

## Use Authorization

In presenting this dissertation in partial fulfillment of the requirements for an advanced degree at Idaho State University, I agree that the Library shall make it freely available for inspection. I further state that permission to download and/or print my dissertation for scholarly purposes may be granted by the Dean of the Graduate School, Dean of my academic division, or by the University Librarian. It is understood that any copying or publication of this dissertation for financial gain shall not be allowed without my written permission.

Signature \_\_\_\_\_

Date \_\_\_\_\_

# Surface EMG and EEG Signal Fusion for Embedded Control of Prosthetic Hand Motions

By

Alex N. Jensen

A Dissertation submitted in partial fulfillment of the  
Requirement for the degree of

Doctor of Philosophy

In

Engineering and Applied Science

**IDAHO STATE UNIVERSITY**

**May 2014**

To the Graduate Faculty:

The members of the committee appointed to examine the dissertation of ALEX JENSEN find it satisfactory and recommend that it be accepted.

---

**Dr. Steve Chiu, Major Advisor**

---

**Dr. Marco Schoen, Advisor**

---

**Dr. Gene Stuffle, Advisor**

---

**Dr. Alba Perez, Advisor**

---

**Dr. Ken Bosworth, Advisor**

---

**Dr. Subbaram Naidu, Advisor**

---

**Professor James Creelman, Graduate Faculty**

**Representative**

I dedicate this thesis to Joyce and Sid Jensen, my parents, my brothers Gavin, Tyson and Trent Jensen, and dear friends for their continuous support, help, encouragement, and motivation for the completion of this research work. I dedicate it also to Amanda J. Clark, for her support, encouragement, and patients in this research and her valuable input to this work. Along with Nathan Spark, for his friendship and technical help with exploring new work on biological signals used in this work. I would have never taken on this adventure without their help and support.

# Acknowledgement

I would like to extend special thanks to my major advisor, Dr. Steve Chiu. We have worked together for many years while working on my bachelors, master and now my Phd. He has been a great help and inspiration while studying and working. Lastly he has become a close friend and colleague. Secondly, I would like to thank my advisor, Dr. Marco Schoen, for his assistance and guidance in completing this dissertation and my master's thesis. I am also thankful to my fellow classmates and friends who helped me on this work. I am grateful for the help of my committee for their advice, and help on this also Dr. Gene Stuffle, Dr. Alba Perez, Dr. D. Subbaram Naidu, Dr. Ken Bosworth, and Jim Creelmen.

I would also like to thank my family for their help and support while working on this research.

# Table of Contents

List of Figures .....	x
List of Tables .....	xiii
Key Words.....	xiv
Abstract.....	xv
Chapter 1: Introduction.....	1
1.1 Problem Statement .....	1
1.2 Dissertation Theme .....	3
1.3 Dissertation Objective and Outline .....	4
Chapter 2: Literature Review.....	7
2.1 Current Prosthetic Hands .....	7
2.2 Prosthetic Hand Controls .....	8
2.3 Anatomy of the Forearm and Thumb.....	10
2.3.1 EMG.....	12
2.3.2 EEG.....	16
2.4 System Identification.....	19
2.4.1 System Identification Algorithm.....	19
2.4.2 State Space .....	22
Chapter 3: Biological Acquisition .....	24
3.1 Overview .....	24
3.2 sEMG .....	24
3.3 EEG .....	26
3.3.1 Emotiv EPOC™.....	26

Chapter 4: Embedded Processor .....	29
4.1 Overview .....	29
4.2 Arduino.....	29
4.3 Major Arduino Functions Used and Their Descriptions .....	31
Chapter 5: Expansion on Previous Work.....	33
Chapter 6: Two Motor Robotic Thumb .....	39
6.1 Proposed EEG Headset Design .....	39
6.2 Key Command Code .....	41
6.3 Motor Code .....	42
6.3.1 Arduino Code.....	43
6.4 Equipment .....	44
6.4.1 Hardware Setup.....	44
6.4.2 Embedded Processor.....	44
6.4.3 EEG Headset.....	45
6.5 Equipment Setup .....	45
6.5.1 Embedded Processor .....	45
6.5.2 EEG Headset.....	48
6.6 Safety.....	48
6.7 Software .....	49
6.8 Experiment .....	49
6.9 System Identification Fuzzy Controller Hybrid .....	53
6.9.1 Experiment Setup.....	53
6.9.2 SI of EEG Signals .....	55

6.9.3	Fuzzy Controller Design .....	58
6.9.4	Validations .....	63
Chapter 7:	Six DoF Prosthetic Hand Experiment and Setup .....	66
7.1	Equipment .....	68
7.1.1	New Equipment .....	69
7.2	Equipment Setup .....	70
7.3	Test Setup .....	73
7.3.1	sEMG Thesis Entropy Method .....	73
7.3.2	EEG Signals .....	77
7.4	Six DoF Hand Tools and Interface Development .....	78
7.4.1	Hand Visualization Tool for a Six DoF Hand .....	78
7.4.2	Control of Virtual Hand .....	79
7.4.3	sEMG Entropy Algorithm Implementation .....	81
7.4.4	EEG Signal .....	87
7.4.5	Text File Format .....	87
7.5	Six DoF Hand Experiment .....	89
7.5.1	Virtual Hand and Embedded System's Prosthetic Hand Code .....	89
7.5.2	Servo Motor and Embedded Platform .....	93
7.5.3	Biological Signal Input .....	94
7.6	Five DoF Robotic Hand Implementation .....	95
Chapter 8:	Results and Analyses .....	98
Chapter 9:	Conclusion .....	102

Chapter 10: Future Work .....	105
References .....	106
Appendix .....	112
Emotiv File Saving Process After an Experiment .....	112
Arduino Startup Check List .....	114
sEMG Processed Signal Text File Sample .....	118
EEG Processed Signal Text File Sample .....	119
AMUSE MATLAB™ Code .....	120
System Identification Steps .....	123
MATLAB™ Figure Formatting and Creating Code .....	127
List of Publications .....	128

# List of Figures

Figure 2.1: Major Flexor Muscle Groups in the Forearm, [13] .....	10
Figure 2.2: Forearm with Zone 1 and Zone 2, [2] .....	14
Figure 2.3: Motor Units, [22].....	15
Figure 2.4: Electrode placement: A) Side profile B) Top profile C) EEG electrode placement map, [25].....	18
Figure 3.1: Emotiv sensor array, [37] .....	27
Figure 3.2: Emotiv cover display, [37] .....	28
Figure 4.1: Arduino Uno Board, [38] .....	30
Figure 6.1: Block Diagram for two Motor Setup.....	40
Figure 6.2: Arduino Processing Button [1].....	46
Figure 6.3: Flow Diagram of Two Motor System [1].....	47
Figure 6.4: SDK Control Panel, [36]. .....	50
Figure 6.5: Emotiv Key Binding Interface, [36].....	51
Figure 6.6: Arduino Servo Motor Setup .....	53
Figure 6.7: Motion Path for Patients.....	55
Figure 6.8: System Identification Toolbox from MATLAB™ .....	56
Figure 6.9: STI Model Output .....	58
Figure 6.10: Fuzzy Controller Membership Function Input .....	59
Figure 6.11: Fuzzy Controller Membership Function Output .....	60
Figure 6.12: Fuzzy Controller Rules.....	61
Figure 6.13: Fuzzy Controller Surface Rules Visualization .....	61

Figure 6.14: System Identification Fuzzy Controller Simulink Model .....	62
Figure 6.15: Fuzzy STI Model Output.....	63
Figure 6.16: Validation Plot.....	64
Figure 7.1: Multi-Motor Setup.....	67
Figure 7.2: Electrodes and Connectors .....	71
Figure 7.3: Electrode Array Placement.....	72
Figure 7.4: Virtual Hand Open .....	79
Figure 7.5: Virtual Hand Closed.....	79
Figure 7.6: sEMG Signal for Multiple Hand Motion .....	82
Figure 7.7: Fuzzy Controller Membership Functions Plot .....	83
Figure 7.8: Fuzzy Controller Output Values.....	84
Figure 7.9: sEMG Fuzzy Controller Flow Diagram .....	85
Figure 7.10: sEMG Actual Hand Motion .....	86
Figure 7.11: sEMG Predicted Hand Motion with Entropy .....	86
Figure 7.12: Processing Code Flow Diagram .....	91
Figure 7.13: Arduino Code Flow Diagram .....	92
Figure 7.14: Six Servo Motor Setup with Embedded System .....	94
Figure 7.15: Five DoF Prosthetic Hand .....	96
Figure 0.1: Emotiv TestBench, [45] .....	113
Figure 0.2: Convert EDF to CSV, [45] .....	114
Figure 0.3: Arduino Board Selection .....	115
Figure 0.4: Unknown Device Detected.....	116
Figure 0.5: COM Issues .....	116

Figure 0.6: Done Uploading .....	117
Figure 0.7: Arduino Front Conceal .....	117
Figure 0.8: System Identification Toolbox .....	125
Figure 0.9: Linear Parametric Model Menu .....	125
Figure 0.10: System Identification Output.....	126

# List of Tables

Table 2.1: EEG Waves and Frequencies, [23] .....	16
Table 7.1: Entropy for Single Motion End Results [2] .....	75
Table 7.2: Summary of Signal Classification by the Intelligent Classifier, [2] .....	77

# Key Words

A	Amp or Ampere
CSV	Comma Separated Value
EDF	European Data Format
EEG	Electroencephalography
EEPROM	Electrically Erasable Programmable Read-Only Memory
EMG	Electromyogram
sEMG	surface Electromyogram
Hz	Cycle per Second
I/O	Input/Output
KB	Kilo Bytes
M	Mega or $10^6$
m	milli or $10^{-3}$
PWM	Pulse Width Modulated
SRAM	Static Random Access Memory
UART	Universal Asynchronous Receiver/Transmitter
USB	Universal Serial Bus

# Abstract

This dissertation focuses on developing an embedded platform for controlling the actuation of a robotic hand and creating a virtual interface environment for it. The work done in this dissertation was built on my master's thesis and multiple publications during my masters and doctorate work.

The first stage of this research focused on creating an embedded system platform that could use Electroencephalography (EEG) to control a two degree-of-freedom (DoF) thumb. This work was successfully done using inexpensive components and an EEG headset for the EEG actuation. The patients could be trained and then control the robotic finger in real-time. This part of the work was so successful that it was published in an IEEE paper, [1].

Another part of this work was to further develop the embedded platform keyboard interface made in, [1]. This interface for that paper was able to control two finger motions. This dissertation expanded that so that it could control a 6 DoF hand. The input was from keyboard keys like in, [1] and added to that was a virtual hand. The virtual hand mimicked what the real hand would do when a command was sent since the embedded system received a similar signal to the virtual hand.

With the knowledge and background in sEMG and EEG signals from previous work, this dissertation developed a platform that can test these algorithms for controlling a prosthetic hand. To do this the controller developed in my thesis along with conference

work on sEMG in [2] and [3] were used and then expanded upon. The expanded work was done because the original controller identified single hand motions at a time and wasn't designed to handle multiple hand motions being sent to it in one continuous file. The EEG control method and signals from [1] were used for the EEG portion.

This dissertation designed a platform that can receive multiple biological signals and can control a prosthetic hand with those signals. This system can be controlled by a plethora of algorithms that use a defined output protocol. This unique hybrid approach shows that the platform is very versatile and can take multiple types of inputs. The design of the embedded platform will allow for expansion to enable it to easily be used in future research since it will be programmed using C code. These properties will allow it to be an ideal test bed for testing and fine tuning control algorithms.

# **Chapter 1: Introduction**

## **1.1 Problem Statement**

Currently there are more than 2 million Americans who have missing limbs, and this number increases by 185,000 people per year [4]. According to USA Today, due to the Afghanistan war, in one month 134 service members lost a limb from mines or war-related injuries, [5]. Current research is focused on creating more intuitive prosthetics to address the rising need for them. From past research, a robotic prosthetic hand should be autonomous, have a high level of functionality, comfort and be easy to use [6]. As part of being easy to use, there should be a natural way of communicating with the robotic limb [7].

Today's robotics have advanced greatly in the last decades but these advancements have not carried over to prosthetic arms and hands. These upper prosthetic limbs have not advanced much and are still using the simplistic hook and claw. The principles and technology developed by the first commercial prosthetic hands at the Central Prosthetic Research Institute of the U.S.S.R in 1964, have not been improved on much since [8]. One of the reasons for this is because of cost. Many of the advanced robotic technologies are still very new and costly. A platform needs to be made that can allow for development of controls and prosthetic hands more cost effectively.

One of the major problems that keeps new prosthetics from being fully used is effective control of the hand. The purpose of this dissertation is to find a cost-effective, non-

invasive method to control a prosthetic hand. Part of the way the cost will be kept low is that a virtual hand will be created that can be used to test algorithms without having the actual hand connect. This will make the platform more versatile since multiple groups can be working in parallel with one prosthetic hand.

Classical control methods for prosthetics have many drawbacks when it comes to controlling smart prosthetics, especially prosthetics with two or more degrees of freedom. Classical control techniques, such as electromyogram (EMG) sensors, have been used ever since the development of the first prosthetics and have been reliable when generating motor movement with electro-potentials from EMG. However, with the introduction of multiple degrees of freedom for artificial hands, classical control techniques are insufficient to account for the added complexity.

A problem with EMG methods is successfully implementing them to control a thumb. Current work with EMG has only been able to model the hand grasping and not performing fine control for each finger or thumb, [2]. This problem will be addressed in this work by investigating the issue of whether or not EEG signals can be systematically used to control the movement trajectory of an artificial thumb.

This research will need to investigate using both sEMG and EEG signals to control a robotic hand. Along with developing control algorithms that can identify the intent of the two signals, a platform will need to be created to receive the signals and then control the robotic hand. This platform needs to be able to work with different programs commonly used to develop algorithms. The platform needs to be versatile and be programmed using a common language. This will allow for future development on this system to be done.

## 1.2 Dissertation Theme

Research is being done to design prosthetic hands that can be controlled by sEMG or EEG signals. sEMG signals occur when a motor unit receives a command from the brain and the motor units that activate the contraction of muscles in the arm. These signals can be measured on the skin as an electric potential by electrodes. sEMG signals are random in nature, and it is difficult to characterize the intent of the signal. EEG signals are electro potentials generated by many neurons that are in the brain. These signals can be measured by surface electrodes or implanted sensors.

This dissertation will further develop an effective process of characterizing sEMG signals and use that to control advanced prosthetic hands. Also, this work will include developing an effective method for acquiring EEG signals that can be used in the control of a robotic thumb to give the hand greater dexterity. In this dissertation, Shannon entropy is used to characterize different hand gestures as sEMG signals. The sEMG signals are processed with an entropy algorithm. Different commonly performed tasks were used such as grasping a water bottle or grasping a key. These tasks were characterized based on entropy values. Each patient was trained for the EEG signals. This training process is important because each person's EEG is unique. After the training process the patient was to move the thumb in a pre-defined set of motions for a given length of time.

The intent of this dissertation is to develop a platform that can take the above biological signals and use them to control a prosthetic hand. A problem in developing control

algorithms is that they are often developed using unique, specialized software packages, making it difficult to interface with a prosthetic hand. This dissertation will develop a platform that can utilize many different control algorithms from different software packages.

To help with the algorithm development, a visualization method was created that allows the users to know when the different motors are being sent commands and in which direction. This visualization tool will also be able to control the hand with different input signals from various controllers.

### **1.3 Dissertation Objective and Outline**

The objective of this dissertation is to create a platform that can control a prosthetic hand with different biological signals. These signals will be processed by different controllers that will be able to identify the motions that will be performed by the prosthetic hand. This platform will also have a visualization aspect that will allow for algorithm development without the need of the actual prosthetic hand.

The organization of this thesis is as follows: Chapter Chapter 2: , Literature Review, gives an overview of what sEMG and EEG signals are and how they are acquired. It also covers some of the anatomy of the hand and brain to help better understand why electrodes were placed in the locations they were.

Chapter Chapter 3: , Biological Acquisition, discusses how sEMG and EEG signals are acquired, on which parts of the body, and the equipment used to do so.

Chapter Chapter 4: , Embedded Processor, discusses the embedded processor that will be used in the experiments conducted in this dissertation.

Chapter Chapter 5: , Expansion on Previous Work, goes in-depth over improvements to the algorithm that was originally created in my thesis and how it has been improved.

Chapter Chapter 6: , Two Motor Robotic Thumb, contains the experiment which was the first stage of the prosthetic hand platform. This part of the dissertation goes into how a 2 DoF thumb is controlled with EEG signals.

Chapter Chapter 7: , Six DoF Prosthetic Hand Experiment and Setup, is the final experiment that takes work done in the previous chapter as well as work from my thesis and fuses them into one system that controls a prosthetic hand. It also shows the virtual hand that is able to mimic the movement of the actual hand whether or not it is attached to the system at the time.

Chapter Chapter 8: , Results and Analyses, covers the finding from the different experiments performed in this work. It also looks at how well the end prosthetic platform fulfills the goal of this dissertation.

Chapter Chapter 9: , Conclusion, ends the dissertation by discussing the need for a better prosthetic hand platform and summarizing the steps that were used to create the platform.

Chapter Chapter 10: , Future Work, discusses some other questions that could be addressed and improvements that could be made on the system.

## **Chapter 2: Literature Review**

Of people who have lost a limb, most turn to prosthetics to regain some mobility and ability to function independently. Today's robotics have developed greatly over the last decade. CBS reported in December 2012, [9], about the recent advancement in human-computer interface robotics. A quadriplegic woman, with an array of electrodes implanted in her brain, has the ability to move a robotic arm, [9].

However within the world of upper limb prosthetics, there is a great potential for improvement. Many current models are based on the same technology and principles as the first commercial prosthetic hand developed at the Central Prosthetic Research Institute of the U.S.S.R. in 1964, [8]. The limitations of the older, purely mechanical designs prevent patients from interacting with them as reported in [9].

### **2.1 Current Prosthetic Hands**

With advancements in modern artificial hands more people are interested in new prosthetic hands. This comes at a cost though and usually requires surgery. With the risks of extensive surgery as was done in [9], many patients prefer other options such as skin surface sensor prosthetics. This includes non-implanted electroencephalography (EEG) sensors and electromyogram (EMG) prosthetics which are simpler for patients to

control. Due to the simplicity, low maintenance and robustness of older models, most patients still currently prefer the older mechanical prosthetic arms.

A part of the newer robotic hands that separate them from the older mechanical hands is the opposable thumb. The thumb allows for more mobility but can be difficult to control. EMG controls are simplistic and cannot handle multiple degrees of freedom well. The design proposed in this paper is to interface EEG signals with an artificial thumb of a prosthetic hand. Examples of the latest robotic hands that would benefit from a more advanced EEG controller include the Shadow Dexterous Hand™, [10], which has 20 actuated degrees of freedom along with position, force, and sensitive touch sensors.

Another example is the Sandia Robotic Hand, [11]. The Sandia hand is considered one of the most cost-effective hands built today. The hand has 12 degrees of freedom and can be manipulated similar to a human hand. The latest breakthroughs in robotics create vast research potential in effectively controlling these types of electromechanical hands. Thus, this research explores interfacing between EEG signals, a computer, and an embedded processor with servo motors for thumb control.

## **2.2 Prosthetic Hand Controls**

One of the modern methods for prosthetics that has been explored is implanting electrodes within the skull, [12], and nerve endings. This method has many disadvantages such as electrode rejection from the body, which can be life threatening.

This approach is still in the experimental phase. Therefore, traditional methods are still being explored due to risk factors of some modern methods. These traditional methods include EEG and surface EMG signal processing. Traditional methods still face the trouble of signal identification of the different hand motions to the original signal.

## 2.3 Anatomy of the Forearm and Thumb

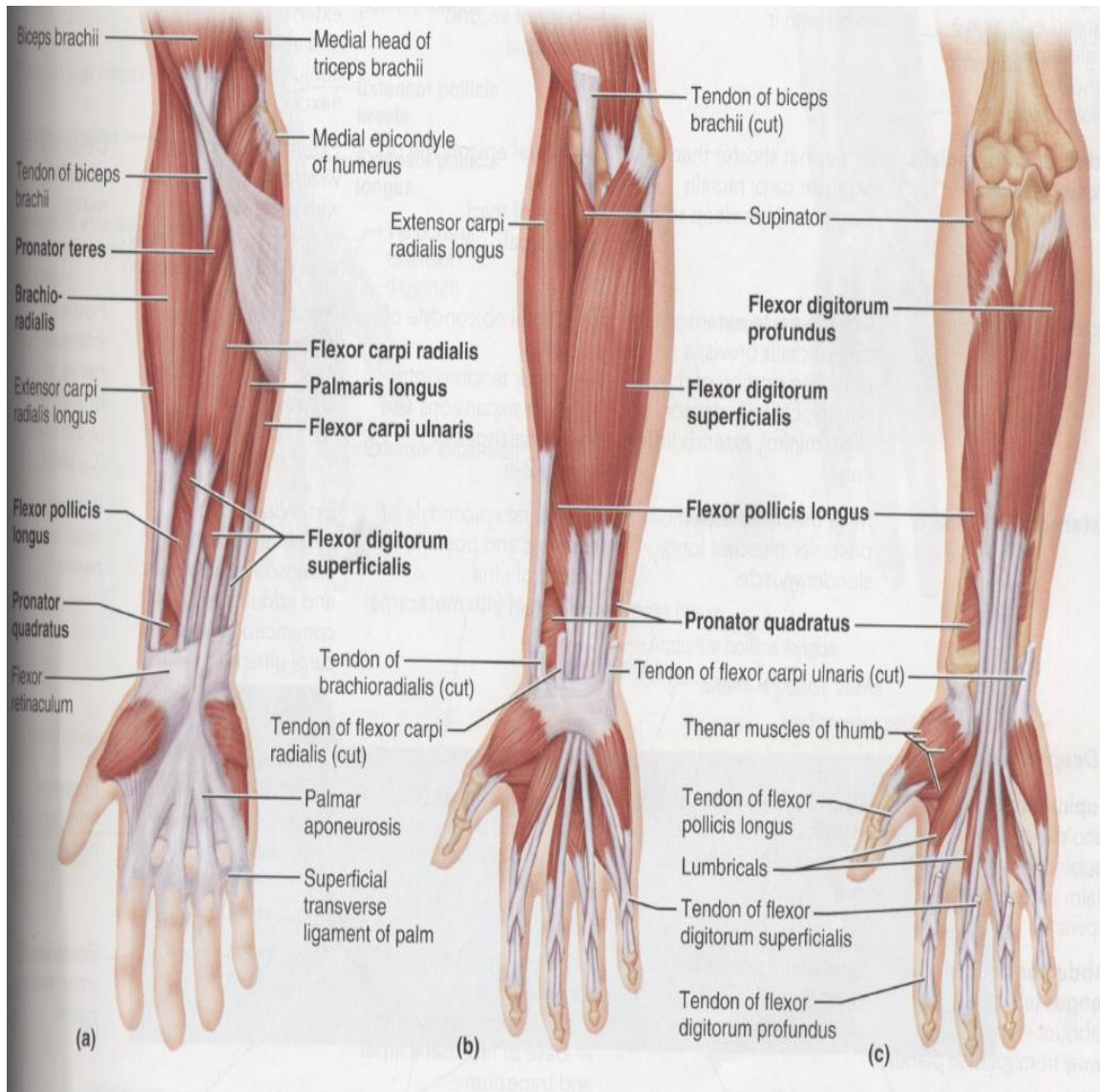


Figure 2.1: Major Flexor Muscle Groups in the Forearm, [13]

It is important to know that actions that require fine motor have small motor units compared to gross or rough motor actions. For example, surgeons need fine motor skills to make precise incisions with their hands whereas running does not require precise

movements. As a result, the motor units for people's hands are much smaller than motor units in people's legs, [2] [13].

The reconstruction of hand kinematics during reach to grasp movements from EEG signals is done through linear modeling and genetic algorithms, [14]. The genetic algorithm selects the EEG channels to be used as inputs to the linear decoder. The results showed that the method developed is a potential design for EEG-based decoders.

Another study utilizes the Auto Regressive (AR) model to convert the time domain signals to the frequency domain, [15]. Through the computation of the Fisher distance, the characteristics are extracted and analyzed. The conclusion of source [15] showed that certain cognitive actions from EEG signals occur within the frequency bands of 14 Hz and 30 Hz. Yet the results were not conclusive about identification through AR modeling.

Other current research in EMG focuses on algorithms to extract the intent of the EMG signals. These EMG methods isolate EMG signals on the forearm at motor units, [16] [17]. Others use an array of electrodes to collect the information from many different motor units, [2] [18]. Some shortcomings of these methods are that EMG signals are taken from the motor units which are created from muscles in the forearm. These methods rely on most of the forearm being intact, which isn't the case in many amputations. To add further complications, most of the muscles that control the thumb are located in the hand and small muscles in the distal part of the forearm, away from the elbow, [13]. Since these muscles are in the distal part of the forearm they are very likely

to be damaged or lost in amputation. This makes it impossible to get these local EMG signals. As a result, alternative means are needed to control an artificial thumb.

The structure of the thumb consists of many muscles that act similarly to a guy-wire structure. The muscle creates tension from all sides of the thumb giving the thumb its unique function. There are ten muscles that give the thumb its motion. Three of these muscles are located in the distal forearm, four intrinsic muscles within the hand, three thenar muscles and the adductor pollicis, [19]. The thumb has four-degree of motion. Those motions include up, down, left, and right. With these four ranges of motion, the function of the thumb is constructed, [20].

Biological signal identification through methods used today still face signal identification issues. Those issues include being able to apply the identification model to a Brain Computer Interface (BCI) and have the model be adaptable to both EEG and EMG. EMG signal models have been created to control prosthetics. They are limited and cannot control the thumb effectively. EEG provides a myriad of control possibilities. Currently there is no cost-effective EEG system that can be used for controlling robotic thumbs.

### **2.3.1 EMG**

EMG signals are electrical impulses that contribute contraction in muscles. EMG signals are present because of neuromuscular activity in the body. The electrical impulses can

be measured by sensors. There are different types of sensors that can be used to measure EMG signals. There are two main groups of EMG sensors; implanted sensors that are expensive but provide cleaner, less noisy output signals. The other type of sensors are surface sensors that are inexpensive but have more noise due to the higher impedances from the skin of the patient, [2].

Surface EMG (sEMG) signals are random in nature, amplitude modulated, and time dependent, [21]. This makes it difficult to classify EMG signals. Currently there is no prosthetic hand at an affordable cost that uses EMG, [16].

In this dissertation the forearm can be broken into two zones. Figure 2.2 shows zone 1 is in the area closest to the wrist. Zone 1 has the largest concentration of tendons in the entire arm. These tendons connect the fingers and the muscles in the adjacent zone. Zone 2 in Figure 2.2 is the proximal part of the forearm; the zone closest to the elbow. Zone 2 is where the major muscle bellies are located and is where the main motor units that drive the function of the hand are located.

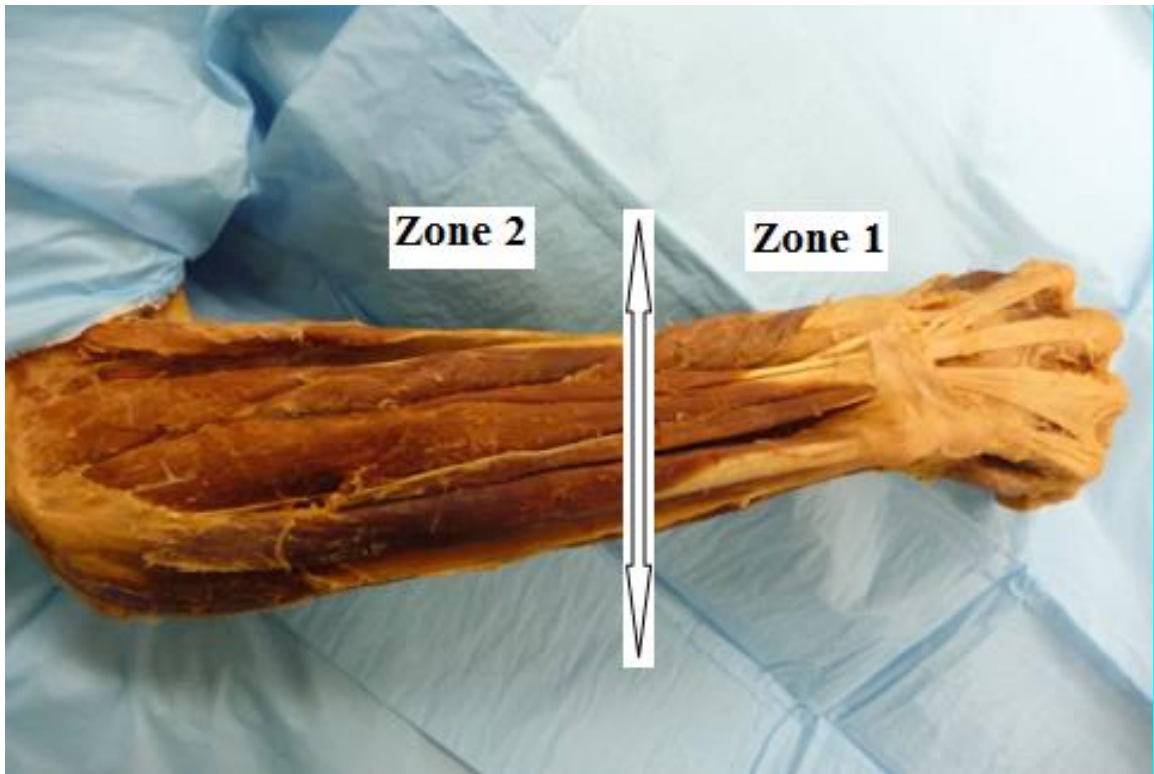


Figure 2.2: Forearm with Zone 1 and Zone 2, [2]

Motor units, like the ones in zone 2, are comprised of motor neurons and all the connected muscle fibers. Motor units use motor neurons to carry impulses from the spinal cord, or central nervous system, to the muscle fibers as shown in Figure 2.3. These action potentials, or impulses, are electrical signals that travel from the spinal cord along the motor neurons to the muscles. When the electrical signals from the motor neurons reach the muscles it innervates the muscle causing a contraction of the muscle, [22].

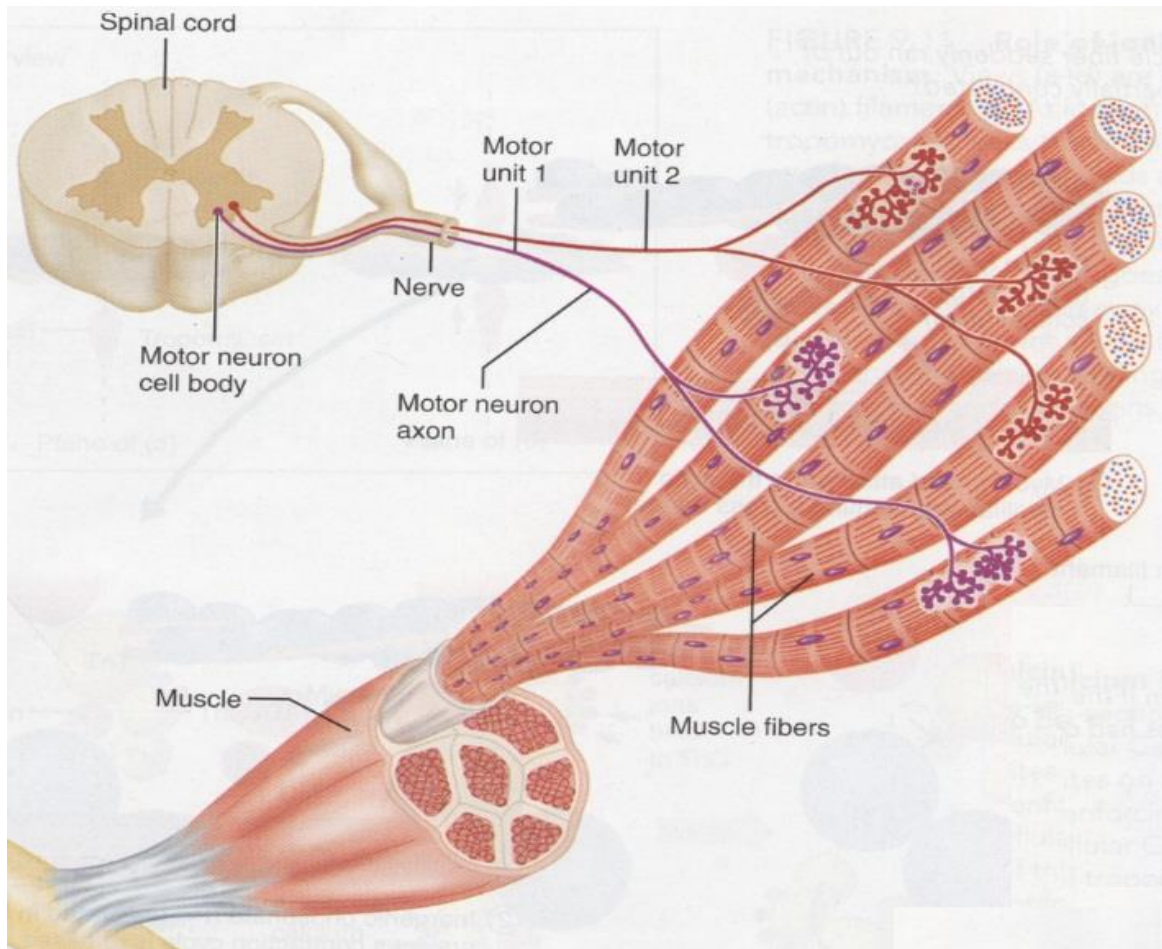


Figure 2.3: Motor Units, [22]

Figure 2.1 (a), (b) and (c) are different views that show the different layers of the flexor muscles of the forearm. The flexor muscle groups contain the majority of the motor units that initiate the muscle contraction. Each layer shown are the muscles used to control the hand. These muscles in Figure 2.1 contain motor units which produce a small voltage that can be measured as EMG signals with electrodes. Zone 1 does not provide significant EMG signals because the movement of tendons does not produce significant sEMG signals, [22].

### 2.3.2 EEG

EEG signals are electro potentials generated by the array of neurons within the brain and can be measured as well. EEG waves have distinct patterns divided into specific frequency ranges. These frequency ranges can be seen in Table 2.1 which contains the four commonly seen frequency ranges when recording EEG signals. These ranges can be recorded and manipulated through signal processing algorithms.

Table 2.1: EEG Waves and Frequencies, [23]

Signal Classification	Frequency (Hz)
<b>Delta</b>	0.5 – 4
<b>Theta</b>	4 – 8
<b>Alpha</b>	8 – 13
<b>Beta</b>	13 – 30

A main handbook on EEG is [24]. This handbook points out the four main items to consider when working with brain recording.

1. Electrodes: They are usually made of silver or gold because they are good conductors. Gold is ideal because it doesn't tarnish but it is more expensive.
2. Amplification: The amplifier needs to be able to operate in the microvolt range.

3. Filters: Must be able to filter on as time rhythm which allows it to remove artifact noise such a blinking or 60 Hz noise from surrounding electronics.
4. Recording Unit: The experiments need to be able to be recorded to allow for data processing.

As with sEMG, EEG electrode placement is important. The system that many laboratories throughout the world use is the 10-20 International System, which stands for the placement of electrodes evenly spaced from each other by 10 or 20 percent of measurements made based on the patient's skull, [18]. A pictorial representation is shown in Figure 2.4, [20].

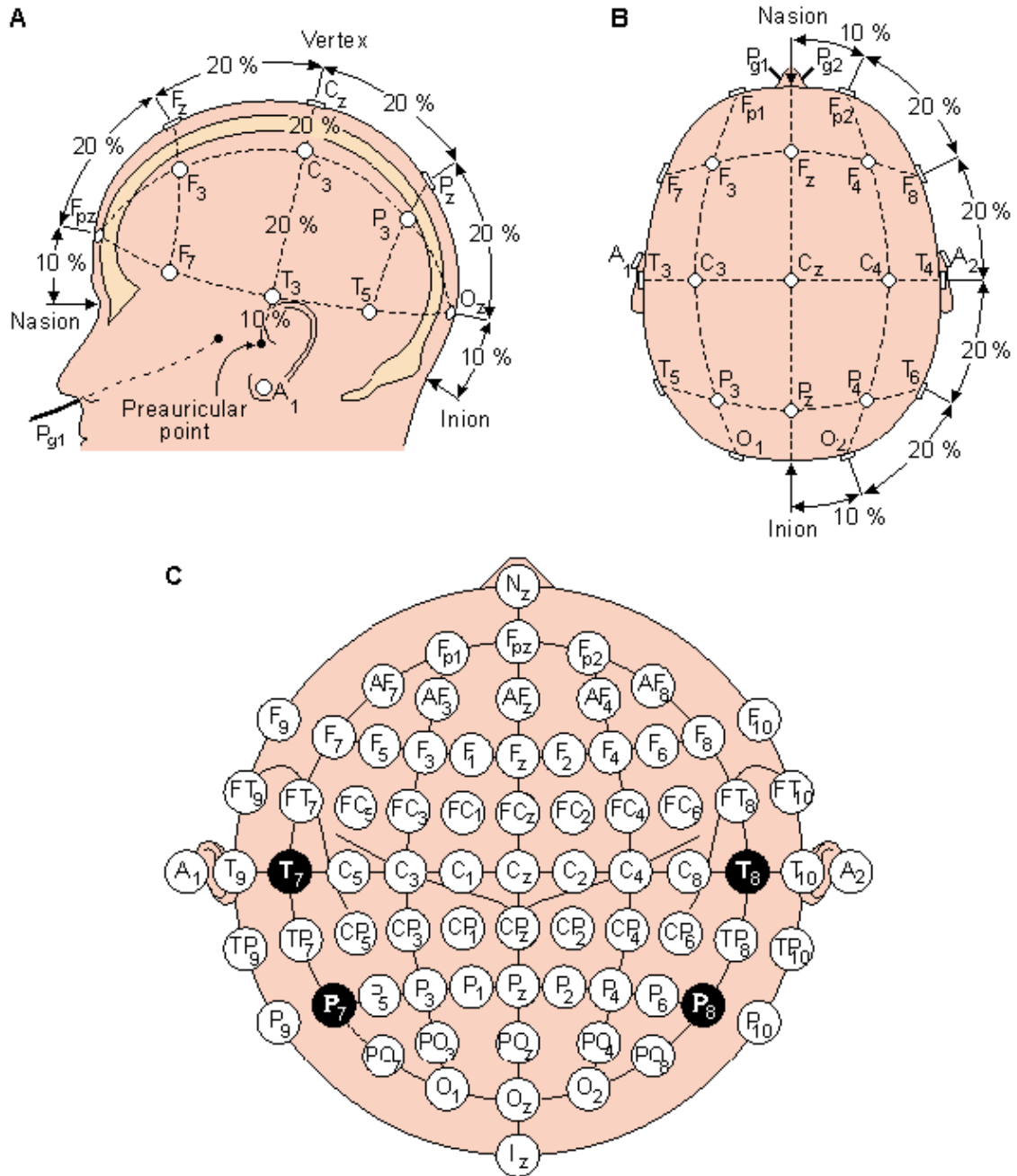


Figure 2.4: Electrode placement: A) Side profile B) Top profile C) EEG electrode placement map, [25]

Another important point is a “ground” or reference electrode. The “ground” electrode is often placed in the middle of the forehead. It is also important to prepare the patient before electrodes are placed to reduce noise of the acquired signals. This is done by

cleaning the location of the sensors with acetone. Electrode paste is often used to hold the electrodes in place and increase conductivity between the scalp and electrodes, [24] [20].

## **2.4 System Identification**

System Identification (SI) is a method that is used to characterize a given physical system, such as a car manufacturing process to biological organisms. SI is useful when little information is known about the entire system. This can range from knowing the input and output to the system to only knowing the output, which is measured or observed. It can be used to model the system and can often be accurately simulated. There are two main types of system identification models: the black-box and gray-box. The black-box model is the most common model and is used when there is no prior information known about the system. It can include some of the following models: Autoregressive (AR), Autoregressive-Moving Average (ARMA), Moving Average (MA), and polynomials, [26]. The disadvantage is that little physical information can be extracted from the black-box model. The gray-box model uses a black-box structure, but allows for limited physical information to be extracted.

### **2.4.1 System Identification Algorithm**

Assume that there is a linear time-invariant system, then:

$$\dot{\mathbf{X}}_t = \mathbf{A}x_t + \mathbf{B}u_t \quad (1)$$

$$\mathbf{Y}_t = \mathbf{C}x_t + \mathbf{D}u_t$$

where  $\dot{\mathbf{X}}_t$  is a time derivative,  $\mathbf{A}$ ,  $\mathbf{B}$ ,  $\mathbf{C}$ , and  $\mathbf{D}$  are matrices,  $x_t$  is the state vector,  $u_t$  is the input and  $\mathbf{Y}_t$  is the output. The general state solution of Equation (1) is given by

$$x(t) = e^{At}x(0) + \int_0^t e^{A(t-\tau)}\mathbf{B}u(\tau)d\tau \quad (2)$$

The discrete-time model of Equation (1) at a sampling rate of  $\Delta t$  is given by

$$x(t-1) = \tilde{\mathbf{A}}x(k) + \tilde{\mathbf{B}}u(k) \quad (3)$$

$$y(k) = \mathbf{C}x(k) + \mathbf{D}u(k)$$

where  $\tilde{\mathbf{A}} = e^{A\Delta t}$ ,  $\tilde{\mathbf{B}} = \int_0^{\Delta t} e^{A(\Delta t-\tau)}\mathbf{B}d\tau$ , and  $k$  is the discrete time index. The output of Equation (3) is given as

$$y(k) = \sum_{n=1}^k Y(k-n)u(n) \quad (4)$$

where  $Y(k)$  are the Markov parameters, shown below as the input and output parameters.

The Markov parameters in Equation (4) are given in terms of the state space model matrices as:

$$Y(0) = D Y(k) = C\tilde{A}^{k-1}\tilde{B}, \text{ or } Y[k] = [D \ C\tilde{B} \ C\tilde{A}\tilde{B} \ C\tilde{A}^2\tilde{B} \dots C\tilde{A}^{k-1}\tilde{B}] \quad (5)$$

with  $k > 0$ . In this work for SI a Hankel matrix will be used because it simplifies later calculation since a fraction of the matrix is need from its singular value decomposition. It will be assumed that  $Y(k) \ 0 \leq k < \infty$  is a pulse response. Using a discrete time shift of the Hankel matrix defined as

$$H(k-1) = \begin{bmatrix} Y(k+1) & Y(k+2) & \dots & Y(k+l) \\ Y(k+2) & Y(k+3) & \dots & Y(k+l+1) \\ \vdots & \vdots & \ddots & \vdots \\ Y(k+w) & Y(k+w+1) & \dots & Y(k+w+l-1) \end{bmatrix} \quad (6)$$

where  $l$  and  $w \in \mathbb{R}$ . The length of the matrix is defined as  $l$  and the width is defined as  $w$ . Using the impulse response, Markov parameters can be used to construct  $Y_0, Y_1, Y_2, \dots, Y_k$ , and this can be done without knowledge of the system matrices  $A, B, C$  and  $D$ . Let  $m$  be the order of the system being identified; choose  $w \geq m$  and  $l \geq m$  to ensure that the matrix  $H(k-1)$  is of rank  $m$ . If  $H(0) = U\Sigma V^T$  is assumed to be the singular value decomposition of the Hankel matrix, then the matrices of the minimal state-space realization are as follows:

$$A = \Sigma^{-\frac{1}{2}} U^T H(1) V \Sigma^{-\frac{1}{2}}, B = \Sigma^{\frac{1}{2}} V^T \begin{bmatrix} I_i \\ 0 \\ \vdots \\ 0 \end{bmatrix}, C = [I_q \ 0 \ \dots \ 0] U \Sigma^{\frac{1}{2}}, D = Y_0, \quad (7)$$

where  $i$  is the number of inputs and  $q$  is the number of outputs, [27,28,29,30,31].

## 2.4.2 State Space

Given a physical system with an  $n^{\text{th}}$ -order ordinary differential equations (ODE) and that has consistent coefficients in continuous-time ( $t \in \mathbb{R}$ ), it can be expressed as,

$$\frac{d^n}{dt^n} x_t + a_{n-1} \frac{d^{n-1}}{dt^{n-1}} x_t + \cdots + a_1 \frac{d}{dt} x_t + a_0 x_t = u_t \quad (8)$$

where  $u_t$  is the input or excitation function. A common method for solving ODEs is to rewrite the system into an equivalent set of  $n$  first-order differential equations with vector components that correspond to the differentials defined as,  $X_j = \frac{d^j x_t}{dt^j}$ , for  $j = 0, 1, \dots, n - 1$  then  $\mathbf{X}_t$  can be written as  $\mathbf{X}_t = \left[ x_t \frac{d x_t}{dt} \cdots \frac{d^{n-1} x_t}{dt^{n-1}} \right]^T$ .

Taking the derivative of  $x_t$  for each component, we obtain

$$\dot{X}_1 = \frac{d x_t}{dt} = x_2, \quad (9)$$

$$\dot{X}_n = \frac{d^n x_t}{dt^n} = -a_{n-1} \frac{d^{n-1} x_t}{dt^{n-1}} + \cdots - a_1 \frac{d x_t}{dt} - a_0 x_t + u_t. \quad (10)$$

The equations then can be written into a vector-matrix form to obtain the following matrices,

$$\frac{d}{dt} \begin{bmatrix} X_1 \\ \vdots \\ X_n \end{bmatrix} = \begin{bmatrix} 0 & 1 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ -a_0 & -a_1 & \cdots & -a_{n-1} \end{bmatrix} \begin{bmatrix} x_1 \\ \vdots \\ x_n \end{bmatrix} + \begin{bmatrix} 0 \\ \vdots \\ 1 \end{bmatrix} u_t \quad (11)$$

And can be expressed in the compact form

$$\dot{\mathbf{X}}_t = \mathbf{A}x_t + \mathbf{B}u_t \quad (12)$$

where  $\mathbf{X}_t, \mathbf{x}_t \in \mathbb{R}^{nx1}$ ,  $\mathbf{A} \in \mathbb{R}^{nxn}$ ,  $\mathbf{B} \in \mathbb{R}^{nx1}$  and  $\mathbf{u}_t \in \mathbb{R}^{1x1}$ .  $\mathbf{A}$  is the state matrix, and  $\mathbf{B}$  is the input matrix.

The output vector, or the measurement vector, can be defined in a similar compact form as:

$$\mathbf{Y}_t = \mathbf{C}x_t + \mathbf{D}u_t \quad (13)$$

where  $\mathbf{Y}_t \in \mathbb{R}^{mx1}$  is the output vector. The corresponding vectors,  $\mathbf{C} \in \mathbb{R}^{mxn}$  and  $\mathbf{D} \in \mathbb{R}^{mx1}$ , [32,33], are the input matrix and the direct transition matrix, respectively. Thus, Equation (12) and (13) are as shown in Equation (1), [27,28,29,30,31].

## **Chapter 3: Biological Acquisition**

### **3.1 Overview**

An objective of this work is to find an inexpensive method of controlling a prosthetic hand. One method this work will explore is an inexpensive electroencephalography (EEG) system to control a robotic thumb. The method used is to train a person to execute different movements with his mind by thinking of an action. After a number of trainings, which vary depending on the person, the patient will become proficient at the actions. At this stage software will be used to send signals to an embedded processor letting it know that the patient has performed a certain action. The embedded processor will interpret the message and then activate a motor based on what action signals were sent. The objective of this research is to control a prosthetic thumb. This control can be done with different types of signals, and this dissertation will be considering EEG or EMG signals.

### **3.2 sEMG**

sEMG signals have the largest amplitude over motor points. But it is important to note that motor units are not isolated in one specific area. The units are spread throughout the muscle so they are not localized. Motor points are where the nerve bundle enters the muscle bellies. Motor points can be activated by using external stimulation, but these usually only causes weak contractions. Past experiments [16] and [34] used a muscle

stimulator, the Rich-Mar HV 1000, to find motor points and then the sEMG signals were measured over these areas. Finding the best spot for muscle contraction is difficult to find and differs by person. Muscle stimulation can generally be used to find motor points that cause fingers contraction; unfortunately it is often painful and time consuming. Motor point location varies for many reasons, such as the subject's muscle mass, sex and other anatomical factors.

The shortcoming of this method is: How would one find the motor points on an amputated limb? The only real way to tell if you activate a motor point is by watching the muscle twitch because they no longer have a hand. Also, depending on the severity of the amputation most of zone 1 and 2 could be lost or damaged if the amputation occurred because of an accident instead of being performed by a surgeon.

To overcome the problem of finding motor points, this dissertation uses electrodes placed perpendicular to the muscles originating from the medial epicondyles of the forearm as shown in Figure 2.1 (a). In past experiments it was verified that zone 1 didn't contain very meaningful sEMG information [2]. Zone 2 was over the general muscle bellies of the forearm. Therefore, the anterior compartment of the forearm was used since it contains the flexor muscle groups, which is where many motor points are located. Three electrode pairs were placed over the muscles in zone 2. The sEMG signals were measured with these electrode pairs during the experiments. Surface electrodes are non-invasive because they do not have to be implanted in the patient to measure the sEMG signals, [22].

The sEMG data acquisition device that was used to collect data from the three electrode pairs was the NORAXON MyoSystem 2000. The data was acquired at a 1,000 Hz frequency (or 1,000 samples a second). The NORAXON measures the potential difference between the two sensors as positive for red and negative for black with reference to ground. The ground, or reference, was placed on the patients elbow. The elbow was used since there are hardly any motor units in that region because of the elbow bones, which provides an excellent reference for the circuit, [2].

### **3.3 EEG**

#### **3.3.1 Emotiv EPOC™**

The Emotiv EPOC™ is an inexpensive EEG device. The headgear with the research software can be purchased for about \$700. The headgear has 14 electrodes with ground/reference nodes. The sensors are in an array on AF3, F7, F3, FC5, T7, P7, O1, O2, P8, T8, FC6, F4, F8, AF4, and 2 ground nodes can be seen in Figure 3.1, [35]. This electrode placement is not as complicated as the 10-20 International System shown in Figure 2.4. This is one of the ways that cost is reduced from many of the larger, more expensive, machines that have to handle many more electrodes. Raw EEG signals can be acquired with the research edition of the software. The raw data can be taken and saved into a Microsoft Excel file which can be imported by other software to do data analysis, such as MATLAB™, [36] [37].

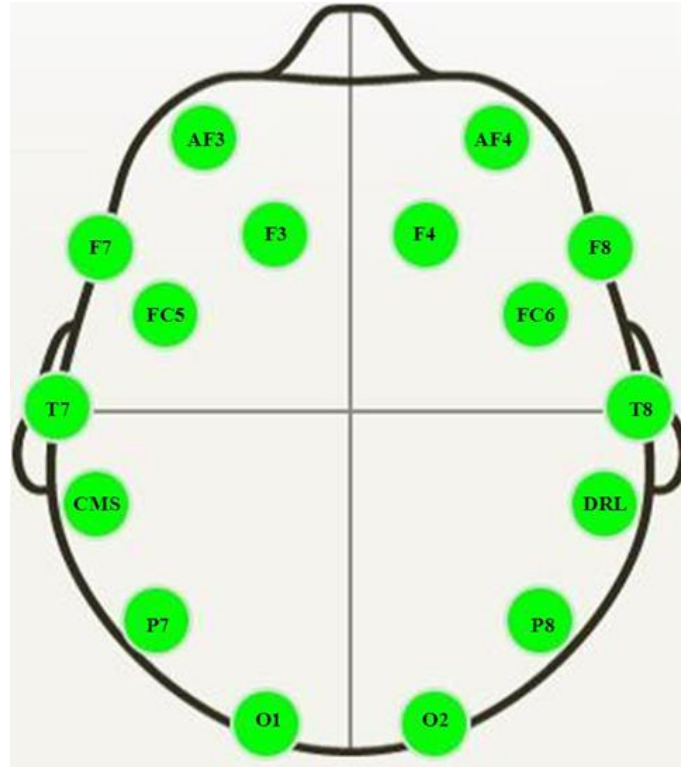


Figure 3.1: Emotiv sensor array, [37]

The 16 points in Figure 3.1 are where the 14 electrode placements are on the headset. The headset can be seen in Figure 3.2 with the electrodes. The electrodes have saline solution applied to them to improve conductivity. The device transmits wirelessly to a computer via a wireless transmitter/receiver (dongle). The EEG signals are displayed graphically on the monitor with help of the software called TestBench™. The Emotiv has a sampling rate of 128 samples per second, [35].



Figure 3.2: Emotiv cover display, [37]

The system allows for many different actions which associate to a thought to be trained. The person trains an action by associating a thought with. The more actions that are trained, the more difficult it becomes to execute the desired motion. Two actions are easy to control but the difficulty increases greatly beyond three motions. The headset will get confused at the action being sent and may do the wrong action. To help with this additional training needs to be done.

Once the patient has been trained, an experiment will be performed and the data can then be saved. For instruction on how to save experiment data, see Appendix of this dissertation: Emotiv File Saving Process After an Experiment.

# **Chapter 4: Embedded Processor**

## **4.1 Overview**

The “brains” of the prosthetic hand is the embedded system. It is used to control the motors based on the biological signals, or key commands, from the computer. The embedded system will control the motor position and the speed of the motors. The hand is driven by servo motors. Servo motors are used since they have a high level of position accuracy. This accuracy will allow the embedded platform, the Arduino Uno or Arduino Duemilanove processor, to control the hand.

## **4.2 Arduino**

Arduino is an embedded processor that is reprogrammable. The startup package is inexpensive for prototyping purposes, and the chip's programming language is straightforward. The Arduino can be programmed using a standard USB cable and a computer. When reprogramming is required, the Arduino does not have to be disconnected from the circuit. It simply needs to have the USB cable connected to a computer. For this work the USB cable will not be disconnected so programming can take place whenever it is needed. This adds flexibility. The USB cable also allows for serial communication. The serial communication allows the user to type numbers and letters and sends them to the Arduino. The Arduino has a 16 MHz clock speed.



Figure 4.1: Arduino Uno Board, [38]

The Arduino main micro-controller is an ATmega 328 that can run off a voltage ranging from 7 – 12 V. It has 14 digital input/output (I/O) pins. This version of Arduino has 6 digital pins that allow pulse width modulation (PWM). All digital pins can supply 40 mA. It has 6 analog input pins and can supply 50 mA. It has 32 KB of flash memory and 0.5 KB are used by the processor's bootloader. It has 2 KB of SRAM. It also has 1 KB of EEPROM memory. It has a 5 V supply pin that is an output from a 5 V voltage regulator on the board. It has a 3.3V pin or the 3V3 which is an output from a 3.3 V voltage regulator which can supply a maximum of 50 mA. The total current that the Arduino can output on all the pins at one time is 200 mA. The physical dimensions of the Arduino are 2.7 inches long and 2.1 inches wide. It has four screw holes for mounting purposes.

One function of the Arduino is Analog Write, which accepts a value from 0 to 255. This function is used for PWM signals. Analog Read will receive an input voltage of 0 to 5 V and converts the voltage to a decimal value of 0 to 1023. These pins are often used for sensors to give feed back to the system. Since servo motors are used, the servo motors will not require feedback, [38].

### **4.3 Major Arduino Functions Used and Their Descriptions**

*analogRead(pin)*. This function works for the analog pins on the board. The pins that this function works for are ANALOG IN pin 0 to pin 5. It reads an analog value that is applied to the pin. The pin is determined by which value is put into the function. This function takes a voltage from 0 to 5 V and converts the voltage to a decimal value of 0 to 1023.

*analogWrite(pin, value)*. This function is used to right to an analog signal to an output pin. The analog signal is specifically a PWM signal. The function is passed a pin value, which is the pin that will be used to output the signal on. The value passed to it determines the duty cycle of the PWM signal. The value is 0 to 255 which corresponds to 0 to 100% duty cycle. The pins that are capable of creating a PWM are 3, 5, 6, 9, 10, and 11, [38].

The Arduino main processor is an ATMEGA 328 AVR microprocessor. It has a USB circuit that allows it to communicate with the terminal without using a UART, which is

typically needed for a microprocessor if the user wants to be able to receive different information from the embedded platform. It also has various circuit protections, generally by means of diodes.

## Chapter 5: Expansion on Previous Work

The author's previous work is provided in [2] and discusses using EMG signals to control the prosthetic hand and involved performing experiments using EMG sensors. The data is collected and stored in large text files with many columns that are not required for the end results. The important information is taken from the text files and imported to MATLAB™. Then MATLAB™ is used to calculate the entropy of the signals whose results can be used to decide what hand motion is being performed.

While doing the work on EEG experiments, I noticed that I was able to optimize the work I did in my thesis. There are a couple of ways that data can be imported. Commonly it is saved in a Excel file but can also be saved as a text file. The needed steps to import the data into MATLAB™ to allow the data to be processed with different algorithms are as follows:

1. Format all the files so the data are uniform
2. Import the file into MATLAB™
3. Assign the file to a variable to allow it to be used in the code
4. Repeat until all the files are imported into MATLAB™

This process was done manually before I began my thesis. This was a time consuming, tedious and often allowed for error. Each file was opened and rows and columns were

deleted such as the titles and other parts of the file that MATLAB™ could not handle properly. In my thesis I took advantage of functions within MATLAB™ to import the data. It required many loops and entering in all the names of the files but it was only needed to be done once and then it could be run repeatedly. If new experiments were performed, the data could easily and efficiently be imported. The code was long though and required time to initially set up. Sample code was as follows, [2] [3].

```

1   for i = 0:163
2
3       if i == 0
4           load a082A1.ASC; % Load the raw sEMG and force data.
5
6           z2_1r = a082A1(:,7); % Column 7 is Z2-1 Rectified.
7           z2_2r = a082A1(:,8); % Column 8 is Z2-2 Rectified.
8           z2_3r = a082A1(:,9); % Column 9 is Z2-3 Rectified.
9           z2_1 = a082A1(:,11); % Column 11 is Z2-1 Unfiltered.
10          z2_2 = a082A1(:,12); % Column 12 is Z2-2 Unfiltered.
11          z2_3 = a082A1(:,13); % Column 13 is Z2-3 Unfiltered.
12
13      elseif i == 1
14
15          load a082A2.ASC; % Load the raw sEMG and force data.
16
17          z2_1r = a082A2(:,7); % Column 7 is Z2-1 Rectified.
18          z2_2r = a082A2(:,8); % Column 8 is Z2-2 Rectified.
19          z2_3r = a082A2(:,9); % Column 9 is Z2-3 Rectified.
20          z2_1 = a082A2(:,11); % Column 11 is Z2-1 Unfiltered.
21          z2_2 = a082A2(:,12); % Column 12 is Z2-2 Unfiltered.
22          z2_3 = a082A2(:,13); % Column 13 is Z2-3 Unfiltered.
23
24          .
25          .
26          .
27
1826      elseif i == 163
1827
1828          load a082D41.ASC; % Load the raw sEMG and force data.
1829
1830          z2_1r = a082D41(:,7); % Column 7 is Z2-1 Rectified.

```

```

1831         z2_2r = a082D41(:,8); % Column 8 is Z2-2 Rectified.
1832         z2_3r = a082D41(:,9); % Column 9 is Z2-3 Rectified.
1833         z2_1 = a082D41(:,11); % Column 11 is Z2-1 Unfiltered.
1834         z2_2 = a082D41(:,12); % Column 12 is Z2-2 Unfiltered.
1835         z2_3 = a082D41(:,13); % Column 13 is Z2-3 Unfiltered.
1836
1837     end
1838
1839 end

```

The code shown above had some limitations. This was time consuming to program the 1800 plus lines and changes the file name for each *elseif* statement but it was much faster than doing it manually. Another shortcoming was the results had to be stored in a matrix and then the next data was loaded in. If previous data need to be looked at again, then that data had to reloaded. It would be better if each set of data had its own unique name. Another shortcoming was the data was in text files which made it necessary to create a variable for each column.

In the work done in this dissertation further optimization was done to allow for more compact code to be created to accomplish the same task. Pre-formatting the data was eliminated which reduced the overall project time dramatically. This reduced time because with the Excel import function specific rows and columns could be selected. For example the title rows and columns could be omitted. This was not commonly done prior to this work.

The function *xlsread* was used to read in the Excel file of the given rows and columns and store it to a variable. The file naming convention that this algorithm is based on is UserNameorPatient Direction NumberofExperiment for example: UserA left1. Further

reduction was done with the *sprintf* function. The *sprintf()* function allows for a statement to be printed with parts that will change, for example, depending on an index from the different for loops. *%s* are used for strings, *%d* are used for decimal numbers. The *sprintf* function needs actual letters so a char command changes the 'word' to word. The first *%s* is the patient's name with the index p, being the second *%s*. Conditions were made based on the progression in the *for* loop to allow for unique variables to be made for each file. The first iteration of the program was created as:

```

1   for a = 1:3
2       for i = 1:3
3           if a == 1 && i == 1
4               in1 = xlsread(sprintf('UserA'%s %d.csv',char(direction(a)),i),'C2:P999999');
5           elseif a == 1 && i == 2
6               in2 = xlsread(sprintf('UserA'%s %d.csv',char(direction(a)),i),'C2:P999999');
7           elseif a == 1 && i == 3
8               in3 = xlsread(sprintf('UserA'%s %d.csv',char(direction(a)),i),'C2:P999999');
9           elseif a == 2 && i == 1
10              in4 = xlsread(sprintf('UserA'%s %d.csv',char(direction(a)),i),'C2:P999999');
11             elseif a == 2 && i == 2
12                 in5 = xlsread(sprintf('UserA'%s %d.csv',char(direction(a)),i),'C2:P999999');
13             elseif a == 2 && i == 3
14                 in6 = xlsread(sprintf('UserA'%s %d.csv',char(direction(a)),i),'C2:P999999');
15             elseif a == 3 && i == 1
16                 in7 = xlsread(sprintf('UserA'%s %d.csv',char(direction(a)),i),'C2:P999999');
17             elseif a == 3 && i == 2
18                 in8 = xlsread(sprintf('UserA'%s %d.csv',char(direction(a)),i),'C2:P999999');
19             elseif a == 3 && i == 3
20                 in9 = xlsread(sprintf('UserA'%s %d.csv',char(direction(a)),i),'C2:P999999');
21             end
22         end
23     end
24
25     for c = 1:3
26         for i = 1:3
27             if c == 1 && i == 1
28                 in10 = xlsread(sprintf('UserB'%s %d.csv',char(direction(c)),i),'C2:P999999');
29             elseif c == 1 && i == 2
30                 in11 = xlsread(sprintf('UserB'%s %d.csv',char(direction(c)),i),'C2:P999999');
31             elseif c == 1 && i == 3
32                 in12 = xlsread(sprintf('UserB'%s %d.csv',char(direction(c)),i),'C2:P999999');
33             elseif c == 2 && i == 1
34                 in13 = xlsread(sprintf('UserB'%s %d.csv',char(direction(c)),i),'C2:P999999');

```

```

35     elseif c == 2 && i == 2
36         in14 = xlsread(sprintf('UserB'%s %d.csv',char(direction(c)),i),'C2:P999999');
37     elseif c == 2 && i == 3
38         in15 = xlsread(sprintf('UserB'%s %d.csv',char(direction(c)),i),'C2:P999999');
39     elseif c == 3 && i == 1
40         in16 = xlsread(sprintf('UserB'%s %d.csv',char(direction(c)),i),'C2:P999999');
41     elseif c == 3 && i == 2
42         in17 = xlsread(sprintf('UserB'%s %d.csv',char(direction(c)),i),'C2:P999999');
43     elseif c == 3 && i == 3
44         in18 = xlsread(sprintf('UserB'%s %d.csv',char(direction(c)),i),'C2:P999999');
45     end
46 end
47 end

```

Notice that the above code was only for two patients and the third patient would just follow suit. The improved method shown before this was better but it was still very tedious to write all the *elseif* statements and fill out the conditions of a and i or c and i. The updated code did reduce the requirement for preformatting the files which was a major part for the creation of the original code. Further work was required to make the code more condensed.

With further refinement the code length was reduced even more. A data object was used to allow for different variable names to be created within the *for* loop. `int2string` is a function that changes an integer into a string. See line 5 of the code ([‘name of the file’ `int2str(index of the loops to count up from 1 until the end)`]). A new matrix for the patient was created with the name `EEGsig#` where # is a number from 1 to the number of experiments. The `xlsread(sprintf('prints the string name of the file%s %s %d.csv',char(patient(p)),char(direction(d)),n),'C2:P999999')`. With these modifications the loop is about 50 to 60 times smaller than the original method used in [2] which was a significant improvement on the manual method used prior to this work.

```

1 patient = {'UserA';'UserB';'UserC'}; % Assigns the matrix patients with strings values
2 direction = {'left';'right';'neutral'}; % Assigns the matrix direction with strings values
3 for p = 1:3
4     for d = 1:3
5         for n = 1:3
6             data.([ 'EEGsig' int2str(9*(p-1) + 3*(d-1) + n)]) = xlsread(sprintf('%s %s
7 %d.csv',char(patient(p)),char(direction(d)),n),'C2:P99999');
8         end
9     end
10 end

```

# Chapter 6: Two Motor Robotic Thumb

## 6.1 Proposed EEG Headset Design

This work explores using a cost effective EEG system to control a robotic thumb. The process for this method begins by training a person on executing different movements with their mind by thinking of an action. After a number of trainings, which varies depending on the person, the patient becomes proficient at the actions. At this stage, software is used to send signals to an embedded processor letting it know that the patient is thinking of a certain action. The embedded processor interprets the message and then activates a motor based on what EEG signal was sent.

Figure 6.1 shows the block diagram of how the system is designed. The EEG headset connects to a computer with a keyboard. Both the EEG headset and the keyboard send information to the computer. The computer has a controller on it to accept and analyze the EEG data. The commands are interpreted as a key press from the software or the actual keyboard. This is read by another part of the controller, and a command is sent to the embedded processor. The embedded processor decodes the command and activates one of the motors clockwise or counterclockwise. The block diagram of the thumb control system is shown in Figure 6.1. This is the overview of the setup of the system and how the components will communicate with each other.

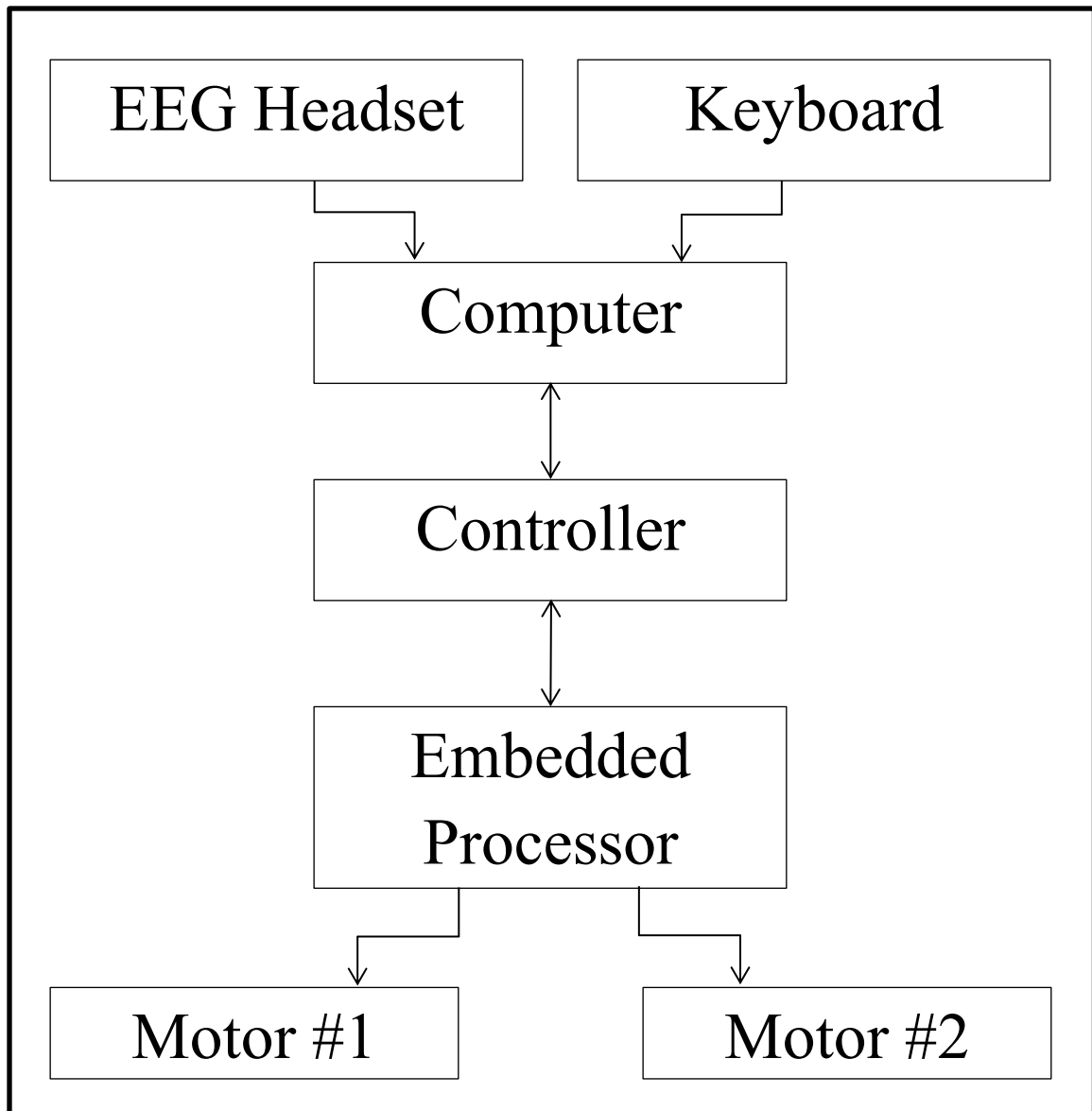


Figure 6.1: Block Diagram for two Motor Setup

The embedded processor in Figure 6.1 controls the thumb motors based on commands sent by the computer. An algorithm was designed to receive key strokes from the keyboard and translate those key signal into motor movement. The pseudo code is split into two different algorithms. The key code receives a command from the keyboard, deciphers the time the key has been held, and outputs which motor should be moved and

by how much. The motor code is set up to run two servo motors; as a result a pulse width modulated (PWM) signal will be used. The pseudo code for the embedded processor is as follows:

## 6.2 Key Command Code

```
1.draw()
2.BEGIN
3. SWITCH key
4. BEGIN
5.  CASE a
6.    SET Servo1TimeA = Servo1TimeA + 11
7.    SET Servo1TimeB = Servo1TimeB - 2
8.  CASE s
9.    SET Servo1TimeB = Servo1TimeB + 11
10.   SET Servo1TimeA = Servo1TimeA - 2
11. CASE z
12.   SET Servo2TimeA = Servo2TimeA + 11
13.   SET Servo2TimeB = Servo2TimeB - 2
14. CASE x
15.   SET Servo2TimeB = Servo2TimeB + 11
16.   SET Servo2TimeA = Servo2TimeA - 2
17. DEFAULT
18.   SET Servo1TimeA = Servo1TimeA - 3
19.   SET Servo1TimeB = Servo1TimeB - 3
20.   SET Servo2TimeA = Servo2TimeA - 3
21.   SET Servo2TimeB = Servo2TimeB - 3
22. END CASE
23. SET Servo1TimeA = adjust(Servo1TimeA)
24. SET Servo1TimeB = adjust(Servo1TimeB)
25. SET Servo2TimeA = adjust(Servo2TimeA)
26. SET Servo2TimeB = adjust(Servo2TimeB)
27. SET diffA = Servo1TimeA - Servo1TimeB
28. SET diffB = Servo2TimeA - Servo2TimeB
29. IF diffA > 0
30.   FOR i = 0 to diffA incremented by ones
31.     IF (i % 3) is 0
32.       send a 1 to motor code
```

```

33. ELSE IF diffA < 0
34.   FOR int i = 0 to diffA decremented by ones
35.     IF (i % 3) is 0
36.       send a 2 to motor code
37. IF diffB > 0
38.   FOR i = 0 to diffB incremented by ones
39.     IF (i % 3) is 0
40.       send a 3 to motor code
41. ELSE IF diffB < 0
42.   FOR i = 0 to diffB decremented by ones
43.     IF (i % 3) is 0
44.       send a 4 to motor code
45.END
46.FUNCTION
47.adjust(num)
48.BEGIN
49. IF num < minimum_pulse_time
50.   SET num = minimum_pulse_time
51. IF num > maximum_pulse_time
52.   SET num = maximum_pulse_time
53. RETURN num
54.END

```

## 6.3 Motor Code

INITIALIZE pulseA and pulseB to midpoint of the servo motors

```

1.loop()
2. IF the user has entered data
3. BEGIN
4.   SET incomingByte = number from processing code
5.   IF incomingByte is 1
6.     IF pulseA < maxpulseA
7.       SET pulseA = pulseA + 1
8.   IF incomingByte is 2
9.     IF pulseA > minpulseA
10.      SET pulseA = pulseA - 1
11. IF incomingByte is 3
12.   IF pulseB < maxpulseB
13.     SET pulseB = pulseB + 1
14. IF incomingByte is 4
15.   IF pulseB > minpulseB

```

```

16.   SET pulseB = pulseB - 1
17.  IF (current_time – lastpulseA >= refreshTime) and
    (incomingByte is 1 or 2)
18.    BEGIN
19.      turn the first servo motor on for the length of pulseA
20.      SET lastpulseA = current_time
21.    END
22.  IF (current_time - lastpulseB >= refreshTime) and
    (incomingByte is 3 or 4)
23.    BEGIN
24.      turn the second servo motor on for the length of pulseB
25.      SET lastpulseB = current_time
26.    END
27.END

```

(Note: Servo motors work by PWM signals, so the position is set by the length of the pulse.)

### 6.3.1 Arduino Code

The code written from the pseudo code shown in Section 6.3 was developed into working code. The full code is shown in Appendix of this dissertation under **Error! Reference source not found.** and **Error! Reference source not found.** . Comments are provided in the code to explain how the code works and specific functions that were used.

## **6.4 Equipment**

### **6.4.1 Hardware Setup**

The servo motors are connected to ground, + 5 V, and an output pin from the embedded processor with a resistor in series to limit maximum current flow to and from the processor. The size of the resistor varies on the maximum output of the embedded processor and for this application 200  $\Omega$  is adequate.

### **6.4.2 Embedded Processor**

The Arduino main micro-controller has digital and analog input/output pins. These pins can supply 40 mA and 50 mA respectively. The processor can be programmed using a standard USB cable and a computer. The embedded system can also communicate serially with the computer via the USB cable it is programmed with. The processor does not have to be released or burned to the chip when running experiments.

Two development environments were used. One was the Arduino programmer and the other is called Processing [39] [40]. Arduino is a programming environment that is generally used to program the Arduino. Processing is another programmer that can be used with the Arduino if it is used in conjunction with the Arduino programmer.

### 6.4.3 EEG Headset

The thumb is driven by servo motors. Servo motors were used since they have a high level of position accuracy. This accuracy allows the embedded platform used in this research to control the thumb more reliably. The patient will have the Emotiv on them and will be trained on how to move the motors which will move the thumb.

## 6.5 Equipment Setup

### 6.5.1 Embedded Processor

To prepare the embedded processor, the motor code is loaded on the chip first using the Arduino programmer. This code sets up the output pins and other configurations that are required to work the key command code. After the motor code is loaded, the key command code is loaded by using Processing. The interface window then opens as shown in Figure 6.2.

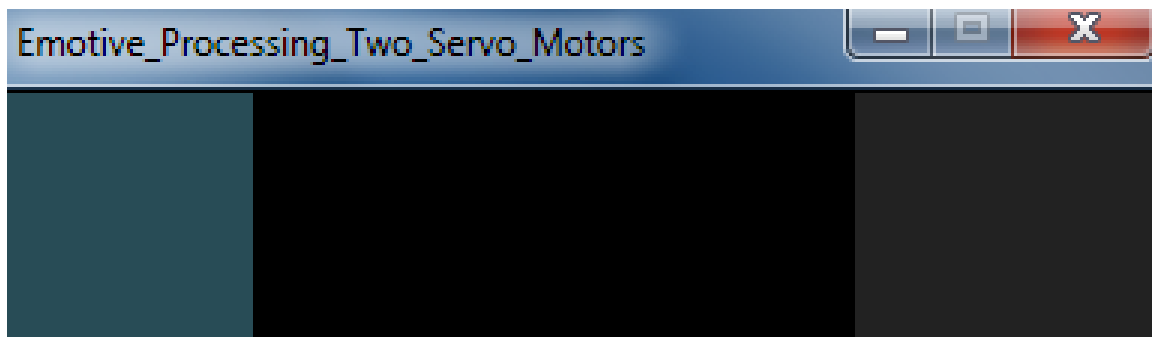


Figure 6.2: Arduino Processing Button [1].

The Arduino Processing user interface is used to train the patient before connecting the motors to the system because the button colors change when a signal is sent to a motor if it is attached. The button on the right is the default color and the button on the left is an activated button. A red button is used when the second motor is activated. The interface is also used when setting up the motors to ensure they are working properly without having to have a patient attempting to move the motors.

The system setup was based on the initial design in Figure 6.1 and was modified using the chosen hardware. A diagram of the final setup is shown in Figure 6.3. The Emotiv has one-way communication with the computer. A USB cable connects the computer to the Arduino for programming and for communication. The computer sends and receives commands from the Arduino based on the controller on the computer. The Arduino sends the desired position to the servo motors. The experiment setup consists of a computer that has the Emotiv for the EEG headset and the Arduino as the embedded processor. Two servo motors were connected to the Arduino.

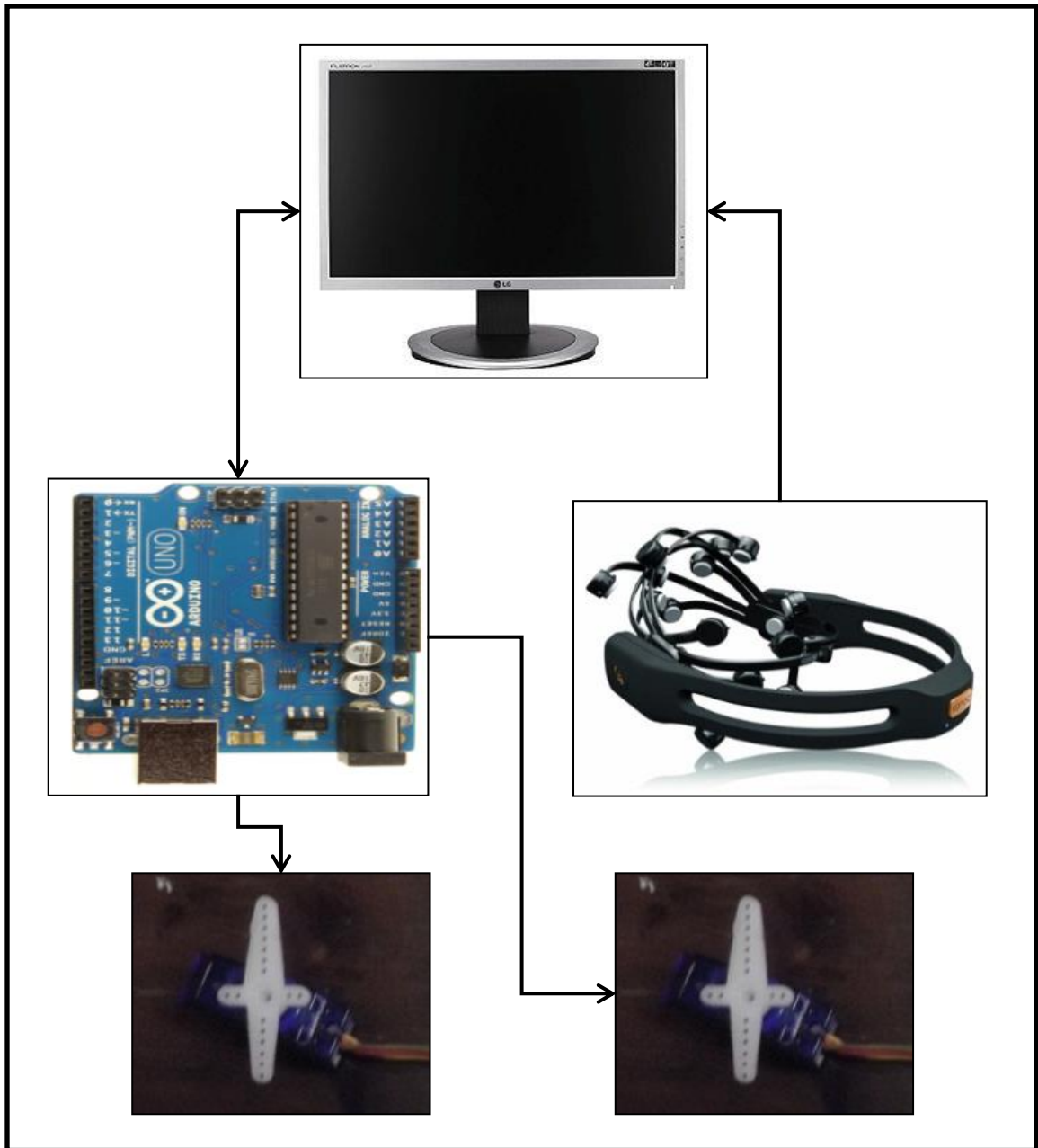


Figure 6.3: Flow Diagram of Two Motor System [1].

### **6.5.2 EEG Headset**

1. Ensure that the headset is fully charged. When the headset battery level gets low, it will randomly lose connection.
2. Wet all the electrodes with one or two drops of saline solution.
3. Once wet, attach each electrode into the Emotiv headgear.
4. Plug the wireless USB dongle into the computer to allow it to connect to the Emotiv.
5. Turn on the headgear and open the Emotiv Control panel program.
6. The system is now ready for experiments.

## **6.6 Safety**

The only safety precaution taken is between the embedded processor and the motors. The embedded processor has a resistor between the output pin and the servo motors. This prevents the processor from getting damaged if there is a fault in a servo motor that causes a short circuit. The Emotiv has its own protection to prevent harmful signals coming from the computer.

## 6.7 Software

The code written from the pseudo code shown in Section 6.3 was developed into working code. The full code is shown in [1]. Comments are provided in the code to explain how the code works and specific functions that were used.

## 6.8 Experiment

To begin the experiment, the patient puts the EEG headset on correctly and establishes connection to the computer. All subjects volunteered for these experiments. The next step is to open the Control Panel, shown in Figure 6.4. The Control Panel is used to train the patient. Once the program is open, a user is selected or a new user is input. With a new user, it is important to train the headgear to the person because each person's EEG signals are unique. To do this, the Expressiv Suite tab is selected as seen in Figure 6.4 on the right. On the left there are two tabs. The training tab should be selected which allows the user to train for different face movements as seen in Figure 6.4. Training can be done for each action along with a neutral training. The Control Panel-Expressiv Suite tab allows the user to train and adjust the sensitivity of the different facial motions. The number of trainings will depend on the patient's unique EEG signals and ability to concentrate, [1] [36].

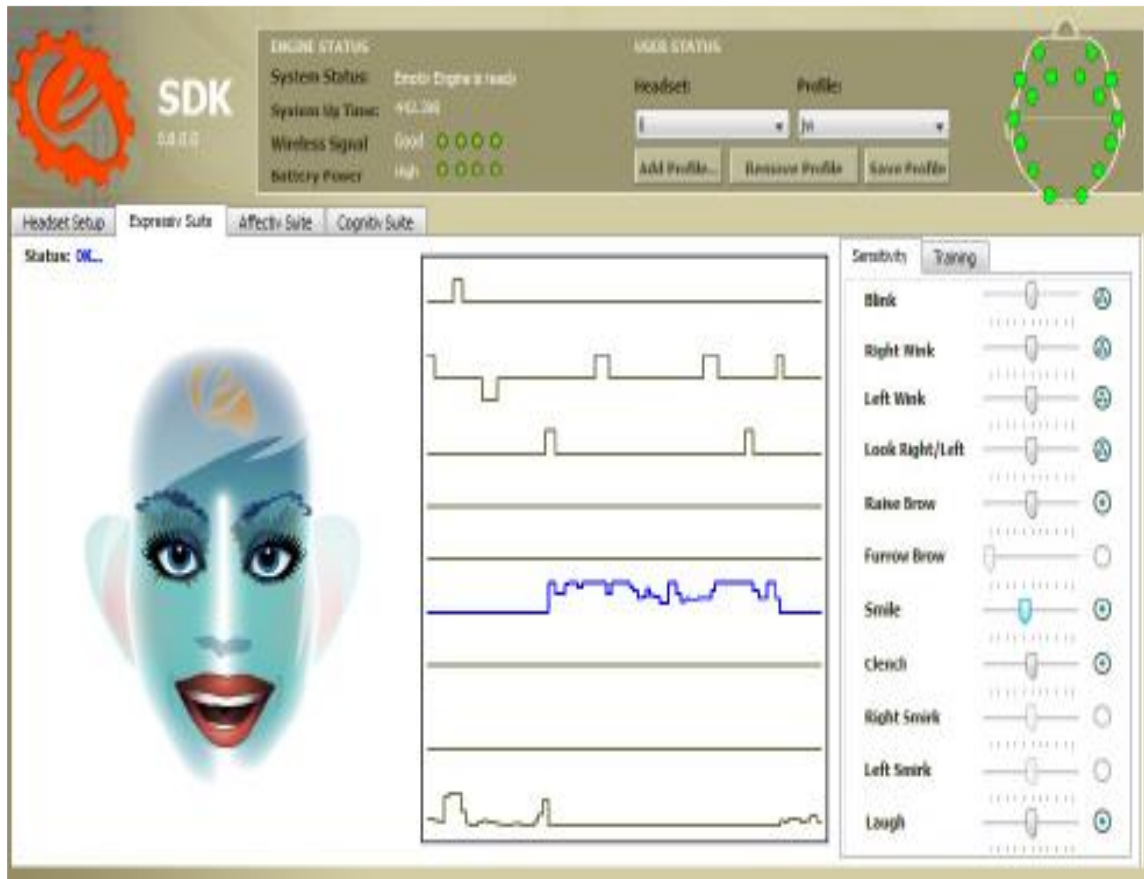






Figure 6.4: SDK Control Panel, [36].

Now that the user has been trained for facial expressions, action training is required. Specific actions can be trained to move a cube on the Control Panel-Cognitiv Suite Tab. The first training required is two neutral trainings. Neutral training is when a subject is relaxed and does not think of a given motion. The first training is a 10 second neutral training and the second training is for 30 seconds.

With the Arduino ready to acquire key signals from the keyboard, the next step is to use the EmoKey to bind actions to certain keys as shown in Figure 6.5. The different rules that can be set have certain key(s) associated with them. The Trigger Condition is where

the different actions that were trained can represent a certain key on the keyboard shown in Figure 6.5. In this figure we can see that for Rule 4 a teeth clench action was selected. The trigger values were set for greater than a 0.2 value. The trigger value can be raised to lower sensitivity to this action, or reduced to increase sensitivity for an action. This helps to fine tune the different motions depending on whether they are easy to identify or are very difficult to identify. Another option that can be changed in the software shown in Figure 6.5 is how long the software presses the key. EmoKey key binding is used to set the key that will be pressed based on a given motion from the Control Panel. Different applications can be targeted also, [1] [36].

Enabled	Player	Name	Key(s)	Behavior	Target Application	
<input checked="" type="checkbox"/>	1	Rule 1	a	<input checked="" type="checkbox"/> Send Once	<To application in focus>	
<input checked="" type="checkbox"/>	1	Rule 2	s	<input checked="" type="checkbox"/> Send Once	<To application in focus>	
<input checked="" type="checkbox"/>	1	Rule 3	z	<input checked="" type="checkbox"/> Send Once	<To application in focus>	
<input checked="" type="checkbox"/>	1	Rule 4	x	<input checked="" type="checkbox"/> Send Once	<To application in focus>	

+ Add Rule - Delete Rule

Trigger Conditions of <Rule 4>

Enabled	Action	Trigger	Value
<input checked="" type="checkbox"/>	Clench	is greater than	0.2

+ Add Condition - Delete Condition

Figure 6.5: Emotiv Key Binding Interface, [36].

Figure 6.2 depicts the user interface for the embedded processor that is run using Processing code. This interface allows signals to be sent to the embedded system using the keyboard. The gray buttons on the left and right change colors when a different key is pressed. Along with the color of the button changing, a signal is sent to the chip indicating which key was pressed and for how long. In this system, to make the motors move for a longer period, repeated signals or motions can be sent from the user, and the servo motors will move further. Sending the signals to the motor at closer intervals increases the speed of the motor.

With the patients trained, they can now experiment in moving the motors. The patients were tasked to move the motors in different orders, different rates, alternating the motors repeatedly. The subjects were able to reliably move the motors in all the different tasks. Fatigue did set in with the patients at varying intervals, typically 15 to 45 minutes, making it more difficult to control the motors; but they were still able to execute the tasks remarkably well even under these conditions.

Figure 6.6 is a picture of the setup of the embedded system with the servo motors connected to it. A USB cable is connected to the Arduino for programming and for communications. The servo motors can be connected directly to the embedded platform because they have their own internal controls to protect the circuit. A 100-200 ohm resistor could be placed in the circuit between the Arduino and the servo motors as protection if the servo motor shorted to ground. For larger motors an external 5 V power supply should be used.

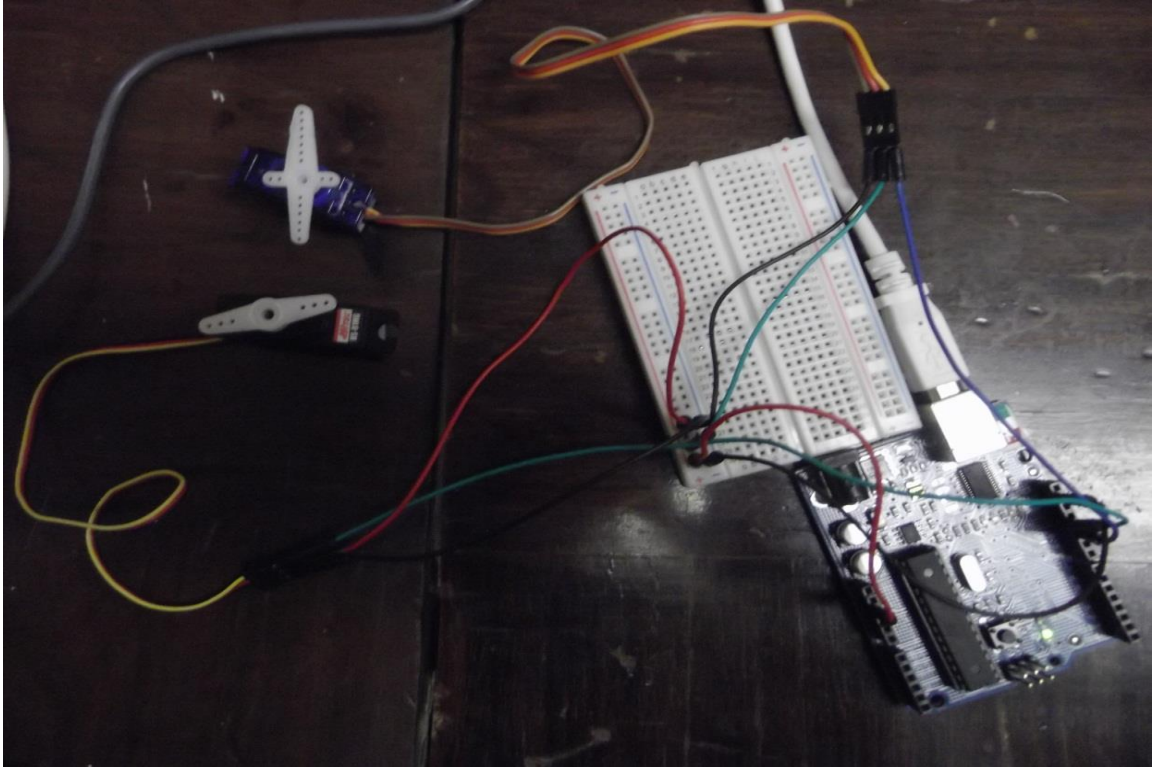


Figure 6.6: Arduino Servo Motor Setup

## **6.9 System Identification Fuzzy Controller Hybrid**

### **6.9.1 Experiment Setup**

There are various methods that have been used to study EEG data. SI has been used in the past to identify different types of systems and would be a good option to study EEG signals. Fuzzy logic has also been successfully used to identify signals. A hybrid SI fuzzy logic controller may prove effective in controlling a prosthetic hand based on EEG signals.

To create an effective SI experiment the EEG signals need to be recorded long enough, with the various motions. The motion time and duration of each movement need to be recorded for an effective SI experiment, such as shown in Figure 6.7.

This experiment will build on the current working setup shown in the previous section, and the patients have been trained on how to use the system. In this experiment, the patient will do specific motions for so long. The EEG data will be recorded, along with the amount of time the patient performed the experiment and the order the motions were executed in. The EEG signals will be analyzed with SI using this information. The output from the SI will then be input to a fuzzy controller to control motor movement.

The patients will be told to send certain movements for a given period of time. The motions and time that the patient will perform the tasks are shown in Figure 6.7. The number 2 correlates to an up motion on the movement paths figure, the value 1 is for right motions and 0 represents the neutral position. The value -1 represents left motions and -2 is for down motions. The x axis is time and each unit is 0.0087 seconds.

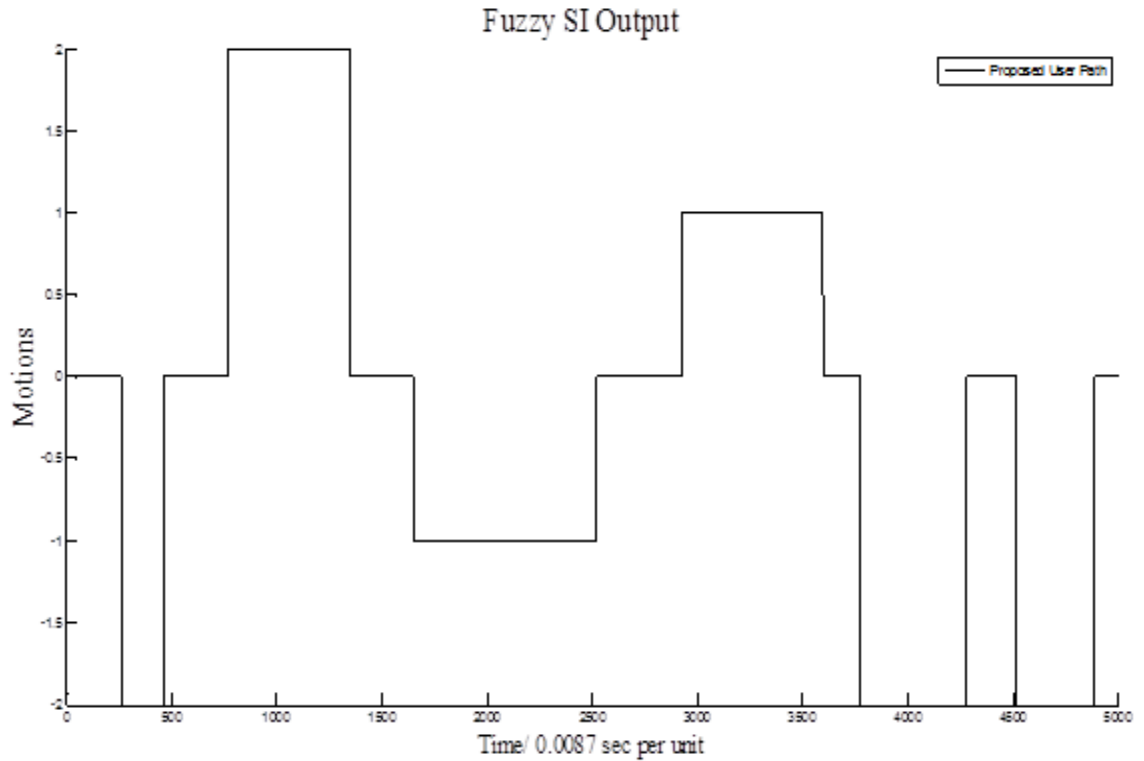


Figure 6.7: Motion Path for Patients

### 6.9.2 SI of EEG Signals

The experiments took many more runs than expected because the user would often have a motion that didn't coincide with the path that was given to them. After the experiments were conducted and the data was collected post processing could be performed. The initial process is doing SI on the EEG signals.

The data from the Emotiv are stored as EDF files which need to be converted as discussed in Section 3.3. The CVS file is a file that can be opened in Excel and then read into MATLAB™. The path information needs to be created too. This was done by recording the computer screen while conducting the experiment. A motion path was then

created for the experiment such as Figure 6.7. With the EEG and motion data imported into MATLAB™ then the SI toolbox can be used to do system identification on the EEG signals. The SI toolbox was used to do the SI on the signals. Figure 6.8 shows the SI toolbox in MATLAB™. Once the data is imported the interface can be used to create a state space model. See Appendix, System Identification Steps, for detailed steps on how to create the model.

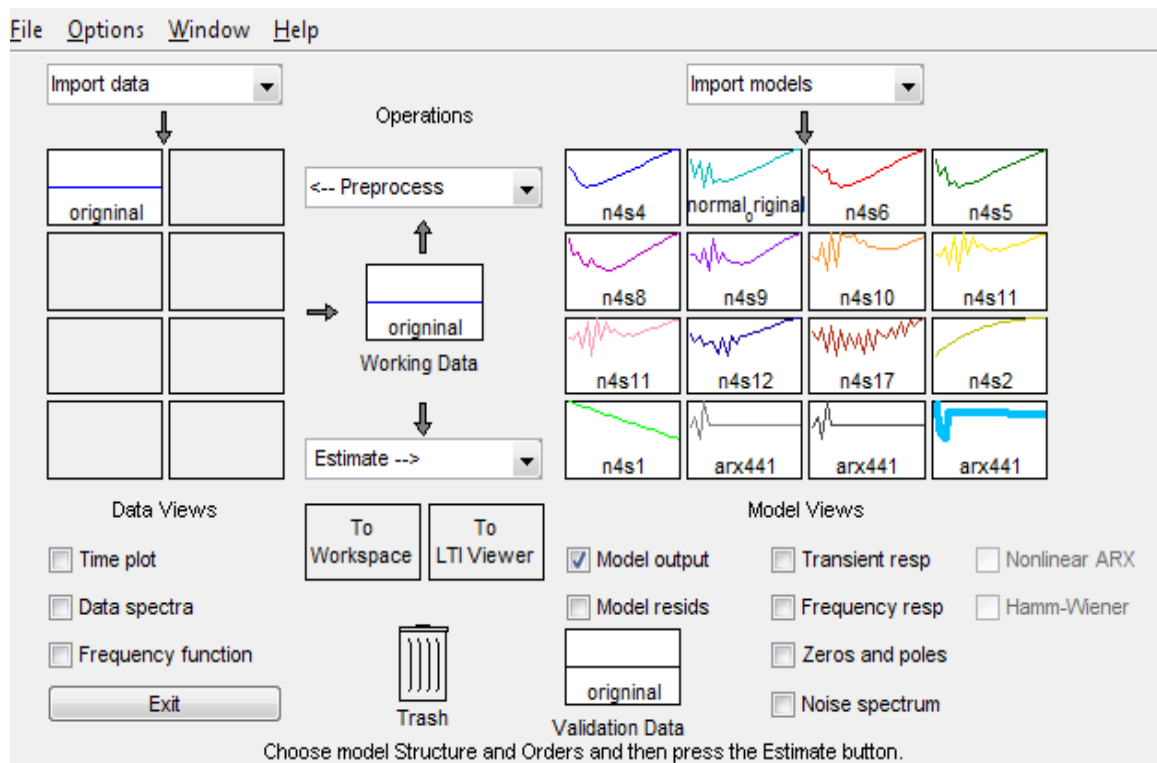


Figure 6.8: System Identification Toolbox from MATLAB™

After the model is created the data can be exported to the workspace. Drag the model to the workspace button on the interface in Figure 6.8. The data can be plotted through the tool or can be plotted directly from MATLAB™ using the plot commands. For the commands to plot from MATLAB™ see Appendix, MATLAB™ Figure Formatting and Creating Code.

Figure 6.9 shows the plot of the SI model of the EEG signals with the exported information from the SI toolbox. The SI model is able to model most of the major movements. The output of the SI model could be used to create signals to be sent to an embedded platform as voltages that would interpret those signal voltages as motions. The embedded process could then send signals to motors connected to a prosthetic hand to move the thumb.

Figure 6.9 shows the best results out of all the experiments done and it miss-identified a few motions which would cause the prosthetic hand to do a movement that the user did not intended. The SI model also overshoots the desired motion in many of the instances which isn't a problem where there is not a movement associated for 3 or -3. If there was it would cause additional incorrect movements. This would be frustrating for the user and so additional work will be done to make the controller respond more accurately in future work.

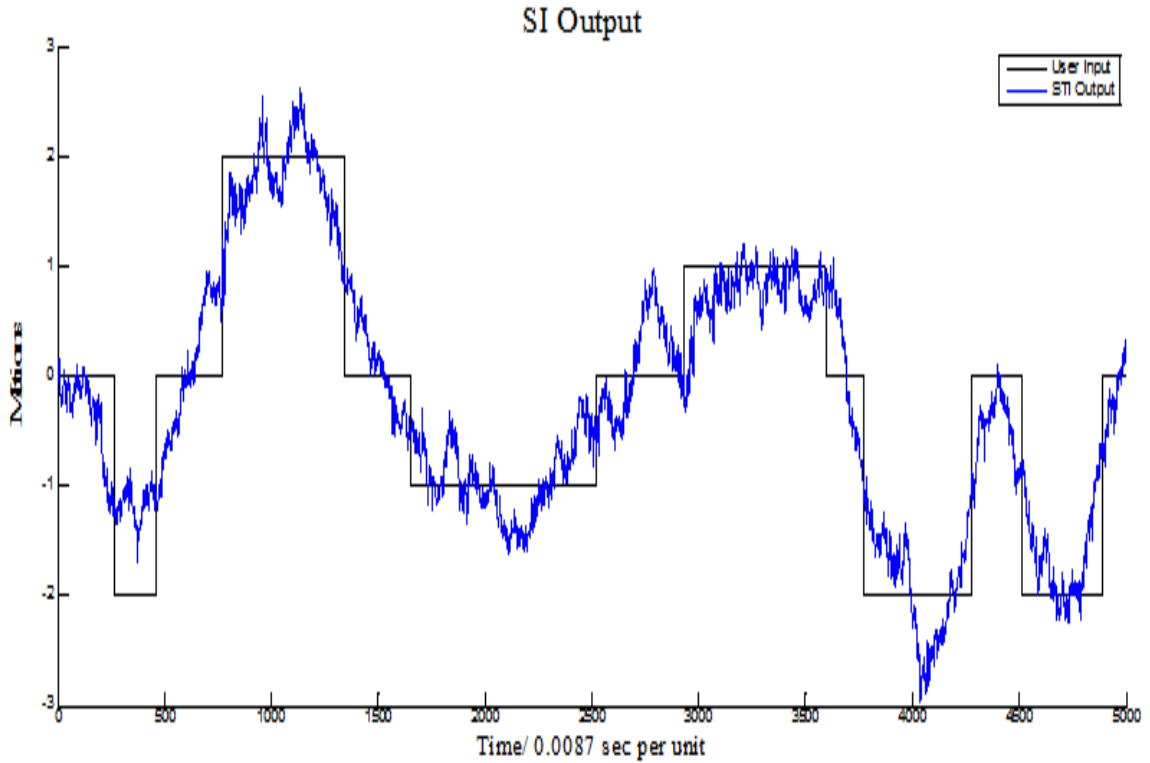


Figure 6.9: STI Model Output

### 6.9.3 Fuzzy Controller Design

The next step to build upon the SI model is using a fuzzy controller. To do this the different motions will have to be programmed into the controller. Using the MATLAB™ fuzzy toolbox a fuzzy controller is developed. Figure 6.10 shows the input membership functions. The membership function range was defined between -4 to 4 as an input. The five motions ranges that were used in the experiments were programmed into the fuzzy controller. The inputs have a range that will be used. From about -1.45 and lower is considered a down motions range, -1.45 to 0.45 is the range for the Left motion, -0.55 to 0.55 is Neutral, 0.45 to 1.55 is a Right motion and greater than 0.45 is the Up motion range. Different fuzzy types were tried but the best was *trimf*.

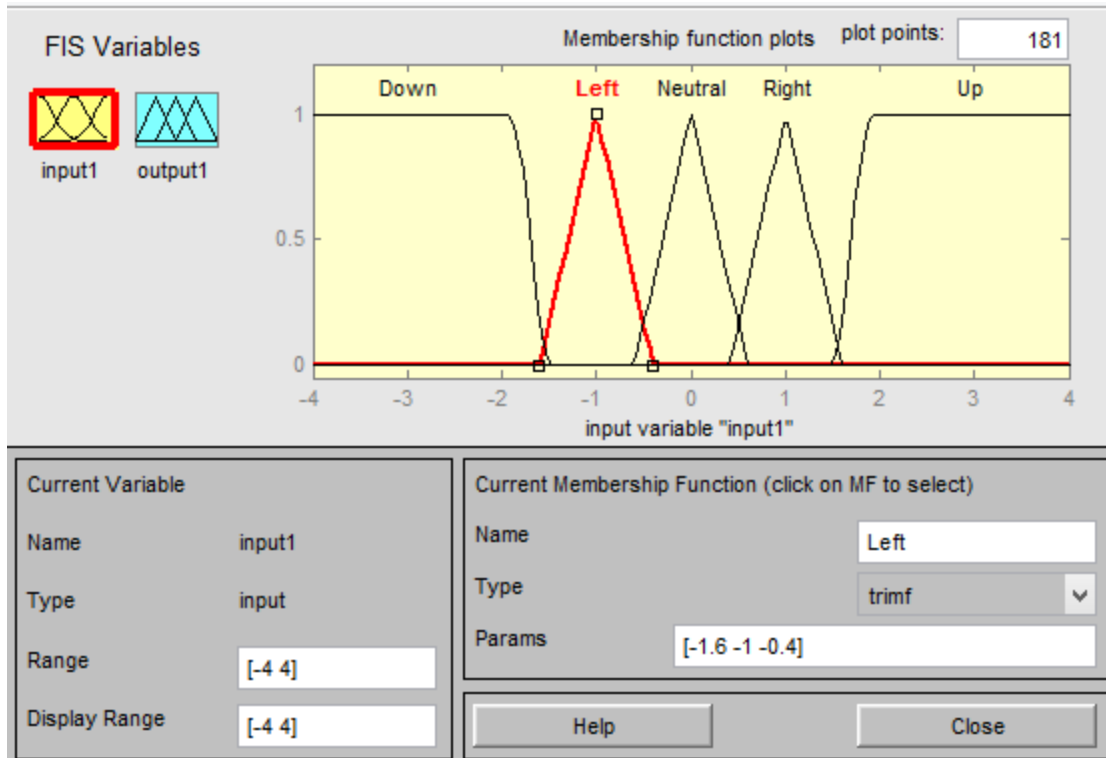


Figure 6.10: Fuzzy Controller Membership Function Input

Figure 6.11 shows the outputs that will be used. In this case the output range will be from -2 to 2. These outputs are the same values as were in Figure 6.7. The outputs can be many different things such as voltages. The output voltages could be used as an input for an embedded processor which in turn control motors based on the output.

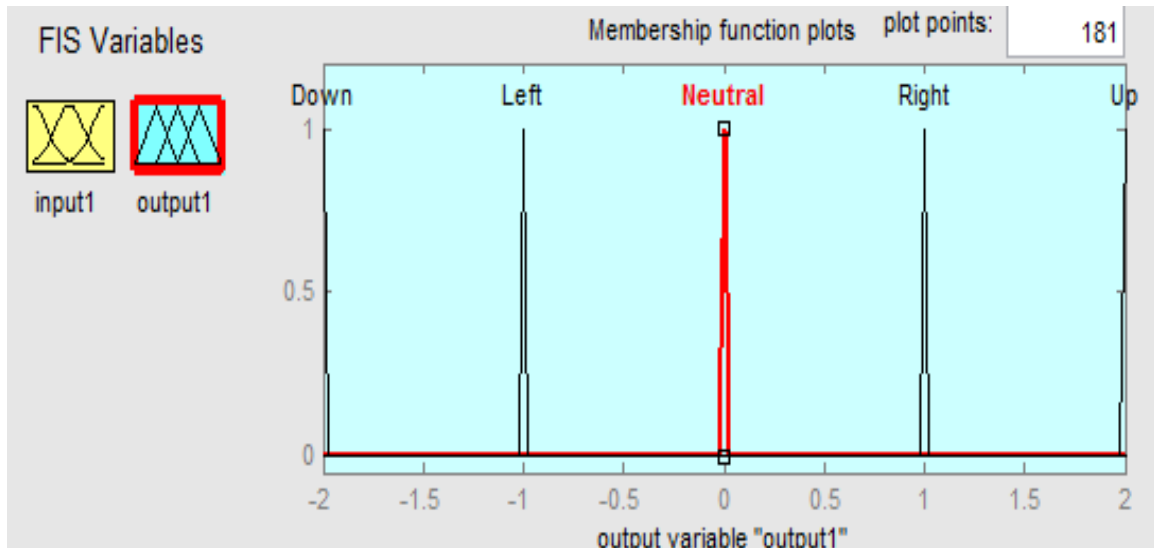


Figure 6.11: Fuzzy Controller Membership Function Output

The rules that were used for the fuzzy controller to correlate the input with the output are shown in Figure 6.12. These are set by the user and can be modified depending on the need of the controller. Another way of looking at the rule is using the surface graph that is shown in Figure 6.13. These are two different ways of visualizing the correlation between the same input and output.

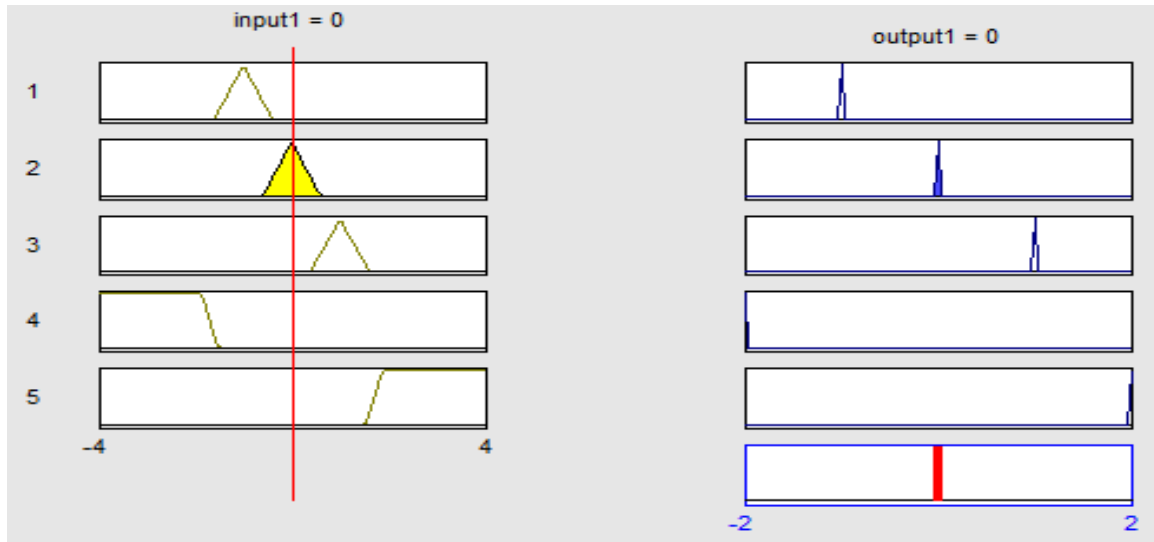


Figure 6.12: Fuzzy Controller Rules

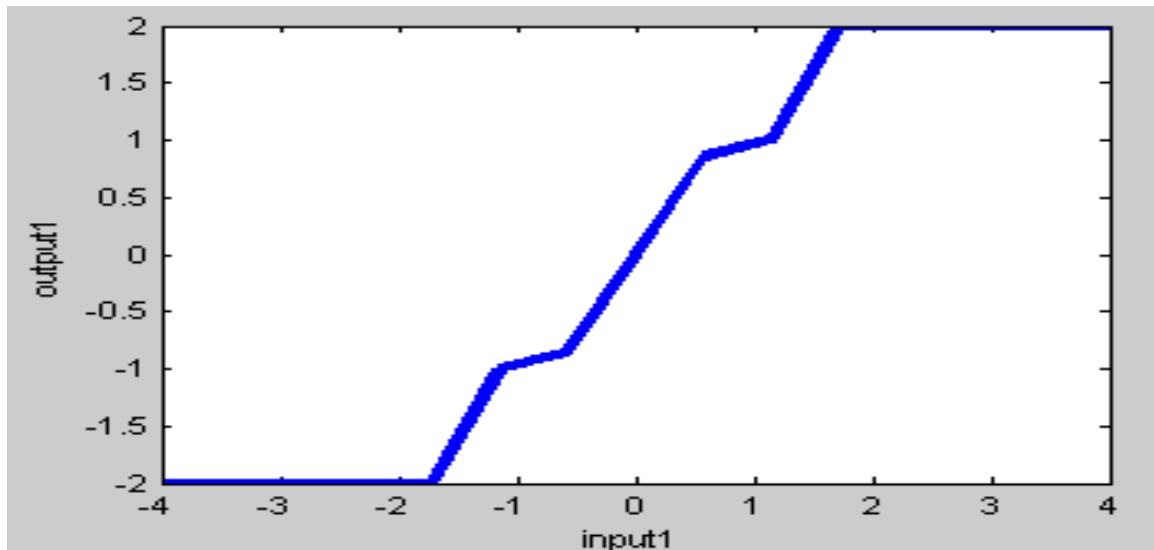


Figure 6.13: Fuzzy Controller Surface Rules Visualization

With the fuzzy controller created it needs to be exported to MATLAB™'s workspace by using file save, then export to workspace. The fuzzy controller has to be exported to the work space to be used in Simulink™. To use the fuzzy controller in Simulink™ the fuzzy logic controller block needs to be placed. Open up the fuzzy block and then enter

the name for the fuzzy controller that was exported to the MATLAB™ workspace. They have to be the same name or it will not work. The input data needs to be imported from the workspace using the block shown in Figure 6.14. The input matrix needs to be a 2D array with one of the columns being the time and the other the actual data from the SI as seen in Figure 6.9. The FuzzyOut1 block exports the results from the fuzzy controller to the workspace. Adjust the time at the top by using play to reflect the amount of time in the input value.

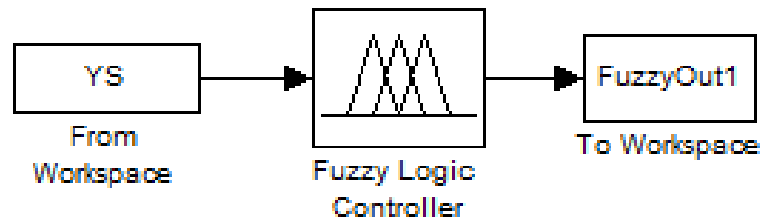


Figure 6.14: System Identification Fuzzy Controller Simulink Model

The results from FuzzyOut1 can be plotted using the method shown in Appendix **Error! eference source not found..** The results are shown in Figure 6.15. This can be compared to Figure 6.9. The fuzzy controller miss-identifies some places but for the most part is able to more successfully identify the motions better than the SI alone. Some of the spots where it drops quickly and rises again can be ignored by the microcontroller since the sampling rate is much slower. The motors will also not respond at that frequency.

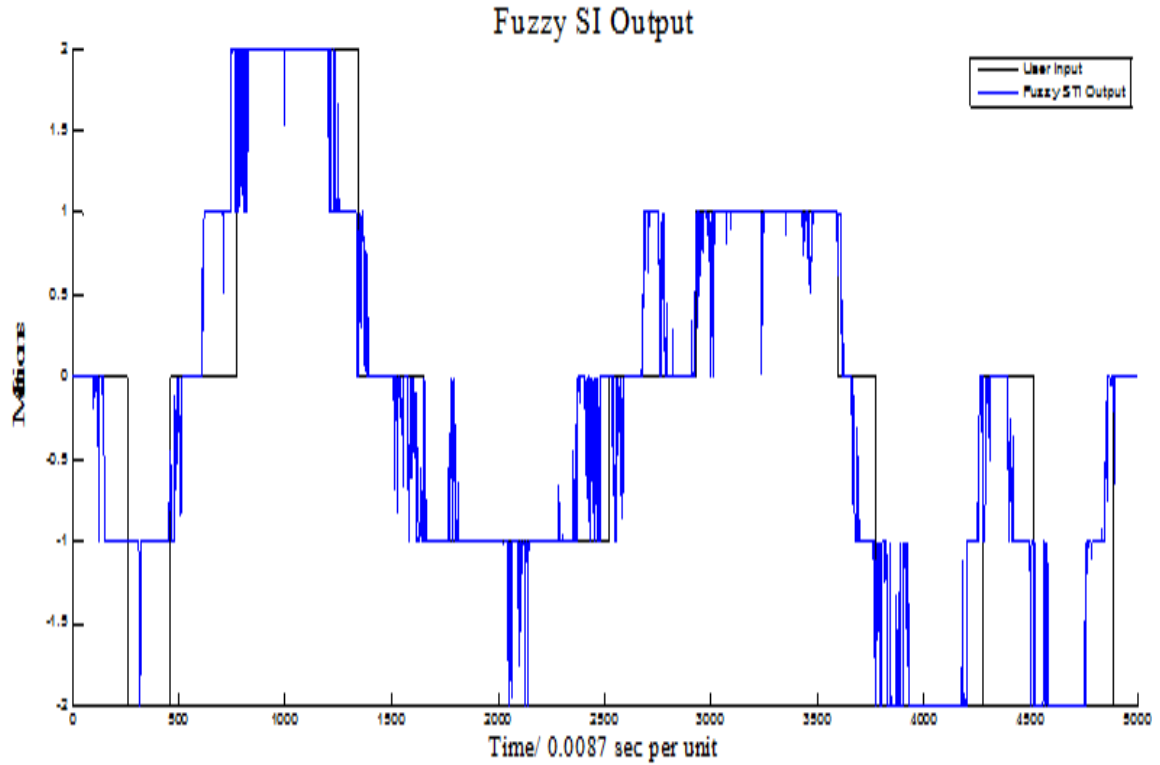


Figure 6.15: Fuzzy STI Model Output

### 6.9.4 Validations

With the fuzzy controller a validation needs to be done to see how effective the controller worked. This was done by using different EEG signals by the same patient. The signal was input to the SI and the output was plotted. The SI was not successful as can be seen from Figure 6.16. The predicted motion does not follow the actual motions' path like it did in the previous section. The magnitude of the SI is also not within the range of -2 to 2 either.

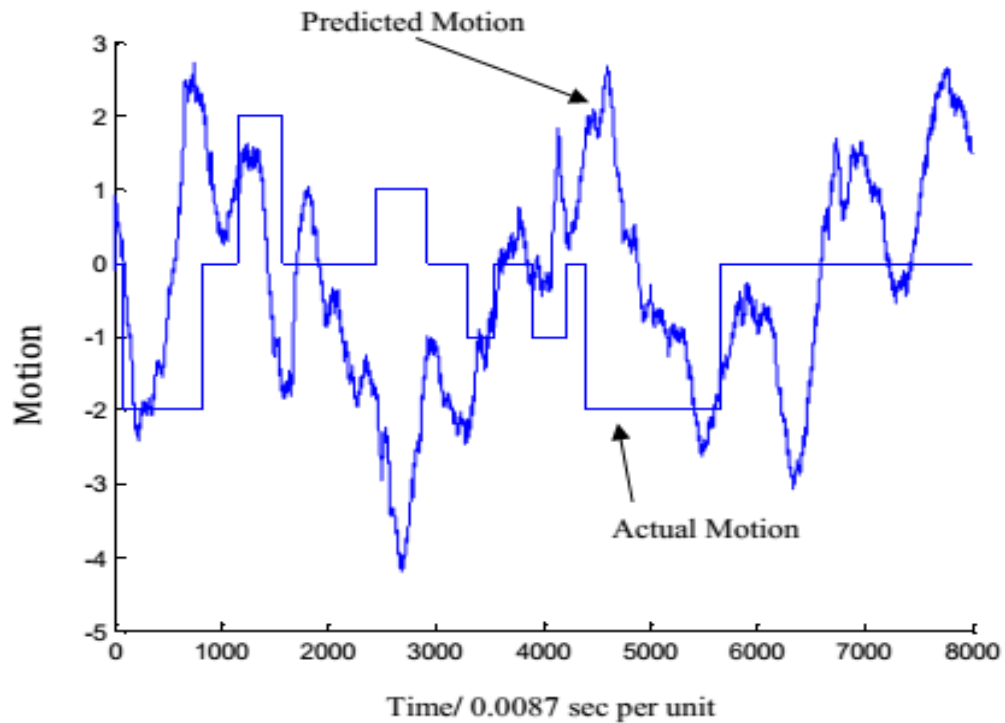


Figure 6.16: Validation Plot

It is unfortunate that the validation was not successful which means that SI was not able to successfully identify the system. But an effective fuzzy controller was designed that was able to more effectively control the output signal than the SI. This can be applied to other problems where SI is successful in identifying the system.

In an attempt to make the SI work better an additional algorithm was tried. It is a Blind source algorithm. It has been known to be used in EEG signal processing before. The blind algorithm that is used is the Algorithm of Multiple Unknown Source Estimation (AMUSE), [41]. The MATLAB™ code for the AMUSE is in Appendix AMUSE MATLAB™ Code.

This algorithm did not perform very well. A blind source toolbox, ICALAB, was used to see if its results were different. It was able to give a little better results than the MATLAB™ version which was based on the original AMUSE paper. The ICALAB toolbox's results did not differ any from the original SI. The differences in outputs from the SI and the AMUSE blind algorithm were negligible.

In conclusion of this section, the SI of the EEG signals was not effective. A success of this part of the experiment though is that a fuzzy controller was successfully created that would work off the SI, [41].

## **Chapter 7: Six DoF Prosthetic Hand Experiment and Setup**

The next phase of this research is to create a platform that can control multiple motors with multiple inputs. The inputs are a sEMG signal and EEG signals. Figure 7.1 shows the flow diagram of how the system is set up. A 3-sensor pair is used to collect the sEMG signals and the Emotiv is used to collect the EEG signals. For this chapter a novel approach is used, where a dual biological signal controller is used. The sEMG signals are used to control the robotic fingers and the EEG signals are used to control a 2 DoF thumb.

One of the motivations for dividing the control of the thumb from the fingers is that sEMG signals for the thumb muscles are very difficult to acquire and what can be measured are only a limited part of what is used to control the fingers. Most of the control for the thumb originates in the hand. To overcome this, EEG signals will be used so that the user of the robotic hand can have more control over the hand when performing a myriad of grasps.

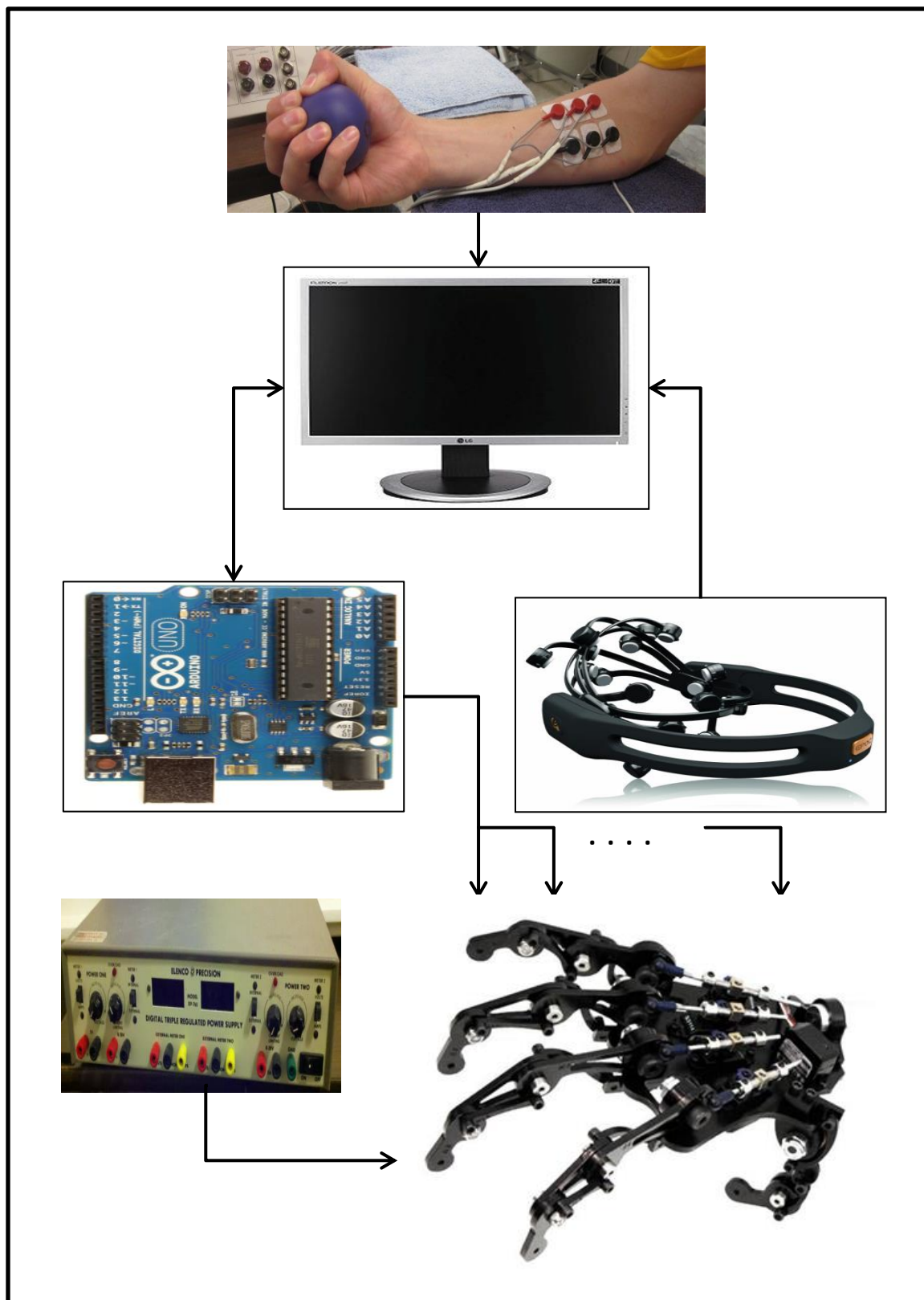


Figure 7.1: Multi-Motor Setup.

## 7.1 Equipment

The idea of this design is that the embedded system will be able to use any controller by having an input file, which is a signal from the user. This will allow the hand to be actuated by controllers designed in MATLAB™ or Simulink™ or a hybrid. This will be an advantage because it will not limit the control design to just the block set supported by Simulink™.

This design will not be real-time, but it will be one step closer to that goal. It will also increase the versatility of the system. Using the Arduino as the embedded processor will significantly expedite the coding process. The code is based on C, and the program is open source. Thus, there is a large reservoir of code examples. It is also straight forward to program compared to previous embedded processors. This will allow for a better transition between people doing research on the hand because a lengthy training process will not be required. This is done because only basic programming skills will be required.

The equipment setup will be similar to the EEG set up in Section Chapter 6: , Two Motor Robotic Thumb. The Emotiv will be used to acquire EEG signals from the patient. The Arduino will be the embedded platform for receiving the different input singles and controlling the motors. The sEMG setup will be similar to the setup in, [2] [3].

The existing embedded circuitry needs to be modified. The embedded system did not have any protection or isolation from the servo motors. If there is a fault in the servo motors, it could cause a surge that could harm the microcontroller.

The button on the desktop of the computer that received commands for the keyboard was used in this platform. See Figure 6.2 for the button; the button changes colors depending on the button that was pressed. This button sends commands to the Arduino through Processing to move the motors. This button allows users to test the motors while the code is running to ensure all the motors are working properly. The button for this work will be expanded to have multiple buttons for the different motors

### **7.1.1 New Equipment**

A couple of additional pieces of equipment are needed for this setup. An external power supply is needed because the Arduino is not able to provide enough power to drive 6 servo motors as in the previous setup. A 5V external power source was used to power all the servo motors.

Circuit protection is implemented in this system. Currently other experiments done by students just attached the servo motors directly to the embedded processor pins. This will not provide adequate protection for a final product used by a patient.

Servo motors' most common failure and worst case failure is a dead short. The servo motors are powered by 5 V. The Arduino max current is 50 mA for a single pin. The protection resistor needs to be greater than  $100\ \Omega$  as shown in Equation (14).

$$R_{\text{protection}} = \frac{5\text{ V}}{50\text{ mA}} = 100\ \Omega \quad (14)$$

The sEMG was measured by the NORAXON MyoSystem 2000. The sampling rate is 1,000 samples a second (or 1,000 Hz frequency). The device did basic high- and low-pass filtering to reduce noise from heart beats and other noise that can corrupt the sEMG signals. The system then outputs that information into an output text file. The files saved the three electrode readings with the raw sEMG data with only the noise removed and another set of data was saved with the filtering and rectification of the data.

## 7.2 Equipment Setup

The EMG signals were measured with sEMG electrodes. The sEMG electrodes were dual GS27 ECG and EMG disposable silver/silver chloride pre-gelled surface electrodes. The electrodes measure the voltage on the skin's surface created by the actuation of the motor units within the forearm, along with other muscles which is why filtering is required. The gel helps the sensors create better contact with the skin which decreases the impedances. The gel is also conductive which allows for a better connection to the skin. Before the sEMG sensors were placed, the subject's skin was prepared according to

International Society of Electrophysiology and Kinesiology (ISEK) protocols [42]. The arm was shaved in the region in Zone 2 where the sensors were to be placed. The area was also cleaned with alcohol prior to electrode placement to allow for better contact with the skin surface and also to lower the impedances. See Figure 7.2 for the electrodes and electrode connectors. The electrodes measure the voltage between a red and black electrode.



Figure 7.2: Electrodes and Connectors

The electrodes were placed in a 2-by-3 array on the surface of the forearm over the main flexor muscle bellies. See Figure 7.2 for the sensor setup. The six sensors were placed in the proximal portion of the forearm, towards the elbow. Using a measuring tape, a mark was made 11 cm from the bottom of the wrist towards the elbow. Then a mark was made half-way between the elbow and the 11 cm mark. The array of 6 sensors was centered at

this mark. Two sensors were placed perpendicular to the muscle fibers and centered at the halfway mark. Then two sensors were placed towards the wrist and two towards the elbow. They were all evenly spaced with enough space so that the sensors didn't overlap each other. For a full explanation of sensor placement rational see the author work in [2] [3]. A reference electrode was placed on the elbow where no significant sEMG signal can be measured. See Figure 7.3 for the electrode placement on the forearm. The electrode pair closest to the wrist is the 1<sup>st</sup> channel, the middle one is the 2<sup>nd</sup> channel and the 3<sup>rd</sup> is the channel closest to the elbow.

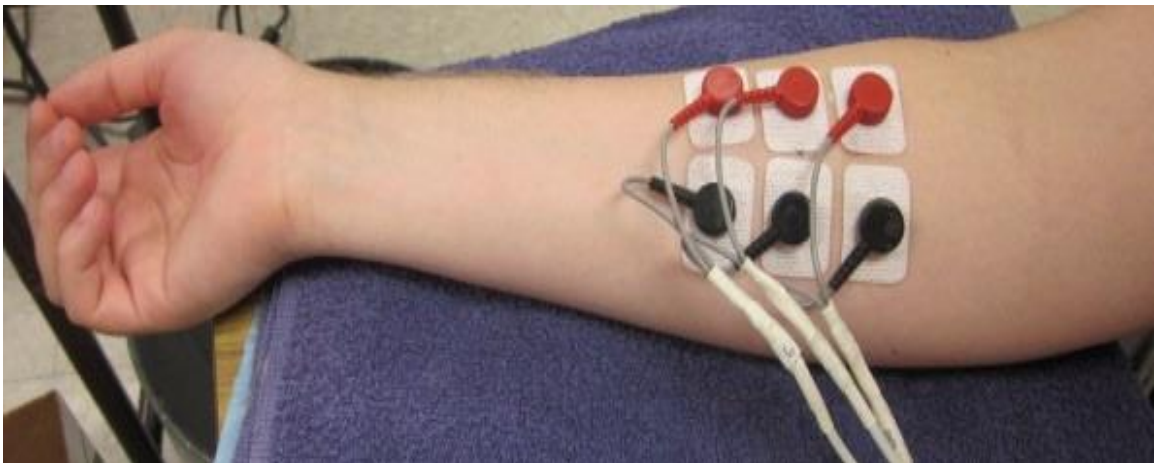


Figure 7.3: Electrode Array Placement

## **7.3 Test Setup**

### **7.3.1 sEMG Thesis Entropy Method**

The prosthetic hand is based on the work that was done in Section Chapter 6: , Two Motor Robotic Thumb, and work done in the author's thesis [2]. The system is expanded so that it is able to control six servo motors instead of two. The thumb is controlled by two servo motors. The other four servo motors are used to control the fingers of the hand.

The four servo motors that represent the fingers are controlled by EMG signals. These signals are collected by three electrode pairs. These signals are a baseline signal, power grasp of a ball, power grasp of a dynamometer, power grasp of a water bottle, power grasp of a soft cylinder, pad to pad pincer grasp of all fingers together, pad to pad pincer grasp of index, middle, ring and pinky fingers separately, key grip, and opening and closing a lid.

The reason that these motions were used was that they are common tasks that people often do in a day. The electrodes that are used for this project are disposable external electrodes. Each electrode has its own adhesive that allows for a strong bond to the skin which reduces the impedance. This results in cleaner and more reliable readings.

These signals were taken from work done in my thesis [2]. The signals were acquired in zone 2 and were then post-processed by a fuzzy controller. These signals could then be input into the Arduino for control of the fingers.

The post-processing on the sEMG signals consisted of multiple steps. The first step was to convert the signals into a format that MATLAB™ could read by removing the titles and other information that wasn't needed. Then the entropy of each hand motion was calculated for a number of patients. With this information, a range of entropy values were created. The different hand motions and entropy values can be seen in Table 7.1.

Table 7.1: Entropy for Single Motion End Results [2]

<b>Baseline</b>		<b>Ball Close &amp; Open</b>		<b>Water Bottle Close &amp; Open</b>	
	Average $z^2$		Average $z^2$		Average $z^2$
Average	0.7159	Average	0.2682	Average	0.2651
Range	0.494 to 0.9378	Range	0.1863 to 0.35	Range	0.2018 to 0.3285
<b>Towel Close &amp; Open</b>		<b>Pad to Pad All Fingers Close &amp; Open</b>		<b>Pad to Pad Index Finger Close &amp; Open</b>	
	Average $z^2$		Average $z^2$		Average $z^2$
Average	0.3309	Average	0.4008	Average	0.4406
Range	0.2916 to 0.3703	Range	0.2747 to 0.5269	Range	0.3393 to 0.5419
<b>Pad to Pad Middle Finger Close &amp; Open</b>		<b>Pad to Pad Ring Finger Close &amp; Open</b>		<b>Pad to Pad Pinky Finger Close &amp; Open</b>	
	Average $z^2$		Average $z^2$		Average $z^2$
Average	0.4645	Average	0.4246	Average	0.4508
Range	0.397 to 0.532	Range	0.3681 to 0.481	Range	0.3955 to 0.506
<b>Key Grip</b>		<b>Lid Open &amp; Close</b>			
	Average $z^2$		Average $z^2$		
Average	0.4172	Average	0.2667		
Range	0.3518 to 0.4825	Range	0.1662 to 0.3672		

In Table 7.1 the baseline has a much higher average than the rest of the experiments. This means that the sEMG signals of people's arms in a relaxed position are more random than when they are doing a task. This is true since most of the signal in a relaxed position is random noise which would have a higher entropy value and helps support that

these results have meaningful information. Using Table 7.1, ranges for the different hand motions can be created. The relaxed position has a high value of entropy; the ball grasp, water bottle, and lid experiments' average spectral entropy are very close to each other, around 0.26. These motions are similar in nature and require a large amount of force while grasping an object. The towel grasp was different from the other power grasps. There are many different factors that could cause it, but one reason may be that the towel did not provide as much resistance as the ball, water bottle, or the lid. This would cause the force required to be less, and as a result it would have a higher entropy value.

Simulations were conducted using the Simulink™ model for all the different hand motions. The overall results from the experiments are summarized in Table 7.2. The different channels are the electrode pairs of positive and negative. The lowest performance was the 1<sup>st</sup> electrode, the group closest to the elbow. It correctly characterizes between the four different motions 50% of the time. The 3<sup>rd</sup> electrode has the highest accuracy, with 75% accuracy. The signals from the 3<sup>rd</sup> electrode will be used for the experiment.

Table 7.2: Summary of Signal Classification by the Intelligent Classifier, [2]

Electrode	Correctly Characterized
1 <sup>st</sup>	50.0%
2 <sup>nd</sup>	58.3%
3 <sup>rd</sup>	75.0%
System Total Accuracy	75.0%

This promising method for determining motion from sEMG signals will be expanded later in this chapter. This will expand the work that was done in [2] to be used on the prosthetic hand.

### 7.3.2 EEG Signals

EEG setup and signals used in the 6 DoF hand experiment will be taken from the experiments in Section Chapter 6: , Two Motor Robotic Thumb. There will be some post-processing that will be done on the fuzzy controller output to make it work with the developed system which will be discussed in a later section.

## **7.4 Six DoF Hand Tools and Interface Development**

### **7.4.1 Hand Visualization Tool for a Six DoF Hand**

To help facilitate prototyping and testing for a prosthetic hand, a platform was created that visualizes the hand and receives inputs the same as the prosthetic hand would. To help in the development of algorithms for a large group of researchers, a virtual hand was developed. Figure 7.4 and Figure 7.5 are pictures of the virtual hand fully opened and closed. This hand has 6 DoF, one for each finger and two for the thumb. Movement is shown with fingers and thumb sliding and the change of color for the second degree of freedom on the thumb which allows the user to quickly see that the hand is responding to commands. The virtual hand was created with Processing. This was the same program used for creating the button in Figure 6.2.



Figure 7.4: Virtual Hand Open



Figure 7.5: Virtual Hand Closed

### 7.4.2 Control of Virtual Hand

The fingers can be moved with the keyboard or with an input from a text file. The keyboard keys are as follows: for the pinky 'a' opens and 'z' closes, for the ring finger 's' opens and 'x' closes, for the middle finger 'd' opens and 'c' closes, and for the index finger 'f' opens and 'v' closes. The thumb has two DoF so for the 1<sup>st</sup> DoF 'g' opens and 'b' closes, and for the 2<sup>nd</sup> DoF 'h' opens and 'n' closes. The colors change so that it is easy to recognize that the position has moved; especially when the hand is fully open and begins to close. When the fingers move on the screen, the embedded system will also send signals to the corresponding pins which will move the motors. Servo motors were used for these experiments.

The code has different options that can be set depending on the testing that will be performed. There is a manual mode where the user can move the fingers with the keyboard keys. Another is that it will read in a text file, and based on the numbers that are in the text file, the fingers will move accordingly. The code will also allow for predetermined hand motions to be used as well. For example, [2] had a pad to pad middle finger grasp which would move the middle finger and the thumb. Sending the command with this motion would move the middle finger and the thumb to their fully closed position while leaving the rest of the fingers open.

Depending on the desired motion, the input numbers may need to be modified. For the default, the following numbers will move the correlating finger. For the pinky 0 opens and 1 closes, for the ring finger 2 opens and 3 closes, for the middle finger 4 opens and 5 closes, and for the index finger 6 opens and 7 closes. For the thumb's 1<sup>st</sup> DoF, 8 opens and 9 closes and for the 2<sup>nd</sup> DoF, 10 opens and 11 closes.

The core code in Processing sends commands to the virtual hand on the computer screen and the Arduino. When the keys are pressed on the keyboard, the signals are used to send positions to the virtual hand and also the embedded processor through the Arduino Programmer. The embedded processor uses the same keyboard command to increase or decrease the PWM signal sent to the servo motors which in turn adjusts the position of the motor/finger.

### **7.4.3 sEMG Entropy Algorithm Implementation**

After getting the virtual hand working with the keyboard and having the servo motors move at the same time, the next stage was to make it so the virtual hand could move with the biological signals. The embedded system will be able to receive two different signals and use them for controlling the hand.

The sEMG work done previously needs to be expanded for this hand. This section will implement the work done in [2] [3] with multiple hand motions in a single signal file. Below, Figure 7.6 shows sEMG signals from a person's forearm for a number of different hand motions. The first half of the signal is of a person holding their hand still, 0 seconds to 13 seconds. The first group with the largest amplitude sEMG signals is a power grasp, 13 seconds to 16 seconds. The next group of signals is a pad to pad middle finger motion, 16 seconds to 18 seconds, and the last group is a key grip, 18 seconds to the end.

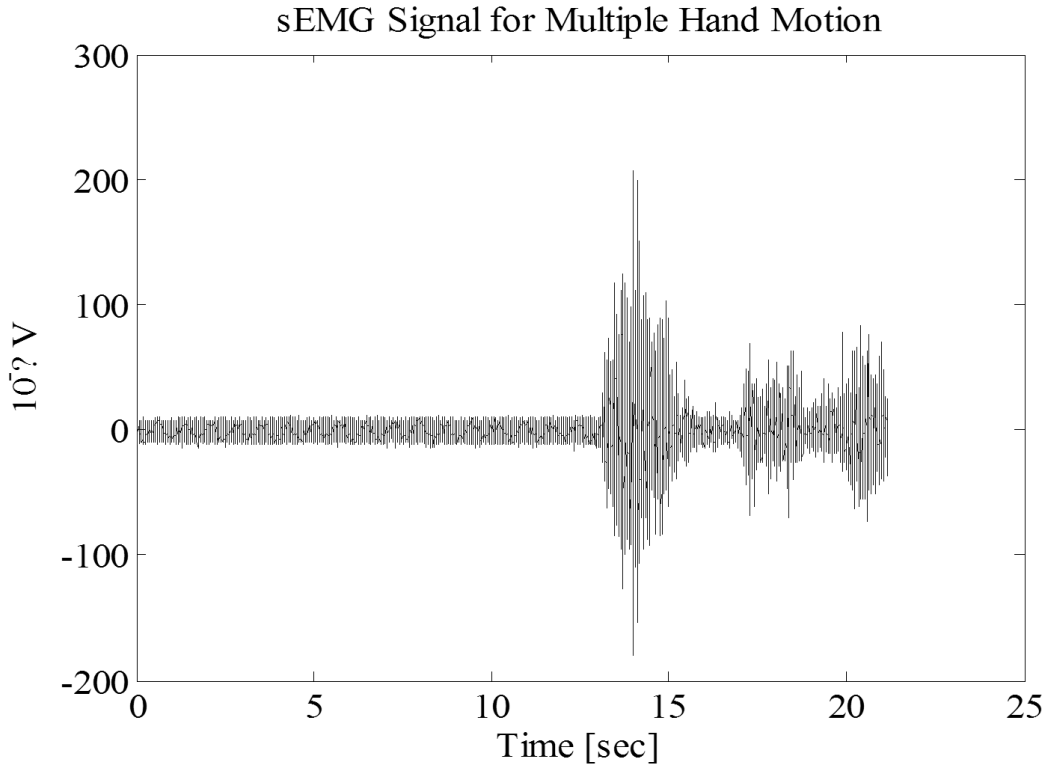


Figure 7.6: sEMG Signal for Multiple Hand Motion

The signal in Figure 7.6 is then input into the entropy calculation algorithm. The algorithm was modified from the original one in [2] [3] because it was only used to identify a single motion at a time and only one motion was given at a time to the algorithm. The algorithm then calculated the entropy of the entire signal in question. The modified algorithm accepts many motions in one signal. To do this, the algorithm had to break the signals into smaller pieces and calculate the entropy for those smaller segments. The main signal is broken into about two second (2.07 seconds) segments, and then the entropy is calculated. This time was used because there has to be enough data points to calculate the entropy to get meaningful results. Also, based on experimenting, around 2 seconds gave the best results for the sampling time of the system used.

After that, the entropy values are then input into the fuzzy controller and the controller determines the motion. The membership functions for the fuzzy controller were modified slightly from the original ones in [2] [3] and are shown below in Figure 7.7. The rules of the fuzzy controller have not been changed and are shown in detail in [2] [3]. Note that the baseline or neutral is when the people are holding their hand still and MFinger means pad-to-pad middle finger grasp.

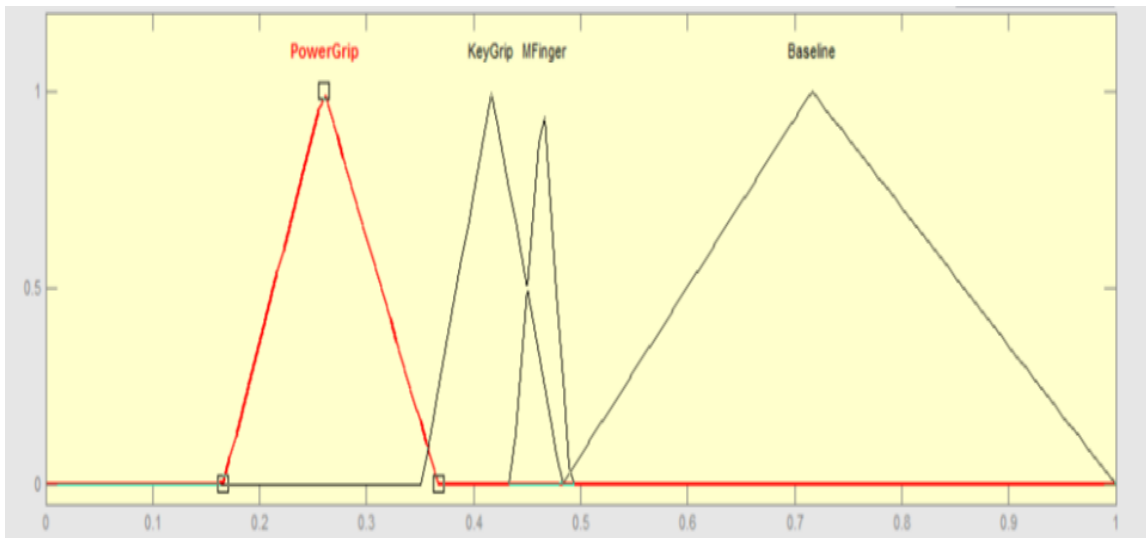


Figure 7.7: Fuzzy Controller Membership Functions Plot

The fuzzy controller then outputs a number depending on the rules and membership functions. The output values are defined as shown in Figure 7.8. This plot shows that if a power grasp is identified, a 2 will be output, a key grip is represented by 4, a pad-to-pad middle finger grasp is shown by a 6, and if the hand is doing nothing or relaxed then a 10 is output.

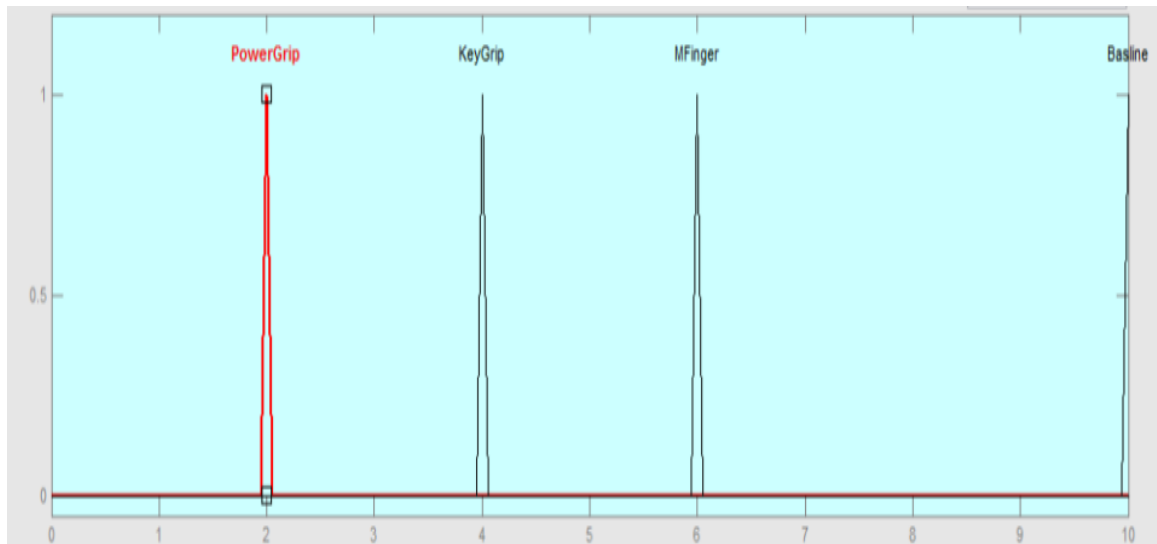


Figure 7.8: Fuzzy Controller Output Values

The flow of the algorithm is shown in Figure 7.9. The signal from Figure 7.6 is input into the fuzzy controller which is stored in MATLAB™'s workspace. The controller reads the file from the workspace controller, determines the motion, and outputs a number based on Figure 7.8. The scope displays the graph and can also be set to save the plot back to the MATLAB™ workspace.

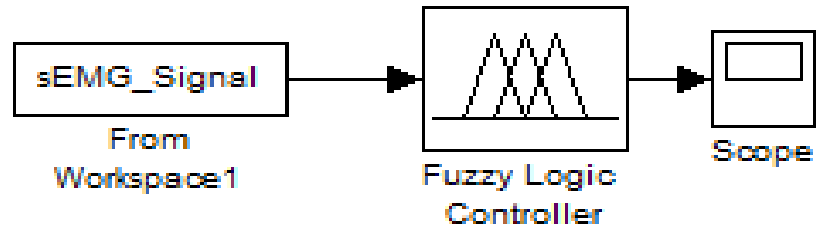


Figure 7.9: sEMG Fuzzy Controller Flow Diagram

Figure 3.1 is a plot of the actual motions that were performed in Figure 7.6 using the fuzzy controller output values. Figure 7.11 is the resulting plot from the fuzzy controller. Comparing the two plots shows that the fuzzy controller was very successful in characterizing most of the signal. The controller was able to successfully identify the correct motions 90 % of the time. These are promising results because of such a high accuracy in correctly identifying the different motions.

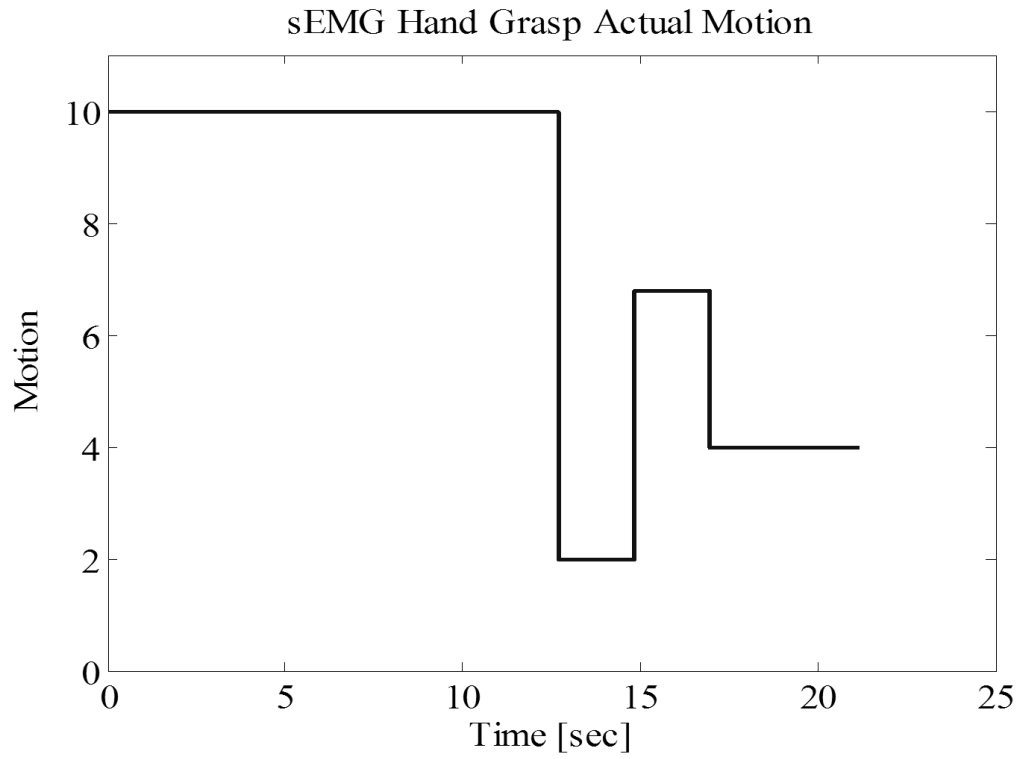


Figure 7.10: sEMG Actual Hand Motion

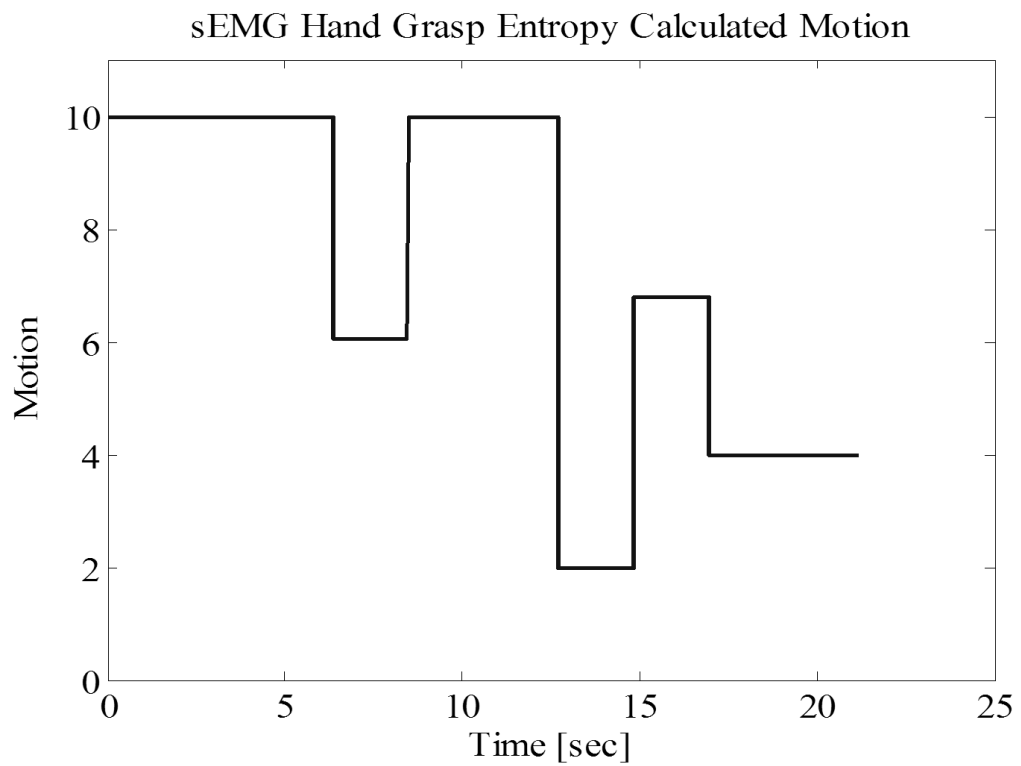


Figure 7.11: sEMG Predicted Hand Motion with Entropy

#### **7.4.4 EEG Signal**

The signals used for the EEG portion of this experiment are similar to the ones used in Chapter Chapter 6: . This method did not need to be expanded on further to be able to run on this platform as did the sEMG signals. These signals are also being used to prove that the controller and embedded processor can receive signals from two different biological signals.

#### **7.4.5 Text File Format**

With sEMG signals and EEG signals processed so that they can be run and an output obtained, an input method to the prosthetic hand platform needs to be created. Different methods were considered and finally it was decided the best method would be to use a text file as the input to the prosthetic hand platform. The advantage of the text file is that most computers can natively handle text files. Many program used to create controls have the option to output the file as a text file. Also, saving as a text file reduces the size of the file which is important when working with embedded processors. File size is important to limit on embedded platforms because memory is often small and it is expensive to add. The add-on memory often has slower execution speeds with larger files.

For the text file method to work properly a standard of how to format the data in the file needs to be set. The simplest and most straightforward method is to make it so only one

number was on a line. The number could be from 0 – 99. For practical purposes this was made for this range. It can be expanded if need.

The sEMG data plotted Figure 7.11 is an array of numbers. This output can be put into a text file that can be read by the Arduino. The numbers from the graph can be put into a text file with carriage return after each number or data point. The Processing code sends the data to the embedded processor and the Arduino uses that information to change the position of the servo motor.

The numbers in the text file need to correlate to the movement numbers listed in Chapter Chapter 7: , Section 7.4.2 Control of Virtual Hand mainly; 0 opens and 1 closes the pinky, 2 opens and 3 closes the ring finger, 4 opens and 5 closes the middle finger, and 6 opens and 7 closes the index finger. For the thumb's 1<sup>st</sup> DoF, 8 opens and 9 closes and for the 2<sup>nd</sup> DoF, 10 moves the thumb up and 11 moves the thumb down if applicable.

To open the entire hand a repeated sequence of odd numbers can be sent to the hand. Likewise to close then hand even numbers can be placed in the text file. To do different hand grasps they can be done by sending a sequence of numbers required to perform the grasp in the text file. Another method is to send a number from 0 – 10 in the Arduino code and when that number is sent have the Arduino move the motors to the desired positions. Both methods are used in the code in Appendix under the Six Servo Motor Arduino and Processing code.

It is important to note that the signal text files that are read using Processing need to be saved in the same file as the Processing main code and Class code. The file name also

has to be the same as the file name in the Processing code and it is best to not use spaces in the text file name and names of code.

## **7.5 Six DoF Hand Experiment**

With the virtual hand created and the motor control from the embedded processor developed, the next stage is to control the motors with actual biological signals. For this part of the process a universal method needs to be implemented. In the past [1] [3] [16] [43] control algorithms were developed. The problem is that the algorithms were developed with different programs. Some were made in MATLAB™ or Simulink™. These are both part of MATLAB™ but the way the code interacts with an embedded processor requires completely different toolboxes. It is also possible to develop control algorithms for a system with LabVIEW™, Micro-Cap, LTSPICE, or even C++. With so many different programs that could possibly be used to develop control for the prosthetic hand a universal approach needs to be taken to ensure that all the different controls could be tested on this platform.

### **7.5.1 Virtual Hand and Embedded System's Prosthetic Hand Code**

To conduct the experiment, an interface between the virtual hand and the embedded processor was created. For the two to interact with each other, first the embedded

processor needs to be programmed with the Arduino Programmer and then the virtual hand code is loaded. The Arduino code receives information that is sent to it from the processing code. Based on those signals the processor moves the appropriate motors and executes other commands. The main Processing code initializes all the variables, uses the class to create the virtual hand, is in charge of controlling the virtual hand, reading the biological signal files and sending signals to the embedded processor. Both these files need to be in the same file for the code to work properly.

The Processing code uses a class to reduce the size of *main()*. The class Shape is in a separate file and is used to create the hand in Figure 7.4 and Figure 7.5. This class code, along with the other Processing and embedded code, uses a matrix to reduce the size of code. Instead of writing *for* loops for each finger and thumb, a single *for* loop was created that steps through each element in a matrix. Each index in the matrix represents a different finger or thumb. This makes the code more compact and elegant. The code would be about double the length - if not more - otherwise. This makes it easier to update the code and change parts of the code too. It also reduces redundant code which is a must in professional programming.

The flow of the code can be seen below in Figure 7.12. This shows the flow of the processing code. The flow diagram of the Arduino code is in Figure 7.13.

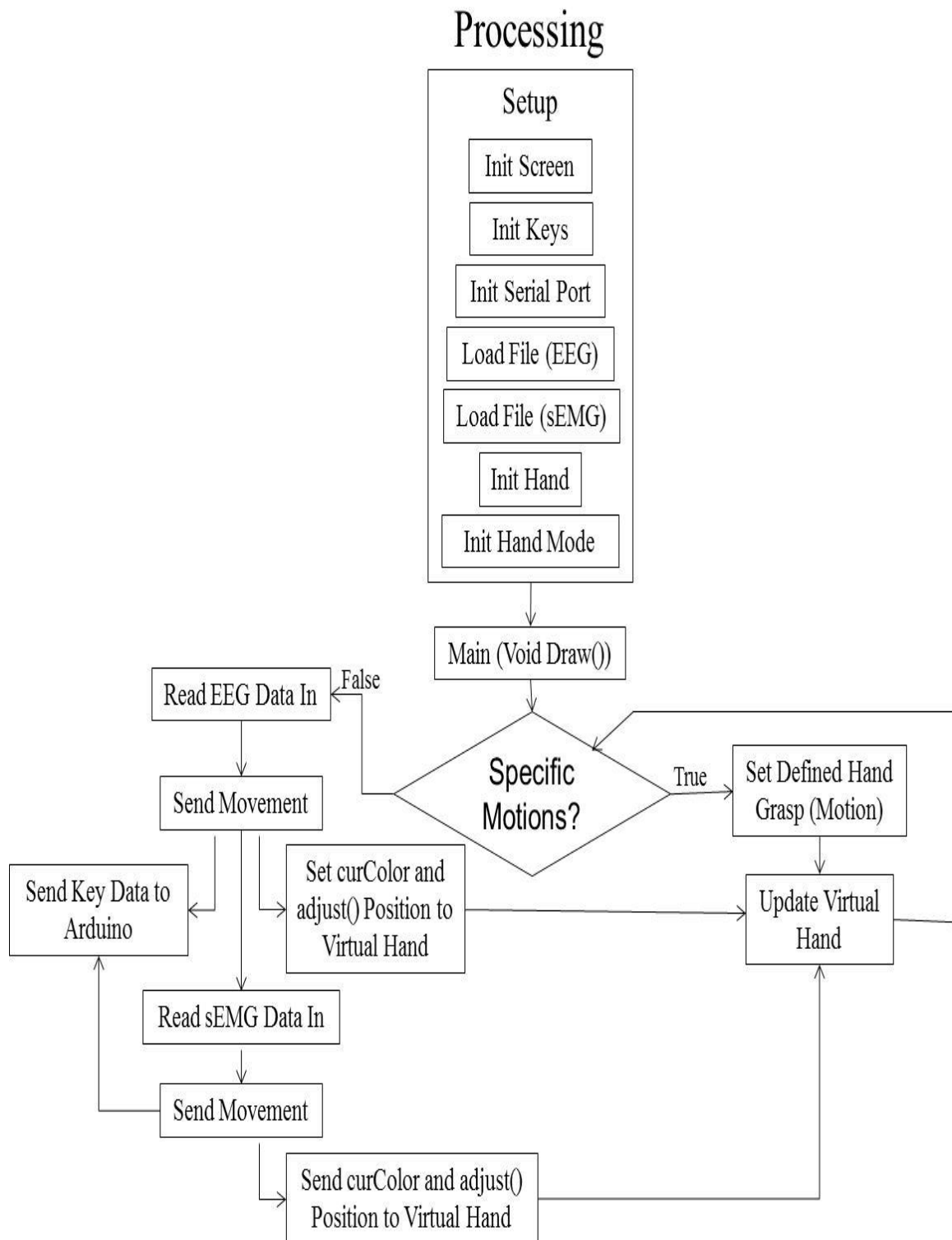


Figure 7.12: Processing Code Flow Diagram

# Arduino

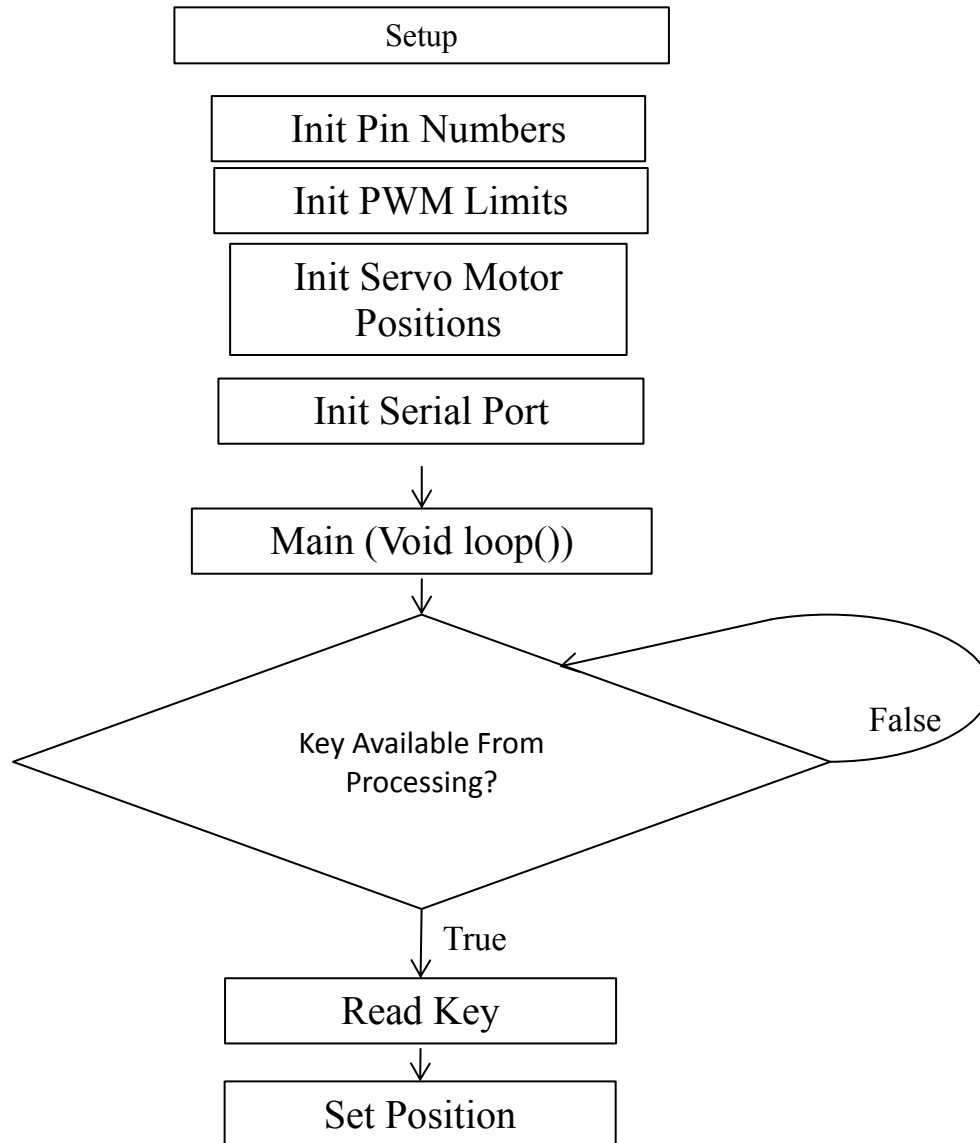


Figure 7.13: Arduino Code Flow Diagram

### **7.5.2 Servo Motor and Embedded Platform**

With the embedded processor and the computer being able to communicate with each other, the motors and circuitry were connected. As stated earlier in this chapter the main circuit protection used were resistors. The resistors were connected in series with the digital output pins of the embedded processor. The other end of the resistor was connected to the signal pin of the servo motors. The power for all the servo motors was provided externally with a power supply. This reduced the loading effect on the embedded platform since it is unable to provide enough current to drive all the servo motors under a loaded condition. All the grounds have to be connected; this means that the power supply, servo motor and embedded processor grounds need to be connected. If not, the embedded processor voltage relative to the servo motors will be different and the motors will not move or will respond in unpredictable ways.

Figure 7.14 shows the setup of the embedded processor with the circuit protection, power supply and the servo motors. These servo motors were successfully moved using the keyboard in conjunction with the virtual hand. They both moved when the keys were pressed, and after some adjustments, the virtual hand and the servo motor reached their limits at the same time.

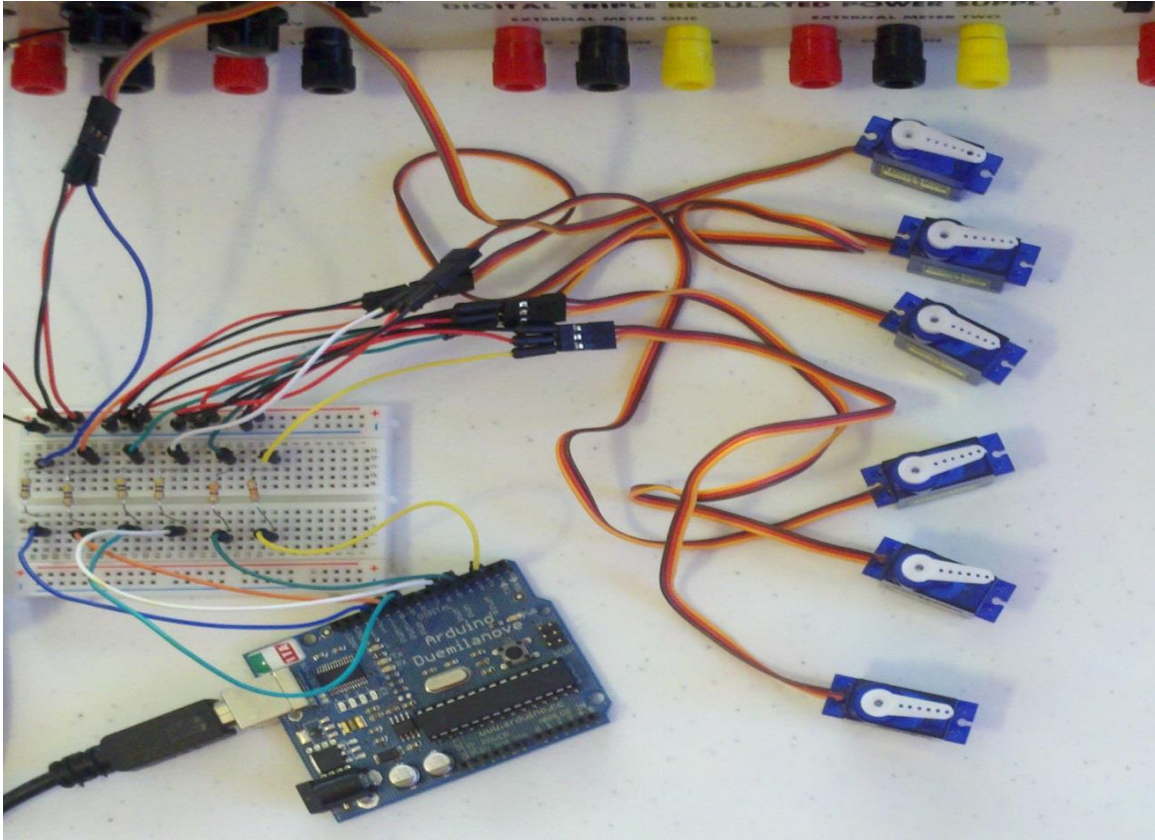


Figure 7.14: Six Servo Motor Setup with Embedded System

### 7.5.3 Biological Signal Input

Once the key command part of the code was validated the biological signal capacity was tested. Two different signals were used for this part: a sEMG signal and an EEG signal. Both signals were acquired with different data acquisition devices and processed using different algorithms. Both algorithms used a combination of MATLAB<sup>®</sup> and Simulink<sup>®</sup>.

The results from the algorithms were then converted into a text file following the rules in Section 7.4.5. Sample of the text file can be found in Appendix, sEMG Processed Signal

Text File Sample, and EEG Processed Signal Text File Sample. These files were much shorter than the original files because they had thousands of data points and often a number may be sent hundreds of times in a row. Notice in the sEMG signal file that close or open commands are sent by a string of even or odd numbers correlating to the fingers. In reality the fingers are not moving in parallel but to the naked eye they appear to move together.

This part of the experiment worked well. There was no buffering or delays while the file was being sent. All six servo motors were able to be moved with the system setup. To finalize this work it was decided to use this code on an actual robotic hand.

## **7.6 Five DoF Robotic Hand Implementation**

The final stage of this dissertation was to implement the working prosthetic hand system with an actually robotic hand. Figure 7.15 is a picture of a robotic hand with 5 DoF. This robotic hand is controlled by 5 different servo motors. Each servo motor controls a different finger or the thumb. The hand has the ability to grasp different objects.

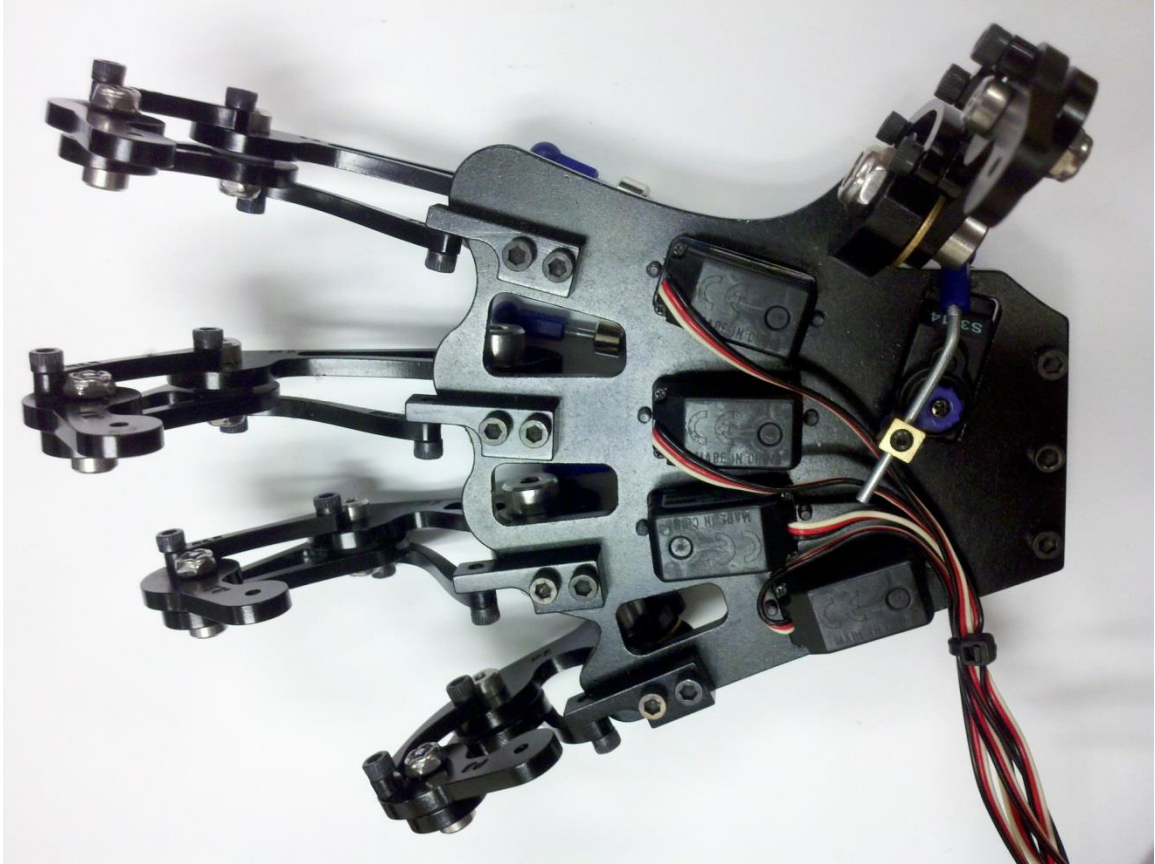


Figure 7.15: Five DoF Prosthetic Hand

The fingers and thumb are connected to the servo motor with a rod. The servo motor then pushes and pulls the rod to move the finger or thumb. Because the servo motors push and pull rods at the extreme ends of motion, the servo motors cannot provide as much torque to the fingers at fully closed or fully open positions. This also limits the range of motion to much less than 160 degree movement that a typical servo motor has. Some motion is as little as 20 degrees.

Even though this hand has these short comings it is a simple and elegant design. For this final experiment it was shown that a robotic hand could be controlled using this system.

The actual hand and the prosthetic hand could be moved in unison. This showed too that a prosthetic hand could be controlled using sEMG and EEG signals at the same time.

## Chapter 8: Results and Analyses

The results are excellent and showed that the entire system could control a robotic hand. The EEG system worked reliably for controlling a robotic thumb along with sEMG signals being used to control the fingers of the robot. The system was successfully able to execute the control of the hand with different algorithms developed in different ways using two programs.

sEMG signals were successfully used to control robotic fingers successfully. The author's work in [2] and [3] was expanded on so that the entropy fuzzy algorithm could be used to receive multiple hand motions in one file and successfully identify the different motions. This advanced the algorithms functionality and made it much more versatile.

For the sEMG signals, some of the variation and unsuccessful characterization may have been due to there being male and female subjects. Research done in [44] shows that sex causes variability in EMG amplitudes. Gender also may affect entropy calculations [2].

Overall the use of sEMG signals was successful. The sEMG signals were correctly identified. Those identified motions were then used to send by the code to control robotic fingers. Some of the hand motions that were used only moved one finger while others moved all four fingers.

Experiments showed that the system successfully identified signals and actuated the motors for the thumb 90.5% of the time. Occasionally the user needed to resend the signal two or three times to obtain a response. This generally depended on the mental fatigue of the person. After an hour of using the system the subjects began to have some difficulties sending the signals. The EEG identification was 80.0% at that time. With the results working well, a pod cast was created showing the system, explaining the setup, and showing the functioning system.

Initially training the patients was critical to the success of the EEG headset system. Sometimes an action would have to be trained numerous times. These repeated trials helped the system better identify the unique user. The users would also become more proficient at executing the actions. Once the training was completed the patients were able to reliably use the EEG system.

The few times the EEG system didn't identify the signal correctly, before fatigue began to set in, was mostly due to excessive eye blinking while the signal was being sent. Blinking is known to cause problems with EEG measurements and is called artifact noise. With practice the subjects were able to send signals without interrupting the signals by blinking. If the patients took time to train, sometimes electrodes needed saline solution reapplied to them. These steps allowed for high accuracy which made controlling the prosthetic thumb feasible.

A virtual hand that represented a 6 DoF hand was developed and implemented. The hand was able to display when signals were sent either by the keyboard or by a biological

signal or signals. The virtual hand has a high degree of accuracy so that if precise hand movements are required it can handle these.

The sEMG and EEG processing algorithms were both implemented in MATLAB<sup>®</sup> and Simulink<sup>®</sup>. These algorithms were processed differently within these programs. The results from the algorithms were then formatted into a text file. This text file was then used for the control of the robotic hand. This part of the project was successful in developing a method that could use a universal input to control a hand rather than one specific program. This will help reduce cost because to run this a design team will not need multiple licenses of expensive program packages.

The system identification effort in this work was unsuccessful. A possible reason that it didn't work was that there is a lot of noise in sEMG signals. It is difficult to filter out all the noise from the surrounding environment and collect a pure sEMG signal. The success of this part of the project was that a fuzzy controller was designed that filtered the output from the SI. The fuzzy controller made the signal much more uniform which would be desirable when implementing on an embedded platform.

The hand was able to work without the actual robotic hand being attached to the system. The ability to work independently of an actual robotic hand connected to the system will allow for it to be used for testing algorithms, or updating the code without the actual prosthetic hand being attached. This will allow for a lab to have multiple people or teams working on a project with only a few prosthetic hands that will lead to cost savings to the labs. The code was also written in C allowing for many users with different backgrounds to be able to work on the source code if needed.

A minor issue with the 5 DoF hand was that for some fingers the limits could not be adjusted to prevent the finger from shaking or making a humming noise. This was also noticed in the 6 DoF experiments. It appears to be a result of a defect with some servo motors. These servo motors are inexpensive and so defects are more likely to occur. For a final design it is suggested that more precise, robust, industrial grade servo motors be used to prevent this from occurring in the final prosthetic hand.

The promising work in [1] [2] [3] was successfully implemented on an embedded platform and could control a robotic hand. The platform could be controlled by either a computer or biological signals. The code for the embedded processor and the virtual hand were both written in a common language to allow for a wider audience to be able to work on modifying or updating the code for their projects. The code was also able to control up to 6 motors. These abilities make this platform a more valuable test bed for experimenting with control algorithms and different prosthetic hands.

## Chapter 9: Conclusion

With the increasing number of people who have missing limbs, the need for prosthetics is increasing. Prosthetic hands are advancing in dexterity and complexity with modern robotics. The controls for the hands are getting more advanced and are allowing for more dynamic inputs. This dissertation looked at how to advance the control of prosthetic hands. EMG signal processing algorithms have been used in the past for controlling simple robotic hands [2], but with the progression of robotic hands newer EEG methods may be able to be utilized in conjunction with sEMG signals to control a prosthetic hand.

This dissertation had multiple objectives. Expand on the author's thesis [2]: specifically further develop the sEMG control method. Develop a method to use EEG signals to control a robotic thumb of a prosthetic hand with 2 DoF. Develop a platform that is able to control a robotic hand using multiple biological signals. Also, this system needs to be able to work with different programming environments used for creating control algorithms.

Non-invasive methods of controlling prosthetic hands are being researched because of the need for a low cost dynamic prosthetic hand because surgery is not need to place the sensors. This dissertation improved on the previous work in [2] of the use of surface sensors to acquire EMG. This work modified an existing entropy algorithm that was a viable control strategy. It was modified so that it can be used to identify different hand motions in continuous signals with a high level of accuracy.

This paper presented a reliable method that is able to control a 2 DoF thumb by using EEG signals. This system uses an EEG headset with surface sensors. This is a cost-effective solution for controlling a robotic thumb. The results showed that surface EEG electrodes can be used to successfully control a robotic thumb. A working model was synthesized that has been demonstrated to work with 90.5% accuracy in executing the user desired motion.

sEMG and EEG control algorithms were developed using multiple programs. Despite having the algorithms developed with different programs the outputs from the algorithms were then converted into text files based on set formatting rules. A platform was developed that use these text files to control a robotic hand.

The platform was developed using C, a common programming language. This platform also had a virtual hand. This virtual hand mimics what the actual hand does. This is useful for testing if the robotic hand is unavailable. The hand, virtual or real, can be controlled by the two different biological signals, or by keyboard commands. The system also has an embedded processor that is in charge of controlling and interfacing with the motors. Previous work didn't implement any protection for the embedded processor so this was modified to provide overcurrent protection.

This dissertation successfully created a system that was able to control an actual robotic hand. This platform can also control a virtual hand and a robotic hand simultaneously. It can take two different biological signals, sEMG and EEG, both created with different algorithms and use them in combination to control the hands. Both of these novel algorithms have shown promising results in controlling a prosthetic hand. This work was

done with a cost effective system, robotic hand, and non-invasive sensors. This worked show that an agile system can be used to prototype different algorithms with or without an actual robotic hand attached to the system.

## **Chapter 10: Future Work**

More experiments can be performed on a multitude of subjects to verify that weight, gender, age and other parameters do not affect the efficiency of the system. If these do affect it, work can be done to overcome these problems. Further research can be done in developing more robust filters to remove the blinking noise that is very prominent in EEG signals. Experiments can be conducted to see how much time it takes for fatigue to set in and how much, if any, the time of the day affects the results. Additional work can be done in looking how to use SI for processing sEMG signals. Further work can be done with the embedded system by developing an independent ATmega 328 circuit board system to reduce the total size of the embedded platform. Work should also be done to determine if this system can be patented.

## References

- [1] Jensen A.N., Clark A., Sparks N., Chiu S., and Schoen M., "Embedded Electroencephalogram (EEG) Processing and Control for the Actuation of a Prosthetic Thumb Prototype," *IEEE International Conference on Electro/Information Technology*, May 2013.
- [2] Jensen A. N., Intelligent Classification of Surface Electromyographic (EMG) Signals Based on Entropy for the Control of a Robotic Hand, 2011, Thesis.
- [3] Jensen A.N., Potluri C., Clark A., Chiu S., and Urfer A., "Intelligent Classification of Surface Electromyographic (sEMG)," *IEEE International Conference on Electro/Information Technology (EIT)*, May 2013.
- [4] (2011) ACA News: National Limb Loss Awareness Month. [Online].  
<http://www.bocusa.org/aca-news-national-limb-loss-awareness-month>
- [5] Zoroya G. (2011, Sept) Injuries cost more troops their limbs. [Online].  
<http://www.usatoday.com/news/military/story/2011-09-19/troops-injuries-war-casualties-amputations-lost-limbs/50472074/1>
- [6] Roth B., and Salisbury J. Zinn M., "A New Actuation Approach for Human Friendly Robot Design," *Int Robot Res.*, pp. 379-398, 2004.
- [7] Heinzmann J. and Zelinky J., "A Safe-Control Paradigm for Human-Robot Interaction," *Intelligent Robot System*, vol. 24, no. 4, pp. 295-310, 1999.
- [8] Sherman E., "A Russian Bioelectric-Controlled Prosthesis: Report of a Research Team from the Rehabilitation Institute of Montreal," *Canad. Med. Ass. J.*, vol. 91,

pp. 1268-1270, 1964.

- [9] (2012, Dec) CBSNews. [Online]. [http://www.cbsnews.com/8301-18560\\_162-57559331/paralyzed-woman-uses-mind-to-move-robotic-arm/](http://www.cbsnews.com/8301-18560_162-57559331/paralyzed-woman-uses-mind-to-move-robotic-arm/)
- [10] Shadow Robot Company. (2013) Shadow Robot Company. [Online]. <http://www.shadowrobot.com/products/dexterous-hand/>
- [11] Sandia National Laboratories. (2012, August) Sandia Labs News Releases. [Online]. [https://share.sandia.gov/news/resources/news\\_releases/robotic\\_hand/](https://share.sandia.gov/news/resources/news_releases/robotic_hand/)
- [12] Artemiadis P. K., Shakhnarovich G., Vargas-Irwin C., Donoghue J. P. and Black M. J., "Decoding grasp aperture from motor-cortical population activity," in *Proceedings of the 3rd International IEEE EMBS Conference on Neural Engineering*, Kohala Coast, Hawaii, 2007.
- [13] Madhow, U., "Blind Adaptive Interference Suppression for Direct-Sequence CDMA," *Proceedings of the IEEE*, vol. 86, no. 10, pp. 2049–2069, 1998.
- [14] Agashe, H. A. and Contreras-Vidal, J. L., "Reconstructing hand kinematics during reach to grasp movements from electroencephalographic signals," in *33rd Annual International Conference of the IEEE EMBS*, Boston, Massachusetts, 2011.
- [15] Mu Z., and Hu J., "Research of EEG Identification Computing Based on AR Model," in *International Conference on Future BioMedical Information Engineering*, 2009, pp. 366-368.
- [16] Potluri C., Yihun Y., Jensen A.N., Anugolu M., Chiu S., Schoen M., Naidu D.S., "Optimal Tracking of a sEMG based Force Model for a Prosthetic Hand," *IEEE Engineering in Medicine and Biology Society*, September 2011.

- [17] Potluri C., Jensen A.N., Anugolu M., Sriram G., Liu S., Chiu S., and Urfer A., "PIC 32 Microcontroller Based sEMG Acquisition System and Processing Using Wavelet Transforms," *World Comp International Conference on Embedded Systems and Applications (ESA)*, 2012.
- [18] Fortuna J. and Capson D., "ICA for Position and Pose Measurement from Images with Occlusion," *Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing, (ICASSP '02)*, vol. 4, pp. 3604–3607, 2002.
- [19] Oja E., Kiviluoto K. and Malaroiu S., "Independent Component Analysis for Financial Time Series," in *IEEE Adaptive Systems for Signal Processing, Communications, and Control Symposium (AS-SPCC)*, Lake Louise, Alta, 2000, pp. 111–116.
- [20] Sparks N. T., Signal Processing Of Electroencephalogram: A study On Blind Source Separation With System Identification And Subtractive Clustering, 2012, Thesis.
- [21] Kandel J. and Scharz E., *Principles of Neural Science*. New York: Elsevier/North-Holland, 1981.
- [22] Hoehn E. and Maarieb K., *Human Anatomy and Hysiology*, 7th ed. San Fancisco, United States of America: Pearson Benjamin Cummings, 2007.
- [23] Teplan M., "Fundamentals of EEG Measurement," *Measurement Science Review*, vol. 2, no. 2, 2002.
- [24] John R. Hughes, *EEG In Clinical Practice*, Second Edition ed. Newton, MA, USA: Butterwork-Heinemann, 1994.
- [25] BCI 2000. BCI 2000. [Online].

[http://www.bci2000.org/wiki/index.php/User\\_Tutorial:EEG\\_Measurement\\_Setup](http://www.bci2000.org/wiki/index.php/User_Tutorial:EEG_Measurement_Setup)

- [26] Candy, J. V., *Model-Based Signal Processing*. Hoboken, New Jersey: John Wiley & Sons, Inc., 2005.
- [27] Juang, J. N., *Applied System Identification*. Englewood Cliffs, NJ: Prentice Hall, 1994.
- [28] Alvin K.F., Robertson A. N., Reigh G. W., and Park K. C., "Structural system identification: from reality to models," *Computers and Structures*, vol. 81, pp. 1149-1176, 2003.
- [29] Ho B.L., Kalman R.E., "Effective construction of linear state variable models from input/output data," in *Proceedings of the Third Annual Allerton Conference on Circuit and System Theory*, Regelungstechnik, 1966, pp. 449-459.
- [30] (2013) State Space Equation. [Online].  
[http://cs.bilgi.edu.tr/pages/courses/year\\_4/comp\\_422/Archive/2006-2007/Week01/Week01.pdf](http://cs.bilgi.edu.tr/pages/courses/year_4/comp_422/Archive/2006-2007/Week01/Week01.pdf)
- [31] Lim R., Phan M., and Longman R., "State-Space System Identification with Identified Hankel Matrix," *Technical Report No. 3045, Princeton University, Princeton, NJ*, September 1998.
- [32] Candy, J., *Model-Based Signal Processing*. Hoboken, New Jersey, USA: John Wiley & Sons, Inc., 2006.
- [33] Matlab. (2011) N4SID. Document.
- [34] Potluri, C.; Kumar, P.; Anugolu, M.; Chiu, S.; Urfer, A.; Schoen, M.; Naidu, D.S., "sEMG Based Fuzzy Control Strategy with ANFIS Path Planning For Prosthetic

- Hand," in *IEEE RAS and EMBS International Conference on Biomedical Robotics and Biomechatronics*, Tokyo, September 2010, pp. 413-418.
- [35] Griffin M. (2011, May) BCI 2000. [Online]. <http://www.bci2000.org/wiki/index.php/Contributions:Emotiv>
- [36] Emotiv. (2013) Research Edition. [Online]. <http://emotiv.com/store/sdk/bci/research-edition-sdk/>
- [37] Emotiv. SDK User Manual for Emotiv. PDF. [Online]. [www.emotiv.com](http://www.emotiv.com)
- [38] (2012, December) Arduino. [Online]. <http://arduino.cc/en/>
- [39] Banzi M., Cuartielles D., Igoe T., Martino G., and Mellis D. Arduino Alpha.
- [40] Fry B. and Reas C. (2001) Processing.
- [41] Cichock A., Amari S., Siwek K., Tanaka T., Phan A., Zdunek R. (2007) ICALAB – MATLAB Toolbox Ver. 3 for signal processing.
- [42] "Standards for Reporting EMG Data," *Journal of Electromyography and Kinesiology*, vol. 9, no. 1, pp. III-IV, February 1999.
- [43] Kumar P., Chen C. H., Sebastian A., Anugolu M., Potluri C., Fassih A., Yihun Y., Jensen A. N., Tang Y., Chiu S., Bosworth K., Naidu D.S., and Schoen M., "Adaptive Hybrid Data Fusion Based Identifications of Skeletal Muscle Force with ANFIS and Smothing Spline Curve Fitting," *IEEE International Conference on Fuzzy Systems*, pp. 932 - 938, June 2011.
- [44] Zurcher U., and Kaufman M. Sung P., "Comparision of Spectral and Entropic Measures for Surface Electromyography Time Series: A Pilot Study," *Journal of Rehabilitation Research and Development*, vol. 44, no. 4, pp. 599-610, 2007.

- [45] Tong L., Liu R., Soon V. and Huang Y., "Indeterminacy and Identifiability of Blind Identification," *IEEE Transactions on Circuits and Systems*, vol. 38, no. 5, pp. 499-509, 1991.

# **Appendix**

## **Emotiv File Saving Process After an Experiment**

Once the user has been trained, setup is complete and an experiment is conducted and the results want to be saved follow the procedures below:

Open the Emotiv TestBench program, see Figure 0.1. Do not close the control panel program. TestBench program, in Figure 0.1, allows the operator to record the EEG signals, and is only a part of the research edition of the software. To save the data, click save, enter the information prompted for and save the data in the desired location.

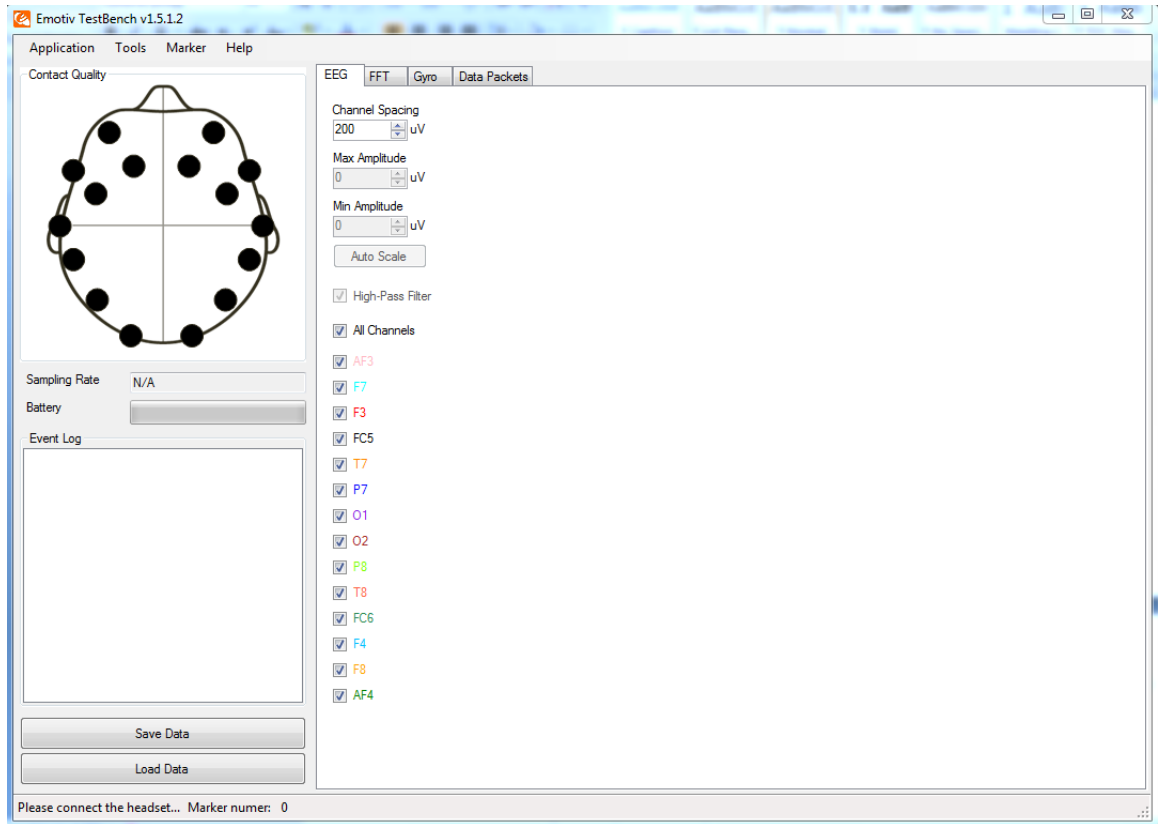


Figure 0.1: Emotiv TestBench, [45]

Finally, once the data has been recorded, the data can be either played back by using the load button in the bottom left hand corner as seen in Figure 0.1, or can be converted to an Excel CSV file. To convert the EDF file, which is an Emotiv output recoding file, to an Excel CSV file, follow these steps: click Tools then click on convert EDF to CSV. A window (see Figure 0.2) will open and the EDF can be converted to an Excel file which can be imported into MATLAB™.

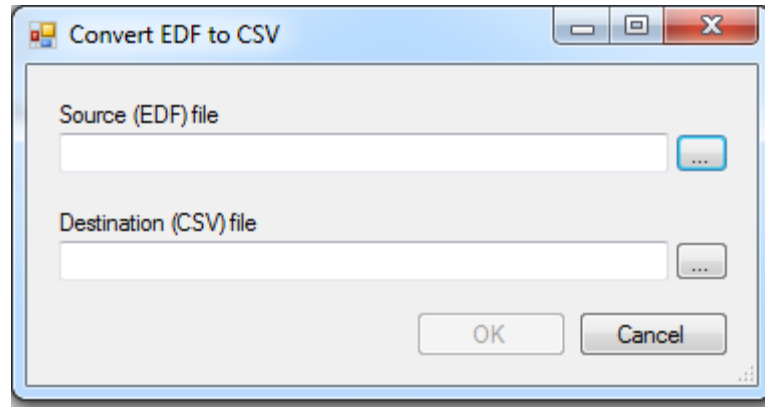


Figure 0.2: Convert EDF to CSV, [45]

## Arduino Startup Check List

Check that the correct board was selected:

- 1) Tools
- 2) Board
- 3) Select the proper board for example the “Arduino Duemilanove or Nano w/ ATmega328”
  - a) When selecting the board, make sure that that the correct chip size is chosen. In the above example 328 was chosen because it has an ATM 328 chip.

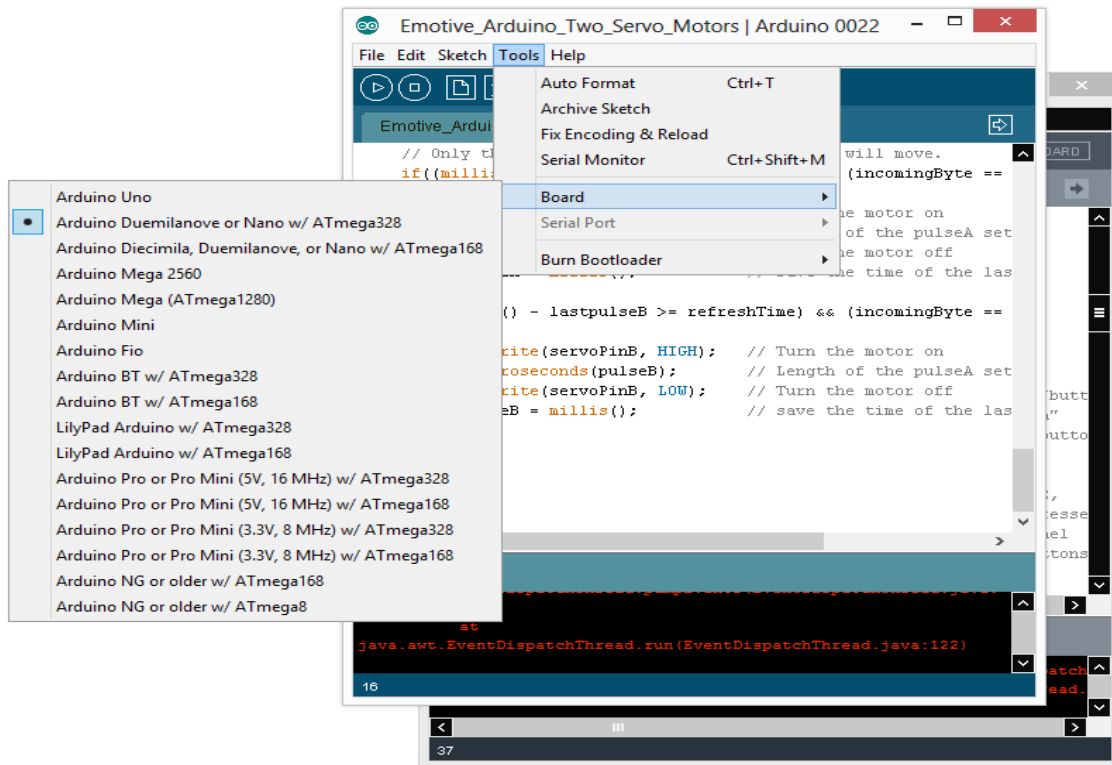


Figure 0.3: Arduino Board Selection

Make sure that the correct serial port is selected if there is a communication problem:

- 1) Tools
- 2) Board
- 3) Serial Port

(Note: If this doesn't work sometimes unplugging the USB cable and moving it to a different USB port will work.)

If the Arduino driver isn't working properly it might be because the correct driver was not installed. To check and see, follow the instructions below:

- 1) Open Arduino environment
- 2) Connect Arduino USB to the computer
- 3) Verify that the drivers are working
  - a) If not open up Device Manager
    - i) Start
    - ii) Control Panel
    - iii) Device Manager
    - iv) Right click on the USB with the error
      - (1) Update Driver
      - (2) Select the driver folder in the Arduino folder
  - b) If driver is installed go to next step

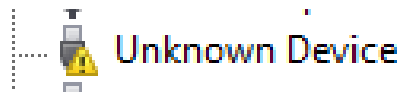


Figure 0.4: Unknown Device Detected

- 4) Open a file
- 5) Uploaded file (use Upload in Figure 0.7 and Figure 0.6 appears if it was a successful upload)
 

If you get a “COM” error try moving the Arduino USB cable to another USB adapter/port. (See

  - a) Figure 0.5)
  - b) Make sure that you selected the correct Arduino under Tools, Board.

Serial port 'COM3' not found. Did you select the right one from the Tools > Serial Port menu?

Figure 0.5: COM Issues

- 6) Wait until it finishes loading (As shown in Figure 0.6)
- 7) Open Serial Monitor (As shown in Figure 0.7)

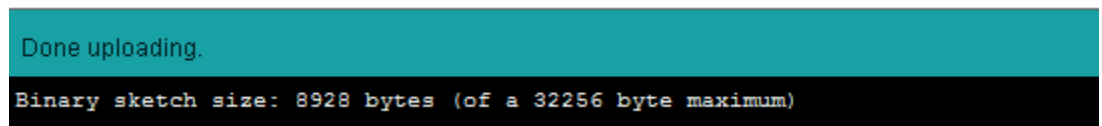


Figure 0.6: Done Uploading



Figure 0.7: Arduino Front Conceal

- 8) Upload Arduino Code from Arduino 22 or other version
- 9) Play Processing code after Arduino 22 is uploaded.

## sEMG Processed Signal Text File Sample

0	4	5	1	5	0
2	6	5	3	7	2
4	0	5	5	1	4
6	2	5	7	3	6
0	4	5	1	5	0
2	6	5	3	7	2
4	0	5	5	5	4
6	2	5	7	5	6
0	4	5	1	5	99
2	6	5	3	5	99
4	0	5	5	5	99
6	2	5	7	5	99
0	4	5	1	5	99
2	6	5	3	5	99
4	0	5	5	5	99
6	5	5	7	5	99
0	5	5	1	5	99
2	5	5	3	5	99
4	5	0	5	5	99
6	5	2	7	5	99
0	5	4	1	5	99
2	5	6	3	5	99
4	5	0	5	0	99
6	5	2	7	2	99
0	5	4	1	4	99
2	5	6	3	6	99
4	5	0	5	0	99
6	5	2	7	2	99
0	5	4	1	4	99
2	5	6	3	6	
4	5	0	5	0	
6	5	2	7	2	
0	5	4	1	4	
2	5	6	3	6	

## EEG Processed Signal Text File Sample

10	8	10	10	11	10
10	8	10	10	11	10
10	8	10	10	10	10
10	8	10	10	10	10
10	8	10	10	10	8
10	8	10	10	10	8
10	8	10	10	10	8
10	8	10	10	10	8
10	8	10	10	10	8
10	8	10	10	10	8
10	8	10	10	10	8
10	8	9	10	10	8
10	8	9	10	10	10
10	8	9	10	8	10
10	8	9	10	8	10
10	8	9	10	8	10
10	8	9	10	8	10
10	8	9	11	8	10
10	8	9	11	8	10
10	8	9	11	8	10
10	8	9	11	8	10
10	10	9	11	8	10
10	10	9	11	8	10
10	10	9	11	8	10
10	10	9	11	8	10
8	10	9	11	8	10
8	10	9	11	10	
8	10	9	11	10	
8	10	9	11	10	
8	10	9	11	10	
8	10	9	11	10	
8	10	9	11	10	
8	10	10	11	10	

## AMUSE MATLAB™ Code

```
clear; % Clears all variables
clc; % Clears the screen
input('How many data points: '); L=ans;
input('Model order p: '); p=ans;
input('How many output signals: '); no=ans;
load('matlab.mat', 'data')
y = load('matlab.mat', 'data');
p = 100;

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%% STEP 1 Estimate Output Covariance Rx %%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
[no,nd] = size(data); % no is the number of output(rows), nd is number (columns)
L = nd;

y = zeros(no,L); % Creates a matrix of defined demintions with random numbers
y = data';

phik = zeros(nd-p,no*p); % Creates a matrix of given size in increase speed
for i=p:(nd-1) % rows
    for j = 1:p % coloms
        col = j;
        phik(i-p+1,col*no-no+1:col*no)= y(:,i-j+1)';
    end;
end;

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%% Autocorrelation Rx = E{x(t)*x'(t)} %%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

[Q1,R1] = size(y);
Rx = zeros(Q1,R1);
for j = 1: Q1
    for m = 1: R1+1
        for n = 1:Q1-m+1
            Rx(Q1,m) = Rx(Q1,m)+y(Q1,n)*y(Q1,n+m-1);
        end;
    end;
end

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%% STEP 2 Compute SVD %%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

```

[U,S,V] = svd(Rx);

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%% STEP 3 Singa^2 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

phik = zeros(nd-p,no*p); % Creates a matrix of given size in increase speed
for i=p:(nd-1) % rows
    for j = 1:p % coloms
        col = j;
        phik(i-p+1,col*no-no+1:col*no)= y(:,i-j+1)';
    end;
end;

R = phik'*phik;

Y = y(:,p+1:L)'; % Estimated yhat from y of p+1 columns to L (the end)
temp = phik' * (phik);
temp1 = inv(temp);

thetaBarHat = temp1 * phik'*Y; % (phik' * (phik))^( -1) * phik'*yhat

yhat2=zeros(no,L);

for k = p+1:L
    for i = 1:p
        yhat2(:,k)=yhat2(:,k)+thetaBarHat((no*(i-1))+1:no*i,:)*y(:,k-i);
    end;
end;

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
Epsilon=y-yhat2;

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%
% Autocorrelation
%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

[Q,R] = size(Epsilon);
%NEED TO FIX INDEXES
Rxx = zeros(Q,R);

for j = 1: Q % Rows for Rxx

```

```

for m = 1: R+1 % Columns for Rxx
    for n = 1:Q-m+1
        Rxx(j,m) = Rxx(j,m)+Epsilon(j,n)*Epsilon(j,n+m-1);

    end;
end;
end
sigma2 = Rxx(1)

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
STEP 4 Preform Orthogonalization Transformation %%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

S1 = S(S~=0)' % Removes the zeros in S and makes it a vector
m = no*p; % m is this length so the code runs but we will
% optimize m later.
for i = 1:m
    di(i) = sqrt(S1(1,i) - sigma2); % di in Algorithm
end;

Us = U(:,1:m);
T = diag(1./di)*Us';

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
STEP 5 Estimate 4th Order Moment %%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

Yt = T*y;%Epsilon;
[Q,R] = size(Yt);
Yxx = zeros(Q,R);
for j = 1: Q
    for m = 1: R+1
        for n = 1:Q-m+1
            Yxx(Q,m) = Yxx(Q,m)+Yt(Q,n)*Yt(Q,n+m-1);
        end;
    end;
end;
M = Yxx*Yxx;

for i = 1:m
    for k = 1:m
        Sum(k) = (di(i)^.2 + 1)/di(k)^2 + 2 * sigma2/di(k)^2;
    end;
    deltai(i) = ((m + 4)*sigma2)/di(i)^2 + sigma2/di(i)^2*Sum(k);
end;
deltaM = diag(deltai);

```

```

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
STEP 6 Singular Valu Decomposition of M - delta M %%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

```

```

[U2,S2,V2] = svd(M-deltaM);

```

```

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
STEP 7 Channel Estimation A0:Ahat = T'*V2 %%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

```

```

Ahat = pinv(T)*V2; % Ahat is pseudoinverse of T times V2

```

```

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
STEP 8 Signal Estimation So(*):Shat(t)=V'*y(t)%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

```

```

for t = m
    Shat(t) = V2(t)*y(t)
end;

```

## System Identification Steps

- 1) Open the System Identification Toolbox, see Figure 0.8
- 2) Import .sid file
- 3) Select Linear parametric model, Figure 0.9
  - a) Choose ARX
  - b) Focus: Simulation
  - c) Initial state: Auto

- d) Covariance Estimate
- 4) Click Estimate
- 5) Select Model from the Model View section on the right
- 6) Check Model output box
- 7) The plot will display as in Figure 0.10
- 8) To extract the data points to use in MATLAB™
  - a) Click and drag the model to the To Workspace square shown in Figure 0.8
    - i) This exports the System Identification model to the workspace of MATLAB™
  - b) Make sure that the input file (one used to make the model) is in the workspace too.
  - c) Use the following MATLAB™ functions to extract the output
    - i) `ys = sim(arx441, EEG4motion)`
      - (1) `ys` is the output file variable name
      - (2) the function `sim(model, U)` where `model` is the S.I. model and `U` is the input)

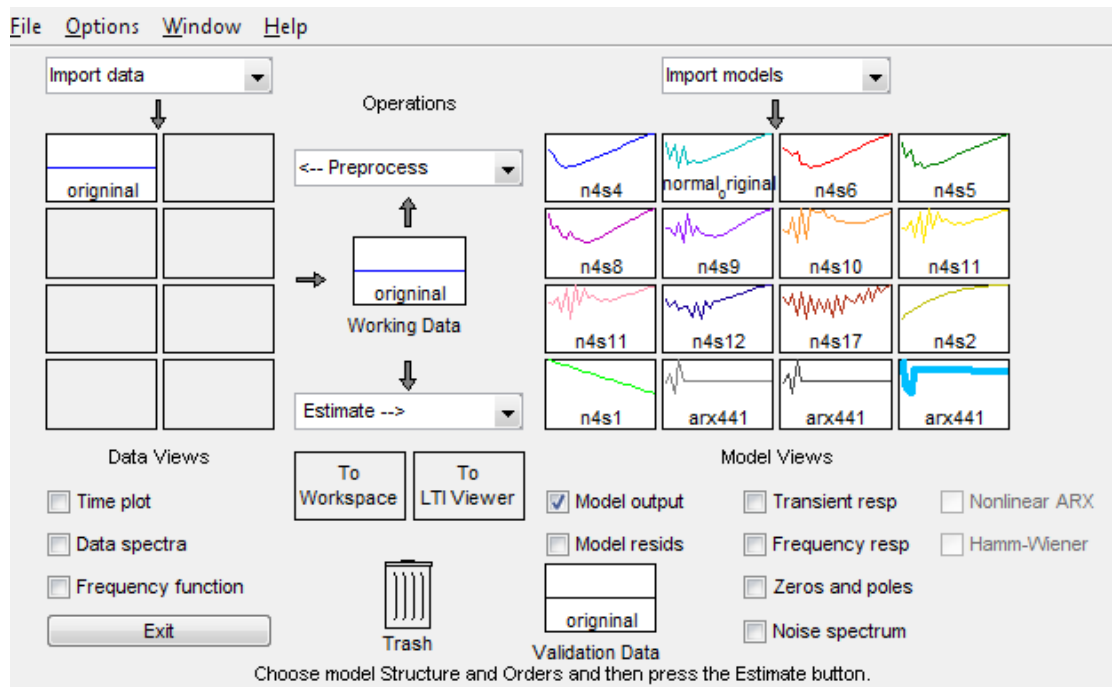


Figure 0.8: System Identification Toolbox

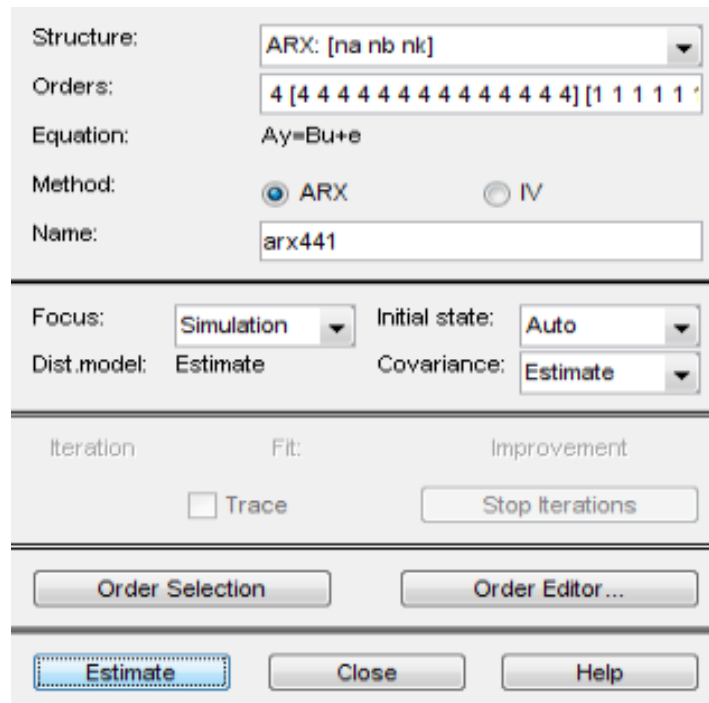


Figure 0.9: Linear Parametric Model Menu

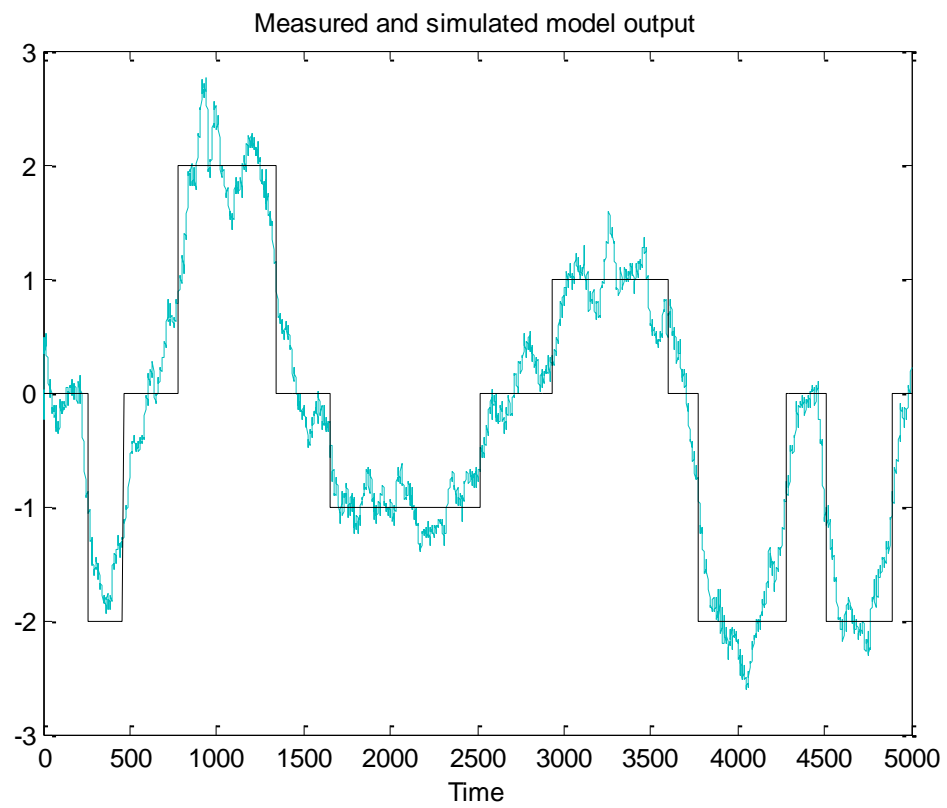


Figure 0.10: System Identification Output

## MATLAB™ Figure Formatting and Creating Code

```
1. % This code creates figures and formats the title and axes making them readable
2. % on a computer screen. This loads in variable from simulink and plots the results
3. FuzzyOut = zeros(5000,2);
4. FuzzyOutTemp = FuzzyOut1.signals.values(56:5055,:);
5. FuzzyOut(:,1) = track2;
6. FuzzyOut(:,2) = FuzzyOutTemp(:,1);

7. ySTI = zeros(5000,2);
8. ySTI(:,1) = track2;
9. ySTI(:,2) = YS(9:5008,2);

10. figure;
11. hold on;
12.    xlabel('Time/ 0.0087 sec per unit','FontSize',16,'FontName','Times')
13.    ylabel('Motions','FontSize',19,'FontName','Times')
14.    title('Fuzzy STI Output','FontSize',22,'FontName','Times')
15.    plot(FuzzyOut(:,1),'color','k','LineWidth',2)
16.    plot(FuzzyOut(:,2),'color','b')
17.    legend('User Input','Fuzzy STI Output');
18. hold off;

19. figure;
20. hold on;
21.    xlabel('Time/ 0.0087 sec per unit','FontSize',16,'FontName','Times')
22.    ylabel('Motions','FontSize',19,'FontName','Times')
23.    title('STI Output','FontSize',22,'FontName','Times')
24.    plot(ySTI(:,1),'color','k','LineWidth',2)
25.    plot(ySTI(:,2),'color','b')
26.    legend('User Input','STI Output');
27. hold off;
```

## List of Publications

- [1] Alex N. Jensen, "Shield Transfer Impedance (STI) Analysis of W5G Cables," PRIME 122680.
- [2] **Alex N. Jensen**, "Shield Transfer Impedance (STI) of Y Cable Aging Surveillance (A/S) P106 Actuator Test Interface Cable," Northrop Grumman, pp. 1-17, 2013.
- [3] John P. Rohrbaugh, Amanda Snyder, Brian Christian, **Alex N. Jensen**, "Request for Hazards of Electromagnetic Radiation to Ordnance (HERO) Certification," Prime 122679, pp. 1-77, 27 March 2014.
- [4] Girish Sriram, Alex N. Jensen, Steve Chiu, "Slippage Control for a Smart Prosthetic Hand Prototype via Modified Tactile Sensory," IEEE International Conference on Electro/Information Technology, Submitted.
- [5] Jason Parmenter, Alex N. Jensen, Steve Chiu, "Smart Irrigation Controller," IEEE International Conference on Electro/Information Technology, Submitted.
- [6] Steve Philips, Alex N. Jensen, Steve Chiu, "Current Protection for Robotic Hand," IEEE International Conference on Electro/Information Technology, Submitted.
- [7] **Alex N. Jensen**, Chandrasekhar Potluri, Amanda J. Clark, Steve Chiu, and Alex Urfer, "Intelligent Classification of Surface Electromyographic (sEMG) Signals," IEEE International Conference on Electro/Information Technology (EIT), May 2013.

- [8] **Alex N. Jensen**, Amanda J. Clark, Nathan T. Sparks, Steve Chiu, and Marco Schoen, "Embedded Electroencephalogram (EEG) Processing and Control for the Actuation of a Prosthetic Thumb Prototype," IEEE International Conference on Electro/Information Technology, May 2013.
- [9] **Alex N. Jensen**, Intelligent Classification of Surface Electromyographic (EMG) Signals Based on Entropy for the Control of a Robotic Hand, 2011, Thesis.
- [10] Chandrasekhar Potluri, **Alex N. Jensen**, Madhavi Anugolu, Girish Sriram, Shiwei Liu, Steve Chiu, Alex Urfer, "PIC 32 Microcontroller Based sEMG Acquisition System and Processing Using Wavelet Transforms," 2012.
- [11] **Alex N. Jensen**, Chandrasekhar Potluri, and D. Subbaram Naidu, "Fusion of Hard and Soft Control Strategies for a Double Inverted Pendulum," International Conference on Mechatronics, Robotics and Manufacturing, pp. 19-23, December 2011.
- [12] Chandrasekhar Potluri, Yimesker Yihun, **Alex N. Jensen**, Madhavi Anugolu, Steve Chiu, Marco P. Schoen, D. S. Naidu, "Optimal Tracking of a sEMG based Force Model for a Prosthetic Hand," IEEE Engineering in Medicine and Biology Society, September 2011.
- [13] Parmod Kumar, C. H. Chen, Anish Sebastian, Madhavi Anugolu, Chandrasekhar Potluri, Amir Fassih, Yimesker Yihun, **Alex N. Jensen**, Yi Tang, Steve Chiu, Ken Bosworth, D. S. Naidu, Marco Schoen, "Adaptive Hybrid Data Fusion Based Identifications of Skeletal Muscle Force with ANFIS and Smoothing Spline Curve Fitting," IEEE International Conference on Fuzzy Systems, pp. 932 - 938, June 2011.

- [14] Chandrasekhar Potluri, **Alex N. Jensen**, Parmod Kumar, Jeff Molitor, Madhavi Anugolu, Kenyon Hart, Steve Chiu, "Multi-Level Embedded Motor Control for Prosthesis," World Comp International Conference on Embedded Systems and Applications (ESA), pp. 37-42, 2010.
- [15] Parmod Kumar, Nikesh Joshi, **Alex N. Jensen**, Chandrasekhar Potluri, Marco P. Schoen, Steve C. Chiu, "Genetic Algorithm Running Time Optimization Using OpenMP Parallel Computing," Proceedings of the International Conference on Parallel and Distributed Processing Techniques and Applications, vol. 2, July 2010.