the EPOC+ is also able to detect the following facial expressions: *Smile, Frown, Clench, Surprise, Blink, Right Wink*, and *Left Wink*. Finally, the headset is also equipped with 9-axis motions sensors including a gyroscope, accelerometer, and magnetometer. These features make the EPOC+ an

ideal device since it provides the capability to generate multiple control signals that can be mapped to a robotic process. The EPOC+ is wireless and connects to the computer through Bluetooth, and has a battery life of up to six hours. This EPOC+ headset is also complimented with a free software application called EmotivBCI. This application is the user interface where the training for mental commands and facial expressions is performed, and will be discussed in depth in Chapter 3.

Figure 2.1. Emotiv's EPOC+ Headset [3].

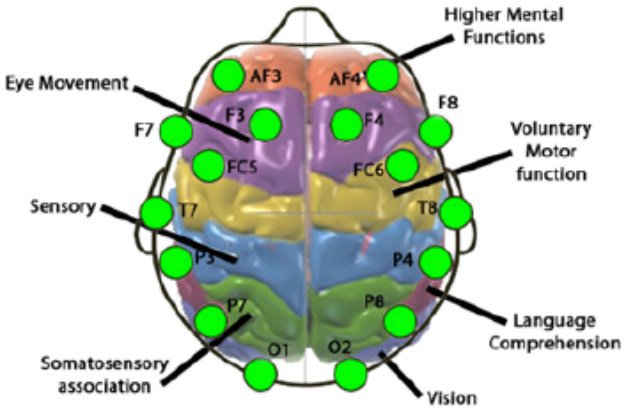

Figure 2.2. EPOC+ Sensor Locations [4].

2.2 Node-RED

Node-RED is a browser-based programming tool developed by IBM's Emerging Technology Services, that is generally used for wiring the Internet of Things (IoT) and robotics [7]. Node-RED utilizes flow-based programming which is a way of building an application by wiring together a network of nodes. Each node has a well-defined purpose in which it is given data, does something with that data, and then passes the result. This is similar to a function in other programming languages. Node-RED uses a graphical interface, similar to LabVIEW, to arrange the network of nodes. Applications are developed by dragging notes from the palette into a workspace where the flow can be build by wiring the nodes together. The base installation of Node-RED includes common programming functions, such as conditional and loop nodes. Once the application is completed, it is deployed for execution. The palette of nodes can be extended by installed new nodes created by the community. In order to utilize the signals from the Emotiv headset, Node-RED was chosen to be used as there is already an Emotiv BCI Node-RED Toolbox available to create BCI applications. This toolbox provides seven nodes that allow communication with the Emotiv headset to extract mental commands, facial expressions, performance metrics, frequency band powers, and data from the motion sensors.

2.3 Robot Platform

A Braccio robot arm from Arduino was the hardware chosen for this research for proof-of-concept work. The Braccio robot has five Degrees of Freedom (DOF) for position and orientation, with one additional motor to control the opening and closing of the gripper. It has a maximum operating distance range of 32 cm, gripper width of 90 mm, and load capacity of 150 grams. The Braccio comes with a shield

that typically interfaces with an Arduino Uno for control, however in this research, LabVIEW was used as described in section 2.4.

Figure 2.3. Braccio Robotic Arm.

2.3.1 Inverse Kinematic Analysis of Braccio Robot Arm

The Braccio robot only comes with Arduino functions that allow the individual motors to be moved to a desired angle, but not a desired position in 3D space. Therefore, the inverse kinematics need to be solved to evaluate the motion for each actuator based on a desired Cartesian position and orientation. To perform the inverse kinematic analysis, we will first find the forward kinematic equations by assigning coordinate systems (frames) to each joint of the robot using the Modified Denavit-Hartenbert (MDH) approach. The MDH approach is followed to assigning joint frames which requires the motion of the joint be along the Z-axis of the joint frame. The X-axis of the joint frame is assigned or defined as the common perpen-

dicular between two consecutive Z-axes. The frame Y-axis is evaluated assuming a right-hand coordinate frame as $Z \times X$. The assignment of these frames on the Braccio robot are shown in figure 2.4, where the left view has the base frame and first frame coincident and the right view has the frames from the second joint to the end effector. The coordinate frames follow the convention where the X-axis, Y-axis, and Z-axis are red, green and blue respectively.

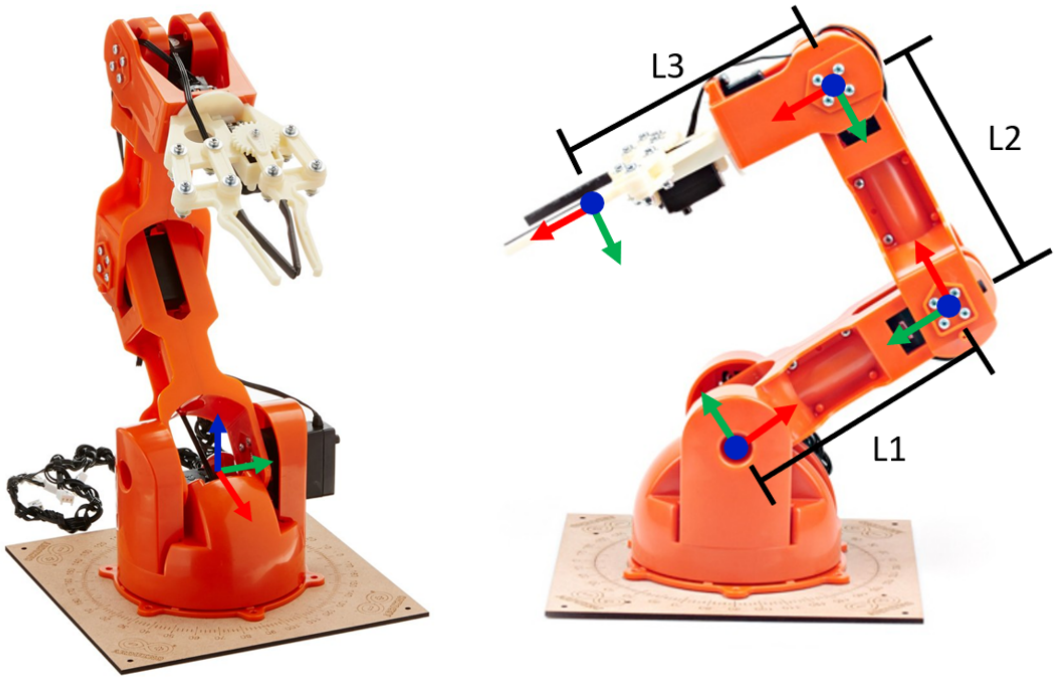

Figure 2.4. Braccio Robotic Arm with Coordinate Frames.

Using this convention, the spatial relationship between the successive joint coordinate frames can be established by defining the following four Denavit-Hartenberg parameters (a, α, d, θ), for successive frames indicated as Current (C) and Next (N) [5]:

1. a: is the distance from z_C to z_N along x_C,
2. α: is the angle from z_C to z_N about x_C,

3. d: is the distance from x_C to x_N along z_C, and
4. θ: is the angle from x_C to x_N about z_C.

The MDH table describing the kinematic relationships of the Braccio robot's coordinate frames is shown in table 2.1. Note the fifth joint that rotates the gripper is ignored for this application.

Frame		Joint			MDH Parameters				Joint Limits	
Current C	Next N	Frame #	Type	Variable	a	α	d	θ	Min	Max
0	1	1	R	θ	0	0	0	θ_1	0°	180°
1	2	2	R	θ	0	90°	0	θ_2	15°	165°
2	3	3	R	θ	L_1	0	0	θ_3	0°	180°
3	4	4	R	θ	L_2	0	0	θ_4	0°	180°
4	E	5	–	–	L_3	0	0	–	-	-

Table 2.1. Braccio Robot MDH Table

The homogeneous transformation matrix is obtained by substituting the values from each row of the MDH table into equation 2.1. This equation is used to evaluate the homogeneous transformation relating two consecutive frames using the MDH parameters. Multiplication of the resulting transformation matrices results in the transformation matrix relating the end effector to the base frame as shown in equation 2.7.

$$
{}^{C}_{N}T = \begin{bmatrix} cos\theta_N & -sin\theta_N & 0 & \alpha_C \\ sin\theta_N cos\alpha_C & cos\theta_N cos\alpha_C & -sin\alpha_C & -sin\alpha_C d_N \\ sin\theta_N sin\alpha_C & cos\theta_N sin\alpha_C & cos\alpha_C & cos\alpha_C d_N \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{2.1}
$$

$$
{}^{0}_{1}T = \begin{bmatrix} cos\theta_1 & -sin\theta_1 & 0 & 0 \\ sin\theta_1 & cos\theta_1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{2.2}
$$

$$
{}^{1}_{2}T = \begin{bmatrix} cos\theta_2 & -sin\theta_2 & 0 & 0 \\ 0 & 0 & -1 & 0 \\ sin\theta_2 & cos\theta_2 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{2.3}
$$

$$
{}^{2}_{3}T = \begin{bmatrix} cos\theta_3 & -sin\theta_3 & 0 & L_1 \\ sin\theta_3 & cos\theta_3 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{2.4}
$$

$$
{}^{3}_{4}T = \begin{bmatrix} cos\theta_4 & -sin\theta_4 & 0 & L_2 \\ sin\theta_4 & cos\theta_4 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{2.5}
$$

$$
{}^{4}_{E}T = \begin{bmatrix} 1 & 0 & 0 & L_3 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{2.6}
$$

The homogeneous transformation relationship for the pose, position and orientation, of the last coordinate frame with respect to the base frame is evaluated according to equations 2.7 and 2.8.

$$ {}^{0}_{E}T = {}^{0}_{1}T\,{}^{1}_{2}T\,{}^{2}_{3}T\,{}^{3}_{4}T\,{}^{4}_{E}T \tag{2.7} $$

$$ {}^{0}_{E}T = \begin{bmatrix} c\theta_1 c\theta_{234} & -c\theta_1 s\theta_{234} & s\theta_1 & L_3 c\theta_1 c\theta_{234} + L_2 c\theta_1 c\theta_{23} + L_1 c\theta_1 c\theta_2 \\ r_{21} & r_{22} & -c\theta_1 & L_3 r_{21} + L_2 s\theta_1 c\theta_{23} + L_1 c\theta_1 c\theta_2 \\ s\theta_{234} & c\theta_{234} & 0 & L_3 s\theta_{234} + L_2 s\theta_{23} + L_1 s\theta_2 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{2.8} $$

Where:

$$ r_{21} = c\theta_4 s\theta_1 s\theta_{23} - s\theta_4 s\theta_1 s\theta_{23} \tag{2.9} $$

$$ r_{22} = -s\theta_4 s\theta_1 s\theta_{23} - c\theta_4 s\theta_1 s\theta_{23} \tag{2.10} $$

To begin the inverse kinematic analysis, we must have the desired position of the end effector.

$$ {}^{0}_{E}T_{Desired} = \begin{bmatrix} r_{11} & r_{12} & r_{13} & p_x \\ r_{21} & r_{22} & r_{23} & p_y \\ r_{31} & r_{32} & r_{33} & p_z \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{2.11} $$

Now, from the geometry of the Braccio robot it can be determined that θ_1 can be calculated by simply using the inverse tangent function as shown in equation 2.12. In the code for the inverse kinematics solution, the *atan2* function will be used to ensure the correct quadrant is determined.

$$ \theta_1 = atan2(p_y, p_x) \tag{2.12} $$

With θ_1 known through the solution of equation 2.12, the Braccio robot is transformed into an equivalent $3R$ planar configuration. The equivalent planar configuration is used to solve for the remaining three joint angles. First, to get the

Braccio robot back to the XZ plane the ${}^{0}_{E}T_{Desired}$ can be premultiplied by the inverse of ${}^{0}_{1}T$ to get a new ${}^{1}_{E}T$ as shown in equation 2.13.

$$ {}^{1}_{E}T = {}^{1}_{0}T\,{}^{0}_{E}T_{Desired} \tag{2.13} $$

The rotation portion of the ${}^{1}_{E}T$ matrix can now be premultiplied by a rotation matrix about the X-axis by -90°, and then reassign p_z as the new p_y value in order to get to the 3R planar configuration, shown in figure 2.5. The forward kinematics for the 3R planar configuration can be expressed as equation 2.14.

$$ {}^{1}_{E}T = \begin{bmatrix} cos\theta_{234} & -sin\theta_{234} & 0 & L_1cos\theta_2 + L_2cos\theta_{12} + L_3cos\phi_{234} \\ sin\theta_{234} & cos\theta_{234} & 0 & L_1Cos\theta_2 + L_2cos\theta_{12} + L_3sin\phi_{234} \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{2.14} $$

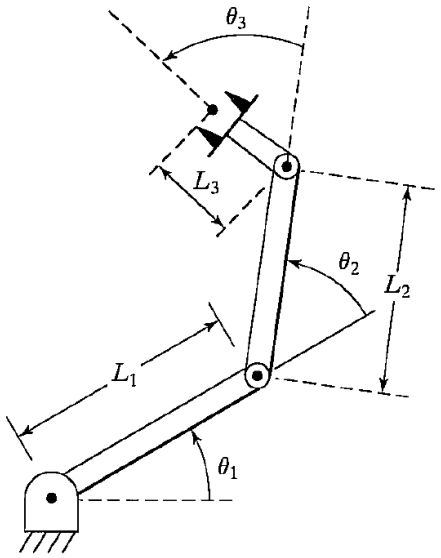

Figure 2.5. 3R Planar Robot [5].

Since the transform ${}^{4}_{E}T$ is fixed, it will not provide more information towards solving the inverse kinematics, so we can reduce equation 2.14 by post multiplying by ${}^{4}_{E}T^{-1}$ resulting in equation 2.15. Equation 2.15 can now be set equal to equation 2.16 to begin solving for θ_2, θ_3, and θ_4.

$$ {}^{1}_{4}T = {}^{1}_{E}T\,{}^{4}_{E}T^{-1} = \begin{bmatrix} cos\theta_{234} & -sin\theta_{234} & 0 & L_1Cos\theta_2 + L_2cos\theta_{23} \\ sin\theta_{234} & cos\theta_{234} & 0 & L_1Cos\theta_2 + L_2cos\theta_{23} \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{2.15} $$

$$ {}^{1}_{4}T_{Desired} = \begin{bmatrix} cos\phi & -sin\phi & 0 & p_x \\ sin\phi & cos\phi & 0 & p_y \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{2.16} $$

Now, by setting the corresponding p_x and p_y components of equations 2.15 and 2.16 equal to each other, we can square and add to obtain equation 2.19.

$$ p_x = L_1cos\theta_2 + L_2cos\theta_{23} \tag{2.17} $$

$$ p_y = L_1sin\theta_2 + L_2sin\theta_{23} \tag{2.18} $$

$$ p_x^2 + p_y^2 = L_1^2 + L_2^2 + 2L_1L_2cos\theta_3 \tag{2.19} $$

Therefore, we can solve for $cos\theta_3$ and $sin\theta_3$ by using the $atan2$ function. Note a positive sign of equation 2.21 will correspond to an elbow-up configuration, and a negative sign to the elbow-down configuration.

$$cos\theta_3 = \frac{p_x^2 + p_y^2 - (L_1^2 + L_2^2)}{2L_1L_2} \tag{2.20}$$

$$sin\theta_3 = \pm\sqrt{1 - cos^2\theta_3} \tag{2.21}$$

$$\theta_3 = atan2(sin\theta_3, cos\theta_3) \tag{2.22}$$

Since θ_3 is now known, it can be substituted back into 2.17 and 2.18 to obtain

$$p_x = k_1 cos\theta_2 - k_2 sin\theta_2 \tag{2.23}$$

$$p_y = k_1 sin\theta_2 - k_2 cos\theta_2 \tag{2.24}$$

where

$$k_1 = L_1 + L_2 cos\theta_3 \qquad \text{and} \qquad k_2 = L_2 sin\theta_3 \tag{2.25}$$

There are different approaches to solve this type of equation. One approach is to use the change of variables method, by changing how k_1 and k_2 are represented.

$$r = +\sqrt{k_1^2 + k_2^2} \tag{2.26}$$

$$\gamma = atan2(k_2, k_1) \tag{2.27}$$

$$k_1 = rcos\gamma \text{ and } k_2 = rsin\gamma \tag{2.28}$$

The change of variable updates equations 2.23 and 2.24 to the following:

$$\frac{p_x}{r} = cos\gamma cos\theta_2 - sin\gamma sin\theta_2 = cos(\gamma + \theta_2) \tag{2.29}$$

$$\frac{p_y}{r} = cos\gamma sin\theta_2 - sin\gamma cos\theta_2 = sin(\gamma + \theta_2) \tag{2.30}$$

As before, the $atan2$ function can be utilized to find θ_2.

$$\theta_2 = atan2(\frac{p_y}{r}, \frac{p_x}{r}) - \gamma = atan2(\frac{p_y}{r}, \frac{p_x}{r}) - atan2(k_2, k_1) \tag{2.31}$$

Finally, θ_4 can be calculated by equating ϕ to θ_{234} from equations 2.15 and 2.16.

$$\phi = atan2(sin\phi, cos\phi) = \theta_2 + \theta_3 + \theta_4 \tag{2.32}$$

$$\theta_4 = \phi - \theta_2 - \theta_3 \tag{2.33}$$

2.4 LabVIEW and myRIO

In order to control the Braccio robot and interface with the BCI, it was decided to use the software LabVIEW and a myRIO controller, by National Instruments, due to their ability to easily interface with hardware [23]. The myRIO device is equipped with three expansion ports (A, B, C), where A and B have an identical set of signals for digital and analog IO. Only the A and B expansion ports are utilized for this research. LabVIEW is advantageous in that it is a graphical based programming application that also provides the capability to design a Graphical User Interface (GUI) and supports parallel processing.

The Braccio robot is normally controlled with a motor shield that interfaces with an Arduino Uno board. A Digilent Shield Adapter was used to interface the motors of the Braccio with part A of the myRIO. A motor adaptor was also needed as there are only three Pulse Width Modulation (PWM) output signals per expansion port of the myRIO. The myRIO with the shield setup is shown in figure 2.6.

The Braccio robot uses SR311 and SR431 servo motors for each of the joints which are controlled using PWM signals. The myRIO Toolkit has a function to control servos, with the input being frequency and duty cycle. Testing on both types of servos showed that a frequency of 50 Hz provided the smoothest motion of the

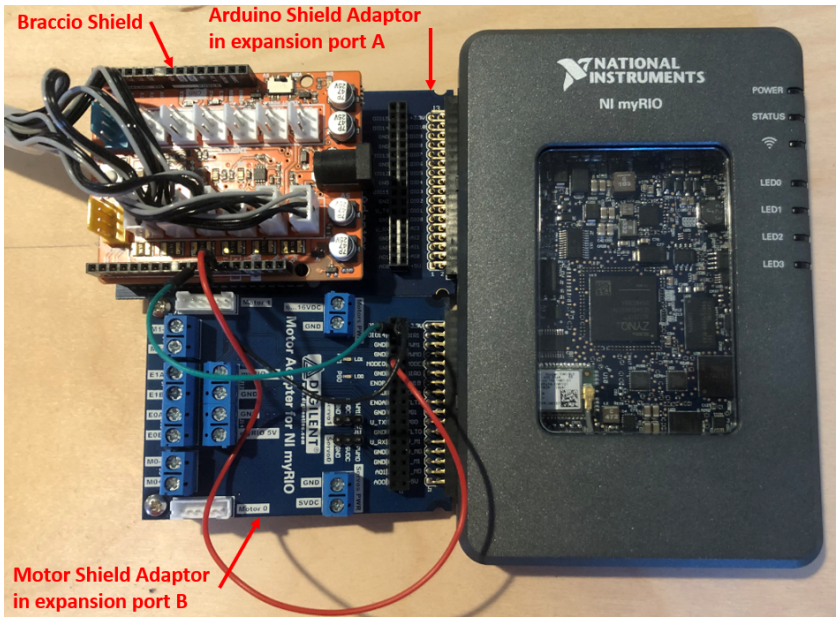

Figure 2.6. myRIO Setup with Arduino Motor Shield Setup.

motors, therefore this input was made a constant. The duty cycle also needed to be calculated for the servo motors to achieve the desired angle. Duty cycle is the ratio of Pulse Width (PW) to the total period (T) of the waveform, and can be represented by equation 2.34.

$$Duty\,Cycle = \frac{PW}{T} \tag{2.34}$$

The servo datasheet states that the range of signal impulse of 0.5-2.5 ms, which correlates to a range of $0 - 180°$. In order to calculate the correct PW signal for a desired angle, θ, a linear relationship was assumed, $PW = m\theta + b$. The calculations for the slope, m, and intercept, b, are shown in equations 2.35 and 2.36, resulting in equation 2.37. From experimental trials with the Braccio's servos, a resolution of approximately 1° is achievable with equation 2.37.

$$m = \frac{2.5 - 0.5}{180 - 0} = 0.0111(\frac{ms}{deg}) \tag{2.35}$$

$$b = 0.5(\,ms) \tag{2.36}$$

$$PW = 0.0111 \cdot \theta(deg) + 0.5(\,ms) \tag{2.37}$$

2.5 MATLAB

MATLAB was used for numerous software needs throughout the course of this research, from reading in and analysing data form the EPOC+ headset to implementing the inverse kinematics derived in section 2.3.1. It was also convenient that MATLAB scripts could be executed from within LabVIEW, therefore allowing for code reusability between the two applications.

One of the most used tools during this research was the MATLAB Robotic Systems Toolbox (RST), which provides tools and algorithms for testing manipulators, humanoid, and mobile robots. The RST uses a rigid body tree robot model which can be imported from a Unified Robot Description Format (URDF) file. A URDF is an XML file format that describes all the physical elements of the robot, such as configuration, inertia, visual, and collision geometry. Many popular industrial and hobby robots usually have their URDF available online, which is where the Braccio robot URDF was obtained. However, in the case of custom robots it must be built. Previous students, for their senior design project, in the Manufacturing Automation and Robotic Systems (MARS) lab, designed and built a 3D printed robotic arm that attaches to a wheelchair to support the assistive robotics research. This wheelchair robot was of interest in our research as it has a greater reach and payload capacity compared to the Braccio robot, which made it more practical for an actual application as opposed to a proof-of-concept. The wheelchair robot is shown in figure 2.7.

Figure 2.7. Wheelchair Robot.

The wheelchair robot was modeled in SOLIDWORKS. An open source SOLIDWORKS to URDF converter was download from the Robot Operating System (ROS) Wiki page. While in SOLIDWORKS, and before exporting to a URDF, a coordinate system was added to each joint. Then the link associated with each joint was defined before exporting to a URDF model. The MATLAB environment with the imported Braccio and wheelchair robot URDF models are shown in figures 2.8 and 2.9 respectively. Having both URDF models in MATLAB was beneficial since it allowed for a robot agnostic inverse kinematics code to be developed.

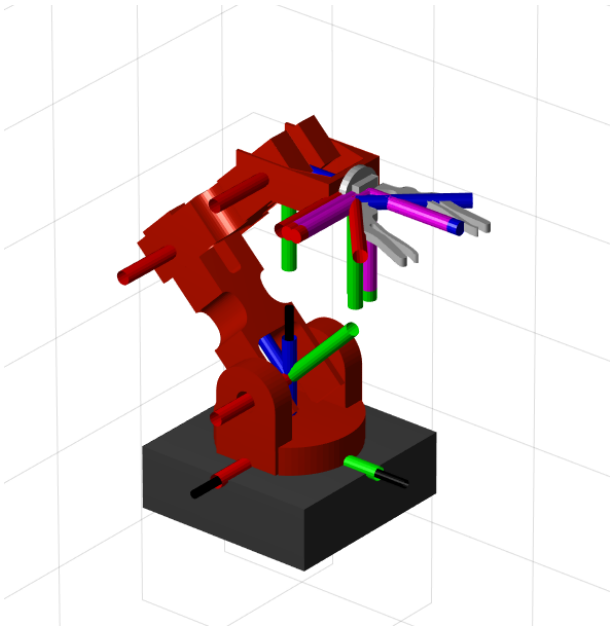

Figure 2.8. Braccio Robot URDF in MATLAB.

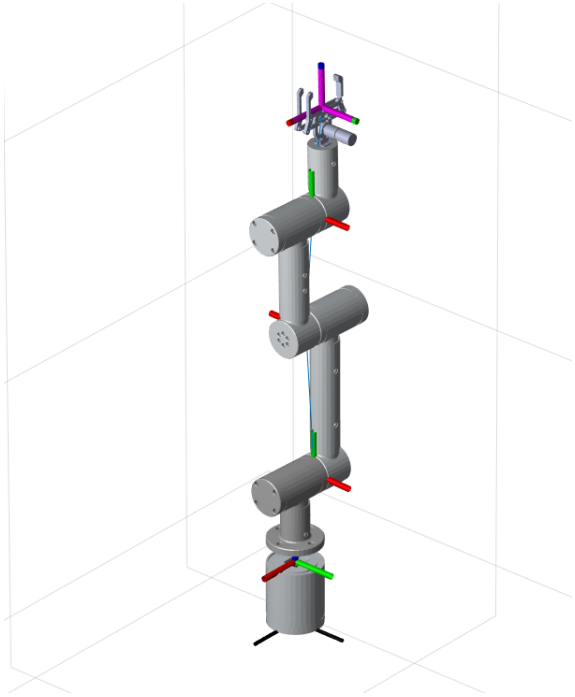

Figure 2.9. Wheelchair Robot URDF in MATLAB.

2.6 Webots

During the course of this research there were several times when it would have been beneficial to perform experiments with the Braccio robot without having to set it up. A number of robotics simulators were considered for virtual experiments before selecting Webots, an open source application by Cyberbotics Ltd. Webots allows controllers to be written in many different programming languages, including MATLAB. This made Webots an obvious choice since a majority of the code that had already been developed for this research was written in MATLAB. The Webots simulation environment is equipped with the Open Dynamics Engine (ODE), which is an open source library for simulating rigid body dynamics. This feature will allow the robot to manipulate objects in its workspace similar to how it would in a real world situation. The ODE is also beneficial when designing a robot manipulator to calculate the torques required to move a desired payload, therefore the appropriate hardware can be chosen. In addition to the physics modeling, Webots is also equipped with several sensors including cameras, lidar, radar, receivers, range finders, accelerometers, and force sensors. The ability to model a camera, and extract information from it, is important in the assistive robotic application presented here as the robot will be in a dynamic environment and will need the ability to distinguish and locate objects that the person controlling the robot will want to manipulate.

In order to test the performance of the programs written for the Braccio robot, it was necessary to import it into Webots. Webots does not accept URDF files, but rather PROTO files. A PROTO file of the Braccio robot was not available online, however there is a "urdf2webots" program written by Cyberbotics freely available on GitHub [24]. This tool was used to convert the Braccio robot URDF to a PROTO file, which allowed us to keep the same MDH parameters. This conversion was crucial

since the MATLAB code was also using this model. The imported model is shown in figure 2.10.

Figure 2.10. Braccio Robotic Arm in Webots.

Another motivation for utilizing Webots is that it provides a platform to test algorithms on additional robotic manipulators without the need to have the physical hardware. Although the Braccio robot serves as a great proof-of-concept device, it lacks the reach and payload requirements to be practical for assistive robotic applications.

2.7 CoppeliaSim

While Webots was the primary simulator used for this research, CoppeliaSim was also being evaluated in our lab. CoppeliaSim is another open source robotic simulator that allows for the controllers to be written in MATLAB. A MATLAB remote Application Programming Interface (API) is used to connect to CoppeliaSim

once the simulation has started. CoppeliaSim also uses the same URDF models that are used in MATLAB's RST, which makes programming the robot straightforward. This simulator is also equipped with four physics engines and the capability to model a number of sensors. This robotic simulator was tested to prove the portability of the developed BCI framework.

2.8 Chapter 2 Conclusion

This chapter provided a description of the hardware and software tools that were utilized to develop the framework for the BCI modality. The BCI hardware, the EPOC+, and the multiple control signals that can be extracted through Node-RED were discussed. The Braccio robot hardware was introduced, along with the myRIO and LabVIEW software used to control it. Finally, a review of the robotic simulation environments used for testing algorithms and integrating virtual sensors was presented.

CHAPTER 3

BCI Interface and Software Architecture

This chapter discusses the procedures for setting up and training the EPOC+ headset. The methods for training mental commands and facial expressions are presented. Section 3.2 discusses the details of the software architecture that was developed to integrate the multiple hardware and software components introduced in Chapter 2. The software architecture describes how the data is collected from the EPOC+ and translated to robotic actions.

3.1 Training the BCI

This section outlines the process for setting up and training the EPOC+ headset for mental command and facial expressions, and is based on material from the Emotiv webpage [6].

3.1.1 Emotiv EPOC+ Setup

Setting up the EPOC+ headset takes approximately 5-10 minutes and consists of soaking the felt pads in a saline solution before installing them on the headset. The felt pads must be properly hydrated in order to achieve quality contact. Once these felt pads are sufficiently hydrated, they can be installed on each of the 14 electrodes. Once the headset is ready, the next step in the training process is to launch the EmotivBCI application. In order to train the EPOC+, an account must be created on the Emotiv website. After this account is created then the user can login to the EmotivBCI application. Under this account many different profiles can be created.

Each of these profiles can be used for testing different methods for training. When first opening up the EmotivBCI application it will have you select the profile to be used for the training session. Then, once the EPOC+ is connected via Bluetooth, it provides instruction on how to fit the headset correctly. This fitting is an important step as having the electrodes as close to or at the same location on the scalp every time will produce better training results. An example of the ideal fitting is shown in figure 3.1. To consistently get the headset into the same position it is recommend to place AF3 and AF4 sensors three finger widths above the eyebrow, and ensure that the reference rubber sensors sit on the bone just behind each ear lobe.

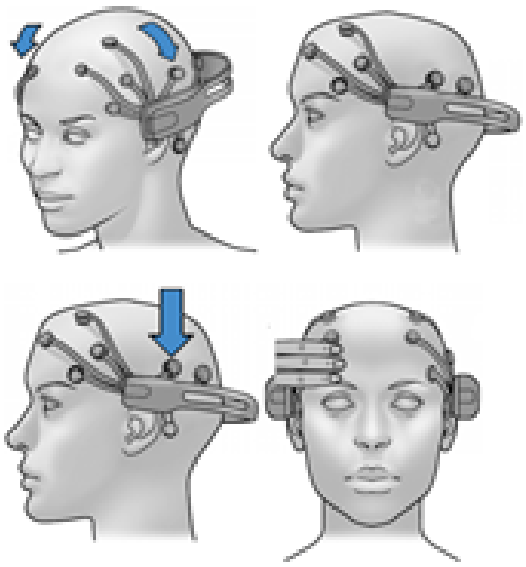

Figure 3.1. EPOC+ Head Placement [6].

Once the EPOC+ headset is fitted properly, the EmotivBCI application will provide feedback on the contact quality as shown in figure 3.2. This screen allows the user to adjust the problem electrodes as needed to strengthen the contact quality. It

is recommend for best results to have at least a 98% contact quality for each training session. For this research, a contact quality above 98% was achieved when first staring to train the headset, however the contact quality degraded over long sessions due to the felt pads drying out.

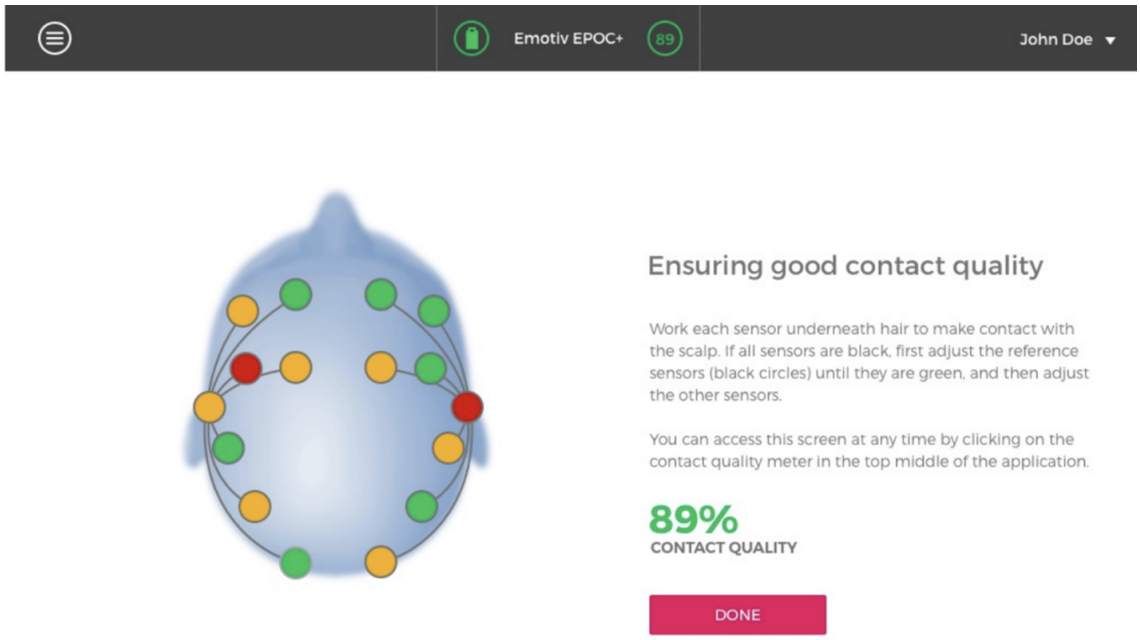

Figure 3.2. Emotiv EPOC+ Contact Quality [6].

3.1.2 Training Mental Commands

When first opening up the mental training page of the EmotivBCI the user is presented the screen shown in figure 3.3. The right side of the screen is where the user can select the desired thought to train. The *Neutral* command will be common to all training profiles, while the others can be selected from the drop down. There is a limit of four mental commands that can be trained for every profile. The *Neutral* state

is the first that should be train. The options of mental commands are: *Push, Pull, Lift, Drop, Left, Right, Rotate Left, Rotate Right, Clockwise, Counter Clockwise, Rotate Reverse, Rotate Forward*, and *Disappear*. Each of these mental commands corresponds to the movement of a floating cube displayed during the training period. When Train is selected, the cube will be displayed for a period of 8 seconds while the headset records the EEG data. During this time, the user should concentrate on a specific thought, without allowing the mind to drift. Immediately following the training, the user will be provided feedback on the quality of training as shown in figure 3.4. If the score of the training is above 75 (on a scale from 0-100), then it is considered a high quality training, otherwise the application will ask if the user wants to accept or reject the training. Once the training is accepted, then it will update the number next to the mental command on this screen to indicated the number of times it has been trained.

For best results, Emotiv recommends alternating between *Neutral* and the mental commands when training [6]. The easiest method seems to be starting with training one thought and master it before training additional mental commands. The quality of the training, shown on the left side of figure 3.3, is represented by a semicircle know as the Brain Space Diagram (BSD). In the BSD, the colored dots represent each mental command already trained. It is ideal for the dots to be spaced as far apart as possible in this BSD, as this indicates that each of the thoughts are distinct and can be easier to triggered independently. A BSD with all the dots closer to the centered *Neutral* dot can lead to false positives. An example of a good and bad BSD is shown in figure 3.5. This interface also has a Live Mode which will display the cube and allow the user to practice by attempting to triggering each mental command and observing if the cube responds as expected.

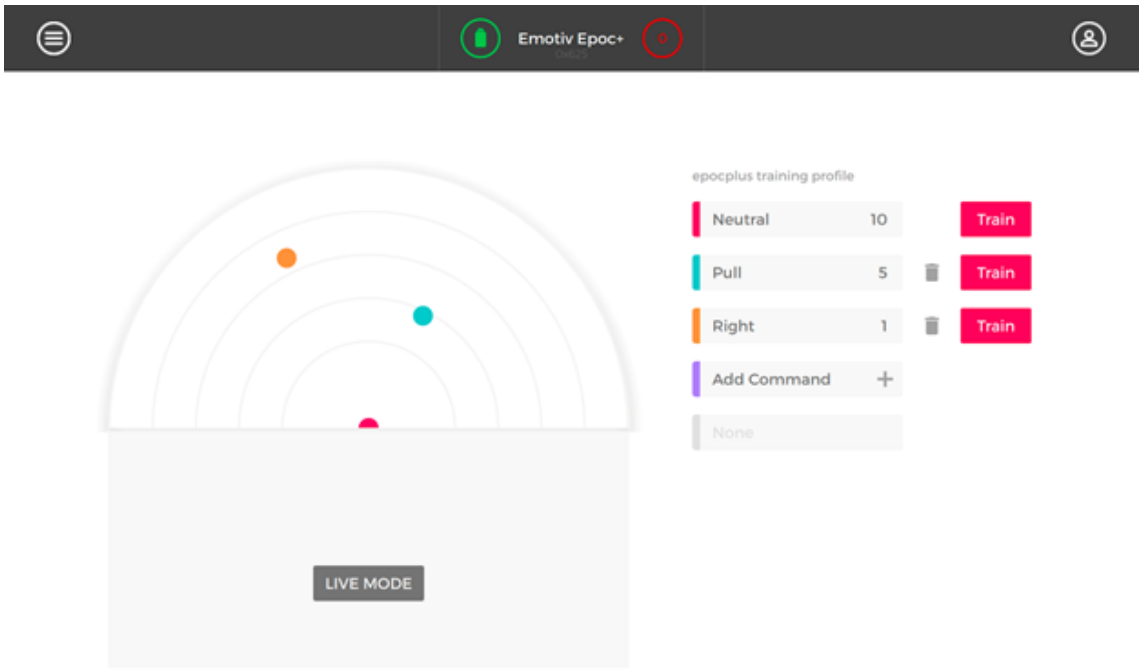

Figure 3.3. Mental Command Training Environment [6].

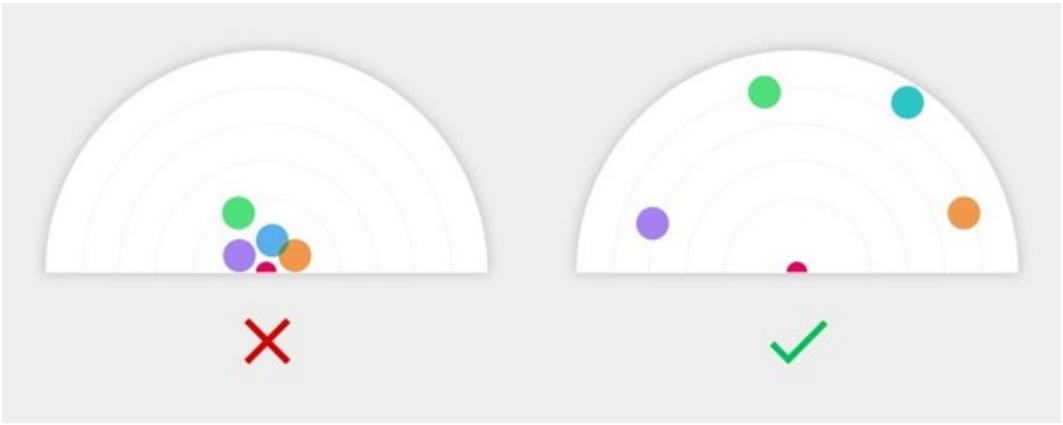

Figure 3.5. Brain Space Diagram Examples [6].

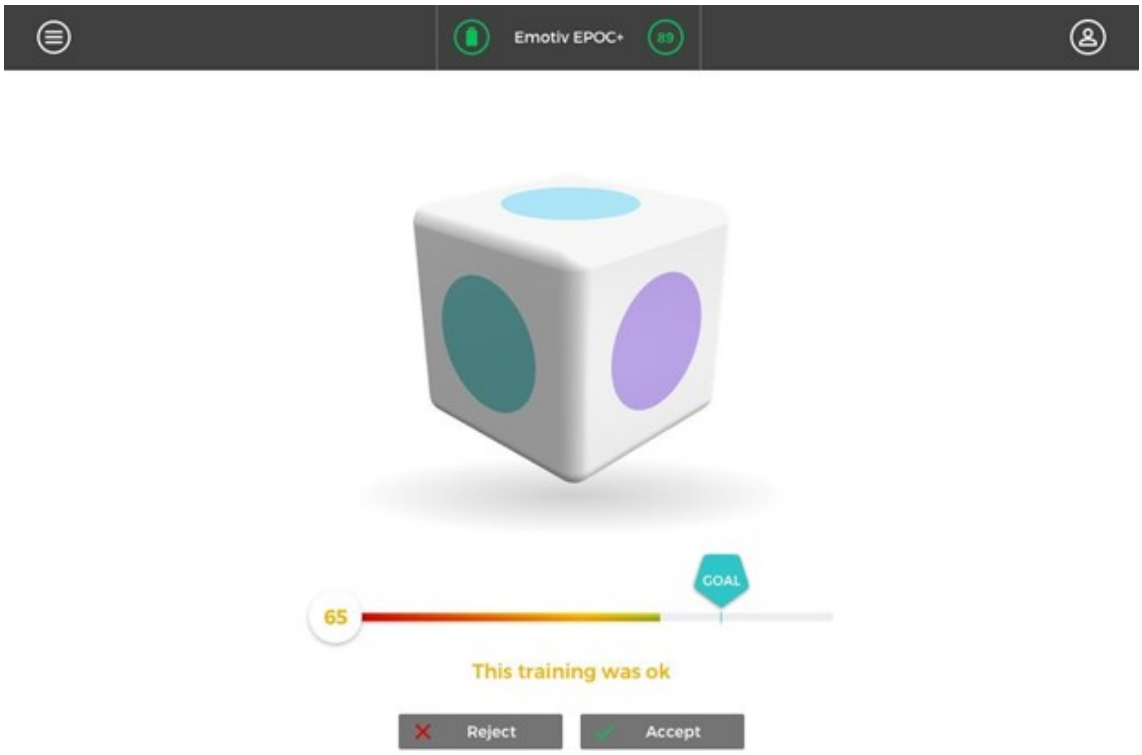

Figure 3.4. EmotivBCI Training Cube [6].

3.1.3 Facial Expression Training

Training a facial expression is quite similar to the mental command interface. In this case, an avatar is presented on the screen, as shown in figure 3.6, that the user will try to mimic during the training period. As before, it is best to start training the *Neutral* expression to establish a baseline. From there each of the facial expressions *Smile, Frown, Clench*, and *Surprise* can be trained. The duration of the training of facial expression is also 8 seconds. A Live Mode allows the user perform the facial expressions and if the avatar mimics the expression then it is trained well.

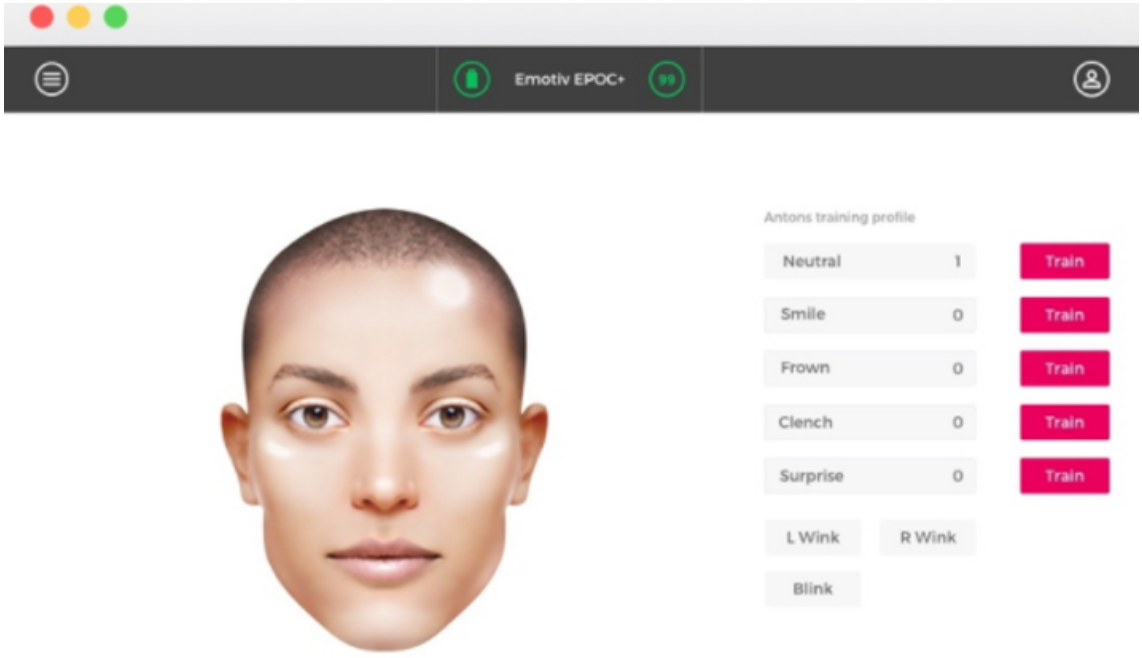

Figure 3.6. EmotivBCI Facial Expression Training Environment [6].

3.2 Software Architecture

In order to control the Braccio robot with the Emotiv headset, it was required that several different software applications communicated with each other as there is no direct interface from the Emotiv software application to LabVIEW or MATLAB. The Emotiv headset has a toolkit that is available in Node-RED, used to extract the signals from the headset, and output them to text files. These text files could be read by LabVIEW and MATLAB. From MATLAB and LabVIEW the robotic process is executed in the Webots simulation and on the hardware, using the data from the Emotiv headset. This software architecture and data flow for the hardware and software tools is shown schematically in figure 3.7.

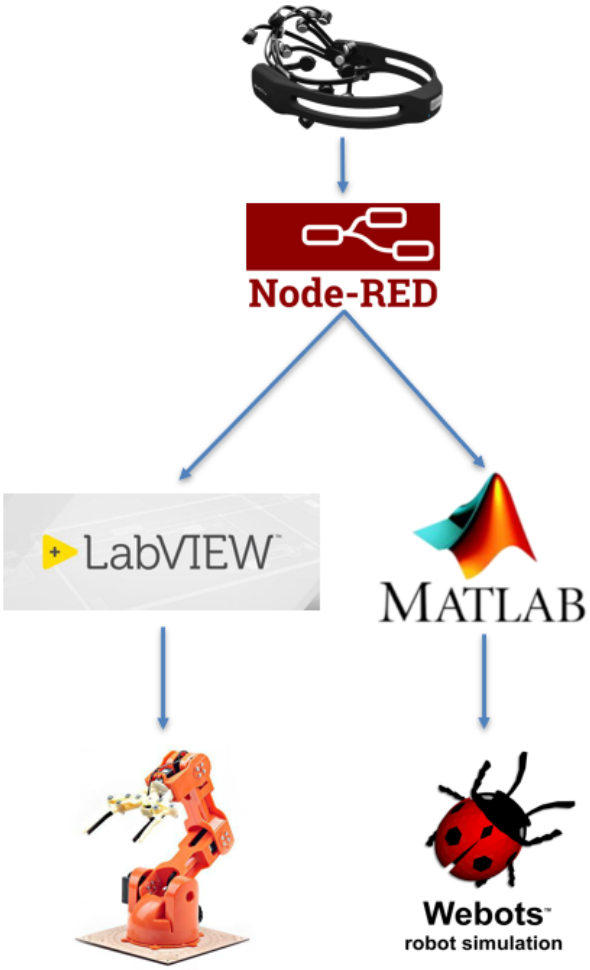

Figure 3.7. Software Architecture Diagram.

3.2.1 Node-RED to LabVIEW and MATLAB

Once the Emotiv headset is connected, the first software application that needs to be running is the Emotiv Cortex application. The Emotiv Cortex application connects to the device and allows communication to begin with Node-RED. Then, the nodes from the EmotivBCI Toolkit, shown in figure 3.8, will begin to extract information from the headset. The nodes from this toolkit have the following functions [7]:

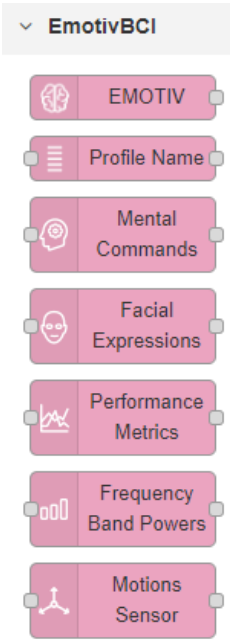

Figure 3.8. Node-RED Emotiv BCI Toolkit Nodes [7].

1. The Emotiv node is an initialization that allows to connection with the Emotiv headset and needs to be placed at the beginning of every Emotiv node sequence. The output of the Emotiv node is an authorization token.
2. The Profile Name node allows the selection of different Training Profiles that are created in the EmotivBCI and associated with the user's EmotivID. This option lets the user pick the profile that has been trained for a specific application.
3. The Mental Commands node is where a specific thought is chosen to trigger another action in the flow. The selected Mental Command must be one that has been trained in the profile in the EmotivBCI application, otherwise an error will occur. The output of this node is an integer from 0-100 that correlates to the intensity of the signal from the headset at a frequency of 8 Hz. This node also contains a Sensitivity setting, an integer from 0-10. Lower values will make the command harder to trigger, but result in fewer false positives. On the

contrary, a higher Sensitivity setting will make the Mental Command easier to trigger, but result in more false positives, an undesirable scenario.

4. The Facial Expressions node is similar to the Mental Command node where only a pre-trained Facial Expression can be selected, and the output is an integer from 0-100 representing the intensity of the expression at a frequency of 16 Hz. However, in this node there is a threshold setting that ranges from 0-1000. The Threshold is analogous to the Sensitivity setting where a lower value results in fewer false positives, and vice versa.
5. The Performance Metrics node allows the user to monitor excitement, interest, engagement, stress, focus, and relaxation. This node does not require input from a Profile node as there is no action to trigger. The output is an integer from 0-100 at a frequency of 0.1 Hz.
6. The Frequency Band node output is the Frequency Band Power at a frequency of 8 Hz and allows for the selection of a band to be monitored. The Frequency Band options are Alpha, Beta, Theta, and Delta. This node does not require a Profile to be selected.
7. The Motion Sensor node provides access to the data streaming from the X, Y, Z axes of the Gyroscope, Accelerometer, and Magnetometer. This node does not require a Profile to be selected.

The Node-RED flow that was set up to send control signals to LabVIEW and MATLAB is shown in figure 3.9. This flow utilizes mental commands, facial expressions, and gyroscope data. The sensitivity setting for the mental commands was set to 5 for each node to limit the possibility of false positives. The threshold setting for the facial expressions was set to 500 for the same reason. Following these Emotiv nodes is a Switch node that reads the intensity from each command, and if that intensity is greater than 50, it will allow for the execution of the following Function node. The

Function node returns a message output of integers ranging from 0-8 depending on the command that is triggered. The output of the Function node feeds into the input of the *pythonshell* node, which is made available once the Python toolkit is installed in Node-RED. The *pythonshell* node allows execution of a Python script with the input to that node being the argument for the script and the output of the script is sent to the output of the node. This Python script creates a 9×1 vector of zeros, then uses the input integer to assign the corresponding row to a value of 1 to indicate the command is triggered. This method ensures that only one command can be trigger at a time. The Python script then writes that vector to two text files, one for MATLAB and one for LabVIEW. These files are read into MATLAB and LabVIEW every half second, and the values correspond to a process that could be executed by the Webots simulation and the physical Braccio robot.

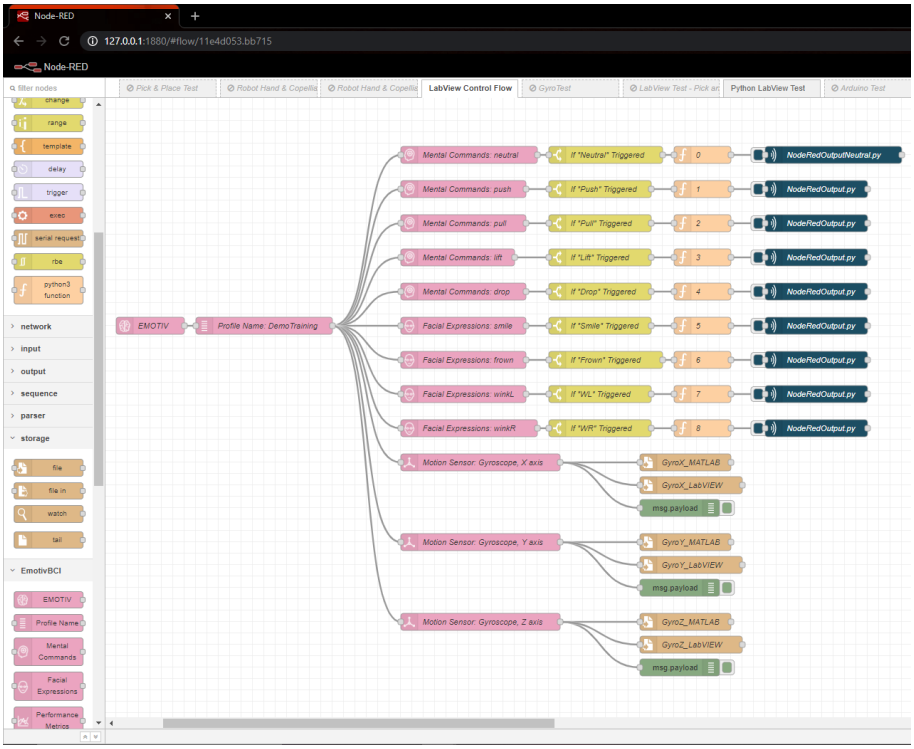

Figure 3.9. A Node-RED BCI Flow Instance.

The gyroscope data was written directly to two separate text files by a File node. This data was streaming from the Emotiv headset at 64 Hz, and each new value was written on a new line in the corresponding text files. In order to turn this data into a control signal, a function was developed in MATLAB to read in the file, analyze the data, and output a binary value based on specific head movement. The type of head movements that seemed to be the most practical to use for generating a control command were *Yes, No*, and *Side to Side* nods. The definition of these nods is shown in figure 3.10. Generating this logic from the gyroscope data can be challenging since a person is almost always moving their head, and may subconsciously nod their head without the intention to control the robot.

In order to address this issue the MATLAB function had four inputs, the frequency of the data stream in Hz, and the desired number of nods for an action to be executed for *Yes*, *No*, and *Side to Side* nods. The function only analyzes the last five seconds of data to ensure only recent intended nods trigger a command. From experimental trials of performing the various nods with the Emotiv headset, it was observed that *Side to Side*, *Yes*, and *No* nods correspond to a spike in amplitude of the gyroscope X, Y, and Z axes respectively. During testing, 1-2 of each nod were performed within a 15 second time frame, with an approximate 1-2 second pause in between. The data from this test was plotted as shown in figure 3.11. This graph was used to determine the amplitude that would identify a firm nod.

Once the amplitude was determined, the output of the function could be generated. The output of this function is a 3×1 vector, which corresponds to the *No*, *Yes*, and *Side to Side* nods. If any of these nods have been executed in the last five seconds, then the corresponding vector value will be set to 1. This logic was then used to generate a robot process. This function could also be called from LabVIEW

using the MATLAB script node, therefore this logic did have to be redeveloped in LabVIEW.

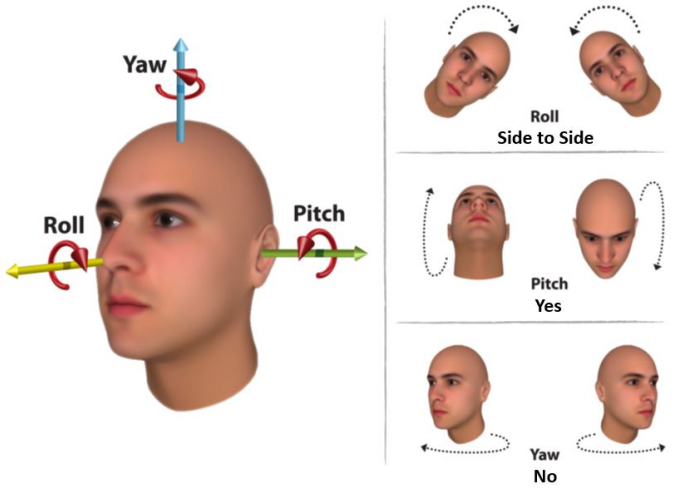

Figure 3.10. Definition of Nods [8].

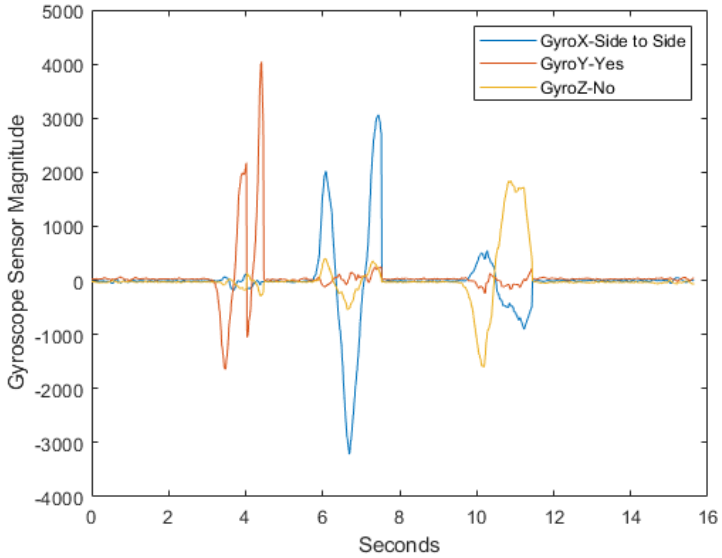

Figure 3.11. Magnitude of Gyroscope Data Based on Nods.

When developing this interface to the hardware and simulation, it was convenient to create a Node-RED flow that allowed for testing of the MATLAB and LabVIEW codes. This flow, shown in figure 3.12, involved Injection nodes, once their button was pressed they would send an equivalent corresponding integer as an action, similar to that from the Emotiv headset. This flow provided adequate algorithmic validation before using the headset for live tests.

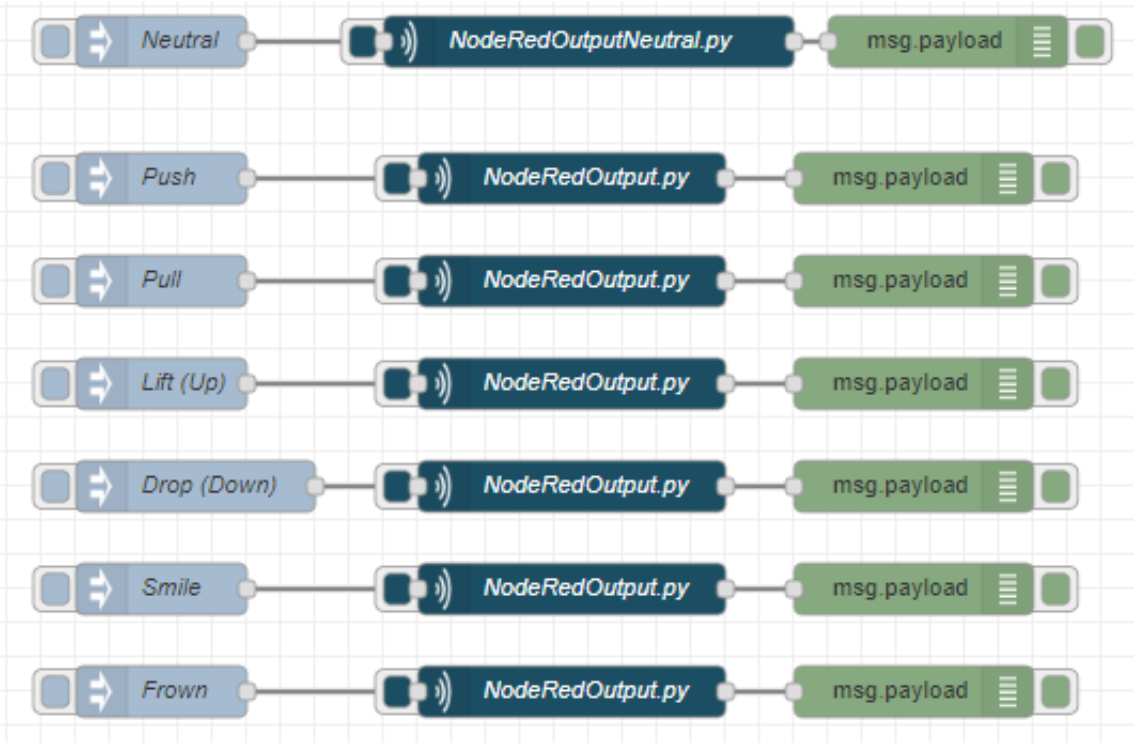

Figure 3.12. Simulated Signal Test Flow.

3.2.2 LabVIEW to Hardware

The first step in setting up LabVIEW to run on the myRIO, was to create a project. LabVIEW projects contain multiple LabVIEW programs, or Virtual Instruments (VI), and supplemental files such as documentation and related links. LabVIEW has a Project Explorer that allows the user to browser all the VIs and

documents contained in the project and add VIs to run on the computer or myRIO. The project for this research is shown in figure 3.13.

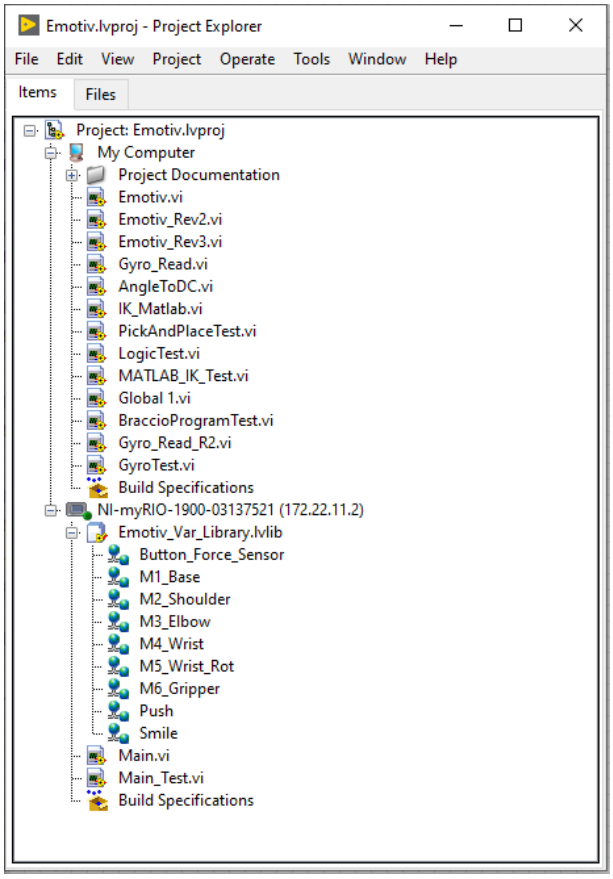

Figure 3.13. LabVIEW Project Explorer.

In order to have LabVIEW to control the robot from the myRIO, it was necessary to have two different VIs running, one on a laptop and the other on the myRIO. The VI running on the laptop had the GUI which allowed the user to select the interaction modality they wish to use to control the robot from the Robot Control Options pull down menu shown in figure 3.14. These modalities include using commands from the Emotiv headset, individual motor control, inverse kinematics, or loading a motion program of predefined joint angles. The *Home* option is the starting position for

the robot. The square LEDs illuminate when a mental command, facial expression, or nod is triggered from the headset. The block diagram of this VI can be used to map an output from the Emotiv headset to the desired action of the robot. This VI was the baseline for interfacing with the Braccio robot using the EPOC+ headset, however a few variations of the VI were created to test the performance of specific use cases.

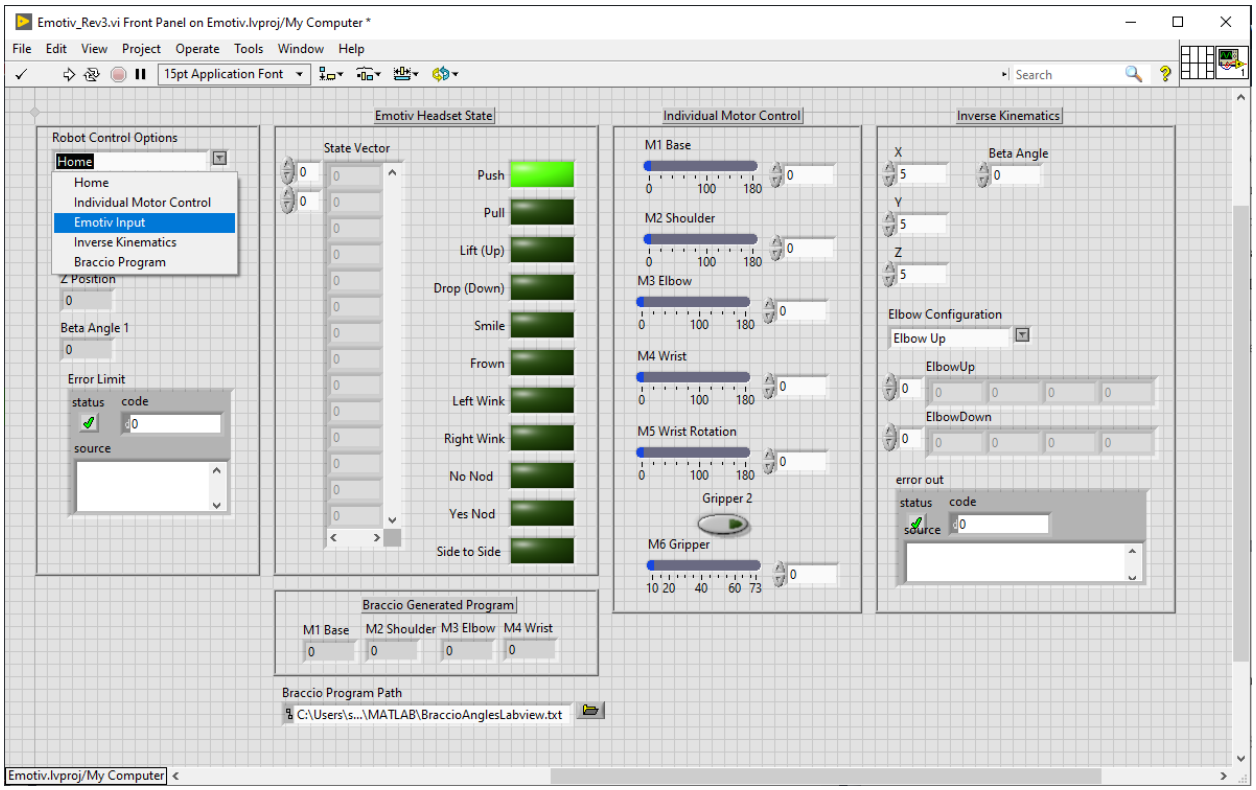

Figure 3.14. LabVIEW GUI for Interacting with Braccio Robot.

This VI running on the laptop is responsible for a majority of the logic, which includes reading in the output data file from the Emotiv headset and converting the desired joint angles to the correct PWM signal found by equation 2.37. The VI running on the myRIO was responsible for sending the PWM signal to the motors. In

order to send the PWM signal from the VI running on the laptop, it was necessary to create Network Published Shared Variables (NPSV). The NPSV are configured from the Project Explorer and allow data to easily be passed between devices on the same network. The combination of these VIs allowed for acceptable control of the Braccio robot.

3.2.3 MATLAB to Webots

As discussed in Chapter 2, the controller for the Webots simulation environment was developed in MATLAB. There were two variations of this controller. The first controller was specifically for pre-programmed routines, where each action of the robot was initiated by output from the EPOC+ headset. The controller does this by monitoring the text file written by Node-RED and executing the pre-programmed routine once a command from the headset is received. The second controller implements a more dynamic approach that can generate a motion plan to pick up an object based on its location in the robot workspace. Even though the inverse kinematic solution, derived in Chapter 2, was sufficient for the Braccio robot, there was a desire to investigate a generic inverse kinematics function to be used with multiple robotic manipulators, such as the wheelchair robot. In order to make this function robot agnostic, an input was the path to the URDF file. The information in the URDF provides the end effector frame, relative to which an offset can be defined to represent the desired Tool Center Point (TCP). The second input to the inverse kinematics function is the position of the object. A simulated camera in the Webots environment identifies the object and returns its coordinates. In order to pick up an object effectively an approach point was defined. This approach point was defined as a set distance from the object, on a line from the robot base to the object. The motion plan to pick up the object is broken up into two parts; a freespace motion

from the robot home position to the approach point, and a linear trajectory from to approach point to the object. The inverse kinematics for the approach point is solved using a function from MATLAB RST, and then the obtained joint values are interpolated to generate the freespace motion. From the approach point to the object, a linear constraint is implemented that keeps the end effector in the same orientation as it moves in to grab the object. An example of this motion is shown in figure 3.15, where the yellow coordinate system represents the position of the object.

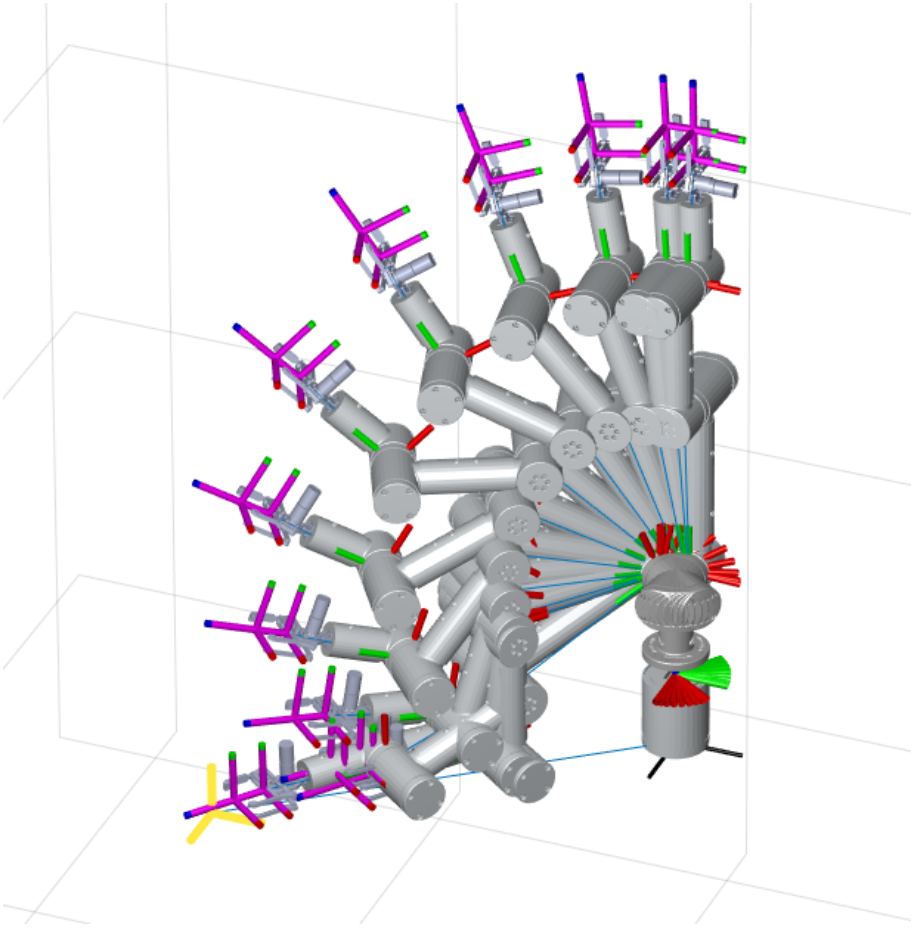

Figure 3.15. Wheelchair Robot Path Utilizing MATLAB General Inverse Kinematic Function.

The third input to the inverse kinematic function defines a file path to the motion file that will be used by Webots. In Webots, robotic manipulators are generally controlled with motion files that are executed during the simulation. The motion files contain a series of poses which include the joint positions and the time at which each pose will be executed. These motion files are generally predefined, however in order to implement the inverse kinematics code to pick up objects observed by the camera, this file would need to be dynamically generated . A MATLAB function program was developed to generate the motion file dynamically based on the object information from the camera. The input to the function is the filename for the motion file, the desired duration for the motion, and the array of joint positions for the motion. The function then writes the motion file in the exact format that Webots needs to execute the robot trajectory in the simulation. This function only needs to be modified when a new robot configuration that has different number of joints is introduced.

3.3 Chapter 3 Conclusion

This chapter described the how to set up the EPOC+ in a repeatable manner that will produce the best training results. The EmotivBCI application was introduced, and the procedures for training mental commands and facial expressions were described. This chapter also provided an in depth description of the software architecture that was developed to extract the intended commands from the EPOC+ headset and map them to robotic actions using Node-RED, LabVIEW, MATLAB, and Webots.

CHAPTER 4

Results and Discussion

In order to verify that the EPOC+ headset could be an effective modality of HRI several tests were performed on robot hardware and in simulation. The procedure for testing the BCI was to start simple and increase the complexity by gradually introducing additional commands after a successful test. For each test, the mental commands, facial expressions, and nods were mapped to a robotic action. This chapter will discuss the several tests that were performed on hardware and in simulation. For the hardware tests, the Braccio robot was set up on a wooden board that provided a solid base and a workspace for testing the robot processes. In addition to the physical setup, a Webots simulation was defined. The software architecture allowed the simultaneous execution of both of these processes.

4.1 Test 1: Grab Object with Braccio Robot

The first experiment performed with the Braccio robot used two commands from the EPOC+ headset, a mental command and a facial expression. The setup for this experiment was simple, with the Braccio robot starting in the Home position; the first command moved the end effector to a predefined (pre-taught) position where an object (a paper towel roll in this case) was located. Once in position, the second command was used to close the gripper on this object. For this routine, a *Push* mental command was used to trigger the Braccio robot to move to the predefined position, and a *Smile* expression to close the gripper.

In order to accomplish this routine, the *Push* mental command was trained immediately before performing this test. The contact quality was 100% for this training. After trying to associate a couple different thoughts with pushing the cube and not producing a quality Brain Space Diagram (BSD), I settled on a thought that became easier to trigger. My thought for training this mental command was visualizing my hands on the cube and imagining pushing it away from me. It took four successful training sessions with this mental command before I could comfortably trigger it during the Live Mode and produced the BSD shown in figure 4.1. Then the training focused on the *Smile* facial expression. The *Smile* facial expression had repeatably been the easiest facial expression to trigger. Once both were trained, the next step was to evaluate on the hardware.

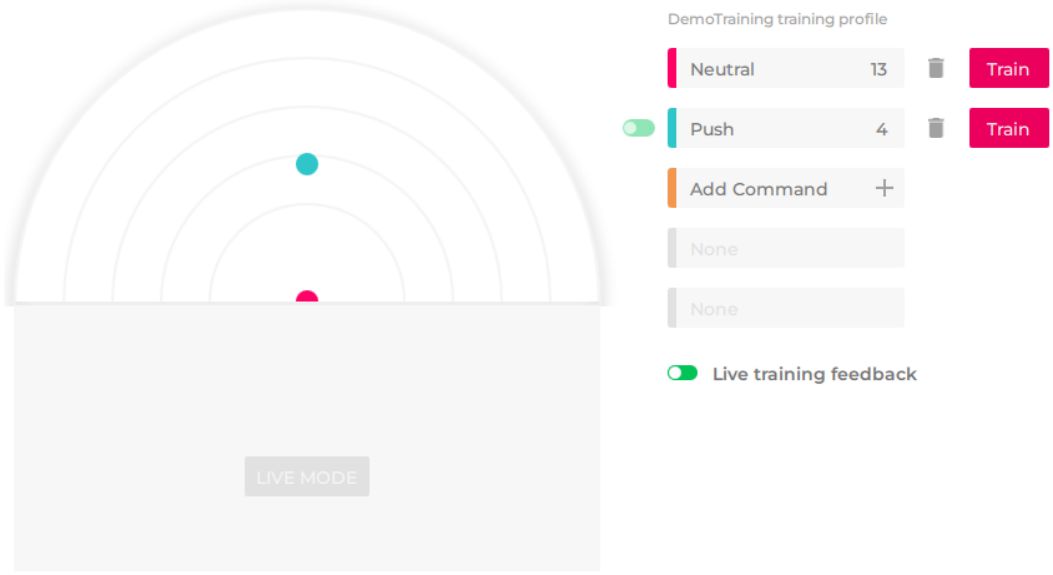

Figure 4.1. Test 1: BSD for *Push* Command.

The setup for this experiment, with the Braccio robot in the Home position, is shown in figure 4.2. For this experiment, I was able to successfully trigger the *Push*

and *Smile* commands to move the robot into place, and close the gripper to grasp the object. The results of this process are sequentially shown in figures 4.2, 4.3, and 4.4.

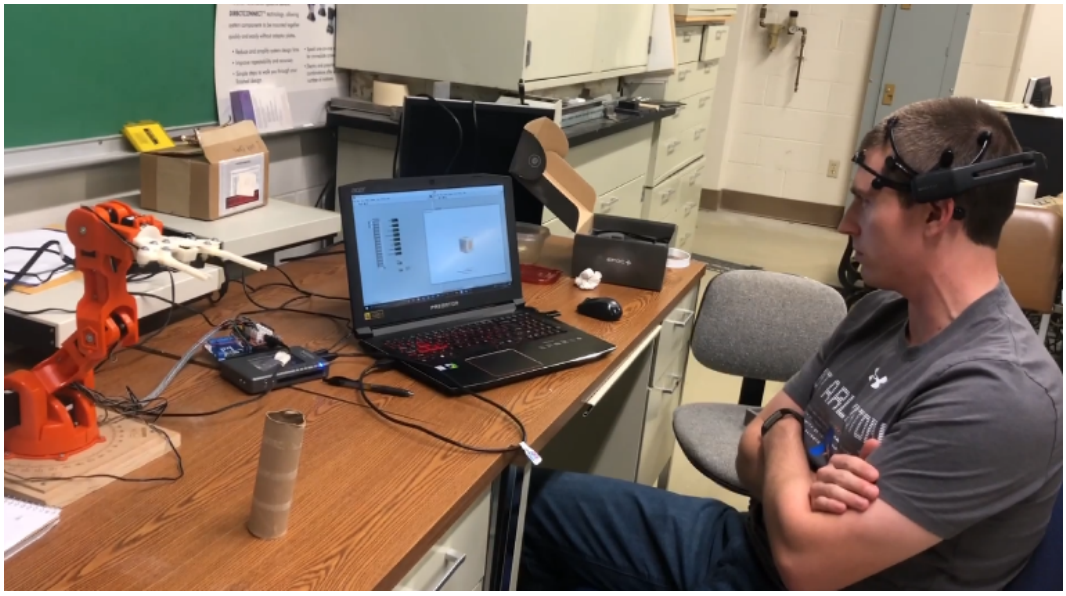

Figure 4.2. Test 1: Braccio in Home Position.

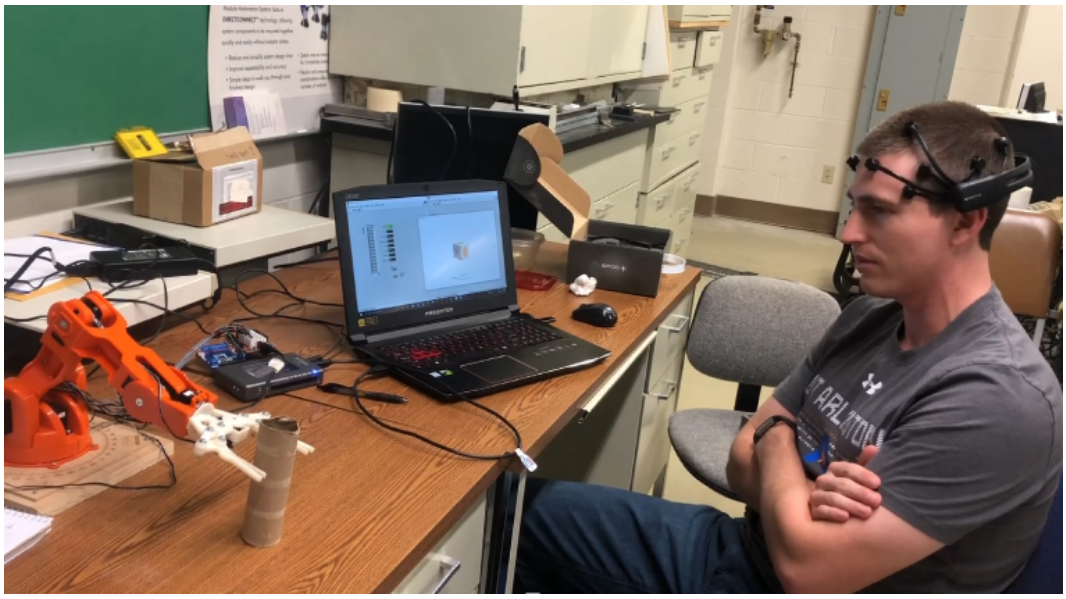

Figure 4.3. Test 1: Braccio in Pick Up Position using *Push* Command.

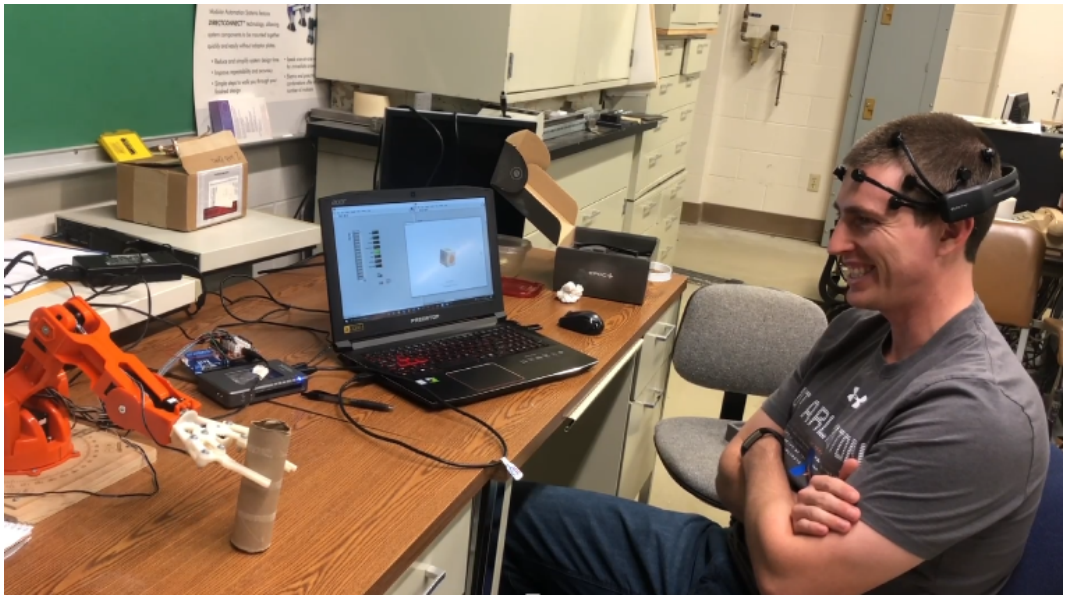

Figure 4.4. Test 1: Braccio Closing Gripper using *Smile* Facial Expression.

4.2 Test 2: Simulation for Verification

After successfully importing the Braccio robot into Webots it was necessary to evaluate that the simulated robot would behave the same as it did on the hardware. For this test, a pick and place routine for a water bottle was programmed utilizing the inverse kinematics solution developed in Chapter 2. The Simulated Signal Test Flow from Node-RED, shown in figure 3.12, was used. The pick and place routine was configured to execute with the following simulated commands:

1. *Push*: Move to water bottle
2. *Smile*: Close gripper
3. *Pull*: Move to place location
4. *Frown*: Open gripper

The Webots simulation and physical robot both executed the pick and place routine when the proper commands were sent from Node-RED. This process is shown sequentially in figures 4.5 to 4.8.

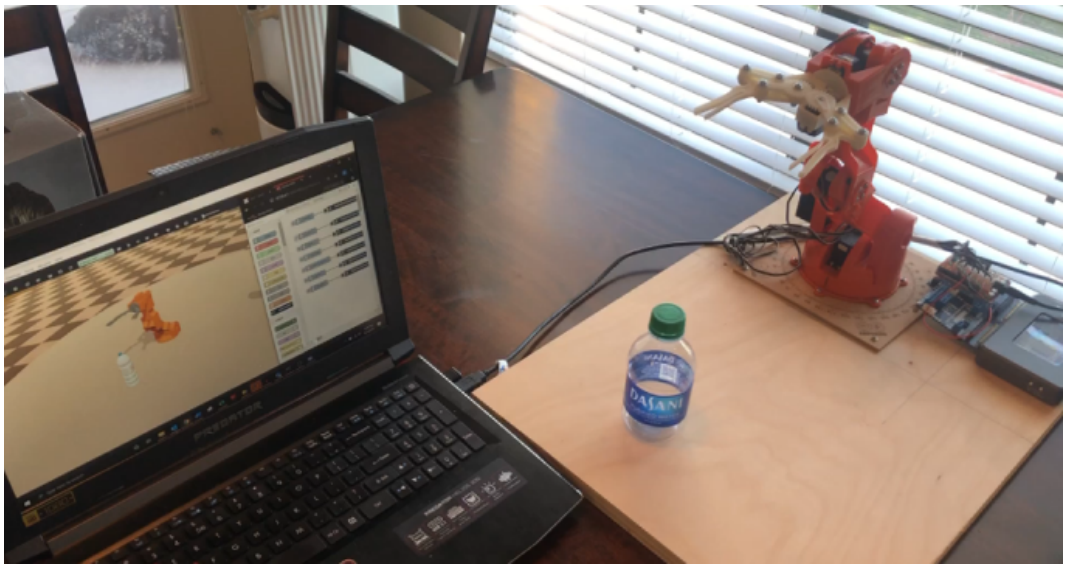

Figure 4.5. Test 2: Braccio Simulation and Hardware in Home Position.

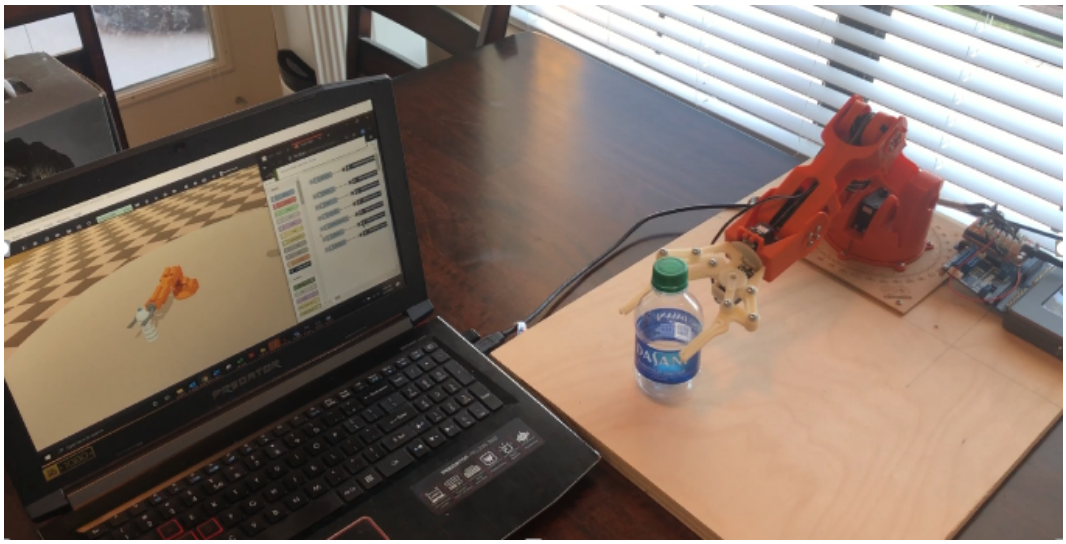

Figure 4.6. Test 2: Braccio Simulation and Hardware in Pick Up Position.

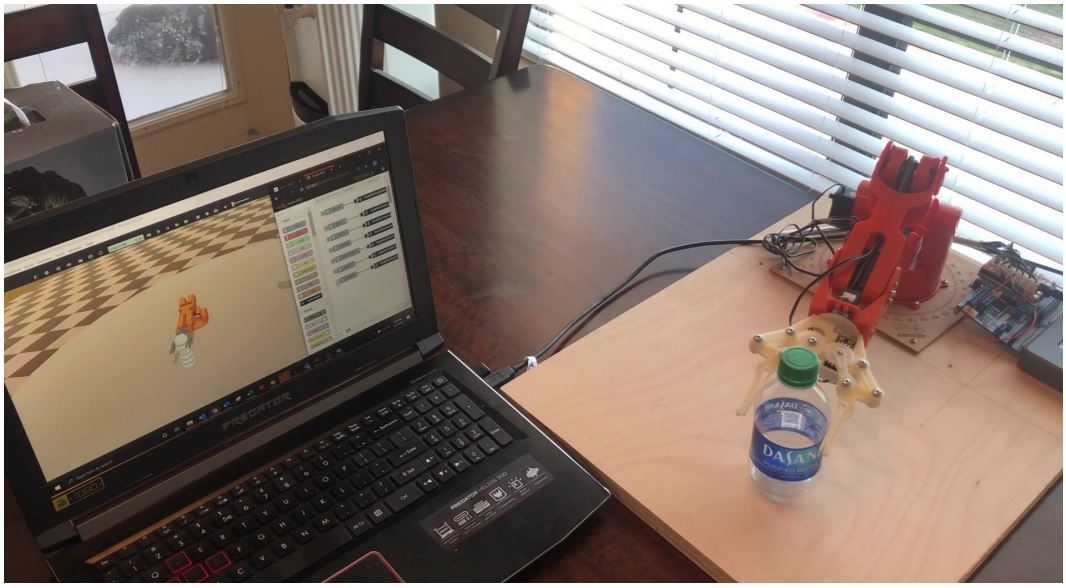

Figure 4.7. Test 2: Braccio Simulation and Hardware Place Position.

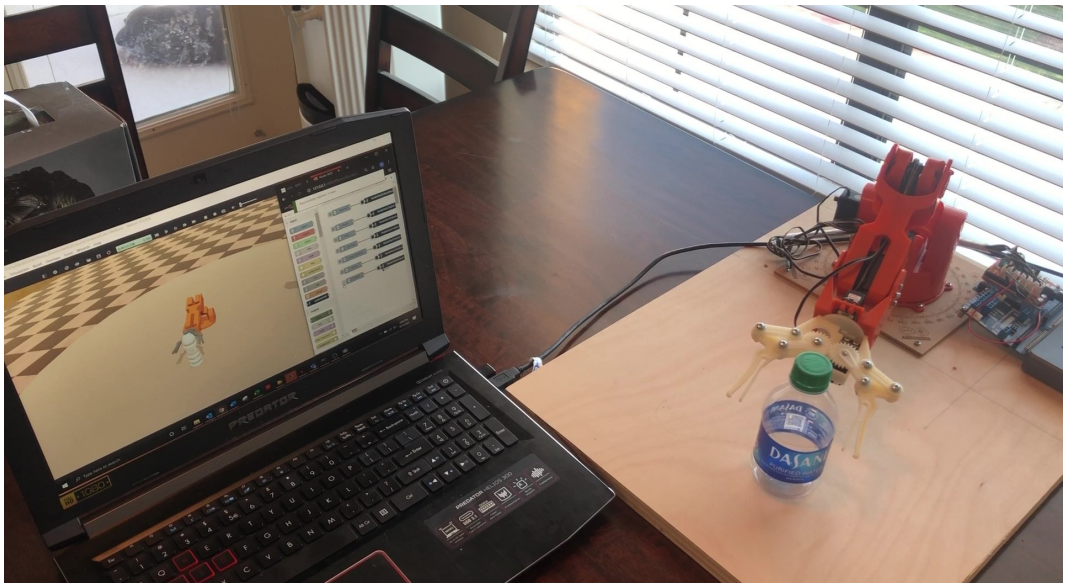

Figure 4.8. Test 2: Braccio Simulation and Hardware Opening Gripper.

4.3 Test 3: Robotic Hand

The third experiment was performed in a Webots simulation with the InMoov robotic hand, for which a hardware prototype is available in the MARS lab [25]. The goal of this experiment was to train a thought associated with physically holding the thumb and index finger as close as possible to each other without touching. This method has proven to be an effective way to consistently trigger a mental command [14]. In this case, the trained mental command is mapped to the simulated robotic hand closing its thumb and index finger to mimic the position of my fingers. For this training session, the thought of holding the thumb and index finger close together was trained to a *Pull* mental command. With this method of training it only took four *Neural* and three *Pull* command training sessions to create a sufficient BSD as shown in figure 4.9. This experiment required only two commands:

1. *Pull*: Close thumb and index finger
2. *Smile*: Return to open hand

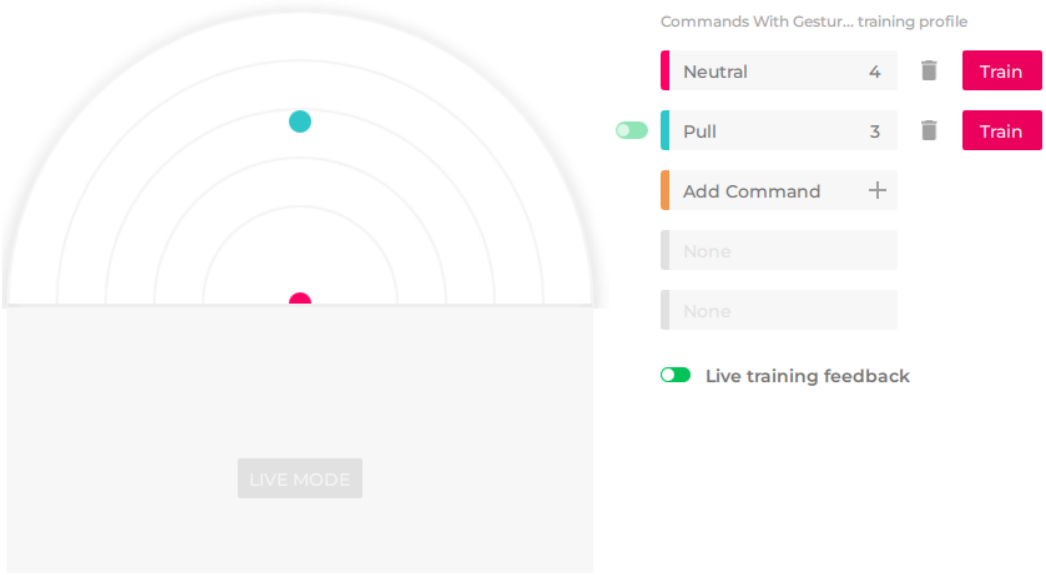

Figure 4.9. Test 3: BSD for Robotic Hand fingers.

This test was executed successfully on the first attempt. Figure 4.10 shows the robotic hand in the starting open state. Once the *Pull* command is triggered, the thumb and index finger close as shown in figure 4.11. Then, the *Smile* facial expression was executed to return the thumb and index finger to their respective open positions.

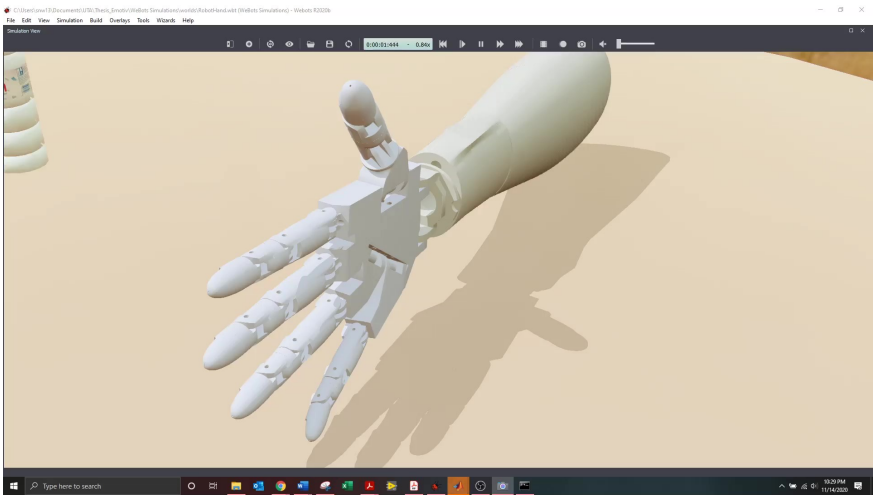

Figure 4.10. Test 3: Robotic Hand Open.

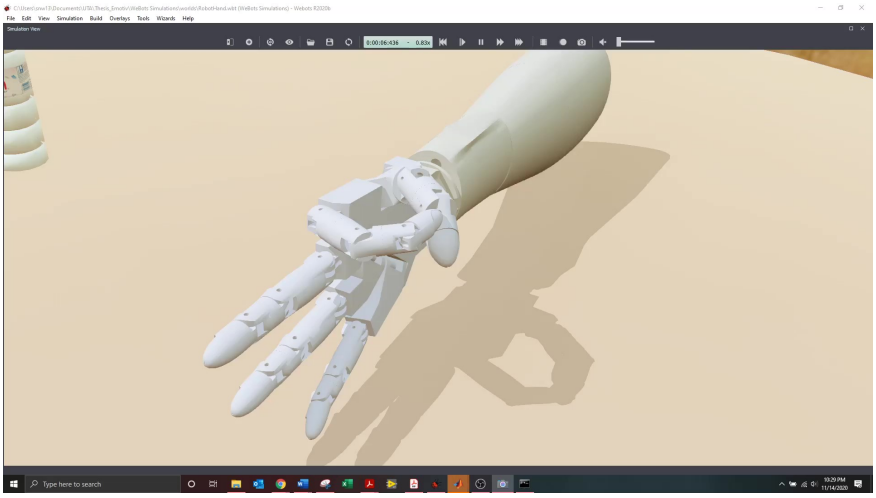

Figure 4.11. Test 3: Robotic Hand with Thumb and Index Finger Closed.

4.4 Test 4: CoppeliaSim Test

Even though Webots was the primary simulator used for this research, there was interest to also demonstrate the ease of portability and code reusability of the developed MATLAB controller to read signals from the EPOC+ to control a simulated

robot in CoppeliaSim, another robotic simulator. In order to accomplish this, a CoppeliaSim simulation environment was created with the Braccio robot. The same training profile from Test 3 was used. Since this test was only a verification that the same MATLAB controller could be used across platforms, only two commands were chosen:

1. *Pull*: Move to pick up position
2. *Smile*: Close gripper.

This test was successfully performed by triggering the two commands from the EPOC+ headset. The starting position of the Braccio robot is shown in figure 4.12. The robot moving to the pick up position and closing the gripper are shown in figures 4.13 and 4.14 respectively.

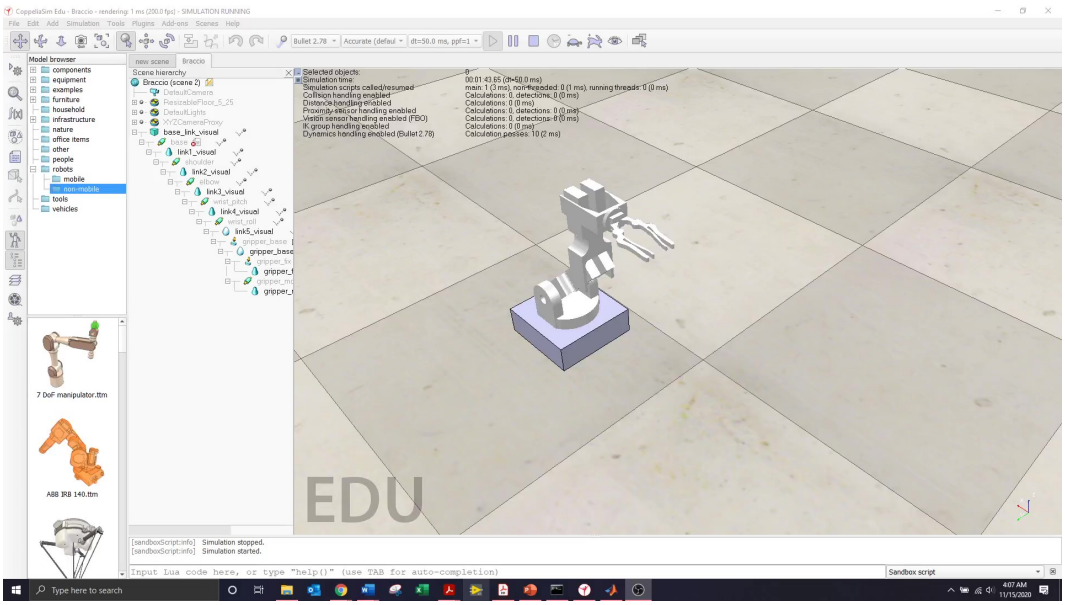

Figure 4.12. Test 4: CoppeliaSim Braccio Home Position.

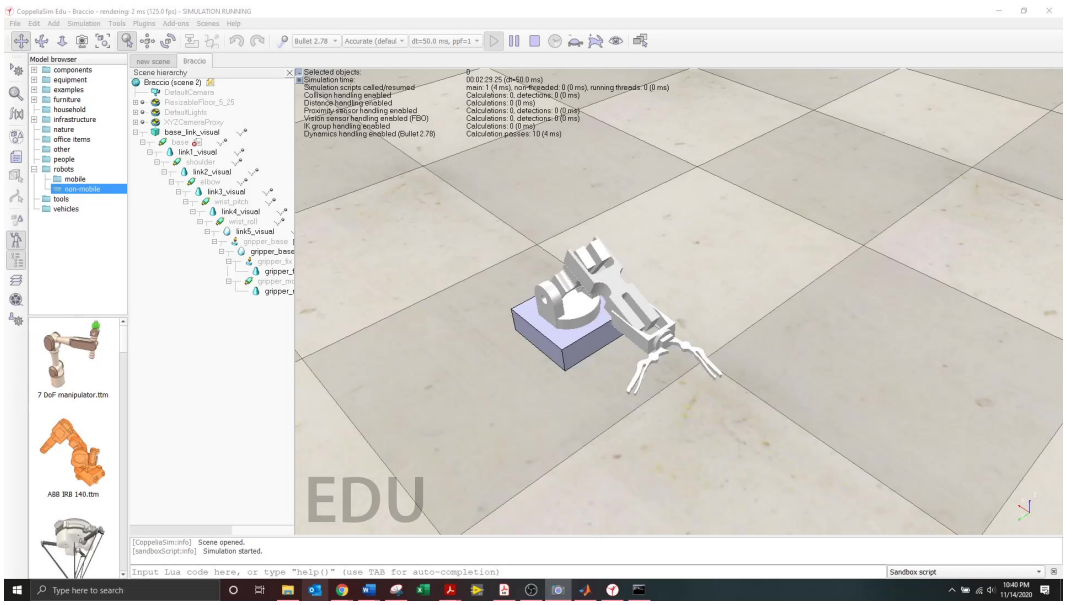

Figure 4.13. Test 4: CoppeliaSim Braccio Pick Up Position.

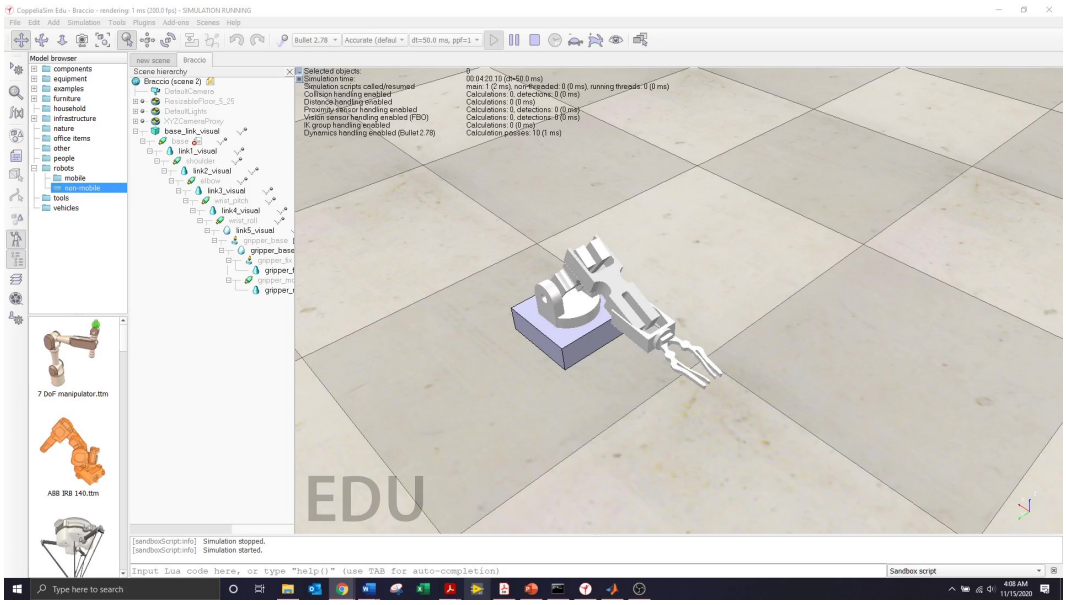

Figure 4.14. Test 4: CoppeliaSim Braccio Gripper Closed.

4.5 Test 5: Pick and Place

The pick and place process was important to demonstrate for this BCI modality of HRI, as it would probably be one of the most used processes in an assistive application. This test was performed with the Braccio hardware robot and in a Webots simulation. For this experiment, a water bottle was placed in a fixed location where the Braccio would reach, and pick it, then lift up it and move it to a predefined location and release. In order to perform this process, a combination of mental commands, facial expressions, and nods were used. The command for each robot action is:

1. *Push*: Move to water bottle
2. *Smile*: Close gripper
3. *Side to Side Nod*: Move to place location
4. *Yes Nod*: Open gripper

The technique for training the *Push* command is the same as in Test 1, where I imagined physically pushing the cube away from me. However, for this this test additional training was performed to ensure reliable triggering of the mental command. The 19 *Neutral* and 11 *Push* training sessions resulted in a high quality BSD, as shown in figure 4.15. The contact quality was 100% for this test.

This test was successfully performed simultaneously on the hardware and in simulation. The LabVIEW GUI and Webots simulation were both executing on the screen during this pick and place routine. The GUI and results of the simulation are shown in figures 4.16 through 4.19. In the LabVIEW GUI, the LED being illuminated corresponds to the triggered signal from the EPOC+. However, the *Smile* LED does not dim after triggered since the text file is only written when a new mental command or facial expression is triggered. The nod LEDs will dim if the user is not performing

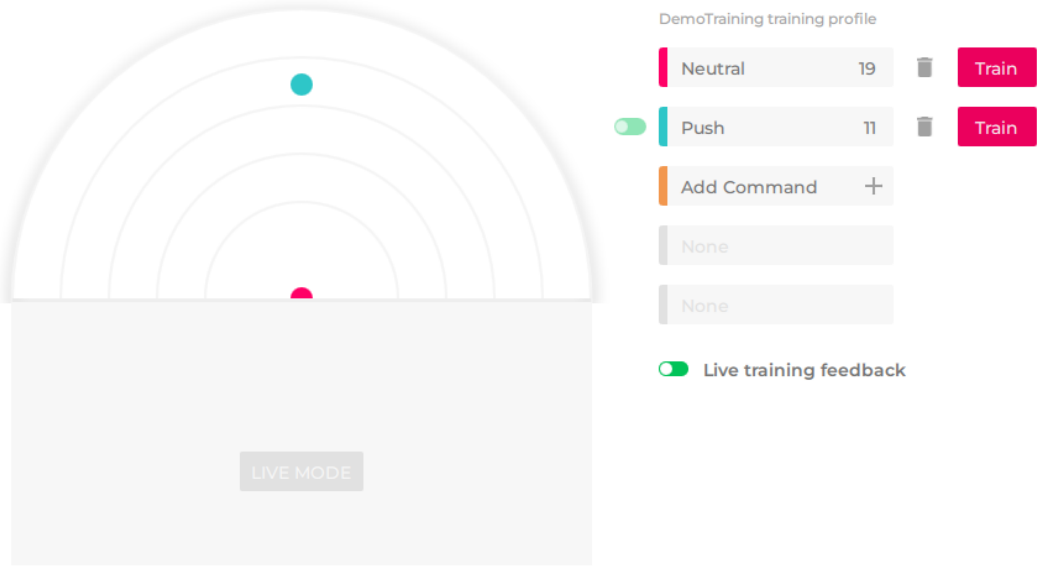

Figure 4.15. Test 5: Pick and Place BSD.

the nod, since the function only looks at the last five seconds of data. The execution of this process on the physical Braccio robot is shown in figures 4.20 though 4.23.

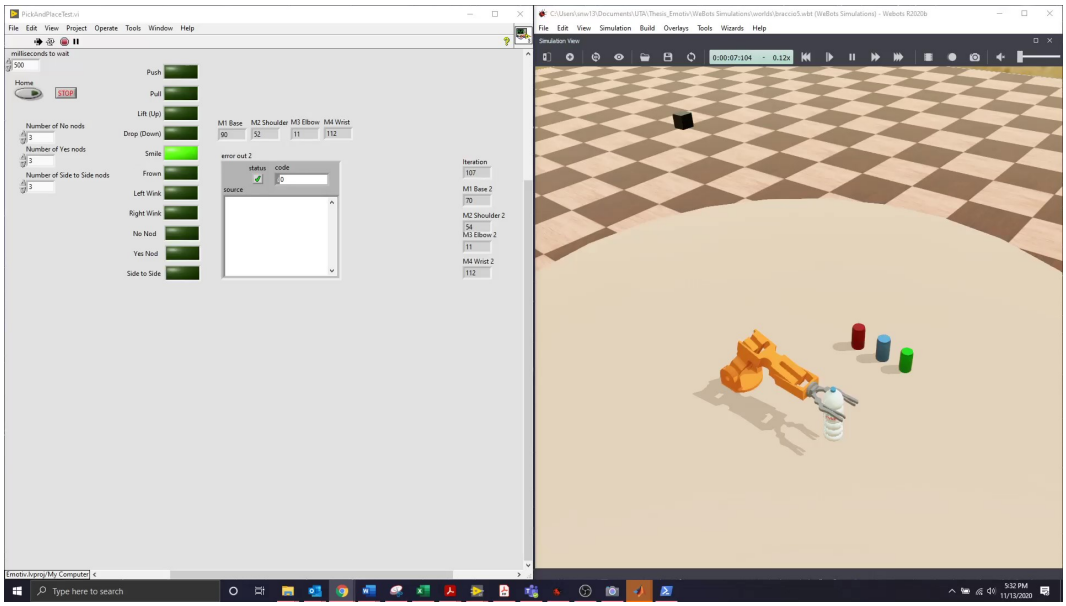

Figure 4.17. *Smile* Triggered to Close Gripper.

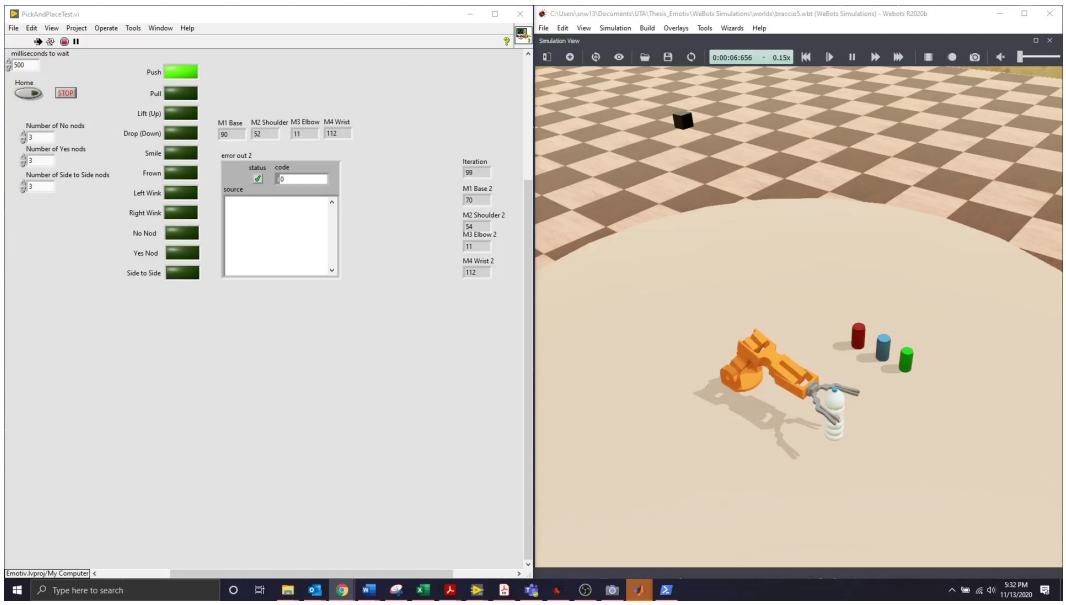

Figure 4.16. Test 5: *Push* Command Triggered for Pick Up Position.

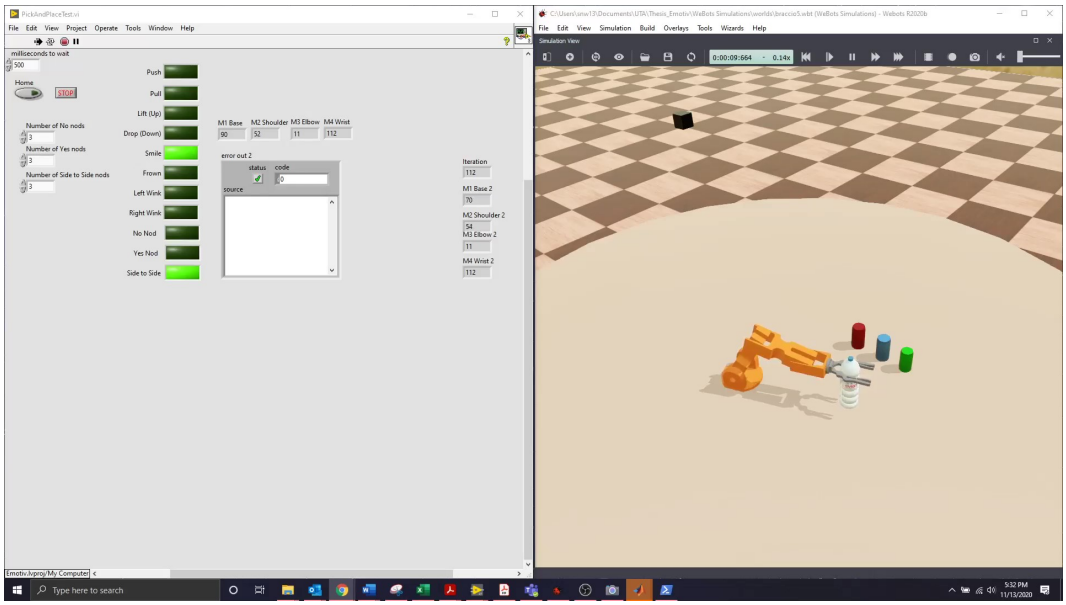

Figure 4.18. Test 5: *Side to Side Nod* for Put Down Position.

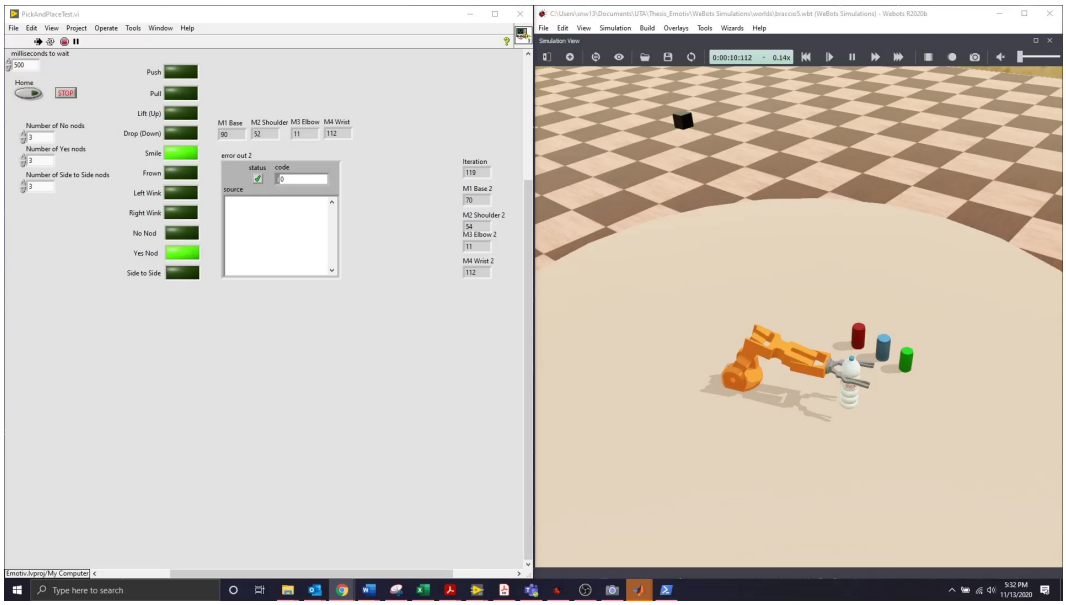

Figure 4.19. *Yes Nod* to Open Gripper.

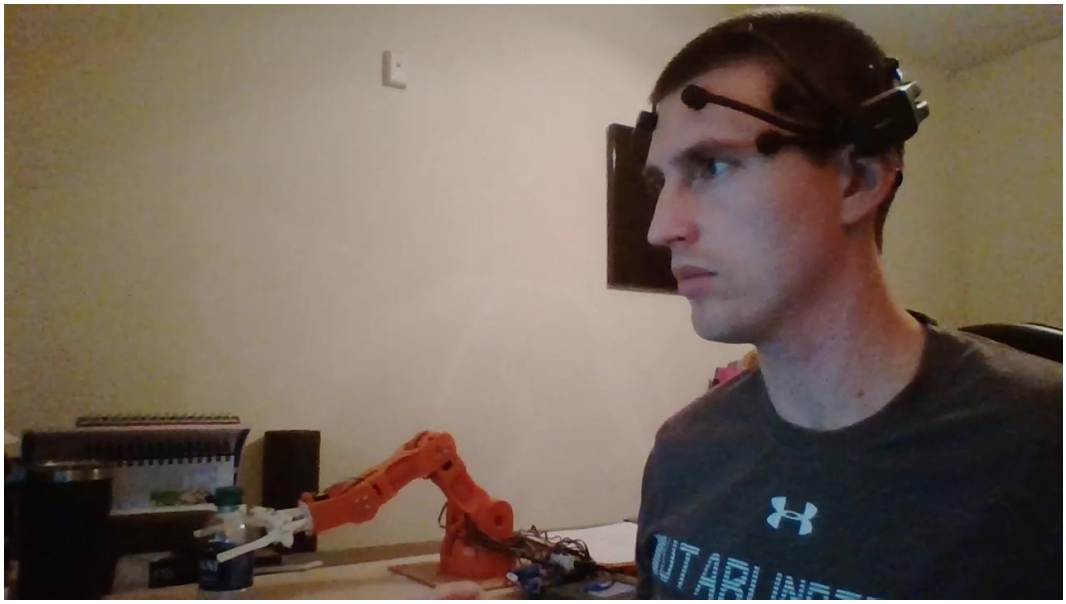

Figure 4.20. Test 5: *Push* Command Triggered for Pick Up Position.

Figure 4.21. Test 5: *Smile* Triggered to Close Gripper.

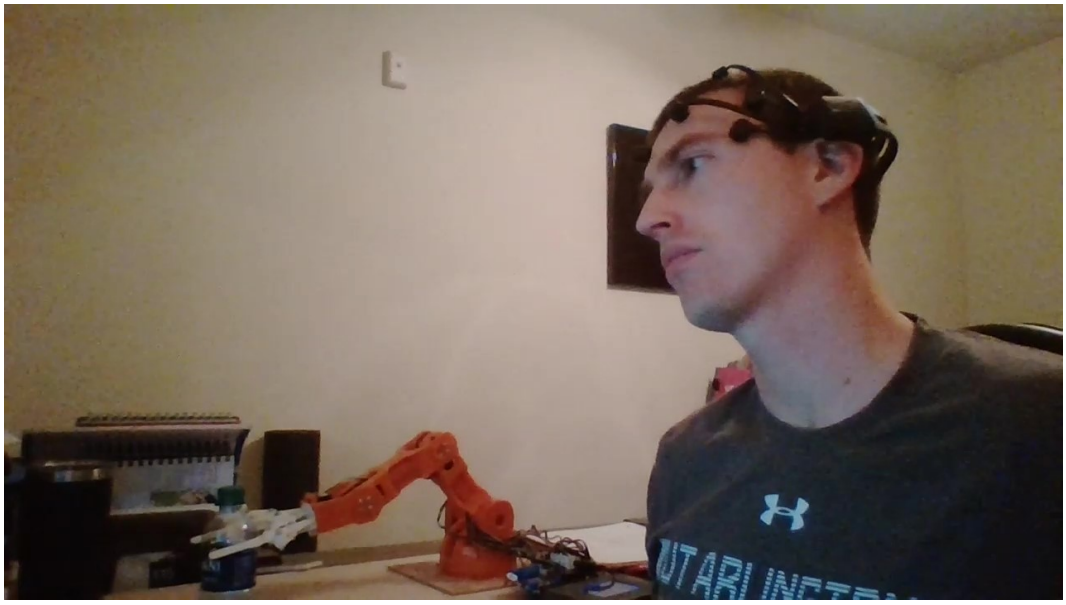

Figure 4.22. Test 5: *Side to Side Nod* for Put Down Position.

Figure 4.23. Test 5: *Yes Nod* to Open Gripper.

4.5.1 Test 5A: Pick and Place

The pick and place routine was repeated with a different mapping of the EPOC+ outputs to robot actions. The test was intended to demonstrate that this mapping procedure is easily configurable, and it can be changed based on a user's preference. The same training profile and BSD from the first pick and place routine were used. This test was successfully executed in the Webots simulation and on the hardware. The commands for each robot action are:

1. *Push*: Move to water bottle
2. *Yes Nod*: Close gripper
3. *Side to Side Nod*: Move to place location
4. *Smile*: Open gripper

4.6 Test 6: Integrating a Camera in Simulation

While the pick and place routine on the Braccio was successful, this fixed routine relies on the object to be in exactly the same location every time. The purpose of Test 6 was to demonstrate that an object could be detected and picked up independent of its location as long as it is within the robot workspace. In order to perform this test, a Webots environment was created with the wheelchair robot since it has a larger workspace. A camera was added so that if could detect the position of an object, a can in this case. The camera estimated the X, Y, and Z coordinates of the can center of mass in its own frame. Then the position of the can relative to the robot base was calculated using the transformation between the camera and base of the wheelchair robot. From that information, the inverse kinematics are calculated, and the robot moves into position to pick up the can. This simulation test was performed several times with the can placed at a different starting location, and each time the robot moved to the new position of the can. The success of this test shows that the EPOC+, in combination with a camera, could be used to grab objects in a dynamic setting. The robot could then bring the can to the human mouth, or any other desired location. A setup of the Webots simulation is shown in figure 4.24.

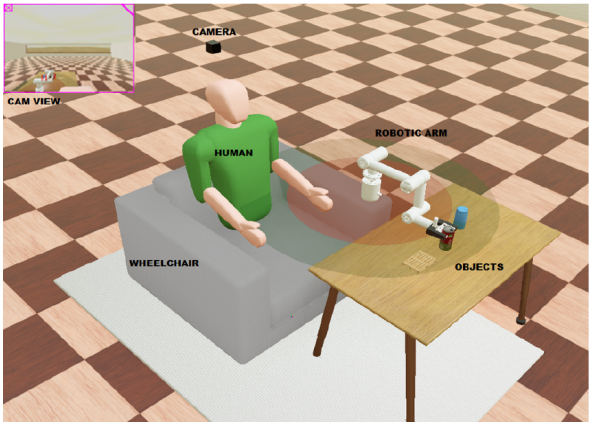

Figure 4.24. Test 6: Wheelchair Robot Simulation (Courtesy of Shubham Gunjal).

4.7 Chapter 4 Conclusion

The successful execution of several tests proves the EPOC+ can be an effective device for controlling a robot. However, there were a few challenges with getting this setup to work. While training one mental command was successful, it was difficult to train two mental commands and achieve a quality BSD. This could be improved by training the headset on a regular basis, and practicing meditation to enhance concentration [3]. The facial expressions easier to train and most repeatable were the *Smile* and the *Surprise*. The *Left Wink*, *Right Wink*, and *Blink* expressions did seem quite easy to trigger without any training needed, however were not used due to the possibility of unintended false positives. The nods were quite easy to trigger, and while only *Yes*, *No* and *Side to Side* nods were used, there are many other types of head movements that can be detected. The felt pads that came into contact with the scalp did tend to dry out after about an hour of use and needed to be rehydrated. Emotiv has made a new EPOC X that address this issue allowing for rehydration while the headset is sill on the user.

One of the other main challenges was reading commands into LabVIEW and MATLAB. While the developed method worked most of the time, there were occasions where it seemed to be trying to read the text file while Python had it open to write the state vector. This led to two issues. The first is that LabVIEW would try to read the file then interpret it as all commands being active at the same time. The second issues with MATLAB led to the vector not being read, and instead showed as an empty variable. In order to alleviate this problem, a delay function was used in both LabVIEW and MATLAB to slow the rate at which they attempted to read the text file. This helped with this issue, however a more robust solution should be developed.

CHAPTER 5

Conclusions and Recommendations for Future Work

5.1 Conclusions

The objective of this research was to develop and demonstrate an HRI framework for controlling a robotic manipulator though a BCI modality for an assistive application. In order to accomplish this, an Emotiv EPOC+ EEG headset and a Braccio hobby robot were used for evaluation. The EPOC+ headset allowed for the collection of several different control signals that were used to execute a desired robotic action. These control signals included mental commands, facial expressions, and nods. The mental commands and facial expressions required training, while the nods were detected from the gyroscope data. Performing the evaluation of this BCI was accomplished using physical and simulated robotic manipulators. In order to perform this evaluation, it was necessary to develop a software architecture that extracted the command signals from the EPOC+ headset and translated them into LabVIEW and MATLAB to control the physical and simulated robot. The Braccio robot was used as the proof-of-concept hardware for simple manipulation tasks to confirm that the BCI is a sufficient modality to control the robot. A pick and place routine was successfully executed on the physical Braccio robot and in a Webots simulation, using a combination of mental commands, facial expressions and nods. While this demonstrated the ability to perform a hard coded pick and place operation for an object from a known location, this capability was expanded to dynamic path planning in simulation by introducing a camera. The simulated camera allowed the detection of an object position relative to the robot, which is used to solve the inverse

kinematics to pick it up. This capability was shown for the wheelchair robot in a Webots simulation.

The successful tests performed during this research proved that an EEG headset can be an effective modality for controlling an assistive robot. While multiple mental commands can be difficult to train when not done on a consistent basis, it is likely that the end user of an assistive robot would be able to train daily and therefore repeatedly trigger these commands. The improved performance due to repeated training was shown in [13], where a group of disable people had a high average success rate of triggering mental commands than the other two groups of individuals.

The facial expressions and nods proved to be relatively easy to trigger, and proved to be a reliable method for controlling a robot. The framework developed during this research for the BCI modality provides a foundation for further enhancement of HRI. With further refinement, this BCI modality would allow the end user to efficiently control a robot to manipulate their environment and reduce dependence on others.

5.2 Recommendations for Future Work

Future work should include the integration of a 3D camera into the hardware pick and place routine. This would help verify that the code used for the camera in the Webots simulation could transfer to hardware and allow the robot to identify an object and its location for manipulation. In addition, it would be beneficial to put new motors on the inexpensive wheelchair robot so that the full HRI modality through a BCI can be tested in a situation similar to how it would be used in the real application.

Another important aspect of this research will be to evaluate how trainable the EPOC+ headset is for a variety of individuals. This could be accomplished by having

several individuals train mental commands and facial expressions on the headset, then evaluate if they can perform the pick and place routine as a base test in simulation before moving to the hardware. The training could be accompanied with reading the raw data from each sensor, which can be accessed with a paid subscription to the EmotivPRO application. Raw EEG data would allow further analysis, and identify which sensors and regions of the brain contribute most to effective training for a user. Applying signal processing and machine learning techniques to analyze the raw EEG data could allow for additional commands to be discovered.

Based on experience from this research, it would be beneficial to utilize a Python API to communicate with the EPOC+ headset. This Python API was only released in the past few months, after extensive research has been completed, and the reason why it was not investigated for this research. The Python API has the capability to develop a more robust method of communicating with MATLAB and LabVIEW, as opposed to writing text files. There is a MATLAB Engine API that can communicate with Python, which would make a majority of the software developed during this research easily reusable.

When deploying this framework to a real assistive application it is necessary to develop a verification step that would allow the end user to confirm the desire robot action before executing on the hardware. The verification could be a simulation of the robot presented on a screen, where the robot is executing the actions interpreted from the BCI output. If these simulated actions match the intent of the user, then the user can approve for the robot to execute the process. Otherwise, the user could reject the presented actions and attempt to re-issue the commands.

APPENDIX A

Braccio Inverse Kinematics Code

The material in the appendix describes how the inverse kinematics analysis was implemented into the LabVIEW controller. The LabVIEW GUI. shown in figure A.1, is where the user inputs the desired position and orientation of the end effector. The information is provided to an inverse kinematics Virtual Instrument (VI) as shown in figures A.2 and A.3. The inverse kinematics VI uses a MATLAB script to turn the inputs into a desired transformation matrix. The inverse kinematics VI then calls the *Braccio_IK_R2* MATLAB function to solve for the joint angles.

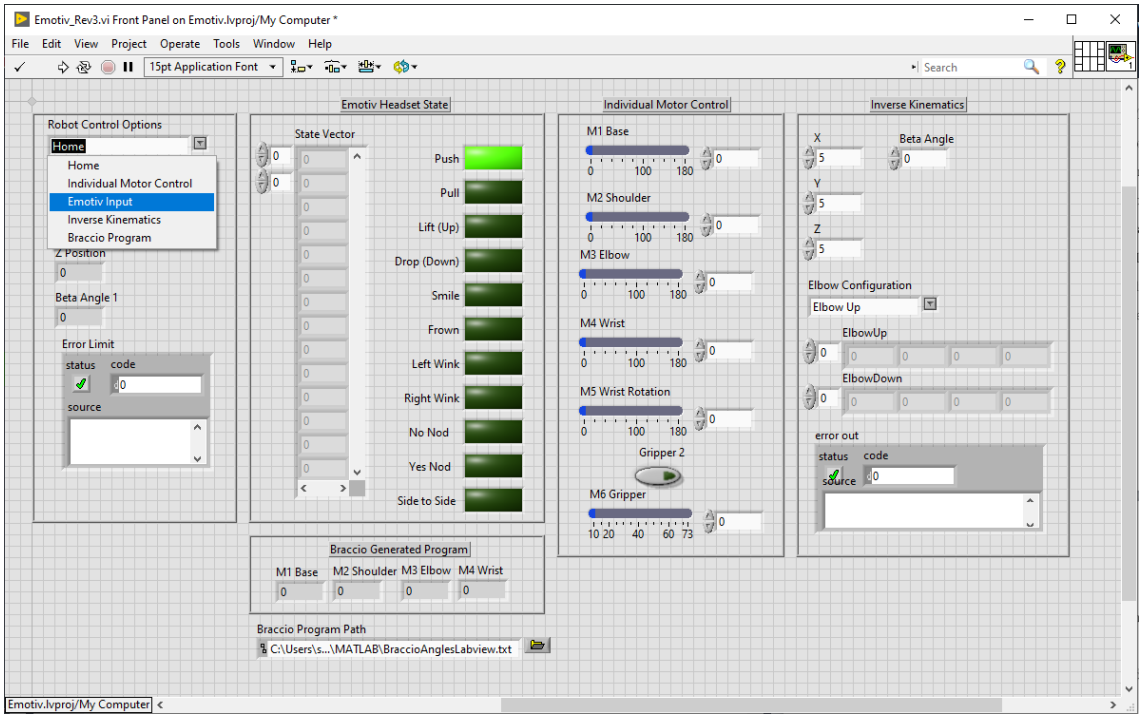

Figure A.1. LabVIEW GUI.

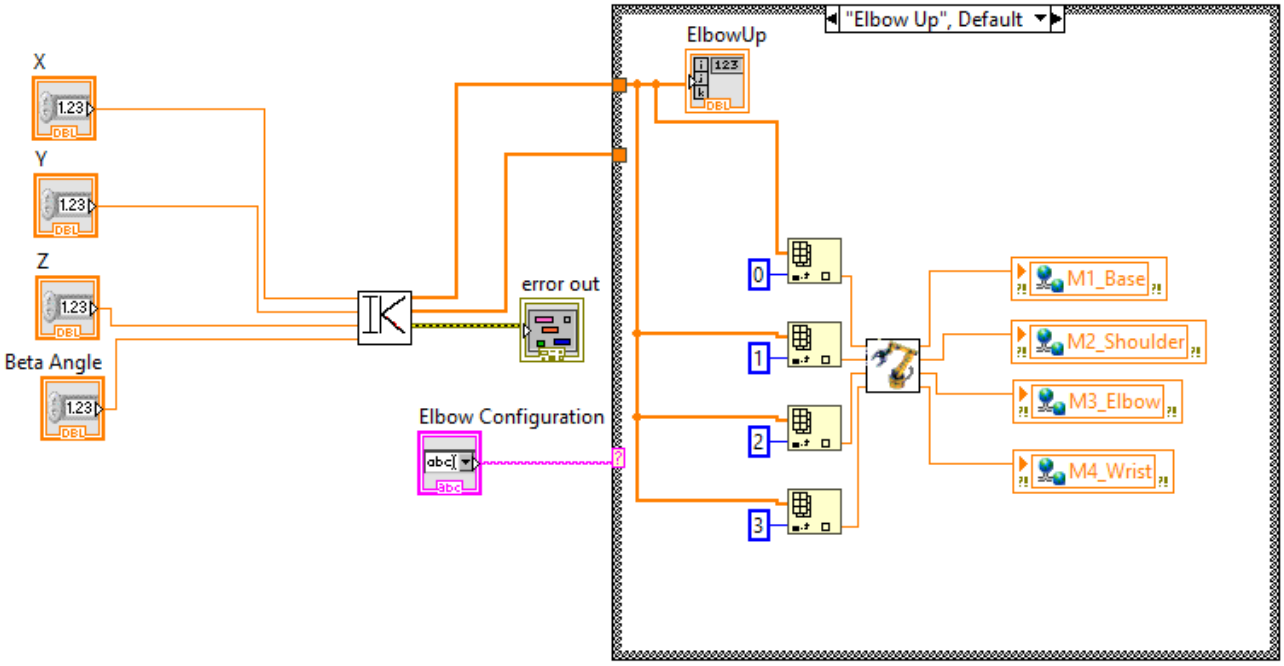

Figure A.2. LabVIEW GUI Block Diagram for Inverse Kinematics.

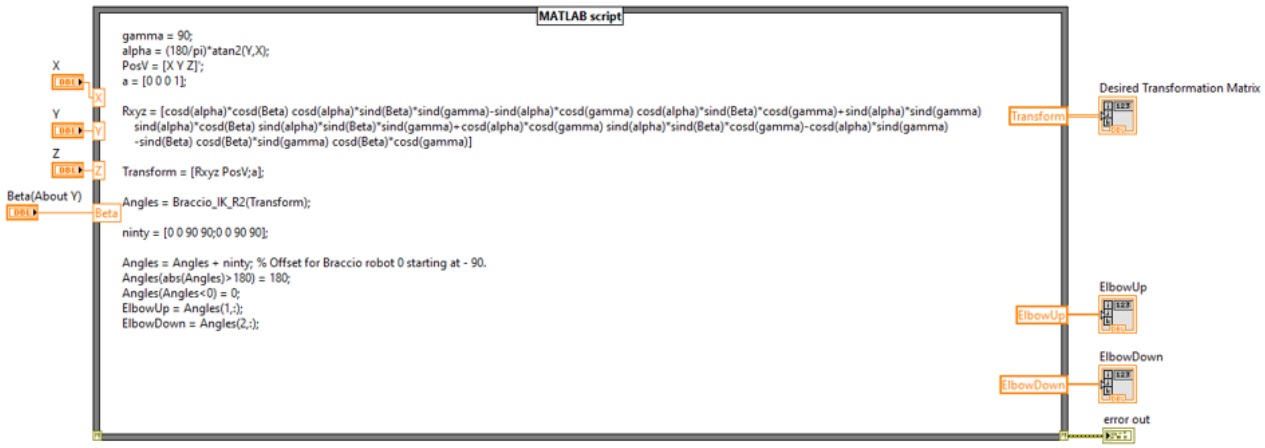

Figure A.3. LabVIEW VI for Inverse Kinematics.

The following two MATLAB functions are the implementation of the inverse kinematics solution for the Braccio robot derived in section 2.3.1.

```
function angles = Braccio_IK_R2(T0H)
x = T0H(1,4);
y = T0H(2,4);
z = T0H(3,4);
a = [0 0 0 1];
theta1 = (180/pi)*atan2(y,x);
t1 = theta1;
%Establish Matrix so it can be turned back to XZ
T0_1 = [cosd(t1) -sind(t1) 0 0
        sind(t1) cosd(t1) 0 0
        0        0        1 0
        a];
 %Turn robot back to XZ Plane
 Planar = inv(T0_1)*T0H;
 x = Planar(1,4);
 OldRot = Planar(1:3,1:3);
 Rotation = Rot(1,-90);
 NewRot = Rotation*OldRot;

 xyz = [x z 0]'; % Planar X and Z are new X and Y
 NPlanar = [NewRot xyz;a];

ThreeAngles = ThreeR_IK_Braccio(NPlanar);

t1 = [t1;t1];
angles = [t1 ThreeAngles];
end
```

```
function Output = ThreeR_IK_Braccio(T0_H)

L1 = 5;
L2 = 5;
L3 = 5.75;
a = [0 0 0 1];

T3_H = [1 0 0 L3
        0 1 0 0
        0 0 1 0
        a];

T0_3 = T0_H*invT(T3_H);

  x = T0_3(1,4);
  y = T0_3(2,4);

c2 = (x^2 + y^2 - L1^2 - L2^2)/(2*L1*L2);

if (c2 > 1)
    disp('The desired location is outside the workspace.')
elseif (c2 < -1)
    disp('The desired location is outside the workspace.')
else

    s2p = sqrt(1-c2^2);
    s2n = -sqrt(1-c2^2);

    theta2p = (180/pi)*atan2(s2p,c2);
    theta2n = (180/pi)*atan2(s2n,c2);
```

```
31
32      k1 = L1 + L2*c2;
33      k2 = L2*s2p;
34  %     rp  = sqrt(k1^2+k2^2);
35      gam = (180/pi)*atan2(k2,k1);
36      theta_1p = (180/pi)*atan2(y,x) - gam;
37
38      k2 = L2*s2n;
39  %     rn  = sqrt(k1^2+k2^2);
40      gam = (180/pi)*atan2(k2,k1);
41      theta_1n = (180/pi)*atan2(y,x) - gam;
42
43      c123 = T0_H(1,1);
44      s123 = T0_H(2,1);
45      phi = (180/pi)*atan2(s123,c123);
46      theta3p = phi - theta_1p - theta2p;
47
48      theta3n = phi - theta_1n - theta2n;
49
50      Sol_1 = [theta_1p theta2p theta3p];
51      Sol_2 = [theta_1n theta2n theta3n];
52
53      Output = [Sol_1
54                Sol_2];
55  end
```

REFERENCES

[1] B. J. Edelman, J. Meng, D. Suma, C. Zurn, E. Nagarajan, B. S. Baxter, C. C. Cline, and B. He, "Noninvasive neuroimaging enhances continuous neural tracking for robotic device control," Science Robotics, vol. 4, no. 31, p. eaaw6844, jun 2019.

[2] J. Meng, S. Zhang, A. Bekyo, J. Olsoe, B. Baxter, and B. He, "Noninvasive electroencephalogram based control of a robotic arm for reach and grasp tasks," Scientific Reports, vol. 6, no. 1, dec 2016.

[3] EPOC+ User Manual, Emotiv, 2018. [Online]. Available: https://emotiv.gitbook.io/epoc-user-manual/

[4] A. Al-Qahtani, A. Nasir, M. Z. Shakir, and K. A. Qaraqe, "Cognitive impairments in human brain due to wireless signals and systems: An experimental study using EEG signal analysis," in 2013 IEEE 15th International Conference on e-Health Networking, Applications and Services (Healthcom 2013). IEEE, oct 2013.

[5] J. Craig, Introduction to Robotics: Mechanics and Control. Pearson, 2018. [Online]. Available: https://books.google.com/books?id=JblZuwAACAAJ

[6] EmotivBCI, Emotiv, 2018. [Online]. Available: https://emotiv.gitbook.io/emotivbci/

[7] EmotivBCI Toolbox, Emotiv. [Online]. Available: https://emotiv.gitbook.io/emotivbci-node-red-toolbox/

[8] E. N. Arcoverde Neto, R. M. Duarte, R. M. Barreto, J. P. Magalhães, C. C. Bastos, T. I. Ren, and G. D. Cavalcanti, "Enhanced real-time head pose estima-

tion system for mobile device," Integrated Computer-Aided Engineering, vol. 21, no. 3, pp. 281–293, Apr 2014.

[9] "Paralysis statistics - reeve foundation," https://www.christopherreeve.org/living-with-paralysis/stats-about-paralysis, (Accessed on 11/23/2020).

[10] K. Ziegler-Graham, E. J. MacKenzie, P. L. Ephraim, T. G. Travison, and R. Brookmeyer, "Estimating the prevalence of limb loss in the united states: 2005 to 2050," Archives of Physical Medicine and Rehabilitation, vol. 89, no. 3, pp. 422–429, mar 2008.

[11] E. Musk, "An integrated brain-machine interface platform with thousands of channels," Journal of Medical Internet Research, vol. 21, no. 10, p. e16194, oct 2019.

[12] "The introductory guide to eeg (electroencephalography) - emotiv," https://www.emotiv.com/eeg-guide/, (Accessed on 11/23/2020).

[13] P. Chowdhury, S. S. K. Shakim, M. R. Karim, and M. K. Rhaman, "Cognitive efficiency in robot control by emotiv EPOC," in 2014 International Conference on Informatics, Electronics & Vision (ICIEV). IEEE, may 2014.

[14] D. Prince, M. Edmonds, A. Sutter, M. Cusumano, W. Lu, and V. Asari, "Brain machine interface using emotiv EPOC to control robai cyton robotic arm," in 2015 National Aerospace and Electronics Conference (NAECON). IEEE, jun 2015.

[15] S. Grude, M. Freeland, Chenguang Yang, and Hongbin Ma, "Controlling mobile spykee robot using emotiv neuro headset," in Proceedings of the 32nd Chinese Control Conference, 2013, pp. 5927–5932.

[16] W. A. Jang, S. M. Lee, and D. H. Lee, "Development BCI for individuals with severely disability using EMOTIV EEG headset and robot," in 2014 International Winter Workshop on Brain-Computer Interface (BCI). IEEE, feb 2014.

[17] S. Aguiar, W. Yanez, and D. Benitez, "Low complexity approach for controlling a robotic arm using the emotiv EPOC headset," in 2016 IEEE International Autumn Meeting on Power, Electronics and Computing (ROPEC). IEEE, nov 2016.

[18] A. Kline and J. Desai, "SIMULINKsup&#x00ae/sup based robotic hand control using emotiv&#x2122 EEG headset," in 2014 40th Annual Northeast Bioengineering Conference (NEBEC). IEEE, apr 2014.

[19] W. Ouyang, K. Cashion, and V. K. Asari, "Electroencephelograph based brain machine interface for controlling a robotic arm," in 2013 IEEE Applied Imagery Pattern Recognition Workshop (AIPR). IEEE, oct 2013.

[20] I. N. Zamora, D. S. Benitez, and M. S. Navarro, "On the use of the EMOTIV cortex API to control a robotic arm using raw EEG signals acquired from the EMOTIV insight NeuroHeadset," in 2019 IEEE CHILEAN Conference on Electrical, Electronics Engineering, Information and Communication Technologies (CHILECON). IEEE, nov 2019.

[21] 10/20 System Positioning, Trans Cranial Technologies Idt., 2012.

[22] A. Cassidy, "Facial expression detections," Aug 2019. [Online]. Available: https://www.emotiv.com/knowledge-base/facial-expression-detections/

[23] NI myRIO-1900 User Guide and Specifications, National Instruments, June 2018. [Online]. Available: https://www.ni.com/pdf/manuals/376047c.pdf

[24] "cyberbotics/urdf2webots: Utility to convert urdf files to webots proto nodes," https://github.com/cyberbotics/urdf2webots, (Accessed on 11/24/2020).

[25] "Hand and forarm – inmoov," http://inmoov.fr/hand-and-forarm/, (Accessed on 11/27/2020).