Royal Holloway University of London

Department of Electronic Engineering

A machine learning approach towards electromyographic control of a prosthetic

EE3000

Individual Project Report

Date of Report : 09/05/2021

Student ID : 2105110

Dr Clive Cheong Took and Dr Steve Alty

Acknowledgments

It comes with sincere joy that I can thank my supervisors; Clive Cheong Took and Steve Alty for the time and detailed guidance they have both instilled into my project. Without their constant help and support, my ability to implement, solve problems and succeed in this project would be drastically limited.

I thank the Lab technicians and staff for the hours they have spent in the laboratories so we can continue to work, extensions of access hours, ordering in components and general assistance making our projects and lives during COVID much easier.

My gratitude extends towards the department of Electronic Engineering, for allowing me to complete this project with ample access to resources, lab space and their time during a difficult year amidst COVID.

Finally, to those of you who didn't have any traditional academic impact, but in fact a greater one than you may imagine; The past year has been something memorable in all of our lives, but especially my own. I have been unfortunate enough to experience some of the deepest falls but simultaneously as a result of them, the highest peaks. The stochastic nature of life, with the ups and downs are something we cannot predict, but we can mitigate, and with that being said, I could not have completed any of the things I have in this project or in my year, without the help of the people around me this year, the new and the old, the loved and the cherished, I thank you from the bottom of my heart as any triumph I experience, is by extension your own and for that you should be proud.

Abstract

Robotic, biosensor controlled prosthesis is the solution to modern loss of limbs. The pushes in modern electronics from smaller and more powerful motors to open-source equipment for bio-sensing coupled with the leaps of intelligence we have gathered in recent history of Biomedical Engineering can lead to ingenious, cheap and futuristic solutions to a historically poorly solved problem.
The following paper delineates a software based application of surface-Electromyogram signal acquisition from an OpenBCI Ganglion device streamed to MATLAB through the Lab Streaming Layer. The signal processing in MATLAB follows a pattern recognition based control technique in order of Conditioning (Raw, Blackmann band-pass and Regular EMG Preprocessing), Data segmentation, Feature extraction and then finally Classification (5 Different models). The signals from MATLAB are output to a prosthetic design constructed in the early stages of this project, utilising servo motors that actuate in deference to the signals processed by MATLAB communicating the classified gesture through the serial port. The prosthesis's goal is to mimic finger flexion, and it can be shown that surface-Electromyogram signal's can be acquired, streamed to MATLAB, processed with our technique of choice (Three different pre-processing choices, Root-Mean-Squared Feature extraction, and the classification types of k-Nearest Neighbours, Support Vector Machines, Decision Trees and Shallow Neural Networks) and in theory output to the Arduino to actuate via the 16 bit Servo Driver.

Contents

Abbreviations

sEMG - Surface Electromyogram
sEEG - Surface Electroencephalogram
MUAP - Motor Unit Action Potential
GUI - Graphics User Interface
TCP - Transmission Control Protocol
LDA - Linear Discriminant Analysis
MAV - Mean Absolute Value
RMS - Root Mean Square
SSI - Simple Square Integral
WGN - White Gaussian Noise
MLE - Maximum Likelihood Estimation
ZC - Zero Crossing
SNR - Signal to Noise ratio
MLP - Multilayer Perceptron
PCA - Principle Component Analysis
LDA - Linear Discriminant Analysis
FSM - Finite State Machine
ANN - Artificial Neural Network
GMM - Gaussian Mixture Model
AR - Auto-regressive Model
WT - Wavelet Transform
STFT - Short time Fourier Transform
LSL - Lab Streaming Layer
kNN - k-Nearest Neighbour
SVM - Support Vector Machine
LMS - Least Mean Squared
MSE - Mean Square Error
API - Application programming interface

List of Figures

List of Equations

(22) - Chebychev distance
(23 - 26) - Least Mean Square derivation

List of Tables

1 Introduction

The world of prosthesis has constantly been evolving with the introduction of new technologies, we are now presented with the battle of creating more anthropomorphic and bio-mimetic designs for users; Tackling all kinds of issues like how to draw information from EEG to EMG, introducing complex degrees of freedom to complete a taxonomy of manoeuvres used in day to day life. Recent developments in robotic prosthesis generally use servo motors linked to wires controlling the fingers, mimicking tendons. With the prevalence of printing and complex fabrication techniques, we have the ability to create incredibly unique designs, and change countless lives.

1.1 Primary Goals

1 - Develop and Implement code that can detect specific muscle movement corresponding to each finger allowing prosthetic to mimic flexion and extension.

2 - Recreate specific predefined gestures.

3 - Undertake background reading and literature survey on electromyogram (EMG).

1.2 Secondary Goals

1 - Investigate hardware upgrades to include extra degrees of freedom.

2 - Explore addition of a wrist servo.

3 - Test the developed EMG methods on a wider range of subjects.

This project will include the production of a prosthetic hand that is controlled by 4 biological inputs from the lower forearm. This endeavour aims to prove that a functioning hand with a specified taxonomy can be produced and ideally used to mimic the real hand as much as the prosthetic design allows.

1.3 Motor Unit

In its simplest form, the motor unit is an assortment of muscle fibres working in tandem to create a contraction caused by an impulse delivered by one nerve connecting all these fibres together. The impulse is more accurately known as an action potential, this is a fast, transitory peak that reaches around 40mV (causing flexion) at its peak. The resting potential of any membrane is around -70mV (causing relaxation), this event is exclusive to muscle and neuronal movement. Action potential is the chemical transfer between channels of a membrane; during the resting potential, only the potassium channel is open and transferring from side to side, however potential increase is the transfer of Sodium down a voltage gate to the other side creating a potential difference thus increasing the voltage, and decline is the movement of Potassium back up through its own K^+ voltage gate. The strict chemistry behind neuronal stimulation in a motor unit is relatively simple or can be explained simply; the axon which ends with the synapse, releases a neurotransmitter called acetylcholine which is transferred between the Vesicles of the pre-synaptic cell to the post-synaptic cells receptor cells. [6]

1.4 Electromyogram

The term Electromyogram can be deciphered by three distinct segments of this word; electro which indicates electrical activity; myo which pertains to the Greek prefix for muscle; and gram which is a documentation of the activity. An electromyogram measures the activity of the depolarisation of a muscle during contraction and the impulse at the nerve cell. Typically, our acquisition of an sEMG signal is the result of a spatially weighted sum of electric activity relative to a large number of motor units at the surface, therefore this method pertains to the ability to understand which muscles co-ordinate what response and map appropriately.

1.5 Brain Computer Interfaces

The 'BCI' is a new implementation of electronics that ultimately connects to the brain with a relatively small processing device.[1] The device provides a method of communication through electroencephalographic signals which can be manipulated to control external devices; connecting a subjective experience to the objective world. The signals are typically run through processing and classification algorithms which decode the purpose of signal. BCI technology is being implemented all over the world, the most renowned iteration can be shown by a biotechnology company called Neuralink [2], they have shown ability to implement data extraction from a small 3,072 channel package surgically implanted to a laboratory rat. The concept of BCI's is being thoroughly used to better the lives of those afflicted with neurological diseases, prosthesis and other forms of motor issues. The technology can be taken in a noninvasive form or an invasive form as shown by the work at Neuralink; We can use sEEG signals which are evidently much more prone to cross talk and degradation due to high Signal to Noise ratio.

1.5.1 OpenBCI Ganglion

OpenBCI is by their name, an Open Brain Computer interface. The company produces an open source, cheap and research grade hardware that allows beginners to enter the field. These bio-sensing tools have been used in many forms, from scientific research to DIY projects. The company offers two main boards; Cyton and Ganglion [3]. The

Ganglion is a robust, and relatively cheap bio-sensing device. The equipment features 4 high impedance differential inputs of interest to Electromyogram work. The sampling frequency is done at 200Hz. The Ganglion can be connected via the a bluetooth dongle (BLED112) to interface with a computer, which transfers the data to a GUI developed for data processing and outsourcing. The GUI presents a variety of methods for data outputting; Serial, LSL, UDP and OSC.

1.5.2 Lab Streaming Layer

LSL is a suite of tools that is built on top of a library. The crux of the library interfaces with many different languages like C, C Sharp, C++, Python, Java and MATLAB. It has general purpose applications for data transmission and acquisition available on the laboratory network. The streaming API is explicitly useful for its streaming ability in chunks, which improves latency and throughput. The library holds a built in clock which allows a unique time stamp per transmitted sample/chunk, whilst achieving a sub-millisecond accuracy on a local stream [11]. Notable features of its reliability are using TCP, Automatic failure recovery, Buffering at both receiver and transmitter end and is type safe whilst allowing type conversion [11].

1.6 Electrode Placement

Electrode placement is instrumental in the success of most iterations a prosthesis development, forming the foundations that are used in signal acquisition and upwards.

1.6.1 Anatomy of the forearm

The forearm muscles can be segmented into two distinct muscle groups [12], Muscles responsible for wrist movement and hand movement. The muscles can further be split into 2 further distinctions; Flexor and Extensor muscles. The large muscles in the forearm seem to be located on the proximal end and narrow down, culminating to the tendons at the wrist. Anterior flexors are largely innervated by the median nerve and the posterior extensors are typically activated by the radial nerve. Finger movement is a result of the forearm muscles causing tendon movement. There are a range of muscles that can be directly correlated for finger movement as shown by [12, 9] along with a summary table below. It would then be the reasonable step to deduce which muscles to use by experimentation in accordance to the array of electrodes available.

1.6.2 Electrode placement methods

Attempting to limit the channel usage to single figures due to OpenBCI's Ganglion only being a 4 channel system presents the only restraining task in this stage. It has been found by Tang et al's [10] proposition of a 6 Electrode array (Figure 3) achieving a 98% success rate in detecting 11 distinct gesture types with all of the individual finger movements as a subset using a new cascaded-structure classifier. This iteration could show high utility as Tang et al has utilised a redundant channel which provides supplementary information but can be removed, it may reduce the accuracy of gesture types however the scope of this paper is covered by less than 11 gestures as it currently stands. The arrangement can be modified to use a 4 channel system by reducing the pair of electrodes measuring Extensor Pollicis (Longus and Brevis) impulses to just one electrode pair. Now in search for one final reduction, there are again two pairs on the Extensor digitorum, this could ideally be reduced to one pair. Investigations into accuracy would need to be completed

Muscle	Muscle Depth and Location	Type of Movement
Flexor Pollicis Longus	Deep, Anterior	Thumb Flexion
Flexor Carpi Ulnaris	Superficial, Anterior	Finger Flexion
Flexor Digitorum Superficialis	Superficial, Anterior	Finger Flexion (2-5)
Flexor Digitorum Profundus	Deep, Anterior	Finger Flexion
Extensor Indicis	Deep, Posterior	Finger Extension
Extensor Carpi Radialis Longus	Superficial, Posterior	Wrist Ext and Ab.
Extensor Carpi Radialis Brevis	Superficial, Posterior	Wrist Ext and Ab.
Extensor Digitorum	Superficial, Posterior	Finger Ext. (Prime Mover)
Abductor Pollicis Longus	Deep, Posterior	Thumb Ab and Ext
Extensor Pollicis Brevis	Deep, Posterior	Thumb Ext.
Extensor Pollicis Longus	Deep, Posterior	Thumb Ext.

Table 1: Muscles related to finger movement [12,9]

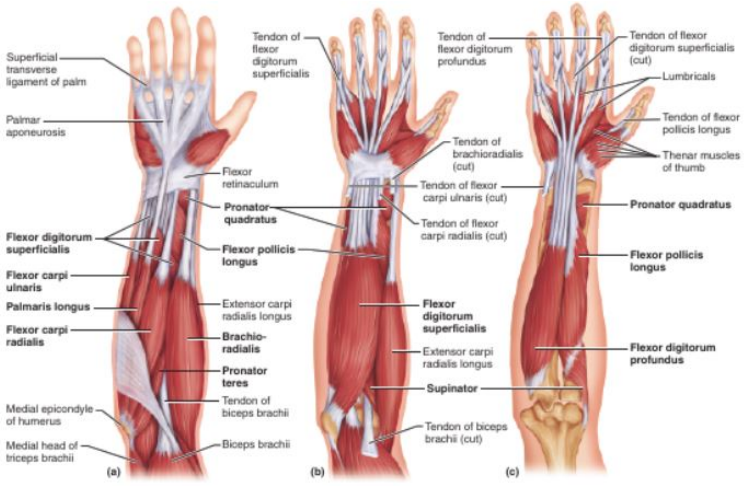


Figure 1: Muscles of the anterior forearm acting on wrists and fingers [12]

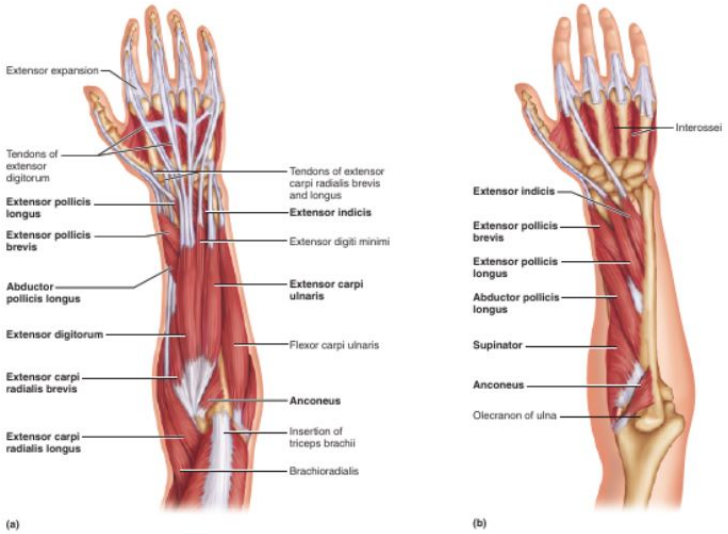


Figure 2: Muscles of the posterior forearm acting on wrists and fingers [12]

however it seems like a possibly fruitful avenue for electrode placement and utilises the major muscles spoken off in the previous section. The electrodes Tang et al used, are 1cm in diameter which is relatively large, and has the electrodes arranged on a ring for both slim and robust forearms on the posterior. It is noteworthy that this study utilises LDA to distinguish the different gesture types.

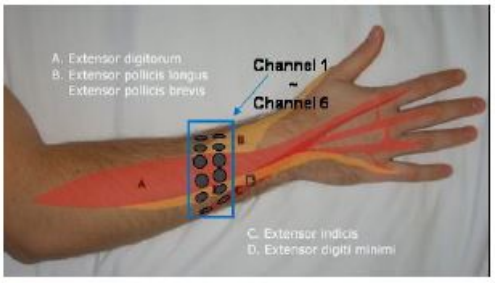


Figure 3: Electrode orientation completed by Tang et al [10]

The following Figures 4 and 5, illustrate the work completed by You et al, in a similar electrode configuration but on the anterior of the forearm achieving a 97.75% accuracy in detection of flexion of all fingers and three multi-finger motions. Figure 5 shows the proposed taxonomy which was achieved and seems to be the types of results this paper is looking to achieve. The paper indicates a 3 second acquisition timing where the first 1.5 seconds were at rest waiting for an acoustic alarm to signal flexion. After 3 seconds elapsed, the EMG recording ended. The subjects underwent eight distinct flexion motions and three multi-finger motions as shown by Figure 5. Adequate rest was allowed between result acquisition to allow minimisation of issues such as muscle fatigue and cramp. In this research, statistical methods are used to model the EMG (Entropy) and subsequently the characteristics of the signal.

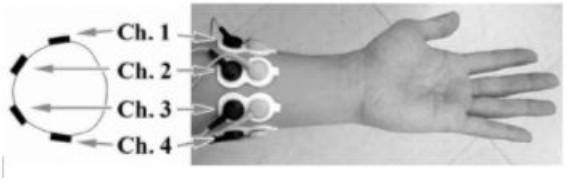


Figure 4: Proposed electrode placement on a 4 Channel system by You et al [13]

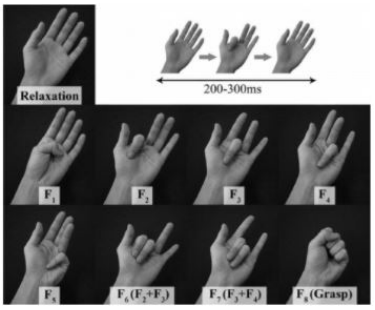


Figure 5: Proposed taxonomy for study by You et al [13]

1.7 EMG Signal Extraction and Conditioning

Capturing muscle activity is typically illustrated in the form of an Electromyogram; measuring the potential generated by the sum of MUAP (Motor Unit Action Potential) at any given location. The signals can be invasive or non invasive in acquisition methods, typically in the healthcare profession surface electromyography is utilised to avoid any unnecessary discomfort, however more bespoke and isolated experiences can include invasive techniques. Electrodes are connected to a patients muscles and a device will engage in signal conditioning; usually amplification, filtering. Filtering can be required due to events such as motion artifacts and attenuating noise (e.g Mains power line interference, within the UK the attenuation would occur at 50Hz. typically Notch filters are used for the frequency reduction). The data can then be fed to a display which outputs raw muscle activity or in addition signal processing can be completed that allows for specific feature extraction and analysis.

1.8 EMG Signal Processing

The aforementioned stage can be broken down by three distinct processes in our pattern recognition based modelling; Data Segmentation, Feature extraction and Classification [7]. The alternative is using non pattern recognition models but this is outside the scope of our efforts.

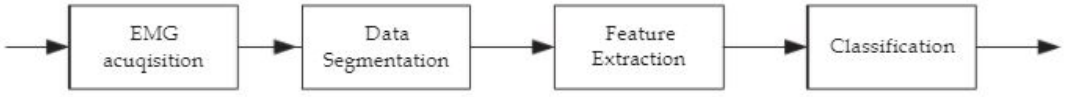


Figure 6: Pattern recognition based control used in [7]

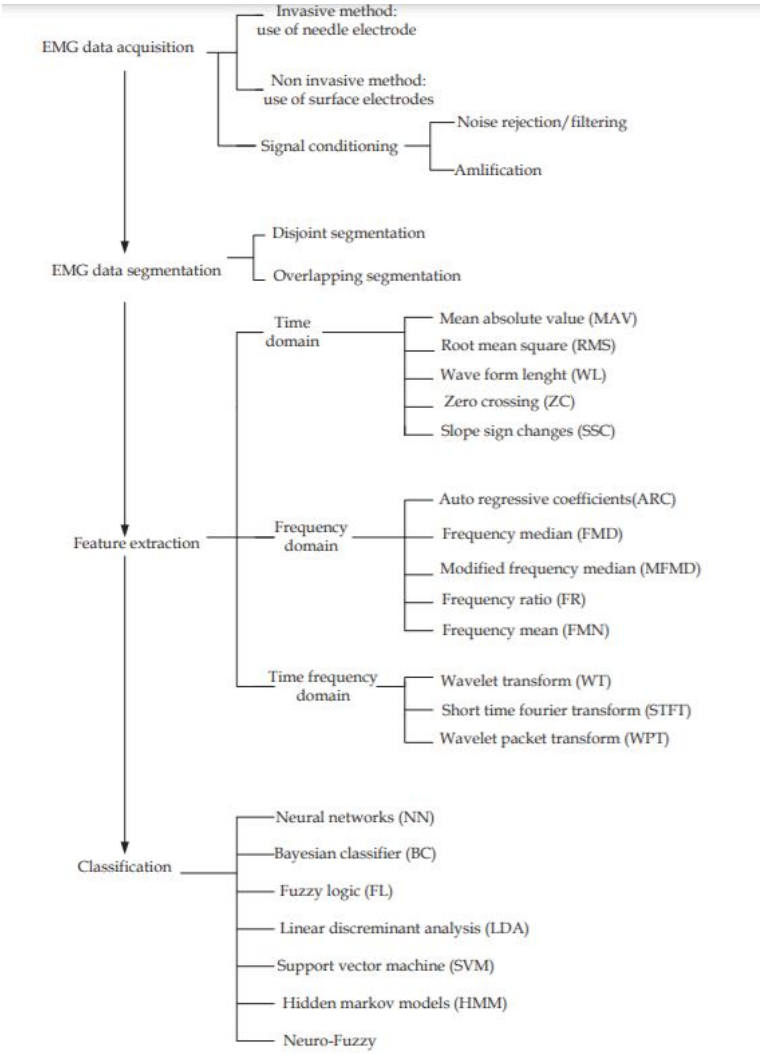


Figure 7: Steps of recognition based processing [7]

1.8.1 Data Segmentation

All EMG signals exist between two distinct states; Transient and Steady state. Transient state involves a muscle transitioning from rest to contraction and constant contraction is visible under steady state. For optimal response from data segmentation, the time slot should be around 300ms or less as shown by You et al and 500ms by Tang et al, this includes any processing time for robotic control output. Data segmentation is typically dispersed into two techniques: Overlapping segmentation and disjointed which takes advantage of already established lengths for feature extraction [15]. With this technique, the signal is now interpreted as steady state. Overlapping segmentation, the most recent sample slides over the previous with a time smaller than the segment length. Adjacent/Disjointed is where segments do not overlap, the signals are divided equally and feature extraction is completed [9].

1.8.2 Feature Extraction

Feature extraction is imperative for deciphering an EMG signal and outputting it into a classifier, therefore the success of pattern recognition depends on feature extraction. Therefore its necessary to compute, pre-established features of the EMG to be used in a classifier or processing algorithm to improve the control system. Feature extraction can be done in Time domain, Frequency or in some cases Time-Frequency (Discrete Wavelet transform). For the purposes of this, we will focus on time domain extraction more than the rest. Typically, when choosing a feature extraction method we should relate it to the utility of the control system, in the case of this experiment it is the control of a prosthesis, therefore features like muscle fatigue, contraction time, power, energy etc. There are an array of methods in Feature extraction, ranging from Integrated EMG, Mean Absolute Value (Similar to Average Rectified), Variance, Root Mean Squared, Waveform Length, Willison Amplitude and many others.

Integrated EMG is a summation of absolute values from the signal. Typically this feature is a good approximation for the envelope of the signal.

$$IEMG = \sum_{n=1}^{N} \mid x_n \mid \qquad [4] \tag{1}$$

RMS of an Electromyogram is to determine the muscle fatigue in the signal and provides the maximum likelihood estimation of amplitude in a constant force. The RMS is modelled as an amplitude modulated Gaussian random process [4] and can be shown by the mathematical demonstration below.

$$RMS = \sqrt{\frac{1}{N} \sum_{n=1}^{N} x_n^2} \qquad [4] \tag{2}$$

Mean absolute value is an easy method of detecting muscle contraction, it is a widely applied feature extraction method. There are different variations of the MAV, one including a window for smoothing.

$$MAV = \frac{1}{N} \sum_{n=1}^{N} \mid x_n \mid \qquad [4] \tag{3}$$

Simple Square Integral takes advantage of the energy of an EMG signal.

$$SSI = \sum_{n=1}^{N} \mid x_n \mid^2 \qquad [4] \tag{4}$$

Variance of an EMG is finding the power of the signal. Typically VAR is the mean of the square deviation but that would be too close to zero so we use the equation below.

$$VAR = \frac{1}{N-1} \sum_{n=1}^{N} x_n^2 \qquad [4] \tag{5}$$

Willison Amplitude is using the MUAP and muscle contraction. It thresholds two segments and counts the number of times the difference between the segments exceeds

a threshold. The equation follows the conditions that f(x) is 1 if x $\geq$ threshold, or 0 otherwise.

$$WAMP = \frac{1}{N}\sum_{n=1}^{N} f(\mid x_n - x_{n+1} \mid) \qquad [4] \qquad (6)$$

Information Entropy can be used to exemplify the information which signals contain. The amplitude is a fundamental quality and increases with force on muscle. Entropy is used to show the dynamic nature of a signal and the information it contains. There are different types, Shannon, Spectral, SVD etc. You et al have utilised Shannon Entropy in their iteration.

$$H(X) = \sum_{m=1}^{M} p(x_m) log_2 \frac{1}{p(x_m)} \qquad [13] \qquad (7)$$

X is the discrete random variable, $p(x_m)$ is the probability of X and when X is equivalent to x_m and the sum of probability of x_m is equal to 1. This model is a probabilistic model in the work by You et al, they obtain probability density functions on each finger motion.

In a survey conducted by Oskoei and Hu [15], MAV and RMS are and have been highly used in EMG feature extraction over the brief history of electronic prosthesis. To recap, MAV and RMS are time domain features and RMS provides a maximum likelihood estimation relative to the amplitude where there is a constant force and non-fatiguing contraction when modelled as a Gaussian. This study indicates that the MAV provides the MLE, as long as it is modelled as a Laplacian random process. The SNR is 32% lower than a Gaussian based model, therefore a Gaussian and by extension the RMS suits better at feature extraction for high levels of contraction whilst a Laplacian model fits for low contraction and tired muscles. The MAV in this case could be useful for addressing motor issues in a supportive prosthesis however RMS in this case would be deal for mimicking high level muscle movement. Clancy et al [16] has indicated in his work that for constant force, non-fatiguing contractions, on average the Gaussian model is much more suitable. The study concludes with the idea that they are both suitable choices and interestingly has shown that multi-channel systems reduce the variance in amplitude features.

Table 2 shows a collection of 5 EMG applications for similar purposes; the classifier and the feature extraction method are of greatest use for our review. Hudgins et al shows the use of an ANN, paired with this the feature extraction entertains Waveform length, Zero Crossing and Mean Absolute Value whilst finally choosing MAV. Classification for the 9 non amputee subjects averaged 91.2% with a standard deviation of 5.6%, while the amputee subjects averaged 85.5% with a standard deviation of 9.8%. Huang et al have looked at the classification rate when using an AV and RMS featured set; the results show that GMM's average classification error is optimal at 3.72%, and MLP and LDA sit at 4.62% and 4.42% respectively. Carrozza et al utilises a state system, a rudimentary application that uses two gestures of open and closed relative to the high-low states of the system. This application could be advanced in this prosthetic, with 4 channels, an attempt could be made at, at least 4 states. Lamounier et al introduces an interesting feature extraction method; Auto-regressive model allowing their neural network to model the coefficients easily. AR can measure the power spectrum of the signal and then neural network can be trained with a small set of data that clearly yields acceptable results. AR is advantageous in this as electrode placement is a nonissue, as well as the information

going into the classifier is reduced and by extension processing time. The mean success rate of classification was 95%.

Authors	Application	Gestures	Channel	Classifier	Feature
Hudgins et al. [17]	Upper Limb Prosthetic	4	2	ANN	MAV
Englehart et al. [18]	Upper Limb Prosthetic	6	4	PCA / LDA	STFT, WT, WPT
Huang et al. [19]	Upper Limb Prosthetic	6	4	GMM / MV	RMS, AR
Carrozza et al. [20]	Prosthetic hand	2	2	FSM	n/a
Lamounier et al. [21]	Training patients	4	5	MLP NN	AR

Table 2: A table of prosthetic works revised from [15] with focus on features

1.8.3 Classification

As explained before, features must become classified in order for the control method to function as required. Classification can help alleviate any undesirable changes in EMG signal like the introduction of perspiration, posture changes, electrode position etc leading to variation in the signal and feature detected. The importance here is classifying the unique feature throughout the time required with external influences coming into effect. Speed is also another important dimension to classification therefore a form of training could be useful in quicker, more informed response. There are a myriad of classification types as shown by Figure 7. The most traditional classifiers are statistical in nature like LDA, K-Nearest, Bayesian Classifiers etc. These are particularly alluring due to its quicker computational time as well as easy implementation in robotic applications. The very clear downfall of statistical classifiers are that as gesture types increase, the accuracy seems to diminish and overlaps (mis-classifications) along with the issue of 'cross talk' come into play at a greater and more difficult level. This is where misclassifications occur, in order to avoid this using a statistical method, is to limit the amount of gestures used. Tang et al [10] has shown a 89% success rate in an 11 gesture identification system for hand motion classification using a Linear Discriminant Method.

Prosthetic	Classification
Exoskeleton Robot [22]	Using MAV and Neuro-fuzzy controllers are used to aid in posture changes
W-EXOS [23]	Neuro-fuzzy controller, rules are used to determine intention
Saga Arm [24]	EMG based fuzzy controller along with kinematic based control through sensors
Manu Hand [25]	3 Level EMG signals (State Machine)
Finger Prosthesis [26]	Post processing classification using Most Vote, LIB-Support Vector Machine and kNN.
EMG based Robotic Hand [27]	ANN classification

Table 3: A table of prosthetic works with focus on classification

It is evident by the Tables 2 and 3 that attempts of classifying EMG signals and prosthesis via pattern recognition techniques are very clearly much more prevalent and useful. There seems to be an emphasis on Neuro-fuzzy models, but statistical analyses like LDA and

Perceptron have been of high utility in classification. It seems by the literature survey, the classification is highly dependant on the feature chosen.

Support Vector Machines are binary classification methods that use the training data to find optimal hyper-planes to differentiate classes. SVM's can be either linear or non-linear. The underlying idea with SVM's is to create linear/non-linear dissecting lines at optimal distances between data sets in order to instantiate what class they belong too.

Linear SVM, can be shown by the figure below. There are two canonical hyper-planes: $H^{\pm}$, which are shown by the dashed lines moving through the furthest data points on the outer end of the classes. **w** is the weight vector that is the optimal H. Mathematically this is shown to be;

$$H_{+} : \langle w, x \rangle + b = 1 \tag{8}$$

$$H_{-} : \langle w, x \rangle + b = -1 \tag{9}$$

$$\text{Optimal H} : \langle w, x \rangle + b = 0 \tag{10}$$

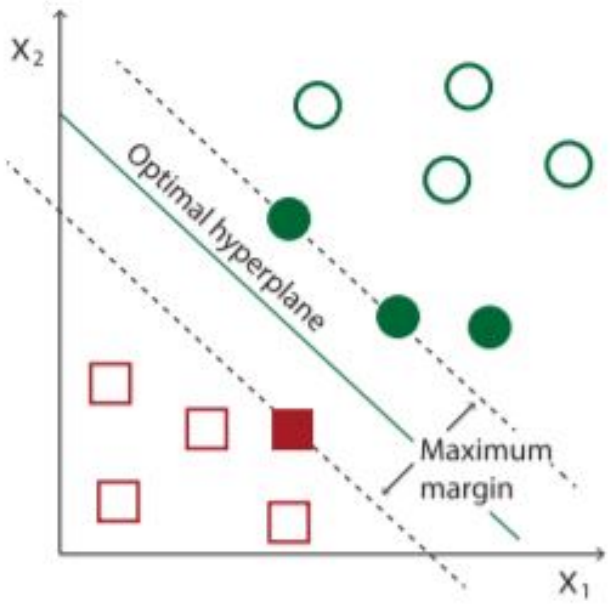


Figure 8: Linear SVM Model [8]

The above equations and figure, aims to satisfy for maximum margin therefore generalises for the data at hand. The mathematical derivation shown by Gutiérrez et al [28] and used in this explanation [29], explains the method to optimise the hyper-plane for maximum margin using Lagrangian methods.

$$\gamma = \frac{2}{\| w \|} \tag{11}$$

$$\min_{w,b} \frac{1}{2} \|w\|^2 + C \sum_{i=1}^{m} \xi_i \tag{12}$$

$$y_i(w \cdot x_i + b) \geq 1 - \xi_i, \xi_i \geq 0, i = 1, ...m \tag{13}$$

The above minimisation task is the inverse of what we understand to be the distance between two support vectors (11). We minimise the reciprocal of (11) which is shown as

(12) along with something we can discuss later as the 'slack' variable. (13) represents the canonical support vectors which we are using as a subject. Creating a initial 'primal' form, taking the partial derivative and finally a 'dual form' [29] is simple and goes as follows:

$$L(w, b, a) = \frac{1}{2}\langle w, w\rangle - \sum_{i=1}^{m} a_i[y_i(\langle w, x_i\rangle + b) - 1] \tag{14}$$

Equation (14) is the Lagrangian taking weight vector (**w**) , bias (**b**) and Lagrangian multiplier (**a**) as coefficients. We then simply negate (12) and (13) from each other with the multiplied Langrangian which results in (14).

$$\frac{\partial L(w, b, a)}{\partial w} = w - \sum_{i=1}^{m} y_i a_i x_i = 0 \tag{15}$$

$$w = \sum_{i=1}^{m} y_i a_i x_i \tag{16}$$

The partial derivatives are then taken, one half shown by (15) which results in the weight vector solution at (16). The other partial explains that if we completed the addition of all Lagrangian numbers it equals 0; What this teaches us is that the data point it is probing is not zero (therefore not a support vector) and by extension of no interest to us.

$$L(w, b, a) = \frac{1}{2}\sum_{i,j=1}^{m} y_i y_j a_i a_j \langle x_i x_j\rangle - \sum_{i,j=1}^{m} y_i y_j a_i a_j + \sum_{i=1}^{m} a_i \tag{17}$$

$$L(w, b, a) = \sum_{i=1}^{m} a_i - \frac{1}{2}\sum_{i,j=1}^{m} y_i y_j a_i a_j \langle x_i x_j\rangle \tag{18}$$

Creating a double summation and simplifying as shown by [29] we can produce the final equation (18).

$$f(x_j) = sgn(\sum_{i=1}^{nSVs} y_i a_i \langle x_i, x_j\rangle + b) \tag{19}$$

Finally, we end up creating the function an SVM model would use. The function shows the test data output, which is deduced by signum allowing for only a ± 1 answer as most machine learning models need. The summation accounts for number of support vectors, and uses the weight vector ($y_i a_i$), multiplied with the data point being checked ($\langle x_i, x_j\rangle$) plus bias.

k-Nearest Neighbour

The k-Nearest Neighbour is a supervised machine learning technique developed for regression and classification but mostly used for classing data. The model stores all data from training and classifies on the proximity of point to point. The kNN is non-parametric which means it does not delve deeper into any extra information hidden like a neural network might. The production of a kNN is simple, begin by defining the amount of neighbours you require for your data, then calculating the different types of distances, from Euclidean, Chebyshev, Minkowski, Spearman, Hamming and many others. All of these use a unique method of finding the distance between two vectors which is uniquely suited to the task we are presented with in a kNN. Below are a few of the formulas.

One dimensional Euclidean:

$$d(p, q) = |p - q| \qquad [31] \tag{20}$$

Two dimensional Euclidean:

$$d(p,q) = \left|\sqrt{(p_1 - q_1)^2 - (q_2 - p_2)^2}\right| \qquad [32] \qquad (21)$$

Chebyshev distance:

$$D_{Chebyshev} = max(|x_2 - x_1|, |y_2 - y_1|) \qquad (22)$$

The next step is finding the neighbours and enumerating data points enclosed the by neighbours making each category/class. This is now a completed model, it can then be used to begin predicting where other data points lie when introducing more co-ordinates. The kNN is very simple to code and it is extremely useful when classifying data that we have little idea about its relationships or positioning. However the computational power is high as finding the distance for every point can be consuming, further as complexity of distance type increases, so does processing type.

Neural Networks - [33]
Feed-forward Neural networks are machine learning models that behave similarly to the brain's neural network, hence the name. Feed-forward NN's as the name suggests have no feed back, otherwise it would be called a recurrent neural network. The traditional layout of a neural network is layered, either single or multi.

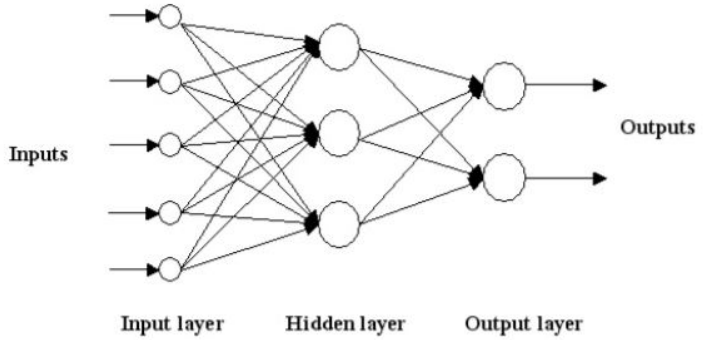


Figure 9: Multi layer feed forward neural network [33]

The figure above shows a multi-layer with a hidden layer inside. The hidden neurons is to engage in the decoding of the 'hidden' information in the form of statistical analysis. The figure above is an example of a fully connected NN as all of the nodes are connected to their maximum capability.

Decision Trees
Decision trees are a very basic and sometimes binary classification methods. There are a few key things to understand about Decision trees; they begin with a Root node, it exemplifies all of the data. This root node is then split into a decision node i.e. Classing a fish by colour (Root node), decision node would be Is it Green, the terminal node to follow would be yes and the subsequent Decision node would be no. This then repeats on every colour until the right one is selected. Each division mediates like a test for a characteristic, this is recursive and repeats. Decision trees use multiple algorithms but for this classification method it is usually CART (Classification and Regression Tree). There

are attribute selection measures which range from Entropy, Information gain, Gini Index, Gain Ratio and more. These are used as, if the data has X attributes, which do you choose for the root. The attribute selection is solved by the criteria above, calculating values for every attribute, these values are used to place a correct hierarchy or root.

1.8.4 Non Pattern recognition Model

Within this model, accuracy is much lower and can suit the initial goals of this effort (Simple individual finger flexion) however creating complex, bio-mimetic movement is unlikely. The main breakdown of non pattern recognition training is known as Proportional Control, Threshold control and others [7]. These are produced by a relatively easy ON/OFF control, there are not a large amount of these implementations due to its clear low level accuracy and limitations. The FSM examples of work produced by [25, 20] suggest that the utility is very low and only allows rudimentary logical implementation which isn't as dynamic and accurate as learning techniques, however from the works presented, it seems a successful fail safe method considering the application we have uses 4 channels, which is 5 pieces of information; Steady state - Rest, Single finger flexion caused by 4 channels and combinatorial logic to make more gestures, this utilises an ON/OFF control system.

1.9 Actuation

Actuation refers to the ability to move the servos and in this case caused by the signals processed from the EMG. MATLAB is a tool that directly interfaces with Arduino through serial and can be taken advantage of in many forms. The practical advantage of MATLAB to Arduino actuation is the devices produced by third parties like the Adafruit Servo Shield or 16 bit servo driver for Arduino. The Servo shield can control 2 servos per board and are stack-able but not for the servos but DC motors therefore you can in theory control as many motors as you want. This board/chip uses I2C 7-bit addresses between 0x60-0x80, select-able with jumpers [14]. Using the inbuilt MATLAB compiler, we can circumvent the need to use the Arduino IDE as alluded to before with the Arduino support package in MATLAB.
Alternative methods and one which ideally should be entertained, could be using the 16 bit PWM Servo Driver by Adafruit, programmed through an Arduino sharing data through the GPIO pins, and receiving the MATLAB data through Serial, only listening for the data. The servo driver uses I2C with only 2 pins. The servo shield has the capability to be stacked up to 62 times therefore controlling up to 992 servos. Finally, the use of a Raspberry Pi microcomputer could be useful for circumventing the use of an Arduino and MATLAB in one package.

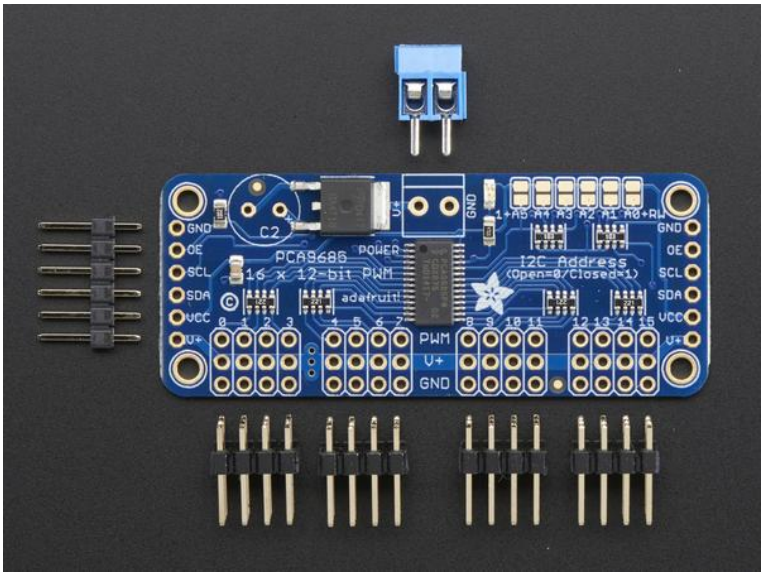


Figure 10: Adafruit 16 Channel Shield [14]

2 Methods

2.1 Prosthetic Preliminary Gesture Testing

Taking into account the works by You et al, the proposed taxonomy of grips will be based upon Figure 5 but inverted as Figure 10 and 11 shows.

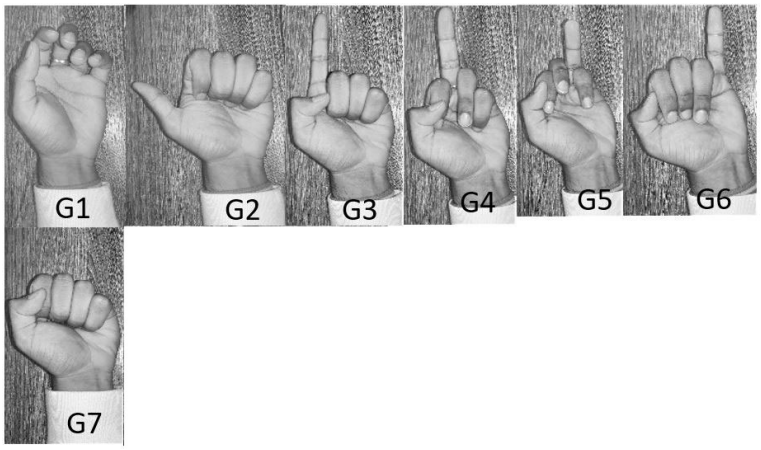


Figure 11: Gestures proposed to be modelled

The motivation behind these basic gestures completion allows me to assess how accurate my signal processing is on a basic level; if the single finger flexion can be reproduced then most gestures can be reproduced later with more training. Resting can be labelled as G1 and G2 as Thumb flexion all the way to G6 for pinky, further works could be done to assess whether a combinatorial nature could allow G2 + G3 = Peace gesture etc. Using the Arduino IDE, the defined gestures have been set up and it can be shown they are clearly reproducible providing the correct stimuli. The Gestures were programmed using the 5 Servo configuration controlled by an Adafruit 16 Channel PWM Servo driver.

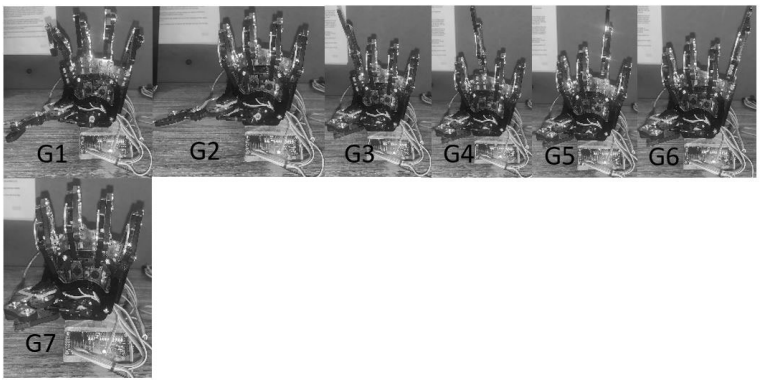


Figure 12: Prosthetic mimicking proposed gestures

Code	Gesture
G1	Relaxed
G2	Thumb Flexion
G3	Index Flexion
G4	Middle Flexion
G5	Ring Flexion
G6	Pinky Flexion
G7	Grasp

Table 4: Gestures code relative to its action

2.2 OpenBCI to MATLAB

The method of this section will include electrode placement, signal conditioning and transmission. This is a simple stage and will utilise the methods proposed by You et al for electrode placement; essentially four equally spaced differential electrodes across the anterior of the lower forearm. EMG signals should then be extracted by the BCI and conditioned with the notch filter and band pass in the GUI or raw for other means of pre-processing or appropriately readying the signal by other means, ready for transmission. Finally, transmission can be completed through the networking tool of the GUI. The tool allows for a few types of data transfers but in this effort, it should be streamed via LSL. MATLAB script should then be run to pick up **chunked** or **single** packets of data. Along with this, the appropriate code/libraries will be used to enable successful receipt of data.

2.3 MATLAB Signal Processing

Signal processing follows a delicate method as shown by Figure 7. We can begin with data segmentation. With these proceedings, disjointed/adjacent segmentation should be utilised sparing any unnecessary overlap of data. Once this is completed to a sufficient standard, Feature extraction of a suitable method. Time domain analysis is ideal as we have spoken about in the background research where the most abundant used are MAV and RMS. In this iteration, RMS shall suffice where we can observe the muscle fatigue from this feature. Classification will follow, this can be completed through the classification learner application within MATLAB or hard coded.Training data will be required in order to train the model, 100 of each finger flexion should suffice for this application with 3 second intervals between each, this can further then be used to test if input randomly. The appropriate classification should be able to recognise with high accuracy with speed,

the best routes for this type of data would be those like the kNN, SVM in theory. This method can be fulfilled with the appropriate testing of classification.

2.3.1 Classification Algorithm

In order to produce a classification algorithm, the most important stage is to acquire training data. The training data should be aquired with the same electrode configuration as live data of course, it should hold 100 of every gesture type with a relatively constant 0 phase for around 1-3 seconds between each gesture execution. Once held, pre-processed accordingly and fed into classification algorithm with a 70/30 holdout validation method.

2.4 MATLAB to Arduino

The final stage should utilise the serial ports for communication; by writing the classification label from MATLAB's newly classed, live data, the Arduino should listen for the numbers and actuate with reference to those labels as shown by Figure 11 and 12. The Arduino will ideally use the 16 bit Servo Driver, backup option will be the Arduino Motor Shield and finally the whole project could be shifted from Arduino Mega focused to Rasberry Pi.

3 Project Experimentation and Results

3.1 System Architecture

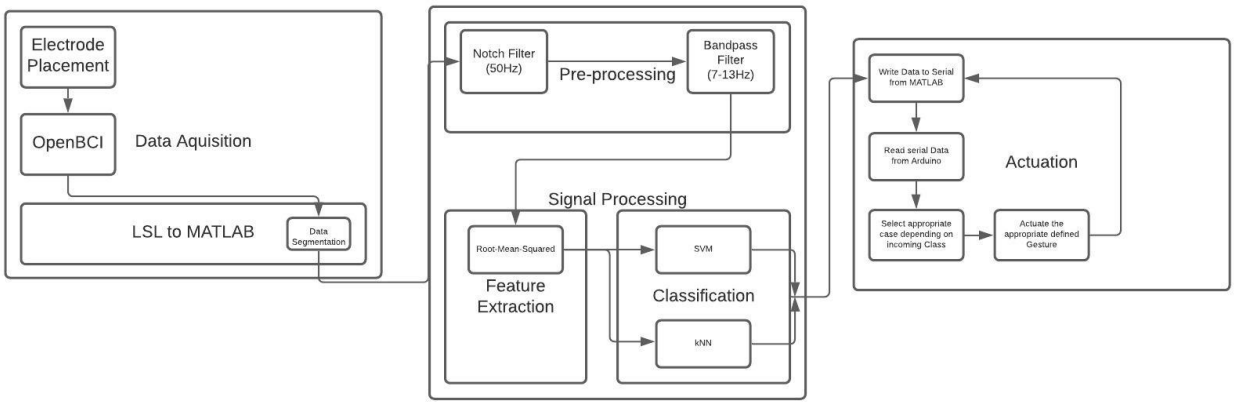


Figure 13: Finalised System Design

3.2 Prosthetic Assembly

The prosthetic in use has been purchased and assembled in the early stages of this project. The focus of this paper is the signal processing in order to move a prosthetic (Servo specifically) with muscle movement therefore this prosthetic has sole use for better exemplification of my algorithm. The prosthetic uses 5 servos for each finger and was purchased on eBay therefore can not be properly reference as the origins are unknown. Figure 12 shows the fully assembled prosthetic with all gesture motions.

3.3 Signal Acquisition

Signal acquisition was completed with an array of 8 Electrodes connected to the mid to upper anterior forearm muscles overlapping the Flexor digitorum superficiallis, Flexor Pollicis longus and the Flexor Carpi Ulnaris. On the posterior, the Extensor Pollicis Brevis. The array matches the configuration shown by You et al as spoken of in the background research along with a final reference electrode attached to the lower bicep just above the Brachiallis as a reference.

Figure 14: Test for Finger flexion

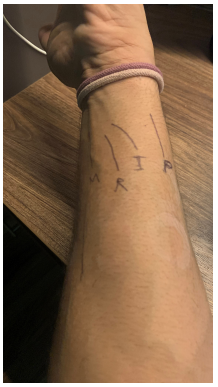


Figure 15: Test for Finger flexion

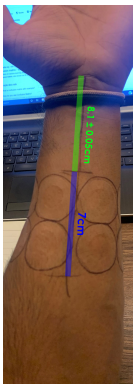


Figure 16: Guide for Electrode Placement

Personal tests were conducted to see where the muscles contract for each finger flexion as shown below, which seemed to justify the electrode placement shown in Figures 14 and 15. Figure 16 shows the guide markings I have used for my own forearm to place the electrodes, this takes into account the placement used by You et al.

Figure 17: Electrode Pad Placement: Angle 1

Figure 18: Electrode Pad Placement: Angle 2

Figure 19: Electrode Pad Placement: Angle 3

Figures 20 and 21 represent the placed electrodes on the guides as spoken of before. This configuration has and should be used for all data acquisition from training data to live data, otherwise the experiment is not consistent.

3.4 OpenBCI to MATLAB

Streaming data from the BCI to MATLAB can be done in a few ways, the chosen method is currently using the Lab Streaming Layer, a system useful for synchronizing streaming data for live analysis and with our objective of live EMG analysis and output, the method is suitable. Explicitly for training data which comes from the text file the GUI saves but

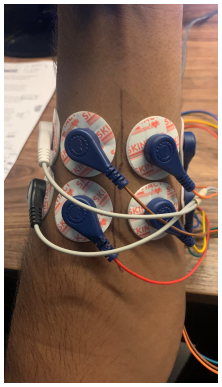


Figure 20: Electrode placement with Electrodes on

Figure 21: Reference Electrode

when sent via LSL it does allow for pre-processing within the GUI, the GUI **does not** allow for signal conditioning before transmission despite what it looks like; I have set the GUI with a 50Hz Notch filter for mains noise attenuation and a 7-13Hz Band pass filter to remove any unnecessary noise or artifacts which will be reciprocated in the MATLAB code for the classification training data but as we can see the Filters button is on which allows for filtered stream data. The representation below shows time series data with a windowed size of 5 seconds (This is only the view we have, observing the first 5 seconds of data that's coming from the EMG). The EMG widget on the top right corner is redundant as we are streaming specifically the time stream data as shown by the networking widget in the bottom right hand corner. The networking widget clearly shows that time series is being sent in Stream 1.

Figure 22: OpenBCI GUI when transmitting synthetic data via LSL

Once the data has been acquired and sent via LSL, the code must read the data in the pipeline and initiate the appropriate data segmentation and pre-processing. Segmentation is completed by the use of chunked data, the GUI picks up a chunk in the pipeline which

contains 10 data points, these are refreshed at a 200Hz rate, matching the sampling from the EMG. Looking at Appendix H which shows the code being used, we can see the LSL code for chunked data that is given to us by the LSL library as an example edited to suit the task of EMG analysis.
Beginning where the library is loaded in. Moving onto the resolution of the specific stream, the useful line which resolves the properties of the EMG stream so it can find the right one if there are multiple which there is not in this experiment.
An 'inlet' is created which can be though of as the pipeline spoken before. The inlet is then associated to the command 'pullchunk' which is a command written to extract the chunk in the pipeline. The square brackets put the corresponding chunk to chunk and stamp which is a unique LSL timestamp which will be irrelevant for our results. The for loop on which states 1:length of stamps is an infinite loop which allows for reading all the data in the inlet until it is nothing.
The chunk is stored in an empty **data** variable which changes size on each iteration as pre-allocation can't be done due to the undefined number of samples we require. The code has been provided for data to be placed in a table which typically works specifically for inputting in the machine learning model but a regular array with the proper headers (headers the training data uses) should suffice. The data is horizontally concatenated every iteration and transposed as the chunk moves horizontally but the data is better manipulated when it moves vertically down.
Preprocessing begins with a real, third order, non-linear Impulse-Invariant notch filter. Beginning with ω_o which is the normalised notch frequency; this is done by division of the required frequency attenuation against the Nyquist frequency. The next is figuring out the bandwidth against a trial and error found Q factor of 15, this is shown in the figures below for which forced the most frequency attenuation as the channels 3 and 4 were not very susceptible to the notch. The notch filter coefficients **b,a** are created by the DSP Package in MATLAB using the function **iirnotch(ω_o,b_o)**. The corner frequency is at roughly 0.48ν as shown by Figure 23. The alternate use of pre-processing uses a Blackmann Band-pass filter due to its ability to refine the data in a method similar to the pre-processing spoken of above.

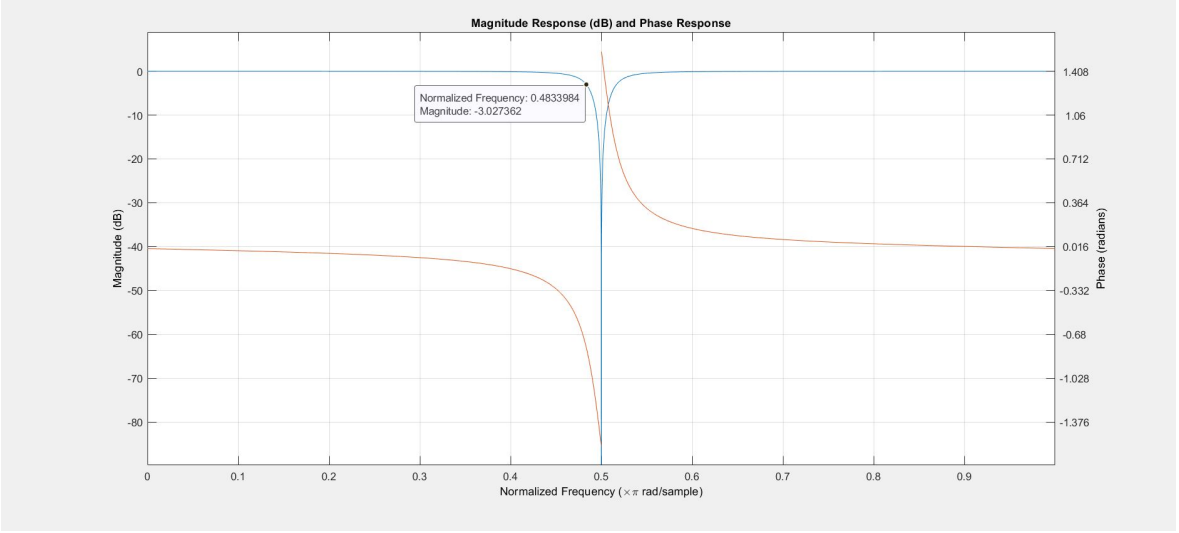


Figure 23: Notch Filter Phase and Magnitude Response

The raw data can then be filtered using simple filter command and the coefficients.

Finally, the band pass filter is a simple function where the Notch Filtered data is used, alongside the instantiated band-pass required (7-13Hz) and finally the sampling frequency.

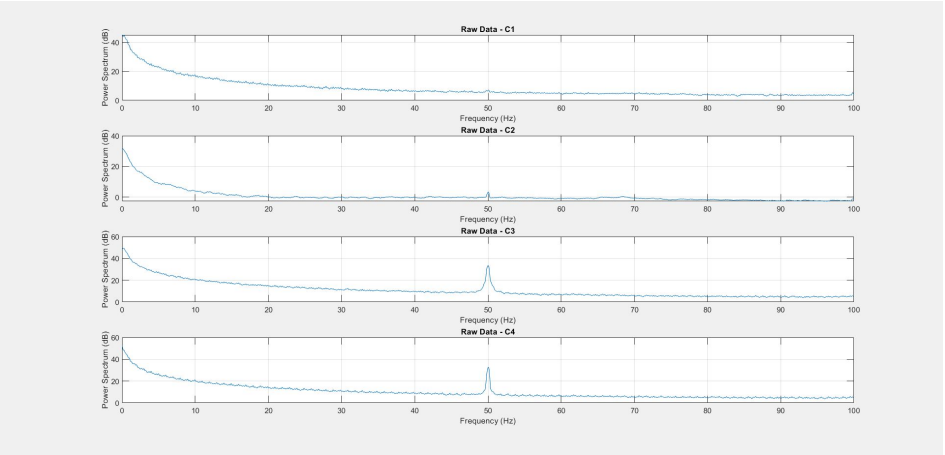


Figure 24: Power Spectrum of the raw data

It can be seen in Figure 24 that the raw data shows clear 50Hz noise and unfiltered as we require the 7-13Hz bandwidth. This can be used as a reference for the coming filtering to see changes.

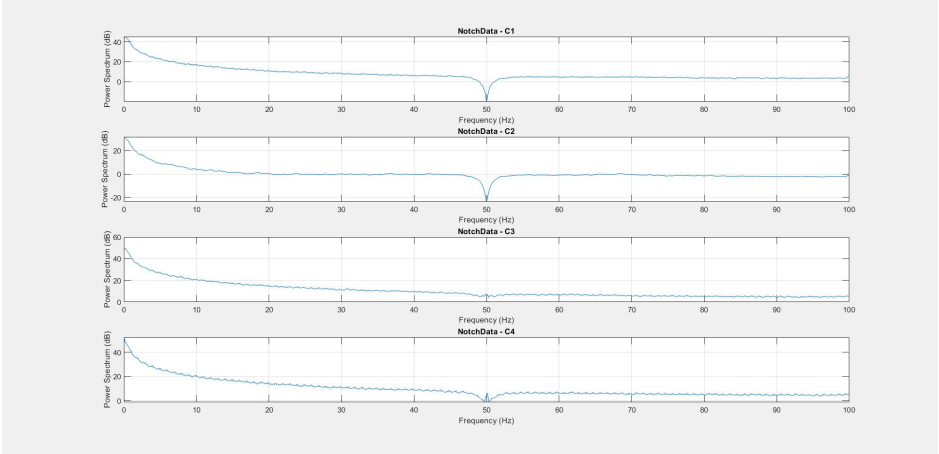


Figure 25: Power Spectrum of the Notch Filtered data

Figure 25 represents the Notch filtered data; Channel 1 and 2 seem to work sufficiently however after extensive probing the best filter was found with a Q factor of 15. This resulted in the channels 3 and 4 having little to no decline in the negative domain. Nonetheless compared to Figure 24, the mains noise is attenuated.

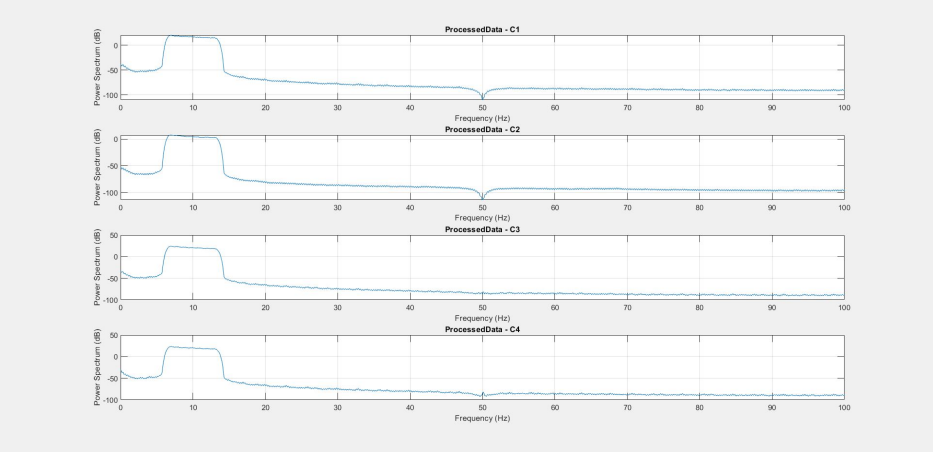


Figure 26: Power Spectrum of the fully Pre-processed data

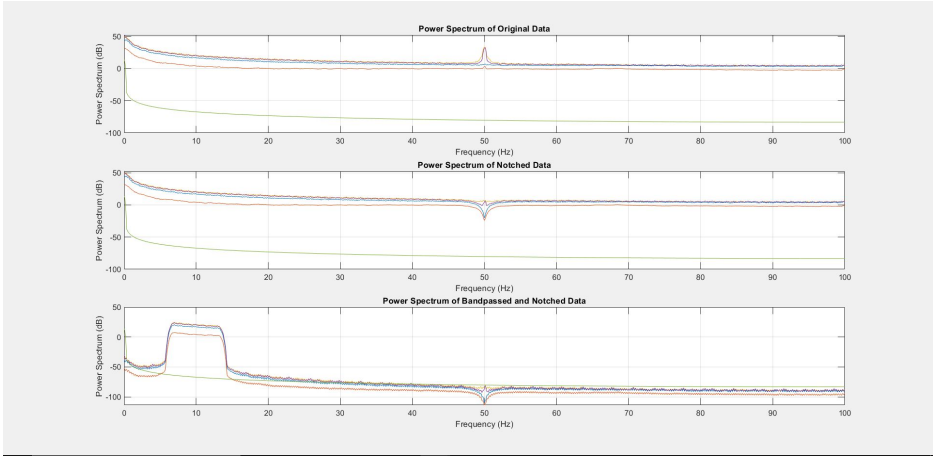


Figure 27: All three stages of pre-processing

The figure above shows a subsequent band pass filter along with Notch filtering, it is clearly shown the data has been sufficiently filtered as needed. There is clearly some room for improvement in the filter design as Channel 4 seems to still have a spike at 50 but simply brought down to be normalised.

The final, alternative method found, which works better for classification is Blackmann Band-pass filtering the incoming data, The Blackmann filter has been found through a Biomedical Engineering book [36] and is not of my own work.

3.5 MATLAB Signal Processing

3.5.1 Training Data

The training data will be shown throughout the course of this report, the acquisition of this data was completed via electrode placement as shown by Figure 16-21. The data is then captured through the OpenBCI GUI, instead of saving with transmission, the data is natively saved to a text file in the OpenBCI source folders therefore, gaining all of the exact raw data from the BCI sampled at 200Hz. Then the data is preprocessed in the proper ways as spoken of below and then used to train the machine learning algorithms. The data is taken in the form of 100 Flexions of each finger with approximately 3 second

pauses between each event. The flexions must run in sequential order from one hundred thumb's to 100 little finger flexions.

3.5.2 Feature Extraction

The chosen method requires segmentation first, but as stated before this has been completed in the transmission and LSL reading. The next stage is Feature extraction, the Root-Mean-Squared which shows muscle fatigue in a system. Appendix H, the Moving RMS section shows a compact method of a moving RMS which has a window size of 5. The windowing occurs as a traditional moving-averager would where it iterates from samples 11-15 then 12-16 and so on, very simply squaring the values, taking the mean of those and finally square rooting. My implementation runs through three distinct trials to test quality of data, as you will see in the Classification portion and in the figures below; I have a test that has not been processed at all (**Raw Data**); in order to not remove any hidden features that *could* result in better machine learning algorithms. Properly pre-processed data with Band pass and Notch filtering and finally using a Blackmann filter with the frequency limits set to 9-10Hz. The Blackmann filter has been found through a Biomedical Engineering book [36] and is not of my own work. These three trials will form the basis for how the training data in my machine learning models will be used.

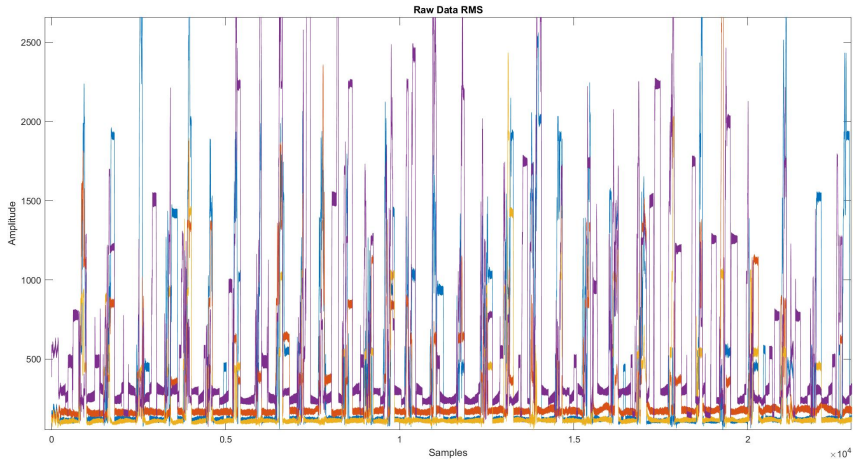


Figure 28: Feature extracted raw, unprocessed data

In the figures above, we can see clear and distinct differences in the data from Figure 28 showing unusual elevation of signal where an event/gesture takes place, compared to relatively similar figures in 29 and 30 which only show difference in the amplitudes and it is apparent there is baseline deviation for each channel. All of these iterations of feature extraction have been tested with the live data and work as intended. The next stage of classification will deem which pre-processing to use by rate of classification.

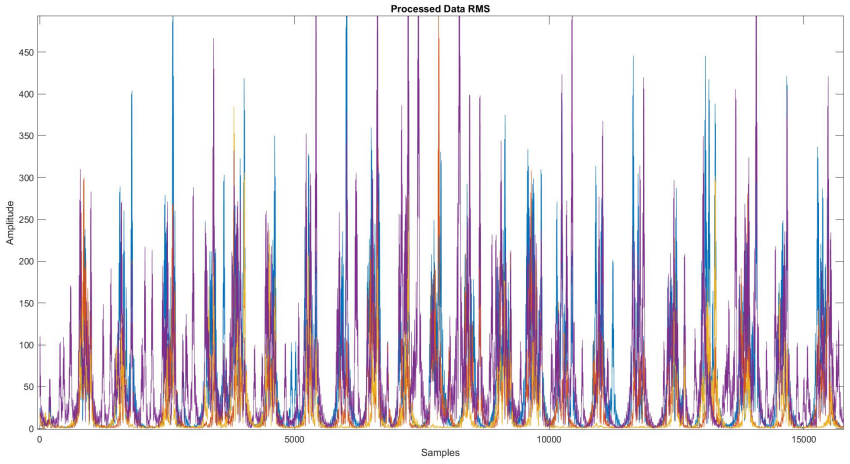


Figure 29: Feature extracted, properly pre-processed data

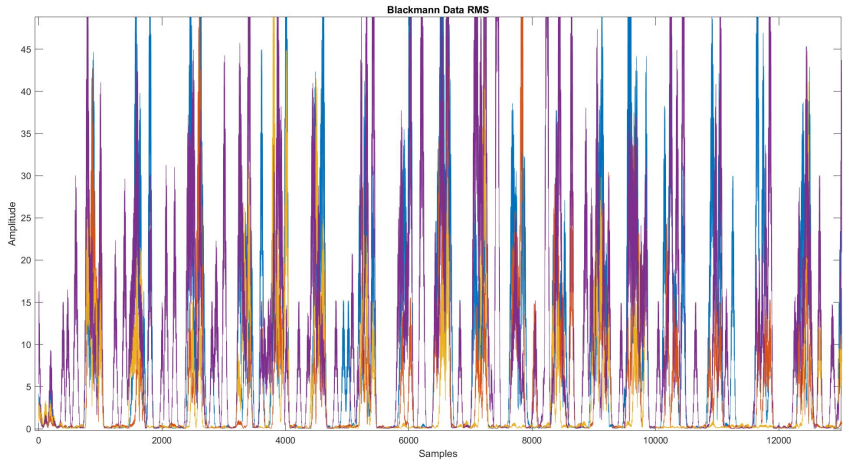


Figure 30: Feature extracted, Blackmann Band passed data

3.5.3 Classification

	Data Type:	**Pre-processed Data**	**Raw Data**	**Blackmann Data**
Model:	**X**	**X**	**X**	**X**
kNN	**X**	85.5%	91.7%	88.9%
SVM	**X**	79.2%	79.5%	76.9%
NN	**X**	80.9%	85.0%	86.9%
Decision Tree	**X**	85.1%	87.9%	88.6%
LDA	**X**	77.1%	81.6%	82.0%
Overall Accuracy	**X**	**81.6%**	**85.3%**	**84.7%**

Table 5: Percentage accuracy's of each machine learning model relative to the training data types input

To begin this section, data must be properly prepared in order to train the appropriate machine learning models. As spoken of before and shown in Table 5, there are 3 different pre-processing types used against 5 machine learning models all showing overall system accuracy's. The classification runs with a 70/30 split where 70% of my 381485 training array is used for training the model and 30% is testing the trained model.

There is 2 universal code used for all of the data preparations, how it is labeled and how it is Root Mean Squared. Using a for loop incrementing up until 381485, the code checks each row whether it is above or below a cut off threshold changing for each processing type, if the row is below the threshold we append a 0 to the label, this works as when training data was acquired, it was done in groups of Gestures therefore from the loading of Gesture 1-6 we can append 1-6 respectively to each one and then fine tune for simply the zeros, this beats an optical method of zeroing by hand.
The method of Root Mean Squaring all the data runs with loading unfiltered data for a .txt file by the GUI, processing it with the right technique: either Notch Filtering and Band-pass filtering, Blackmann filtering or simply just raw filtering. The data is then feature extracted with the RMS. Preparation of data runs differently for the Neural Network, which is trained with Inputs (the four channel training data) with the Targets (6 Class's which run as binary rows: Class 1 = 100000, Class 2 = 010000, Class 3 = 001000, Class 4 = 000100, Class 5 = 000010, Class 6 = 000001), the preparation file that was written uses the processed data from a main file, loads it into another as Inputs, the Target file is zeroed in a matrix of 6 columns x 381485 rows and an if loop is used to decide which data points are class 1,2,3,4,5,6 to match the binary definitions stated above. The neural network will look like the figure below:

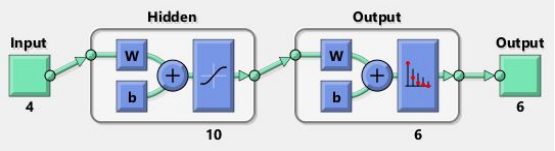


Figure 31: Neural Network Diagram

The SVM data preparation is tricky, the problem lies with the limited processing power in my machine which means the classification of 381485 data points is hopeless in a Support Vector Machine, taking in excess of 3 hours per mode, baring in mind these would be tests and need to be optimised so its not feasible to be using this model for me. However, an method has been devised at reducing the data size by limiting a training set to 120000 samples, 20000 samples of each class which reduces the time to train to roughly an hour. This is the only real preparation required for SVM data but only because of the limited power of my hardware.
Finally, the data for kNN and Decision Tree are done by the filtering above and finally labelled via the method stated.
All of the methods outlined use holdout validation with a 70 Training and 30 Testing data split. The kNN used 30,20,30 neighbours for Preprocessed, Raw and Blackmann models respectively as these seemed to yield the optimal accuracy's. The Decision tree all have a maximum of 100 splits, in theory there should only be 6 splits but allowing a large amount can help group the signals that may be the same but not actually look the same therefore it allows its own 'folder' for lack of a better term.
Unfortunately, due to lack of processing power which seems to be a recurring issue for this project, I could not work on optimising the different variables in models I have trained, they would begin to train and after a while they would exceed the memory space I have and cancel however the figure below will show for a kNN optimisation with different neighbour numbers tested and different distance types:

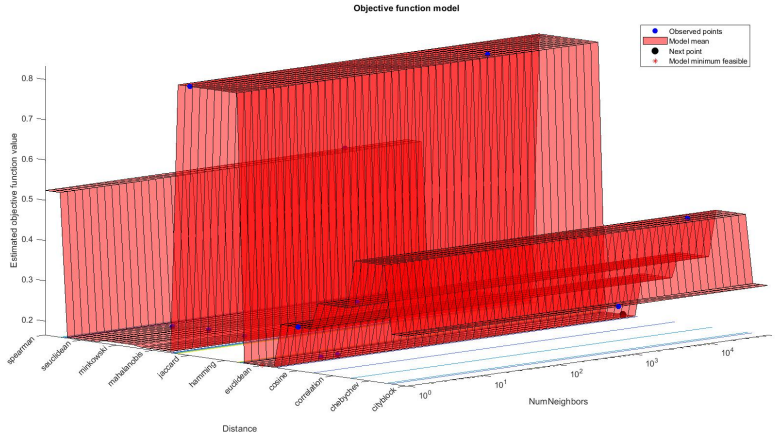


Figure 32: Visual Representation of the Objective function (kNN)

Iter	Eval result	Objective	Objective runtime	BestSoFar (observed)	BestSoFar (estim.)	NumNeighbors	Distance
1	Best	0.1671	61.169	0.1671	0.1671	16	seuclidean
2	Best	0.16542	1.762	0.16542	0.16585	6	euclidean
3	Accept	0.5243	731.22	0.16542	0.1836	9248	spearman
4	Accept	0.83333	161.27	0.16542	0.20242	2894	hamming
5	Accept	0.16598	2.1614	0.16542	0.16572	10	euclidean
6	Accept	0.27375	80.309	0.16542	0.16571	1	cosine
7	Accept	0.17167	2.4799	0.16542	0.16571	16	minkowski
8	Accept	0.1683	821.83	0.16542	0.16571	16	mahalanobis
9	Accept	0.33905	79.307	0.16542	0.16571	2	correlation
10	Accept	0.28833	23.021	0.16542	0.16571	638	cityblock
11	Accept	0.45752	19889	0.16542	0.16571	15629	chebychev
12	Accept	0.83315	102.63	0.16542	0.16571	1	jaccard

Figure 33: Results before failure of the optimisation task (kNN)

Raw Data: kNN and Decision Tree

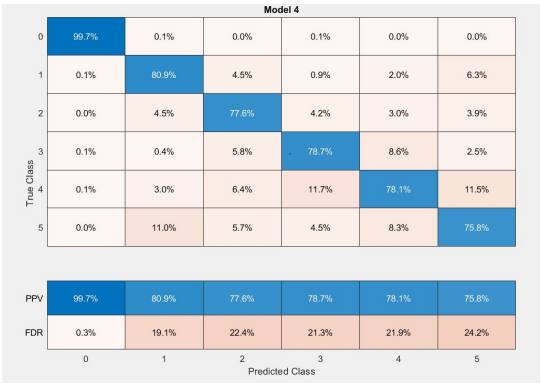


Figure 34: kNN Confusion matrix for the Raw Data at 91.7%

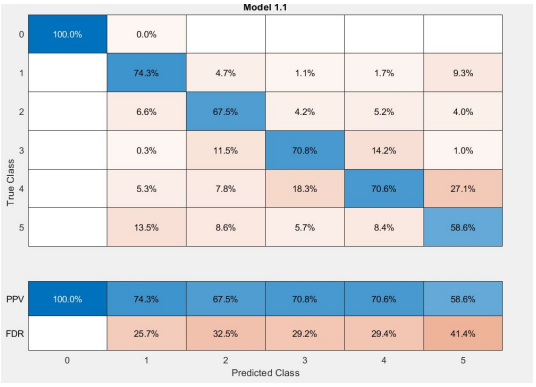


Figure 35: Decision Tree confusion matrix for Raw data at 87.9%

The above classification showns kNN at 91.7% and Decision tree at 87.9%. Both of these are highly skewed by the Gesture 1 which is relaxed hand, which of course has a higher accuracy due to the considerably higher amounts of data for Gesture 1. Nonetheless the accuracy's are suitable to our application.

Raw Data: 10 Neuron Neural Network

Figure 36: Raw data neural network confusion matrix at 85.0%

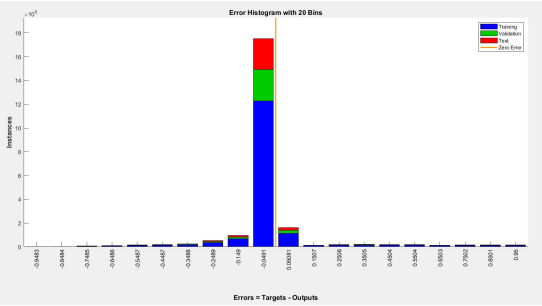


Figure 37: Error Histogram of Model for Raw data

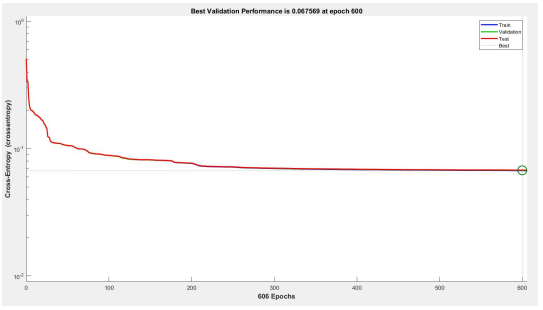


Figure 38: The training performance over epochs for raw data

The figures above show a overall 85% accuracy trained at 606 epochs, taking 3 minutes to complete. The model along with all NN's are trained using a Scaled Conjugate Gradient, the best validation performance is 0.067569 as shown by Figure 35. Finally, we can see that the Error histogram holds the largest bin height at the central error of 0.0491 which is the lowest error.

Raw Data: SVM

Class	1	2	3	4	5	6
1	**99.8400**	0.1300	0.0250	0.0050	0	0
2	0.1200	**79.1900**	7.3050	2.1050	4.2700	7.0100
3	0.1150	4.9800	**79.7800**	6.9950	3.7050	4.4250
4	0.0250	0.2950	7.7750	**82.7300**	8.0100	1.1650
5	0.0350	2.6400	3.9750	14.6850	**69.7600**	8.9050
6	0.0500	13.4800	9.0600	7.8800	4.0450	**65.4850**

Table 6: Raw data SVM confusion matrix

The overall model classification is 79.5%, it seems the confusion matrix is fairly weighted

with only columns 5 and 6 reducing the accuracy.

Pre-processed Data: kNN and Decision Tree

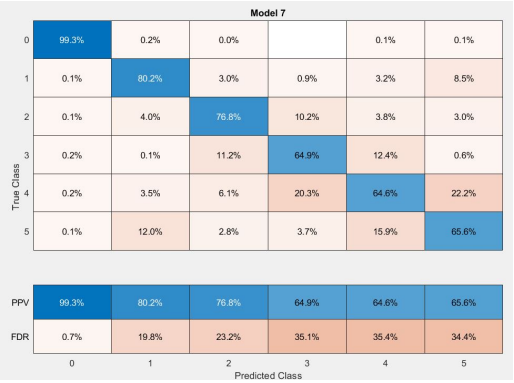


Figure 39: Pre-processed kNN modelat 85.5%

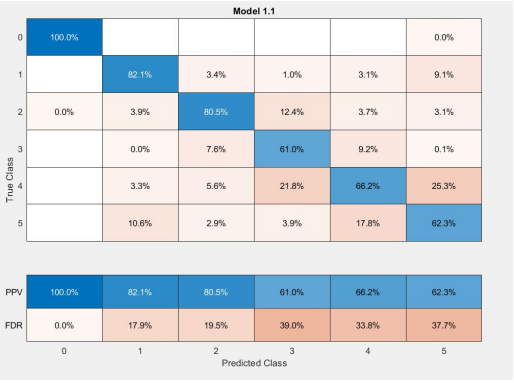


Figure 40: Pre-processed Decision treeat 85.1%

Pre-processing which uses Notch filtering and Band-pass filtering when passed through a weighted kNN and Decision tree yield a 85.5% and 85.1% accuracy respectively. The training accuracy's show similar predictive accuracy's throughout Classes 1-6. It seems the decision tree for resting class shows no inaccuracies which is unusual.

Pre-processed Data: 10 Neuron Neural Network

Figure 41: Pre-processing Neural Network Confusion Matrix at 80.9% Overall

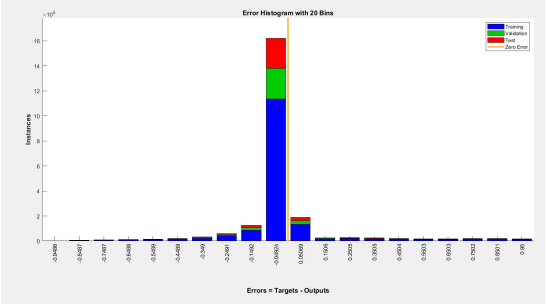


Figure 42: Error Histogram of Model for pre-processed data

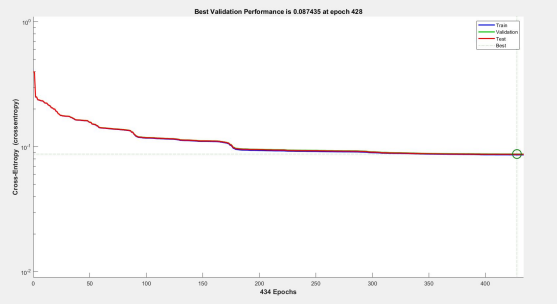


Figure 43: The training performance over epochs for pre-processed data

The neural network trained for this model runs at 80.9% as shown by the confusion matrix, taking 434 epochs in 3 minutes. The histogram shows a lowest error bin (near no error) at 0.04924. The best validation performance is 0.087435 at the training epoch 428. The confusion matrix shows clear biasing in the overall classification rate due to the near 100% at Class 1 and rest of them are 60-70%

Pre-processed Data: SVM

Class	1	2	3	4	5	6
1	**99.4250**	0.3250	0.0850	0.0150	0	0.1500
2	0.2400	**89.2850**	3.7300	1.2350	0.8000	4.7100
3	0.0450	3.0650	**76.6350**	14.1700	1.8700	4.2150
4	0.0500	0.0150	8.2250	**84.5050**	6.4750	0.7300
5	0.0400	1.2350	5.4200	26.0150	**46.6650**	20.6250
6	0.1400	8.9000	3.1050	5.6550	3.8150	**78.3850**

Table 7: Pre-processed data SVM confusion matrix

The table above shows the confusion matrix of the pre-processed data Support Vector Machine. It seems the classification average lies at 79% which is likely brought down due

to the Class 5 classification at 46.6%.

Blackmann Band-pass Filtered Data: kNN and Decision Tree

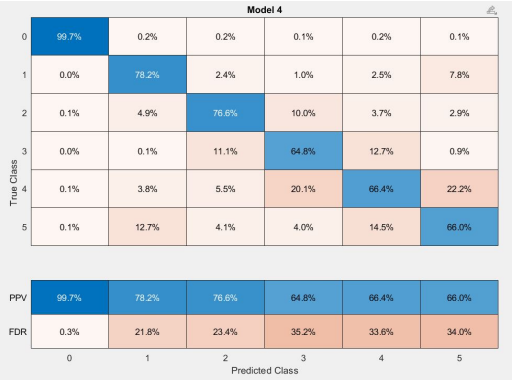


Figure 44: Blackmann Filtered kNN model at 88.9%

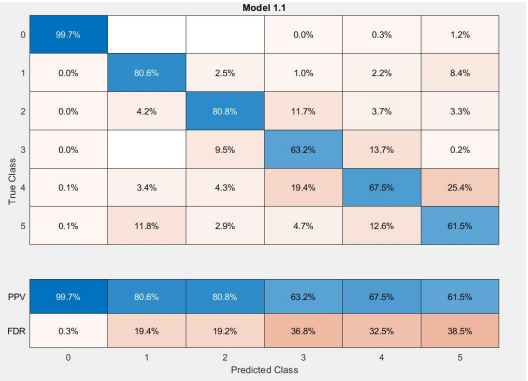


Figure 45: Blackmann Filtered Decision tree at 88.6%

The two models described above by their confusion matrix are the 88.9% kNN and the 88.6% Decision tree. The two trees hold very high and consistent classification percentages and seem to fit a general trend throughout all models where they are strongest for Class 1, 2 and 3 then subside, lowering for Class 4,5 and 6.

Blackmann Band-pass Filtered Data: SVM The SVM confusion matrix above is at

Class	1	2	3	4	5	6
1	**99.5900**	0.2950	0.0650	0	0.0100	0.0400
2	0.0600	**82.8700**	2.6650	2.8600	0.7400	10.8050
3	0.0100	4.3000	**74.4400**	15.3100	1.2100	4.7300
4	0	0.2750	9.4000	**85.9950**	2.9950	1.3350
5	0.0100	2.3800	5.9450	27.1750	**41.7850**	22.7050
6	0.0150	10.4000	4.3650	5.1550	3.8200	**76.2450**

Table 8: Blackmann filtered SVM confusion matrix at 76.9%

76.9% classification, considerably high rates comparative to the other models except for Class 5 which is at a very low 42%.

Blackmann Band-pass Filtered Data: 10 Neuron Neural Network

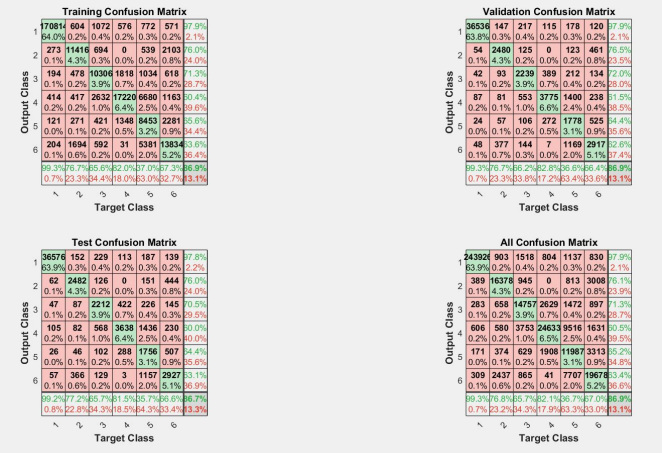


Figure 46: Blackmann - Neural Network confusion matrix at 86.9% overall

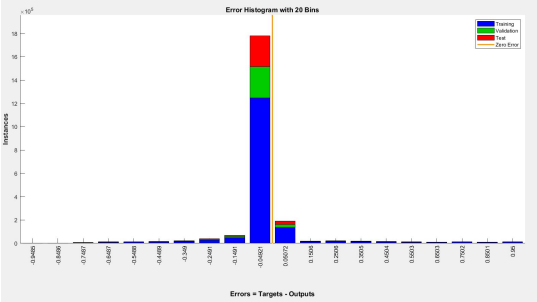


Figure 47: Error Histogram of Model of Blackmann data

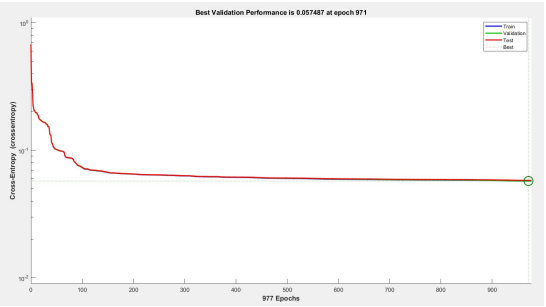


Figure 48: The training performance over epochs of Blackmann data

The neural network on Blackmann filtered data shows an overall accuracy of 86.9%, it completed with 977 iterations in 14 minutes with the best validation at 0.057487 at epoch 971. The figure above also shows lowest error at bin 0.04921 which is also where most of the data resides.

Using the data that has been aquired from classifying the three data types in 5 different machine learning models, it is evident that the Raw Data has the highest classification average over 5 models at 85.3% and likely due to the fact we are not ridding useful information when we pre-process, this is interesting as pre-processing is a norm in EMG analysis but it can be shown that this may be redundant in prosthetic signal processing. It is noteworthy that all of the classification results are skewed higher due to Class 1 being near 100% every time, Class 1 has the highest amount of clear and simple data in abundance therefore it is understandable our models seem higher than they really are.

The best trained model has proven to be the kNN at 91.7%. The confusion matrix for the raw kNN shows a high classification of nothing below 75.8% throughout classes 1-6 which is unusual considering the cross talk between the middle, ring and little finger is enormous, this can be shown in other models where the data is less distinguishable as shown by the Neural network for this data type. According to MATLAB documentation, classifiers have been given average speeds and memory used: it seems the best fitting for our use is the classifiers which are Fast (0.01s for predictions) and Small (use 1MB of data). The classi-

fiers that fit this mould are: Decision Trees (Fast and Small), kNN (Medium and Medium) and NN (Fast and Medium). Medium is 1 second prediction time and 4MB memory. It seems the SVM's we have tested can all be ruled out as it has the longest training time as stated before (excess of 2 hours) along with large data usage and deceptively lowest classification rate as shown by table 5 which will be spoken about below. kNN's are all the highest performing models which for the pre-processed and Blackmann filtered data both seem to be lower classification explicitly due to the last 3 fingers which as we know are linked together, this provides a good explanation as to why the three gestures are less distinguishable.
Neural Networks seem to be all be very good on first glance but this is actually untrue; The raw data neural network classified at 85% but the accuracy's for Class 2-6 are actually around 62%, the skewing of Class 1 leads to very misleading information but the network seems to make distinctions between different gestures which is good. NN are the most efficient predicting method along with Decision Trees and kNN's therefore this will be kept in mind when making a choice of which route to take, this model took 3 minutes with 606 epochs but as the performance figure shows, it converges towards the best line at around epoch 200 which is extremely quick. The model uses a cross entropy performance as a loss function which increases, as the classified label of the data moves further away from its actual label, the closer to 0 the better the model and here we can see best is 0.067 at epoch 600. The confusion matrix for the properly pre-processed data is much less appealing with Class's 2 to 6 just slightly higher at 63% which shows it to be the least desirable of them all, the model again only took around 3 minutes to train at 434 epochs at a performance of 0.0861, worse than the raw data.
The final NN confusion matrix of Blackmann filtered data shows an average if 67% from Class 2-6 which is marginally better than the previous two data types. The best performance was 0.057 at epoch 971 out of 977, training in 14 minutes. This model outperforms the other NN's but it seems to show much longer training times which is problematic. The SVM's we have trained have shown, apparently low overall scores, ridiculously long training times and reduction of training data due to its large quota for memory however it seems these models are actually very good except for one class, Class 5. To begin with the Raw Data SVM, it seems that rounded to nearest integer we have accuracy's of 100,80,80,83,70 and 66, these are exceptional rates for my application of course excluding Class 6. Moving on to pre-processed SVM we get similar numbers if not higher at 100,89,77,85,47 and 78. The 47 is of greatest concern here at the ring finger, and this is due to the major misclassifications at Class 4 and 6 which we have previously discussed to be expected with the last three fingers showing a slightly higher misclassification. The nerves for the ring finger and the little finger are intertwined and share most of the muscles therefore it is understandable that Class 5 and 6 are mixed together, along with the middle which share tendons. Finally, the Blackmann data SVM shows 100,83,74,86,42 and 76; again very good accuracy's besides the 42 which is explained before and is unlikely to be linearly separable with an SVM, for this reason I expect a Neural network with more hidden layers to detect the separable information as it did better than the SVM.
Finally, the Decision tree beginning with Raw data. Class 1 in Figure 32 needed almost no misclassification as shown by the empty boxes, which allowed for more accurate representation for the rest of the classes, all of the accuracy's reside around 70 besides Class 6 (Little Finger), this is odd however could be put down to quality of information being poor as the ring and middle seem to be classified well. The overall accuracy is 87.9% which is very high. The preprocessed and Blackmann decision tree seems to follow the same rules as the Raw data's Class 1 but the ring, middle and little finger are shown to be much less

separable compared to the 82 and 81% at Class 2 and 3 respectively. This shows cross talk is clearly occuring and that these models may not be ideal compared to the rest.
Overall it is clear that the kNN's and Decision Tree's are best suited for this task due to simplicity and speed, followed by Neural networks and finally SVM. To delve deeper, it is seems that the Raw data kNN is the optimal classifier but this could be changed with the introduction of live data.

3.5.4 Testing the classifiers

In order to check the trained model, code has been written to test online and offline classification.
Beginning with offline classing, the implementation uses a random number generator to give us a random data point from my initial training data set, now this is fitted to my classifier and the code checks whether the response is the same. For the purposes of this testing, only the models for Preprocessed kNN at 86%, Raw Data kNN tree at 92% and the Blackmann Decision tree at 89%. Every time this was tried, the right response came out as the value as it should, but of course it is the exact training data with no changes. Now, the next step was introducing white noise to the data in various stages and the average of **5 tests** for each processing type is shown below

When the Signal to Noise ratio is increased to 10 for white Gaussian noise function, 100 data points have been tested to see how they fair. The results on the kNN came out with a 56% positive classification rate which is quite poor, however the noise introduced is significant. When the SNR is increased to 20, the correct classification comes out to 88% which is a significant improvement. The two tests show that the models are clearly great at handling low amounts of noise but of course not high amounts as it has not been trained on this type of data.
The Decision tree using Blackmann data comes to a 67% positive predictive rate at an SNR of 10, when increased, of course the classification increases at a drastic rate; the SNR now at 20 yields a 90% positive classification. The two tests clearly show a better resistance to noise on the Blackmann Decision tree than the preprocessed kNN.
Finally, the Raw data kNN. Unfortunately, with an SNR at 10 the positive classification rate is subpar at 42% however when increased to 20 SNR, the correct classing shows 75% accuracy.

Now for testing the live data; it seems that with live data, classing accuracy degrades heavily. An understanding of why it does this is stated in the limitations and problems in the conclusions but in brief, the training data is distinguishable from the data that comes in live which means the model is not accurate for the new incoming data. Three tests have been taken to show the inaccuracies, many more have been checked and unfortunately all of them yielded a poor rate of positive classification. Below shows what the code written holds in terms of the data input and then classification from 0-5. Figure 49 shows fingers 1 and 2 in order of occurrence, Figure 50 shows 3,4 and 5 in order, and finally Figure 51 shows 1-5 in that order. It is abundantly clear from the brief figures below that classification is almost exceptionally flawed for live data, however the silver lining is that it is clear how to fix this issue.

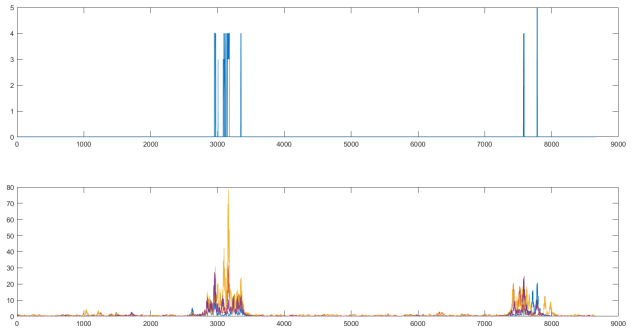


Figure 49: The incoming live data against classifier response of Fingers 1 and 2

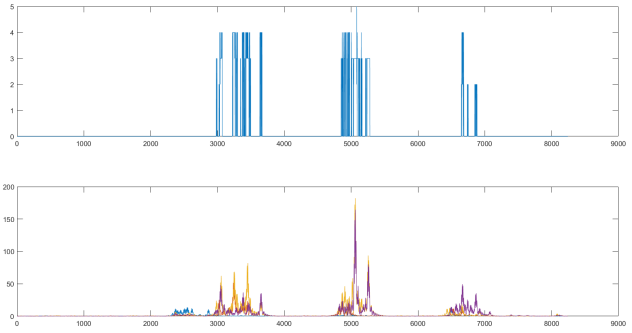


Figure 50: The incoming live data against classifier response of Fingers 3,4 and 5

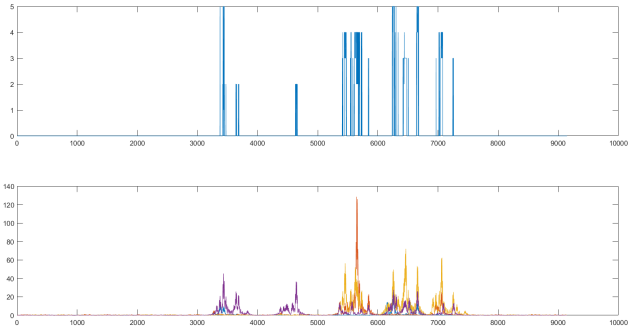


Figure 51: All finger flexion against classifier response

3.6 MATLAB to Arduino

Appendix I shows the Arduino code using the PWM Servo Library and data rate set at 9600 which corresponds to the MATLAB code. The code holds a variable that stores the incoming serial data from 0-6 in **incomingClass**. 6 Gestures have been defined in functions that are called upon in the case statements in the main loop for ease of coding. This code is very simple, it checks the serial port to see whether data is being written and if it is greater than 0, the data is fed into the variable stated above. Using a switch case structure, the code decides which part to execute depending on the incoming serial data.

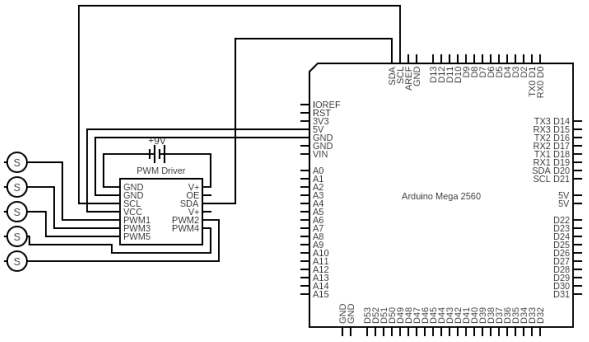


Figure 52: Circuit Diagram wiring for Arduino to Shield to Servos

The Figure above shows the very limited wiring included in getting the Arduino to interface with the shield. The shield is powered by a separate 9V battery supplying enough power to the servos.

```
%MATLAB Code for Serial Communication with Arduino
fclose(instrfind);
delete(instrfind);
x=serial('COM4','BaudRate', 9600);
fopen(x)
go = true;
while go

a = input('Enter 0:6 to turn ON LED or 7 to exit, press: ');
fwrite(x, a, 'char');
if (a == 7)
  go=false;
end
end
fclose(x)
fclose(x)
```

```
Command Window
  Enter 0:6 to turn ON LED or 7 to exit, press: 1
  Enter 0:6 to turn ON LED or 7 to exit, press: 2
  Enter 0:6 to turn ON LED or 7 to exit, press: 3
  Enter 0:6 to turn ON LED or 7 to exit, press: 4
  Enter 0:6 to turn ON LED or 7 to exit, press: 5
  Enter 0:6 to turn ON LED or 7 to exit, press: 6
  Enter 0:6 to turn ON LED or 7 to exit, press: 0
  Enter 0:6 to turn ON LED or 7 to exit, press: 7
```

Figure 53: MATLAB to Arduino stand-alone working file

The code that was developed separately to the main file is shown above. The code shows implementation of MATLAB looking for the Arduino and sharing a input given by the user (in this case, a prompt in the command window given by myself). This small compact piece of code coupled with the Arduino IDE case code shown in Appendix I. This implementation is efficient and works at a very quick speed.
When introduced to the main file it looks like Appendix H, in the last section before the code ends. This is where the problems occur, through experimentation I have found that this section is definitely the problem for my whole implementation.
Now to elaborate on this issue, the code works by MATLAB and Arduino sending and receiving, but the issue lies where two devices cannot be latched onto a serial port simultaneously. Now in order to circumvent this, every time my primary loop executes, the serial is opened and closed in a very inefficient way causing a large amount of time to be consumed, coupled with time it takes to read the data from the Arduino and finally actuate. Finally, the classification response comes in at a very high rate, every chunk from the inlet holds 10 data points which are sampled at 200 Hz from the pipeline therefore the data inflow is high, and that means alot of data must be output quickly which seems to not work in my project, despite working in the code I have shown in Figure 50 when the data comes as a sole input from a user, allowing all the gestures to be complete.
It seems that the there is the glaring solution of down-sampling, and of course this has

been attempted to which two implementations were trialed. The first being simply changing the pause rate at the bottom of the code to allow for a longer time between loops which is doesn't work as the data comes in chunks, therefore when the packet comes in and is analysed, the classification response can change a maximum of 10 times in one which is extremely quick and the servos cannot actuate in time let alone when the pause rate is normal and the data comes way too quick for actuation to occur. The second try included using a secondary classification response variable that only outputs to the Arduino, the idea behind this is, if the issue is that the data changes too quickly for the servos to respond, then what if when the classifier outputs 1, we put one hundred ones into the outputting variable therefore every time the Arduino checks its serial, it sees 1 for enough time to actuate. Despite this effort, the trial did not seem to succeed, for reasons that seem obvious now, that the code that was trialled does not do anything but extend the issue to another set of code, the one hundred ones will still change as quickly as the classification response changes in the yfit variable therefore this is redundant, a simple downsample from LSL should fix this issue but it appears the data still updates quickly. Therefore, it can be shown that in a **stand alone file** the data can easily be sent to establish gestures, however when linking it to a classifier the system is set up in a way that it simply cannot work regardless how I down-sample due to the nature of my incoming LSL data.

4 Conclusion

To properly analyse the success of this project, we can cross reference achievements to the goals outlined from the beginning of this paper. The initial, primary goal is to develop and implement code that can detect specific muscle movement corresponding to each finger allowing prosthetic to mimic flexion. The code that has been developed provides the exact solution for this goal, the machine learning algorithms can detect muscle movement and classify (of course with low accuracy on live data but with offline data it is perfect) but can not tell the prosthetic to move **in this system architecture** due to the MATLAB to Arduino problems in my main code, of course it proves to work in my stand alone file. The next goal states it should recreate specific predefined gestures; these gestures shown by Figure 11 and recreated by Figure 12, it can also be shown via the Arduino code that when it receives the appropriate number, the corresponding gesture is initiated in the stand alone code.
Finally, the last, main goal asks to undertake sufficient background reading into electromyography. This has been completed in the introduction section; this begins with the Motor unit and how the aggregate action potentials are what we come to learn as the signal at a muscle. It is then explained how proper locations for electrodes can be deduced when looking for optimal EMG signals per task, in this case, forearm finger flexion. This section analysed the muscles in the arm that allow for the human hand to manoeuvre in the intricate ways it does and simultaneously allows us to take inspiration and understand how we can reciprocate this.
With retrospect, the secondary goals were very naive for this current iteration as you will read in the limitations below and future works, they were written before the project had properly begun therefore is representative of the limited knowledge I held at the time. There are much better, and useful goals to be written as I have found throughout this endeavour; therefore it is my mistake for including these. Nonetheless, all three of these goals were **not** met, and I feel for good reason.
To finalise this conclusion, it has been shown that data can be streamed from a BCI device to MATLAB, and the data can then be properly processed to be analysed by a machine learning algorithm and output to a set of servos in a prosthetic (in the separate file). I have learnt through experimentation that Raw data might be the best and easiest option for prosthesis as it requires less processing power and retains any patterns that are lost from processing. I have learnt that Arduino is incredibly difficult to interface via serial nonetheless can be done, a kNN seems to be the most efficient at mapping data where you have little understanding on what kinds of patterns to look for, Support vector machines require an immense amount of processing power and time, along with proper optimisation showing a clear flaw in my idea here. The project has experienced a variety of problems from original system architecture change from Arduino to MATLAB, the initial intended model was to use a FSM to work the servo but instead a machine learning model has been used; different types of pre-processing have been introduced in order to add quality to the data as opposed to simply the stereotypical notch and band pass filtering and finally the solution to using the Adafruit 16 Channel servo driver with the use of interfacing the Arduino IDE and MATLAB together which of course do not work in my attempts, not that it cannot work at all, but it was a better solution than the alternative of the Motor Shield Driver.
The efforts in this project completed the primary goals that were required and set out from the beginning, showing that this system could be improved exponentially aswell as already showing utility in its current form, it has been an incredibly fruitful experience and has shown how robotic prosthesis could very well be, not the future but the present

of modern limb loss.

4.1 Limitations and Problems

4.1.1 BCI to Arduino

Initial attempts during the beginning of this project were made at exporting data from serial (OpenBCI GUI) to Arduino. Many problems were sparked in this attempt but for starters, data cannot be written and read from the same serial port therefore it is necessary that 2 Arduino boards are required or the use of a microcomputer.
On the same topic, receiving data via serial is much more problematic when it comes to coding, and it would not work despite a large amount of time and effort; The arduino receives the data but could not parse it properly. However it seems there is a way to get it working, with the Cyton board OpenBCI offers according to the forums. Further the examples use a 'Focus' experiment that works by exporting binary data from OpenBCI to Arduino by serial therefore it must be possible in some way to do the EMG data via serial.
OpenBCI Ganglion cannot transmit float/char/string (as i attempted to do) to Arduino (Documentation claims it is possible but the developers claim it is not) only binary can work, therefore the idea of BCI to Arduino was stopped.
The documentation for the Ganglion is misleading when it comes to the on-board micro-controller "Simblee", there is a limited way of accessing with the Ganglion and neither the forums nor documentation help in accessing therefore it is not possible to export as far as I understand. There is also no easy way to use the GPIO pins on the Ganglion which is again another halted avenue.

4.1.2 OpenBCI to MATLAB

The data being received via LSL shows practically and in theory, a latency which is too long for a good prosthesis. The LSL data gets placed in a pipeline, which is taken in chunks but the need to pre-process, feature extract and classify means the 'live' data comes significantly later than needed. Even the BCI GUI operates by sending the last piece of information to reach the left side of the GUI from the right, meaning if we use a 20 second window, it seems theoretically there should be a roughly 20 second delay from just the LSL transmission.
Classification further has some issues; it seems Neural Networks for this iteration cannot be used due to the limited processing power and its lack of finding patterns althought showing strong promise on the confusion matrix's. Along with Support Vector Machines which take on my personal machine over 2 hours to train. The system I have designed has a blatant flaw which is being reliant on the processing power of my own machine for training good models limiting me to what I can have, therefore this system could be much better if I simply had better hardware which is clearly not optimal. The alternatives of course were to utilise other machine learning methods like kNN as shown in the results.
Finally, there seems to be a clear difference between the training data and the streamed LSL data. Training data used in this project is straight from the GUI, which writes to a text file at the sampling frequency of 200 Hz, unprocessed. Conversely, when the data is transmitted via LSL, it is being taken from the pipeline at a rate of 200 Hz but it is unclear at what rate the data is filling the pipeline, therefore different amounts of data come to MATLAB. Summing this up, it seems there is a clear discrepancy (no matter how small, it is a discrepancy) between the training data and live data therefore this reduces the accuracy of live classification, the training data is not trained on what it will see live.

A solution again would be to create a hardware EMG and acquire data the same way for training and live classing which is theoretically the most accurate method.

4.1.3 Matlab to Arduino

As stated before, the standalone file for MATLAB-Arduino conversation works to show every gesture required, **however** it cannot work when integrated into my main code, for reasons i believe has to do with the LSL transmission. It seems repeatedly that the system i have designed holds a large amount of flaws.

4.1.4 Arduino Motor Shield

The arduino motor shield is not stack-able which I was not aware of (for the servo pins, it is stack-able for the regular stepper motors), this means we have to take a longer route coding on the Arduino IDE to receive data as opposed to MATLAB being able to control the Motor Shield. Nonetheless, the 16 bit driver works better however is just more a) processing power b) time spent coding c) longer actuation times reducing the overall reaction time of the whole algorithm.

4.2 Future Work

4.2.1 Own Analogue EMG/Alternate device and software

It seems that the largest issue with my system is the latency from OpenBCI GUI to MATLAB via LSL. I believe this could all be circumvented with the design of my own EMG which can be fed into the analogue pins of 1 of 2 Arduinos, this arduino would take data and input it for us to manipulate in MATLAB or in the Arduino IDE, and rewrite that data down into a second Arduino. Either this or the 2 Arduinos could be switched for a Microcomputer like a Raspberry Pi and all the acquisition and processing can be done in one neat package, saving time, resources and money.
Alternatively, according to the OpenBCI developers; the Cyton board is much better suited for this task with more channels and a better interfacing with serial despite its much higher price tag. The BCI developers further recommend using Python instead of MATLAB as Python has much better applications for machine learning, or using the most recommended if we still want to use MATLAB is the Brainflow API to interface with MATLAB. According to developers, it is a better refined and easier alternative to LSL, maybe relieving some of the issues I have spoken of in limitations.

4.2.2 Better system architecture

As shown by this project, the system runs from an EMG that sends data to a GUI, which then transfers to MATLAB and then sends down to Arduino. I think it would be ideal if a machine learning approach could be fit onto a micro-controller or a microcomputer allowing for a compact, portable model. This iteration requires a computer (because of MATLAB), finding a method of machine learning on Arduino (which is possible on specialised boards), using a hard wires EMG to interface, maybe the use of tools like Myowear EMG devices. I state it requires a computer, but Arduino have products which allow machine learning to be completed on their boards as micro-controllers.

4.2.3 EEG Data

During the course of this project, I have been frequently glaring at the issues of EMG data, how it is not robust for different people/scenarios of applications. Working on using real Brain-Computer interfacing would increase the utility and adopt-ability of this project which is the ideal end goal, having a prosthesis that could be worn and used with a high accuracy and and easily as a real hand. Taking surface-EEG data simply makes much more sense, why would one take signals from the arm when it can be directly read from the source.

4.2.4 Combination Gestures

Further works could be done to see if extra, useful gestures one would need in basic, day to day life could be added. This could be done in the same method as the project has done already, by taking data and training a models on it.

4.2.5 Peer Reviewed Work

Finally, to explain the most important avenues of future works. During the course of this project, I have been corresponding with a PhD student who has been working on similar applications of this technology using my training data-set acquired during this project. The so-far, successful implementation completes as a multi-channel Least-Mean-Squared algorithm as a classifier with a soft-max by Karl Moore [34]. The LMS is an adaptive filtering algorithm, developed in 1959 by Widrow and Hoff. As explained by Bernard Widrow in a video presentation about the LMS algorithm along with the Adeline hardware he and his student developed [35]; **Adaline** is a piece of hardware that uses knobs and dials to represent the biological classifier formally known as a neuron. The Adaline has a 4 by 4 binary (which can, in theory be analogue as stressed by Widrow) input set which are connected to weights which learn to class patterns. The mathematical explanation for this system holds much better which can be shown as we go along.
X is the input, a pattern which is associated with a set of components. The inputs (patterns) have a weight and that weighted sum is represented as:

$$y = x_n \times w^t \tag{23}$$

It is important to note that the above equation is the same in reverse for transposition; x transposed multiplied by the weight is equivalent. A bias weight is added at the summing point where it is for these purposes +1. Now when training, we give 'x' a set of input patterns, for each pattern there is a desired response. The desired response is half of the answer, the other is the actual output and when the difference of the two is taken, we can find the error. Widrow has explained that the coefficient of the weights are adjustable and this is exceptionally interesting as it leads us to the adaptability portion of this filter. If the error is linked to the coefficient of the weights, we can take a minimisation approach to the Mean Square Error for the average training pattern and find an optimal system with error close to 0.
The mathematical explanation Widrow has shown is as follows [35]:

$$\begin{aligned} Error(E) &= Desired(d) - Output(y) \\ &= d - (x^t \cdot w) \\ E^2 &= d^2 - y^2 \equiv d^2 - 2(d \cdot x^t w) + (w^t x \cdot x^t w) \end{aligned} \tag{24}$$

Now to show the MSE algebra, the weights of the coefficients must be kept fixed. The MSE is the average of square of error over the set of training patterns.

$$MSE = \xi = \varepsilon[E^2] = \varepsilon[d^2] - 2\varepsilon(d \cdot x^t)w + w^t\varepsilon[x \cdot x^t]w \tag{25}$$

The epsilon is the expected, i.e. Expected Error squared. The next portion is simply a finalisation of algebra to show what type of pattern we will see when approaching our desired weight value.

$$\begin{aligned} \xi &= \varepsilon[d^2] - 2P^t w + w^t R w \\ P^{\triangle} &= \varepsilon[d \cdot x] \\ R^{\triangle} &= \varepsilon[x \cdot x^t] \end{aligned} \tag{26}$$

The output is fed into a sigmoid function as shown by Widrow and this allows us to represent the data from -1 to 1.
As explained before, this leads us to how it can be used to find a minimal MSE which simultaneously gives us the weight vectors. If we map the MSE as a function of the weight vector it becomes a quadratic of course, as shown by 26. Now Widrow has shown, if we begin with the weights tuned at 0, and gradually change the weights in the direction of 0 error, the minimal possible MSE can be found. The weight that gives the minimum MSE value is called the Weiner solution. This is a recursive task as all machine learning algorithms are but as it seems, a very simple idea with a complex implementation that is successful in the work I have been shown so far by Karl Moore [34].

Using the training data I have acquired, the model was written by Karl Moore and shows impressive testing times of 0.03553 seconds average of 9 tests to identify the correct signal. A notable difference in the works of Karl Moore [34] is that a soft max transfer function is used as opposed to the sigmoid function proposed by Widrow.

The implementation as far as I understand it, shows immense prospects considering the speed and accuracy, furthermore its a simple neural network that requires limited processing power as my project has constantly run into as a problem, and seems to be an easy code to integrate into online classification.

5 References

[1] - B. He, Neuromodulation, Second Edition (Page 339). Minnesota, Minneapolis: Academic Press, 2018.
[2] - E.Musk, "An integrated brain-machine interface platform with thousands of channels," Neuralink., San Francisco., California, 2019.
[3] - OpenBCI Developers. (2020, August. 6). Documentation[Online]. Available: https://docs.openbci.com/docs/08FAQ/FAQLanding
[4] - A. Phinyomark., C. Limsakul., P. Phulpattaranont., "A Novel Feature Extraction for Robust EMG Pattern Recognition," Journal of Computing [Electronic]. Volume 1., Issue 1., Page 71 - 80., Available: https://arxiv.org/ftp/arxiv/papers/0912/0912.3973.pdf
[5] - X. Bao., Y. Zhour., Y. Wang., J. Zhang., X. Lu., Z. Wang., "Electrode placement on the forearm for selective stimulation of finger extension/flexion," PloS ONE 13(1): e0190936., 2018
[6] - K. Najarian, R. Splinter, "Electromyogram," in Biomedical Image and Signal Processing, Second edition, Boca Raton: CRC Press, 2012, 217-225
[7] - R.C. Gopura, S.V. Bandara, M.P. Gunasekera, Electrodiagnosis in New Frontiers of Clinical Research (pp.32), London: InTech, 2013.
[8] - D. C. Toledo-Pérez, J. Rodríguez-Reséndiz, R. A. Gómez-Loenzo, and J. C. Jaureegui-Correa, "Support Vector Machine-Based EMG Signal Classification Techniques: A Review," Applied Sciences, vol. 9, no. 20, p. 4402, Oct. 2019.
[9] - J. Keating, "RELATING FOREARM MUSCLE ELECTRICAL ACTIVITY TO FINGER FORCES," Bachelor Thesis, Dept. Elec and Comp Eng, Worcester Poly. Inst., Worcester., 2014.
[10] - X. Tang, Y. Liu, D. Sun,. "Hand Motion Classification Using a Multi-Channel Surface Electromyography Sensor," Sensors., vol. 1, pp 1130 - 1147, December 2012.
[11] - C. Kothe, D. Medine, C. Boulay, M. Grivich, T. Stenner., (2013). Introduction [Online]. Available: https://labstreaminglayer.readthedocs.io/info/intro.html
[12] - E. N. Marieb, "10: The Muscular System." Anatomy and Physiology. 5th ed. Boston: Pearson, n.d. N. pag/ Print.
[13] - K.J. You, K.W. Rhee, H.C. Shin., "Finger Motion Decoding Using EMG Signals Corresponding Various Arm Postures.", Experimental Neurobiology 19 (2010): 54-61. Print.
[14] - Adafruit. Adafruit 16-Channel 12-bit PWM/Servo Driver - I2C interface [Online]. Available: https://learn.adafruit.com/16-channel-pwm-servo-driver
[15] – M.A. Oskoei, H. Hu, "Myoelectric control systems – A survey, "Biomedical Signal Processing and Control", vol. 2, issue 4, pp 276-292, Oct 2007.
[16] - E.A Clancy, E.L. Morin, R. Merletti, "Sampling, noise-reduction and amplitude estimation issues in surface electromyography, "Journal of Electromyography and Kinesiology", vol. 12, issue 1, pp 1-16, 2002.
[17] - B. Hudgins, P. Parker, R. Scott, A new Strategy for multifunction myoelectric control, IEEE Trans. Biomed. Eng. 40 (1) (1993) 82–94.
[18] - K. Englehart, B. Hudgins, P.A. Parker, A wavelet-based continuous classification scheme for multifunction myoelectric control, IEEE Trans. Biomed. Eng. 48 (3) (2001) 302–310.
[19] - Y. Huang, K. Englehart, B. Hudgins, A.D.C. Chan, A Gaussian mixture model based classification scheme for myoelectric control of powered upper limb prostheses, IEEE Trans. Biomed. Eng. 52 (11) (2005) 1801– 1811.
[20] - M.C. Carrozza, G. Cappiello, G. Stellin, F. Zaccone, F. Vecchi, S. Micera, P. Dario, On the development of a novel adaptive prosthetic hand with compliant joints: experimental platform and EMG control, in: Proceedings of the IEEE/RSJ International Conference

on Intelligent Robots and Systems, April 2005, pp. 3951–3956.
[21] - E. Lamounier, A. Soares, A. Andrade, R. Carrijo, A virtual prosthesis control based on neural networks for EMG pattern classification, in: Proceedings of the Artificial Intelligence and Soft Computing, Canada, 2002.
[22] - Kiguchi, K, Tanaka, T, Fukuda, T. Neuro-fuzzy control of a robotic exoskeleton with EMG signals. IEEE Transactions on Fuzzy Systems. Aug, 12(4), 481-90.
[23] - Gopura RARC, Kiguchi K. Electromyography (EMG)-signal based fuzzy-neuro control of a 3 degrees of freedom (3DOF) exoskeleton robot for human upper-limb motion assist. Journal of the National Science Foundation of Sri Lanka [Internet]. 2009 Dec. Available from: http://www.sljol.info/index.php/JNSFSL/ article/view/1470
[24] - Kundu, S, Kiguchi, K. Design and Control Strategy for a 5 DOF Above-Elbow Prosthetic Arm. International Journal of ARM. (2008). , 9(3), 61-75.
[25] - Pons, J. L, Rocon, E, Ceres, R, Reynaerts, D, Saro, B, Levin, S, et al. The MANUS-HAND Dextrous Robotics Upper Limb Prosthesis: Mechanical and Manipulation Aspects. Autonomous Robots. (2004). , 16(2), 143-63.
[26] - Khushaba RN, Kodagoda S, Takruri M, Dissanayake G. Toward improved control of prosthetic fingers using surface electromyogram (EMG) signals. Expert Systems with Applications. 2012 Sep 15;39(12):10731–8.
[27] - Pulliam CL, Lambrecht JM, Kirsch RF. Electromyogram-based neural network control of transhumeral prostheses. The Journal of Rehabilitation Research and Development. 2011;48(6):739.
[28] - M. León, J.M. Gutiérrez, L. Leija, R. Muñoz, "Multiclass Motion Identification using Myoelectric Signals and Support Vector Machines," Department of Electronic Engineering., CINVESTAV., Mexico, 2011.
[29] - S. Alty. EE4080. Class Lecture, Topic: "Support Vector Machines" Department of Electronic Engineering, Royal Holloway - University of London, Surrey, UK, February. 8, 2020.
[30] - JavaTPoint. K-Nearest Neighbor(KNN) Algorithm for Machine Learning [Online]. Available: https://www.javatpoint.com/k-nearest-neighbor-algorithm-for-machine-learning
[31] - K. Smith, (2013), Precalculus: A Functional Approach to Graphing and Problem Solving, Jones Bartlett Publishers, p. 8, ISBN 978-0-7637-5177-7
[32] - D. Cohen, (2004), Precalculus: A Problems-Oriented Approach (6th ed.), Cengage Learning, p. 698, ISBN 978-0-534-40212-9
[33] - M.H. Sazli , "A brief review of Feed-Forward Neural Networks" Dept. Elect. Eng., Ankara Univ., Turkey, Series A2-A3, pp 11-17 (2006).
[34] - K. Moore, S. Ibunu Mohomad, C Cheong Took, "An adaptive filter based autoencoder", work in progress to be submitted to Elsevier Neural Networks, May 2021.
[35] - B. Widrow. "The LMS algorithm and ADALINE. Part I - The LMS algorithm," YouTube, 30 July, 2012. [Video file]. Available: https://www.youtube.com/watch?v=hc2Zj55j1zU. [Accessed: May 3, 2021].
[36] - J. L. Semmlow and B. Griel, Biosignal and Medical Image Processing (CRC Press, ISBN: 9781466567368, 2004).

6 Appendices

6.1 Appendix A - Project Specification Form

1 Project Approach

The project will be run in a specific order, and in an Agile format. The project begins with optimal electrode placement decisions, moving on to algorithm development which will need to constantly be tested and refined until satisfactory. I plan to use a Karnaugh Map to aid in representing my logic for which fingers will move. Due to cross talk occurrence, Information must be taken from multiple electrodes where Logic can be written to deduce when the right finger needs to be raised. The next stage would include testing of implementation; testing out all predefined grip types making sure they all work.

2 Quality Assurance

The project has a very specific goal set in mind therefore I can assure quality if my project
A) Mimics a human hand's movement with signals drawn from the muscle.
B) Can complete the taxonomy of grips
Therefore the ability to quality check only lies within two distinct criteria, as most of this project is coding, quality of code can be upheld from readability to clear commenting which is something that can be enforced. In a project like this, or any project; errors and faults are inevitable, and knowing this information I can enter with the correct tools/information to mitigate my margin for error, simply reading and learning more material and being sure small mistakes do not domino into large code errors.

3 Resources Needed

1 - Lab Space
2 - Test Benches and equipment i.e. Batteries, Voltmeter, Oscilloscope, OpenBCI Ganglion and Bluetooth Dongle, Arduino, Wires etc

Supervisor Signature:	C. Cheong Took S. Alty
Date:	17/10/2020
Student Signature:	Rishan Patel
Date:	17/10/2020

1

6.2 Appendix B - Monthly Progress Forms

EE3000 - Individual Project Supervision and Progress Log

Date:

Student Name: Rishan Patel

Project Title: Prosthetic Project

Supervisor: SA / CCT

What tasks have you been assigned during the last month?	1. Create Gantt Chart 2.Fill Ethics Form 3.Fill HR Risk Assessment form 4.Project Brief
Which of the above tasks have you completed?	2,3,4
Any problems encountered? How do you intend to resolve them?	Lack of ideas of which Clive and Steve were both asked for input in regards to what kind of hazards could occur.
What tasks have you been assigned for the next month?	1.Gantt Chart 2.Establish transmission from BCI to Arduino 3.Create ability to read the code in float form 4.If unable to establish transmission wirelessly, research GPIO output from BCI.

EE3000 - Individual Project Supervision and Progress Log

Date:

Student Name: Rishan Patel

Project Title: Prosthetic Project

Supervisor: SA / CCT

What tasks have you been assigned during the last month?	1.Gantt Chart 2.Establish transmission from BCI to Arduino 3.Create ability to read the code in float form 4.If unable to establish transmission wirelessly, research GPIO output from BCI.
Which of the above tasks have you completed?	Gantt chart complete. Transmission from BCI to Arduino has been unsuccessful (2,3,4).
Any problems encountered? How do you intend to resolve them?	Serial monitor problems cannot access the data if theres being anything sent. Once figured out how to see information, the data is not what I expected or want. Attempted the Arduino Focus experiment but that's for binary data so has some use but not much. Attempted to look at the microcontroller " Simblee" but the docs are insufficient for access, along with the GPIO pins on the BCI. It is clear that if I want to communicate via serial its not going to work as serial can only be read/read by one device: need two arduinos for this. Tried to check if I can output to text file but again it's the wrong information being written. It has been spoken off that we will move to MATLAB as opposed to Arduino IDE.
What tasks have you been assigned for the next month?	1. Access OpenBCI data from MATLAB 2.See if you can write from MATLAB to Arduino 3. 4.

EE3000 - Individual Project Supervision and Progress Log

Date:

Student Name: Rishan Patel

Project Title: Prosthetic Project

Supervisor: SA / CCT

What tasks have you been assigned during the last month?	1. Access OpenBCI data from MATLAB 2.See if you can write from MATLAB to Arduino
Which of the above tasks have you completed?	Both tasks have been completed.
Any problems encountered? How do you intend to resolve them?	LSL transmission allows for communication from BCI to MATLAB. Problem lies that we cannot save any of the incoming data, simply output to command window but data comes in and can be manipulated.
What tasks have you been assigned for the next month?	1.Moving RMS or MAV 2.Attempt to find specific cut-offs for actuation 3.See if I can solely define a clench to output 4.

EE3000 - Individual Project Supervision and Progress Log

Date:

Student Name: Rishan Patel

Project Title: Prosthetic Project

Supervisor: SA / CCT

What tasks have you been assigned during the last month?	1.Moving RMS or MAV 2.Attempt to find specific cut-offs for actuation 3.See if I can solely define a clench to output
Which of the above tasks have you completed?	Moving RMS completed
Any problems encountered? How do you intend to resolve them?	Seems like I cannot use the 16 channel servo driver via MATLAB, could wire the servos directly to the Arduino and power them externally. Arduino can be powered with 6-20V, 1 servo needs 4.8v to 6V each. Could buy multiple Arduino servo shields that work with MATLAB.
What tasks have you been assigned for the next month?	1.Finish the Lit Review 2.Find optimal electrode placements 3.Begin the classification; take large datasets of hand gestures (100 of each finger flexion) and label them. 4.Work on the project poster presentation

EE3000 - Individual Project Supervision and Progress Log

Date:

Student Name: Rishan Patel

Project Title: Prosthetic Project

Supervisor: SA / CCT

What tasks have you been assigned during the last month?	1.Finish the Lit Review 2.Find optimal electrode placements 3.Begin the classification; take large datasets of hand gestures (100 of each finger flexion) and label them. 4.Work on the project poster presentation
Which of the above tasks have you completed?	Most of Lit Review, optimal electrode placements found, data is being collected and project poster presentation done.
Any problems encountered? How do you intend to resolve them?	6 gestures, all 5 fingers and 1 resting hand set out. The data is being collected just a long task.
What tasks have you been assigned for the next month?	1. get signal processing ready for the data 2.begin to look at classification methods 3.document any patterns with the flexions 4.

EE3000 - Individual Project Supervision and Progress Log

Date:

Student Name: Rishan Patel

Project Title: Prosthetic Project

Supervisor: SA / CCT

What tasks have you been assigned during the last month?	1. get signal processing ready for the data 2.begin to look at classification methods 3.document any patterns with the flexions
Which of the above tasks have you completed?	Signal processing/prep for the training data is ready. Patterns have been documented.
Any problems encountered? How do you intend to resolve them?	
What tasks have you been assigned for the next month?	1.Look at Feed forward NN 2.SVM (both 70/30 splits) 3.Test if live data can be classified by model 4.Work on report more

EE3000 - Individual Project Supervision and Progress Log

Date:

Student Name: Rishan Patel

Project Title: Prosthetic Project

Supervisor: SA / CCT

What tasks have you been assigned during the last month?	1.Look at Feed forward NN 2.SVM (both 70/30 splits) 3.Test if live data can be classified by model 4.Work on report more
Which of the above tasks have you completed?	Feedforward NN, SVM
Any problems encountered? How do you intend to resolve them?	Feedf NN seems to give bad accuracies. Accuracies of less than 30%. SVM takes too long to train
What tasks have you been assigned for the next month?	1.Send data to Classifiers : kNN, k-means, SVM. Research methods 2. Work on sending data from classifier to Arduino

EE3000 - Individual Project Supervision and Progress Log

Date:

Student Name: Rishan Patel

Project Title: Prosthetic Project

Supervisor: SA / CCT

What tasks have you been assigned during the last month?	1.Send data to Classifiers: kNN, k-means, SVM. Research methods 2. Work on sending data from classifier to Arduino
Which of the above tasks have you completed?	Data sending to Arduino works separately to main file. This is without any boards just using a breadboard and external power.
Any problems encountered? How do you intend to resolve them?	
What tasks have you been assigned for the next month?	1. Retake data with a different hand 2.

EE3000 - Individual Project Supervision and Progress Log

Date:

Student Name: Rishan Patel

Project Title: Prosthetic Project

Supervisor: SA / CCT

What tasks have you been assigned during the last month?	1. Retake data with a different hand
Which of the above tasks have you completed?	None, but other models for classifier have been made. kNN, SVM with cross validation, LDA, Decision trees.
Any problems encountered? How do you intend to resolve them?	SVM again are too long to train.
What tasks have you been assigned for the next month?	1. Try to use cross validation on SVM

EE3000 - Individual Project Supervision and Progress Log

Date:

Student Name: Rishan Patel

Project Title: Prosthetic Project

Supervisor: SA / CCT

What tasks have you been assigned during the last month?	1. Try to use cross validation on SVM
Which of the above tasks have you completed?	None
Any problems encountered? How do you intend to resolve them?	SVM takes way too long
What tasks have you been assigned for the next month?	1. Optimise my functions 2. Make the models live classify 3. Integrate actuation code

EE3000 - Individual Project Supervision and Progress Log

Date:

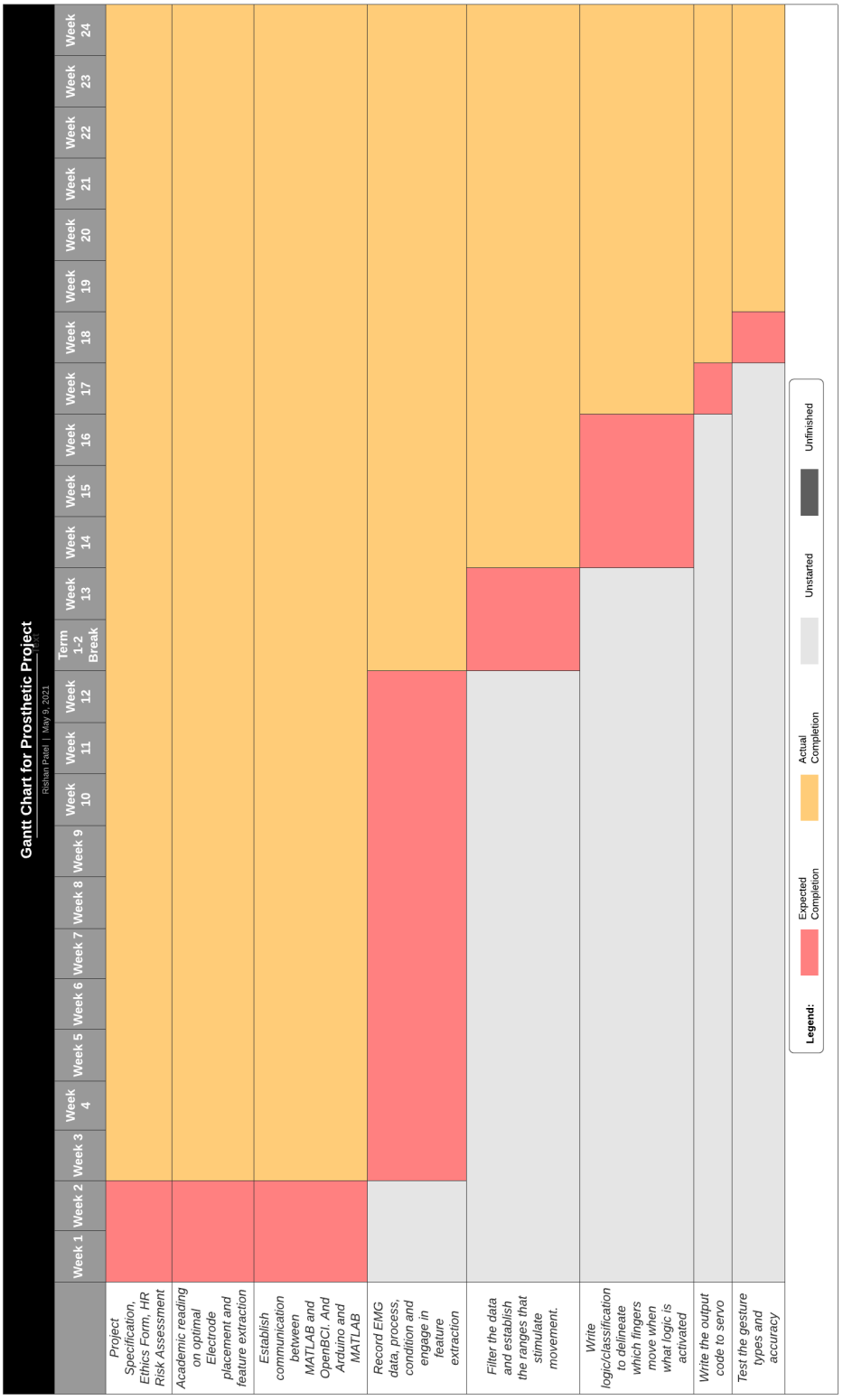
Student Name: Rishan Patel

Project Title: Prosthetic Project

Supervisor: SA / CCT

What tasks have you been assigned during the last month?	1. Optimise my functions 2. Make the models live classify 3. Integrate actuation code
Which of the above tasks have you completed?	All
Any problems encountered? How do you intend to resolve them?	Optimisation took the most time, using three separate preprocessing methods, redone all of the models and compiled a table of efficiencies. Using the highest classification along with easiest implementation to actuate. Found a method to use the 16 channel servo driver which compactifies the whole system therefore actuation works easily.
What tasks have you been assigned for the next month?	Working on the report, Working with Karl (pHd student) to peer review each others work.

6.3 Appendix C - Gantt Chart // Latest iteration



6.4 Appendix D - Ethics Form

EE3000 – Prosthetic Project Ethics Form

Rishan Patel

2020-2021

	Please answer the following
1	(a) Are you using personal data human specimen and/or animals? (e.g., personal information that allows you to identify individuals or company confidential information.) (b) If yes, are you collecting this data? **No data is collected that is sensitive in its nature nor is any Data is stored in memory.**
2	Who might be adversely affected by your project and why? **The project only concerns the ability to aid/better an existing issue (amputees), as for any misuse, there are no immediate adverse effects of this endeavour.**
3	Are there any environmental, societal, legal or sustainability issues? If yes, how will you address these potential risks? **There are any issues that concern legality or society. However, the plastic we will be using to either laser cut, 3D print, PCB's, Soldering must be used with efficiency to avoid waste and the material itself used should be reusable. The software end poses no risks. To address waste, a simple resolution would be to set a cap on spending for production or ensure that all production is done as efficiently as possible by myself.**
4	Could your project and its results be misused by third parties in any way? If yes, explain what measures you have taken to counteract this. **This project could take many forms of future misuse but only if worked upon to greater depths. As of now, it poses no real threats as it concerns misuse as it is only a hand controlled by impulses, a proof of concept and finally, the Intellectual property is retained by the University therefore no misuse could legally be conducted.**

6.5 Appendix E - Risk Assessment Form

Name of Person Undertaking Assessment		Date Conducted	Department / Area (including description of what is being assessed)					
Rishan Patel		06/11/2020	Risk Assessment for working at Home and at the Shilling Lab for the project of a Servo driven Prosthetic controlled by sEMG signals.					
Ref No	Hazard under review	No & Description of Staff/Students/ Others Involved	Existing Controls	Assessed Level of Risk			Further Action Required	By (Date) + Review Date
				L	M	H		
1	General electrical faults – Electrocution by devices, mains, faulty/misuse of power supplies, misuse of components.	Student	Have been trained since 1st Year in the proper use of electronic devices, test bench equipment and components to avoid burning out of components. Power supplies have channels which are voltage and current protected/fixed therefore any misuse can be limited. The equipment is periodically checked to ensure no faults. All the components available draw a very limited amount of voltage (Typically 5V or the drawing of 9v between 5 Servos) and generally all wires are well protected with the use of Jumper cables.	☒	☐	☐	None	06/11/2020
2	Harm due to entrapment – prosthetic pinching skin, closing with force, trapping fingers.	Student	The Servos in question are low torque and have a very limited ability to cause any real damage.	☒	☐	☐	None	06/11/2020
			The material of the prosthetic (acrylic) would not be strong enough to cause any real harm. The Prosthetic will typically be tethered to the computer and EMG broadcasts information therefore a safe distance can be kept avoiding entrapment.					
3	Electrode pads – causing skin irritation, pain (due to the stickiness, tugs on hairs will cause pain on removal and may aggravate skin).	Student	The use of medical grade electrode pads mean they have been tested and made with safety in great question. First aiders in the department to assist if necessary.	☒	☐	☐	None	06/11/2020
4	Health issues due to sitting at computers for prolonged amounts of time.	Student	Following NHS Protocol on 'How to sit at your desk correctly' found at - https://www.nhs.uk/live-well/healthy-body/how-to-sit-correctly/	☒	☐	☐	None	06/11/2020
5	COVID-19 Risk – Infection, transmission.	Students, Staff.	Masks, Hand Sanitizers, Workbenches actively sanitized and cleansed.	☒	☐	☐	None	06/11/2020

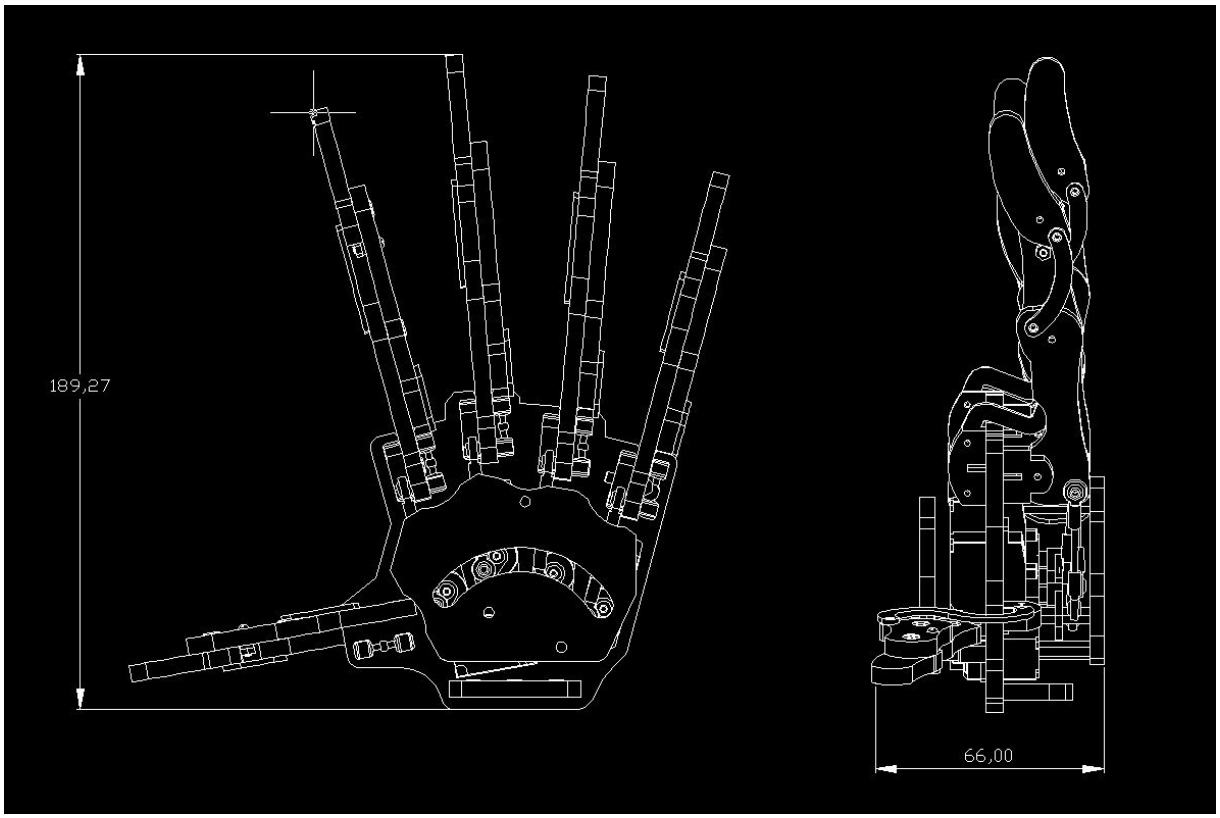
			All workstations have been separated with the appropriate 2-meter distancing All students given masks and protective goggles for safety. Provided our own cables for test bench equipment, limiting cross contamination.					
6	General Injuries – cuts, falls, slips etc.	Student, Staff, Visitor	Generally, slips can be avoided with proper signage. Cleaning up any spills and reporting any leaks to the department. We have dedicated first aid trained staff (Alex) who can come of assistance.	☒	☐	☐	None	06/11/2020
7				☐	☐	☐		
				☐	☐	☐		
				☐	☐	☐		

06/11/2020

6.6 Appendix F - Software and Hardware listing

H - Hardware S - Software
OpenBCI Ganglion (H)
OpenBCI GUI (S)
MATLAB (S)
Arduino IDE (S)
Elegoo Mega 2560 (H)
Prosthetic (H)

6.7 Appendix G - Prosthetic Design and Dimensions



The design for Prosthetic in use

6.8 Appendix H - MATLAB Code

```
clear all
%clearvars -except trainedClassifier validationAccuracy %Try use this so
%you dont have to keep retraining everytime you run the code
clear clc

x=serial('COM4','BaudRate', 9600);

%% Load Classification model for classing
load('PPTrainingData'); % Load the data file with training data for
%%Preprocessing
Z = TrainingData;
[trainedClassifier, validationAccuracy] = kNN86(Z); % Choose between
%%Tree85(Z) model or kNN86(Z) model


%% instantiate the library
disp('Loading the library...');
lib = lsl_loadlib();

%% resolve a stream...
disp('Resolving an EEG stream...');
result = {};
while isempty(result)
    result = lsl_resolve_byprop(lib,'type','EEG'); end

%% create a new inlet
disp('Opening an inlet...');
inlet = lsl_inlet(result{1});

disp('Now receiving chunked data...');
while true
    % get chunk from the inlet
    [chunk,stamps] = inlet.pull_chunk();
    for s=1:length(stamps)
        % and display it
%         fprintf('%.1f\t',chunk(:,s));

        %% Store

        sz = [100000 4];
        sv = [4 100000];

        data = [];
        %data = zeros(sv);
        % T = array2table(chunk,'VariableNames',{'column_1','column_2','column_3',...
        'column_4'});
        data = horzcat(chunk,data);
        data = transpose(data);
```

```
%% Pre-processing
fs = 200;

wo = 50/(fs/2);
bw = wo/15;
[b,a] = iirnotch(wo,bw);  % Notch Filter at 50Hz

NotchData=filter(b,a,data(:,1:2:3:4));

ProcessedData = bandpass(NotchData(:,1:2:3:4),[7 13],fs); % Bandpass Data at 7-13
as BCI Suggests

%% Labelling data as variables
c1 = ProcessedData(:,1);
c2 = ProcessedData(:,2);
c3 = ProcessedData(:,3);
c4 = ProcessedData(:,4);


%% Moving RMS (Square then mean then root) 5 point centered moving RMS (1-5,
%2-6, 3-7 etc)
C1 = sqrt(movmean(c1 .^ 2, 5));
C2 = sqrt(movmean(c2 .^ 2, 5));
C3 = sqrt(movmean(c3 .^ 2, 5));
C4 = sqrt(movmean(c4 .^ 2, 5));

column_ = horzcat(C1,C2,C3,C4); %Name it column_ unless i change the training
%data variable

%% Putting data in table/matrix (Depends on need) for proper classification
%        T2 = array2table(column_);

%         T3 = table(([0;0;0;0]),[0;0;0;0],...
%             [0;0;0;0],[0;0;0;0],...
%             'VariableNames',{'column_1','column_2','column_3','column_4'});

T3 = zeros([3 4]);

T3 = [T3;column_];


%% Classification
       yfit = trainedClassifier.predictFcn(T3);

%                fprintf('%.1f\t',yfit(end));
%                fprintf('%.1f\n',s);

%% MATLAB Code for Serial Communication with Arduino
```

```
        fclose(instrfind);
        delete(instrfind);
        x=serial('COM4','BaudRate', 250000);
        fopen(x)
        go = true;
        while go

            fwrite(x, yfit, 'char');
            if (a == 7)
                go=false;
            end
        end
        fclose(x)
        fclose(x)
    end
    %pause(0.005); %200Hz refresh rate
    %pause(0.5); % This seems to work as a better alternative in practice
end
```

6.9 Appendix I - Arduino Code

```
1    #include <Adafruit_PWMServoDriver.h>
2 Adafruit_PWMServoDriver pwm = Adafruit_PWMServoDriver();
3
4 // our servo # counter
5 uint8_t servonum = 5;
6 int incomingClass = 0;
7
8 void setup() {
9   Serial.begin(9600);
10   for (int i=5; i<=12; i++){
11     pinMode(i, OUTPUT);
12   }
13
14  pwm.begin();
15  pwm.setPWMFreq(60);  // Analog servos run at ~60 Hz updates
16
17              }
18
19 void Gesture0 () {
20         //relaxed
21    pwm.setPWM(11, 11, 300);   //150 = closed, 500 = open
22    pwm.setPWM(12, 12, 400);   //150 = closed, 500 = open
23    pwm.setPWM(13, 13, 400);   //150 = closed, 500 = open
24    pwm.setPWM(14, 14, 400);   //150 = closed, 500 = open
25    pwm.setPWM(15, 15, 400);   //150 = closed, 500 = open
26    Serial.println("relaxed");
27    Serial.write(incomingClass);
28    delay(2000);
29 }
30
31 void Gesture1 () {
32         //Thumb
33    pwm.setPWM(11, 11, 500);   //150 = closed, 500 = open
34    pwm.setPWM(12, 12, 150);   //150 = closed, 500 = open
35    pwm.setPWM(13, 13, 150);   //150 = closed, 500 = open
36    pwm.setPWM(14, 14, 150);   //150 = closed, 500 = open
37    pwm.setPWM(15, 15, 150);   //150 = closed, 500 = open
38    Serial.println("thumb flexion");
39    Serial.write(incomingClass);
40    delay(2000);
41 }
42
43 void Gesture2() {
44    //Index
45    pwm.setPWM(11, 11, 150);   //150 = closed, 500 = open
46    pwm.setPWM(12, 12, 560);   //150 = closed, 500 = open
47    pwm.setPWM(13, 13, 150);   //150 = closed, 500 = open
48    pwm.setPWM(14, 14, 150);   //150 = closed, 500 = open
```

```
49    pwm.setPWM(15, 15, 150);   //150 = closed, 500 = open
50    Serial.println("Index Flexion");
51    Serial.write(incomingClass);
52  delay(2000);
53 }
54
55    void Gesture3() {
56    //Middle
57    pwm.setPWM(11, 11, 150);   //150 = closed, 500 = open
58    pwm.setPWM(12, 12, 150);   //150 = closed, 500 = open
59    pwm.setPWM(13, 13, 560);   //150 = closed, 500 = open
60    pwm.setPWM(14, 14, 150);   //150 = closed, 500 = open
61    pwm.setPWM(15, 15, 150);   //150 = closed, 500 = open
62    Serial.println("Middle Flexion");
63    Serial.write(incomingClass);
64  delay(2000);
65 }
66
67    void Gesture4() {
68    //Ring
69   pwm.setPWM(11, 11, 150);   //150 = closed, 500 = open
70   pwm.setPWM(12, 12, 150);   //150 = closed, 500 = open
71   pwm.setPWM(13, 13, 150);   //150 = closed, 500 = open
72   pwm.setPWM(14, 14, 560);   //150 = closed, 500 = open
73   pwm.setPWM(15, 15, 150);   //150 = closed, 500 = open
74   Serial.println("Ring Flexion");
75   Serial.write(incomingClass);
76 delay(2000);
77 }
78
79  void Gesture5(){
80   //Pinky
81   pwm.setPWM(11, 11, 150);   //150 = closed, 500 = open
82   pwm.setPWM(12, 12, 150);   //150 = closed, 500 = open
83   pwm.setPWM(13, 13, 150);   //150 = closed, 500 = open
84   pwm.setPWM(14, 14, 150);   //150 = closed, 500 = open
85   pwm.setPWM(15, 15, 560);   //150 = closed, 500 = open
86   Serial.println("Pinky Flexion");
87   Serial.write(incomingClass);
88 delay(2000);
89  }
90
91   void Gesture6() {
92   //grasp
93    pwm.setPWM(11, 11, 150);   //150 = closed, 500 = open
94    pwm.setPWM(12, 12, 150);   //150 = closed, 500 = open
95    pwm.setPWM(13, 13, 150);   //150 = closed, 500 = open
96    pwm.setPWM(14, 14, 150);   //150 = closed, 500 = open
97    pwm.setPWM(15, 15, 150);   //150 = closed, 500 = open
```

```
98    Serial.println("Grasp");
99    Serial.write(incomingClass);
100  delay(2000);
101   }
102
103 void loop() {
104   if (Serial.available() > 0){
105     incomingClass = Serial.read();
106     Serial.println(incomingClass);
107     switch (incomingClass){
108      case 0:
109       Gesture0();
110       break;
111      case 1:
112       Gesture1();
113       break;
114       ////////////////////////
115      case 2:
116       Gesture2();
117       break;
118      case 3:
119       Gesture3();
120       break;
121      ////////////////////////
122     case 4:
123       Gesture4();
124       break;
125     case 5:
126       Gesture5();
127       break;
128      ////////////////////////
129      case 6:
130       Gesture6();
131       break;
132      ////////////////////////
133   }
134 }
135 }
```

6.10 Appendix J - Training Data preparation file

```
    %% Processing Training Data Properly

%% Importing Data
load('OriginalData.mat');

data = vertcat(G1,G2,G3,G4,G5);

fs = 200;

wo = 50/(fs/2);
bw = wo/15;
[b,a] = iirnotch(wo,bw);

fvtool(b,a);

NotchData(:,1)=filter(b,a,data(:,1));
NotchData(:,2)=filter(b,a,data(:,2));
NotchData(:,3)=filter(b,a,data(:,3));
NotchData(:,4)=filter(b,a,data(:,4));
NotchData(:,5) = data(:,5);

ProcessedData(:,1) = bandpass(NotchData(:,1),[7 13],fs);
ProcessedData(:,2) = bandpass(NotchData(:,2),[7 13],fs);
ProcessedData(:,3) = bandpass(NotchData(:,3),[7 13],fs);
ProcessedData(:,4) = bandpass(NotchData(:,4),[7 13],fs);
ProcessedData(:,5) = data(:,5);

NotchTest3 = filter(b,a,ProcessedData(:,3),[7 13],fs);
NotchTest4 = filter(b,a,ProcessedData(:,4),[7 13],fs);
%% Feature Extraction on data
% Moving RMS
movrmsA = sqrt(movmean(ProcessedData(:,1) .^ 2, 5));
movrmsB = sqrt(movmean(ProcessedData(:,2) .^ 2, 5));
movrmsC = sqrt(movmean(ProcessedData(:,3) .^ 2, 5));
movrmsD = sqrt(movmean(ProcessedData(:,4) .^ 2, 5));

%Adding up Feature Extracted data into one variable
TrainingData = zeros(381485,5);
TrainingData(:,1) = movrmsA;
TrainingData(:,2) = movrmsB;
TrainingData(:,3) = movrmsC;
TrainingData(:,4) = movrmsD;
TrainingData(:,5) = data(:,5);

%% Adding Label to data for resting hand
for i=1:1:381485
    x=20;
    if TrainingData(i,:) < x
```

```
        TrainingData(i,5) = 0;
    end

end
Label = TrainingData(:,5);
```

6.11 Appendix K - Blackmann Filter

```
Fs =200;%Sample Frequency.
fh = 13/Fs;%Upper bound cutoff, upper EMG range.
fl = 7/Fs;%Lower Bound cutoff, lower EMG range.
L = 129;%Number of weights
j = -floor(L/2):-1;
c = sin(2*pi*fh*j)./(pi*j)-sin(2*pi*fl*j)./(pi*j);%Negative b[k] filter impulse
c = [c 2*(fh-fl), fliplr(c)];%Rest of the b[k]
c = c .* blackman(L)';%Multiplication of the previous output and a blackman window.
```

6.12 Appendix L - Classifier Testing Code for Raw Data

```
% load RawTrainingData.mat;
% [trainedClassifier, validationAccuracy] = kNN92(FeatureExtractedData);
TrainingDataNoise = awgn(FeatureExtractedData(:,1:2:3:4),20,'measured');

for i = 1:100

r = randi([1 381485],1);

TestPoint = TrainingDataNoise(r,:);
ActualLabel(i,:) = FeatureExtractedData(r,5);
InputData = TestPoint(:,1:2:3:4);


PredictedLabel(i,:) = trainedClassifier.predictFcn(InputData);
% PredictedLabel = predict(SVMModels{j},InputData);
% plot([TrainingDataNoise TrainingData]);
fprintf('%.1f\t',ActualLabel);
fprintf('%.1f\t',PredictedLabel);

if ActualLabel(i,:) == PredictedLabel(i,:)
    Score(i,:) = 1;
else
    Score(i,:) = 0;
end

end
plot([TrainingDataNoise FeatureExtractedData]);
```

6.13 Appendix M - Equipment and Price List

Component	**Price**
Elegoo MEGA 2560*	£12.99
Battery Snaps*	£4.19
16 Channel PWM Driver*	£5.99
OpenBCI Ganglion	£188
Mechanical Hand*	£64
Bluetooth Dongle for Ganglion	£10
9V Batterys	£5
Electrode Pads - Ag/AgCl (30 Pack)	£5.84
Total	**£296.01**

'*' - Signifies the item has been funded myself therefore should not count towards the project budget. The items: **OpenBCI Ganglion + Bluetooth dongle**, **9V Batteries** and **Electrode Pads** were **provided** by the University and did not incur any extra charges for the purposes of this project.