ExoAD

EMG Controlled Hand Exoskeleton Assistive Device

Final Report

Michele D'Mello, Diana Harasym, Natalie Hazzouri, Sara Jamil

Electrical and Biomedical Engineering Design Project 4BI6
Department of Electrical and Computer Engineering

McMaster University
Hamilton, ON, Canada

Table of Contents

Background

Loss of hand control is an issue affecting a wide range of individuals as a result of injury, neuromuscular conditions or the effect of age. Common symptoms include loss of coordination and loss of strength. Hand and fine finger control become a problem. It becomes more difficult to pick things up and hold onto objects firmly. These symptoms progressively lead the at-risk population to drop objects more frequently. In order to maneuver the hand as desired, electromyography (EMG) signals have been explored to control an exoskeleton device. Those with loss of hand control are unable to produce enough action potentials to carry out movement for themselves. An exoskeleton device can be utilized to carry out daily tasks or function as a method of rehabilitation for the individual. The objective is to design a device fitting the user to control the desired gestures of grasp and pinch using their EMG signals as input.

A motor unit consists of a single peripheral neuron and its innervated muscle fibers. They serve as the 'final common pathway' for all motor commands. [7] Motor units are slowly lost over the first sixty years of one's life, and afterwards at a faster pace. More variable discharges of motor units during muscle contractions and lower amplitudes are both associated with aging and may cause weak EMG signals.

EMG is a diagnostic procedure to assess the health of muscles and the nerve cells that control them (motor neurons). Motor neurons transmit electrical signals that cause muscles to contract. [7] In essence, motor unit action potentials (MUAPs) from numerous motor units are superimposed to make up EMG signals. Thus, less motor units, less action potentials to superimpose leads to reduced amplitude. Table 1 below shows the results from a study on the effects of aging on EMG variables. [8]

Asterisks show significant differences between the young and elderly subjects.

	Young	Elderly		All subjects
Force (kg)	22.78 ± 2.72	17.05 ± 4.68	**	19.91 ± 4.75
ARV (μV)	413.76 ± 147.22	274.06 ± 109.39	*	343.91 ± 145.53
MDF (Hz)	63.68 ± 7.15	52.29 ± 7.52	**	57.98 ± 9.24
MFCV (m/s)	4.73 ± 0.99	3.85 ± 0.56	*	4.29 ± 0.90
CC	0.87 ± 0.05	0.92 ± 0.02	**	0.90 ± 0.04

Valus are Mean±S.D. *p<0.05, **p<0.01

Table 1: Muscular force and EMG variables during maximum voluntary contraction

From this table, we can observe that the amplitude of the EMG signal for an elder is reduced by a significant amount and can be claimed statistically significant. The frequency does not vary by a notable quantity.

Literature Review

Numerous hand exoskeleton devices have been developed or are in the process of development, by research groups in order to solve the problem of loss of hand control. These devices function for rehabilitative or assistive purposes, with various differences in the type of actuators used to cause the movement of the exoskeleton, the sensing method of the intended motion, and the transmission of the actuator force to the exoskeleton itself to induce movement. The type of actuator used ranges from electric (servo, stepper, and DC motors) to pneumatic (pistons) and even includes shape memory alloys. The transmission of the actuated motion to the exoskeleton or the mechanical structure used also varies, with there being direct actuation, cable systems, linkages systems, and belt driven systems. In order to accurately sense the user's intended motion, force sensors, flex sensors, pressure sensors, electromyography, and mechanomyography were used [11]. The following are reviews of notable exoskeleton devices.

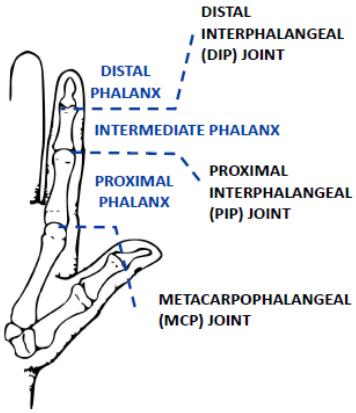

Figure 1: Finger Joints Labelled

In the mechanical design of the hand exoskeleton, it is important to recognize the natural movements of the hand in order to produce a comfortable motion of the finger joints. The human fingers have three joints with four degrees of freedom per finger. From the distal end, the joints are the DIP (distal interphalangeal), the PIP (proximal interphalangeal), and the MCP (metacarpophalangeal). The DIP and PIP joints have the flexion/extension degree of freedom, while the MCP joint has both the flexion/extension and abduction/adduction degrees of freedom.

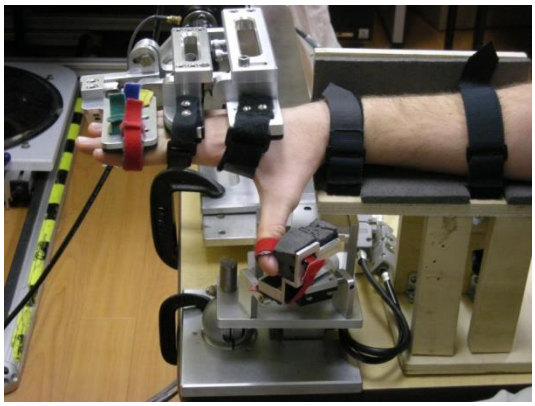

Figure 2: HEXORR

The Hand Exoskeleton Rehabilitation Robot (HEXORR) developed by Schabowsky et al. was designed mainly for stroke patients, for both continuous passive movement aimed at rehabilitation and for active force assisted movement. It is made up of two stationary modules, one for the fingers, and one for the thumb. The finger module has a four-bar-linkage system, which is able to rotate the fingers at all the joints, including the MCP and PIP joints. The modules themselves are driven using electric motors, and the sensing of the intended motion is based on torque sensors. The advantage of this design is its ability to provide almost full range of motion of every joint of the hand [6]. However, the current design is based on a stationary exoskeleton that remains on a table, which is sufficient for rehabilitation purposes, but is inadequate as a day-to day assistive device. Also, it is not able to move fingers individually, meaning it can only do a grasping or closing hand motion, not the more specialized hand motions needed for day-to-day tasks such as pinching.

Hasegawa et al. developed an assistive exoskeleton that controls the hand and wrist. The device is a cable driven system that actually controls each joint independently, allowing for a wide range of motion. The index finger and thumb are controlled individually, and the middle ring and little finger are controlled together, with no individual motion. The cables are driven using motors and activated with EMG signals. The advantage of this device is the wide range of motion of each individual joint as well as wrist movement, which has a total active DOF of 11. It also uses a sophisticated algorithm that estimates the grasping force based on the amplitude of EMG signals without the use of a force sensor and is able to differentiate between when the user would need assistance for tasks that require stronger grasping compared to other hand motions where they do not need assistance [12]. However, including the wrist in the range of motion for the hand complicates the design, and might not be possible in the time frame of this project.

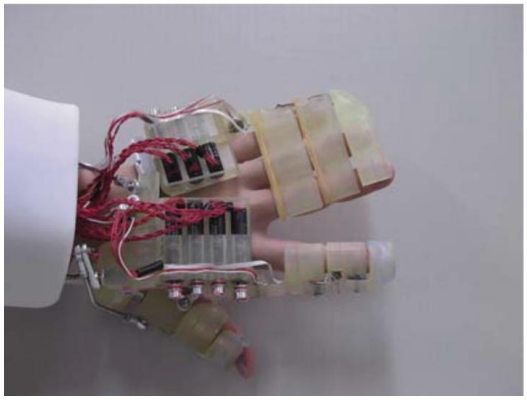

Figure 3: Assistive Exoskeleton

The EMG-Controlled Hand Exoskeleton for natural pinching developed by DiCicco et al. was designed to target those with spinal cord injuries which result in partial paralysis while the brain stays fully functional. The lightweight orthotic exoskeleton is designed to restore dexterity to paralysed hand by using the user's available EMG signals to assist in the execution of daily activities. Some of their design goals include providing the pinching motion by articulating the index finger to the thumb, ensuring normal interaction with real world objects by minimizing the exoskeleton material on the palmar side of the hand, and minimizing the weight of the device. We decided to adopt these design goals for our implementation of the hand exoskeleton. In terms of their mechanical design, the flexion/extension of the DIP and PIP joints are coupled, but the DIP/PIP and MCP flexion/extension are independent. Passive abduction/adduction movement of the fingers are allowed and aid the finger in conforming to its target object.

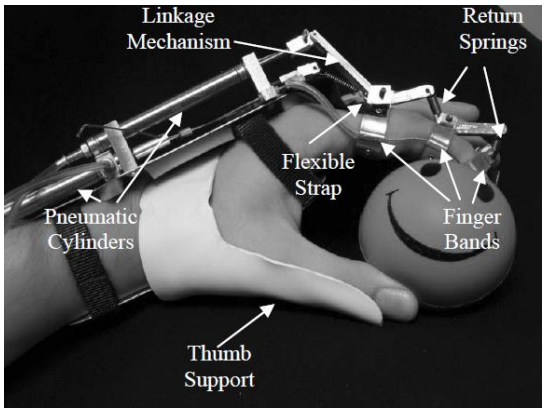

Figure 4: EMG-Controlled Hand Exoskeleton

There is also a wide variety of algorithms being used in EMG signal analysis to better categorize different hand motions and individual finger motion, in order to control the exoskeleton more accurately. It is important to realize that EMG signals are biological signals that have a lot of noise and variability associated with them which makes the classification difficult as the EMG signal will never be identical for the same gesture. Many current approaches use artificial neural networks and support vector machines (SVM) as the main classification algorithm to tackle this problem. Some applications also include preprocessing of the signal to prepare and amplify the signal for the subsequent steps and to reduce noise artifacts. This process consists of a feature extraction step followed by a dimensionality reduction technique, principal component analysis (PCA), to simplify the input to the classifier.

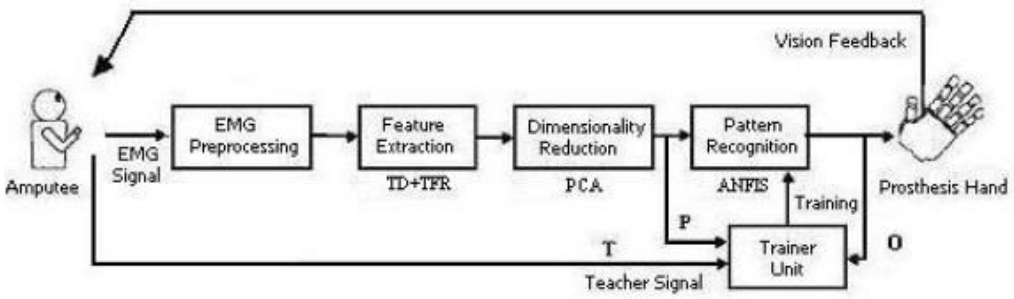

Figure 5: Signal Processing

Proposed Solution

The ExoAD is our proposed solution. It is an exoskeleton that uses EMG signals to assist with hand movement. This assistive device amplifies the existing (weak) EMG signals, and the exoskeleton helps reinforce the desired hand motion. The aim of this device is to assist the aging population and partially paralyzed patients in day-to-day activities. This device can also be used for therapy and rehabilitation purposes by helping to strengthen the hand muscles. This may help prevent or delay the further onset of muscle atrophy.

Most day-to-day activities can be completed using two major hand movements: grasping and pinching. For the purpose of this project, grasping is defined as seizing and holding something tightly. Thus, our focus is to close the hand around an object firmly. Pinching is defined as the act of gripping between the index finger, the middle finger and the thumb simultaneously. For rehabilitation purposes, we focus on the flexion and extension of the hand as a whole, which is essentially the grasping motion without an object. Flexion and extension of individual fingers is beyond the scope of this project.

The initial design is comprised of a cable system driven by motors. Two sets of cables will run through ring-like linkages between the joints of the fingers. The first set of cables will be parallel to the finger to induce flexion, while the other set will be directly on the top surface of the finger to induce extension. An alternative to the extension cables is the use of elastics on the top surface that automatically extend the fingers once the motors stop flexion. The ring linkages couple the movement of the DIP and PIP, allowing passive abduction and movement of the MCP joint. Our goal is to keep the palm and underside of the hand free of cables and motors, to ensure normal interactions with real world objects. The ring linkages will be designed in AutoCAD and 3D printed to allow the incorporation of the cables as well as a low cost design. Different motors such as the servo, step, and DC motor will be tested based on their size, power, and ease of integration into the design. Based on the available space on the top of the hand, as well as the specific hand motions the device will accomplish, one motor will control the index and middle finger together, one motor will control the ring and little finger together, and one motor may possibly control the thumb, or the thumb will remain in a locked position. However, the separate ring linkages on each of the fingers were designed to allow for individual finger control in the future, or if time permits.

In order to prevent the device from straining the user's hand or possibly breaking an object, feedback sensors will be placed on individual fingers to determine the maximum force that can be applied to the object during the hand motion and to stop the hand at that position. This maximum force applied to an object can be found experimentally, or can be a configurational setting. Another option is to use a power threshold on the motors that allows fingers to flex only to a certain position.

Our aim is to collect and process the EMG signal of the grasping, pinching and extension of the hand and process the signals to be able to differentiate between those three movements in real time. The Myo armband is used to record the EMG signals and send it via Bluetooth. The final product will be designed to have all the components attached to a glove that can easily be put on and removed. The goal is to have a light and non-bulky design, in order to promote day-to-day use, as well as for comfort. A low cost design is also important to allow availability for all patients.

System Description

The overall device consists of signal processing components and mechanical design. The ExoAD acquires EMG data from the user's arm using a Myo armband. The data is then processed in the server and accurately classified as one of our defined gesture classes: nothing, grasp, pinch, or release. The microcontroller activates the actuators on the exoskeleton to execute the user's desired motion.

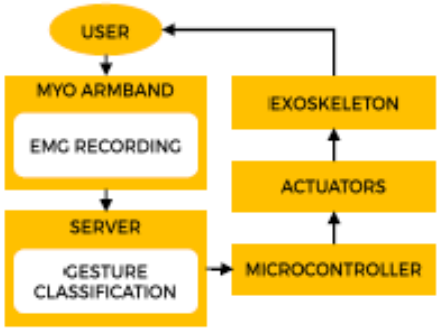

Figure 6: Overview of ExoAD Control

The server processes the data from the Myo in MATLAB and then sends the classified results to the Arduino microcontroller. The Myo Armband communicates in C only. Thus a Myo-MATLAB binding is required to use C code in MATLAB to record and analyze data from the Myo. The Myo transfers the data it records to MATLAB via a built-in Bluetooth connection. In order to make our device wireless, we have implemented Bluetooth connections between MATLAB and the Arduino using an HC-05 Bluetooth Module. Figure 7 illustrates our communication platforms; the Myo is able to send the EMG data to MATLAB via Bluetooth and MATLAB is able to send the desired motor output to the Arduino also using Bluetooth.

Figure 7: Communication Platforms

Mechanical Design

The device uses two servo motors to control a cable and ring linkage system that acts as an exoskeleton. Rings are placed on the phalanges of each finger. For simplicity, the thumb is fixed using a thumb splint. Wire is threaded through from the top (to control extension) and from the sides (to control flexion). One motor controls the first two fingers (index and middle) and the other controls the last two fingers (ring and little). When the first motor is activated the pinch movement can be executed and when both motors are activated at the same time, the grasp movement can be completed.

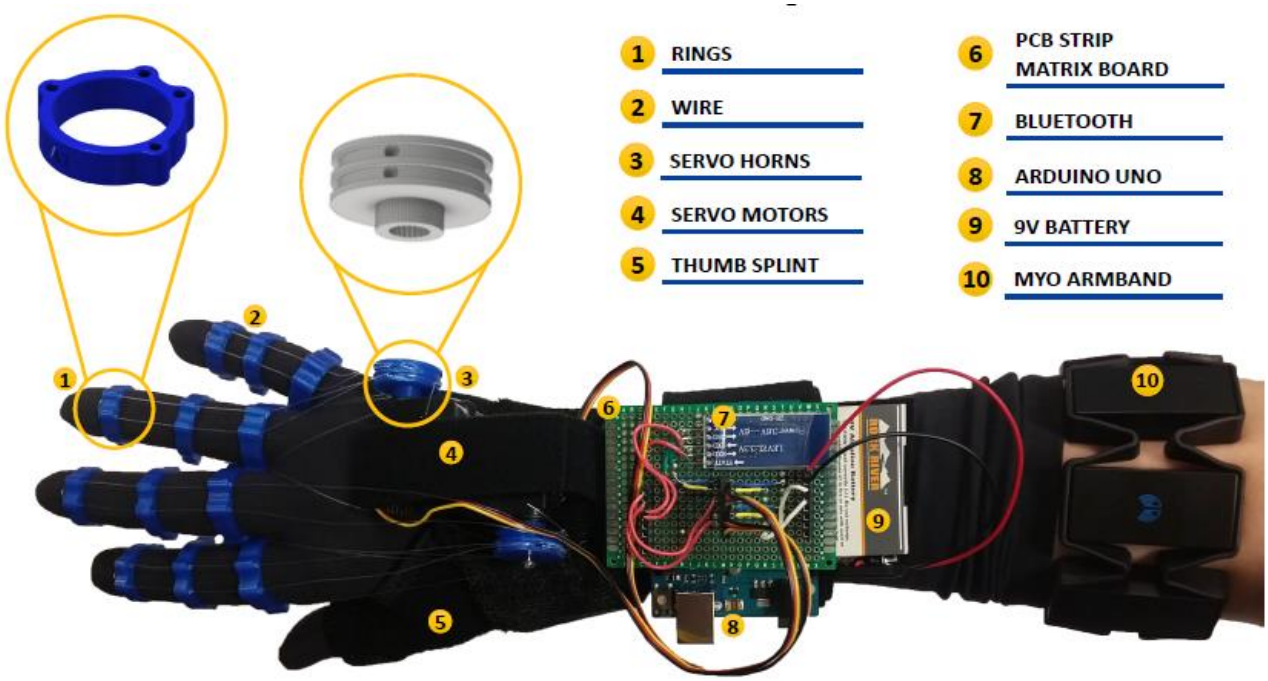

Figure 8: ExoAD Device

Two sets of cables (each for flexion and extension) are used for the exoskeleton rather than one cable (flexion) and an elastic (automatic extension), due to the added strain on the motor to counteract the extension force from the elastic. To overcome this problem a higher torque motor would need to be used. However, the more power a motor has the larger its size, which would not fit on the hand. Fishing wire is a good option for the cable due to its strength, durability and low cost.

We tested multiple ring designs for both durability and comfort. Our goal was to keep the palm and underside of the hand free of cables and motors, to ensure normal interactions with real world objects, yet still bend the fingers in the correct direction. The 'Mickey Mouse' design as seen in Figure 8 was able to achieve our design goals. The design was also created to be easily 3D printed by generic printers (not complicated or too small/thin) to make sure costs stay low. We also tested different angles between the flexion loops on the sides of the ring to choose the optimal design. There was a tradeoff between angles that would cause the rings to interfere with each other resulting in discomfort and interference with objects held at the palm of the hand. Also, certain angles of the loop positions would require a larger torque from the motors to be able to produce the same force at the finger tips.

We chose a servo motor over a stepper or DC motor because it requires less current (doesn't need an external power supply), its angle can be controlled precisely, and it produces the best torque for its size. We chose to control two sets of fingers because we couldn't fit enough motors on the hand to control fingers separately. Also, it is more complicated to characterize the EMG for individual fingers. The thumb is too complex to be controlled with a basic servo, due to its wide range of motion. Hence, a thumb splint is used to fix the thumb in place. The thumb splint also allows for the wrist to be stabilized, in order for the wrist angles not to interfere with the tension of the cable system. Custom servo motor horns were developed as seen in Figure 8 and 3D printed as well. They allow the cables to be organized in such a way that prevents knotting or any other type of cable interference during motor movement. The servo horns also increase the force or pressure of the hand gesture, to allow the user to actually grip an object, by pulling the cables a longer distance.

The glove used is made from a thin breathable material (not a thermal glove) and can be comfortably worn for long periods of time. The glove also protects the users hand from friction due to cable contact with the surface of the hand in some areas, mainly between the MCP joints. Pressure sensors were not used, as they were an unnecessary addition due to the current power of the motor, which does not provide enough force or pressure to damage an object or cause any injury to the user. The power source for the device is a 9V battery. A rechargeable USB power bank was tested, but did not have sufficient current to power the Arduino and the 2 servo motors at the same time. We chose the Arduino microcontroller, because it is a well-known and versatile microcontroller and is able to directly communicate with MATLAB. The Myo is a less bulky alternative to surface EMG electrodes, and it can easily be removed and placed on which makes this device more realistic for everyday use.

Design Cost and Equipment

The following parts and equipment are required for the realization of the ExoAD, which totals to approximately $375, as shown in Table 2. Due to the generous support from Thalmic Labs and McMaster's EPIC lab, we were able to alleviate the total cost of the project.

3D Printed Material (Rings, Servo Horns)	Approx. 20$
Glove	27.16 $
Thumb Stabilizer	18.50 $
Wires (Fishing Line)	Approx. 1 $
Microcontroller (Arduino UNO R3)	Approx. 16 $
Hitec HS-85BB+ Servo Motors (x2)	56.25 $
Myo Armband	Approx. 200 $
Interlink 400 FSR Pressure Sensors (x2)	19.05 $
HC-05 Bluetooth Module	13.55 $
PCB ProtoBoard and Headers	5.74 $

Table 2: Cost of Equipment

Signal Processing

After examining the raw EMG data pertaining to the specific hand gestures we were trying to classify, we found that the signals are indistinguishable and cannot feasibly be resolved using simple magnitude thresholding techniques. Figure 9 shows some examples of the pinch and grasp gesture motions and Figure 10 shows the EMG data after enveloping. Since thesholding techniques were inadequate in differentiating between these signals, machine learning was employed to examine other features of the data that can be used in classification.

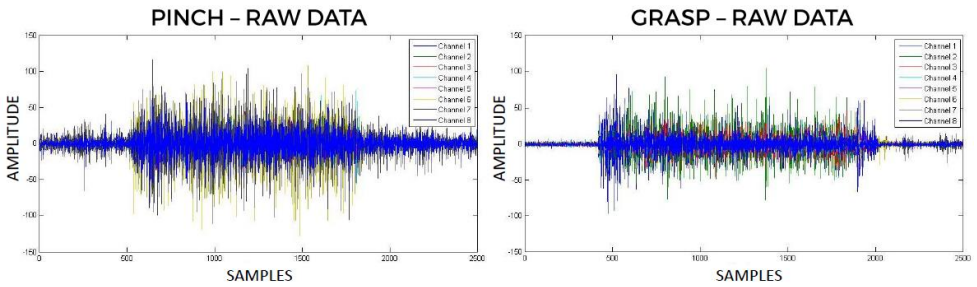

Figure 9: Raw EMG Data for Pinch and Grasp

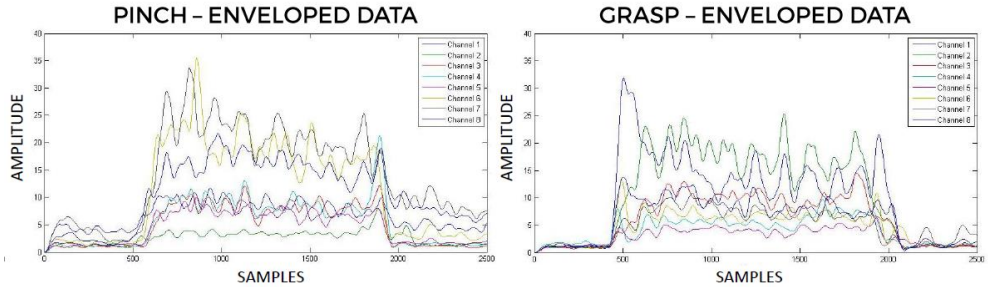

Figure 10: Enveloped EMG Data for Grasp and Pinch

The overall signal processing system that was used in gesture classification is shown in the figure below. After the EMG signals were recorded, the preprocessing steps played a crucial role in the correct classification of the signals. Feature extraction involved a process of finding and testing the key features of the data that would provide the best classification rate using machine learning algorithms. Feature selection involved existing algorithms that were used to determine which features were most relevant in distinguishing between gestures and which were redundant and can be removed in order to improve classification. The machine learning algorithm used to perform gesture classification was a support vector machine (SVM) algorithm. Furthermore, the process was implemented in a state machine that helped to improve the classification rate by separating the classifiers. The data was split into training and testing sets where 10-fold cross-validation was performed to obtaining the accuracy rate of the classifiers. While this figure shows the process involved for training the data, preforming real-time classification of the signals only requires calculation of the selected features and the output is determined from a set of weights that were obtained from the training process.

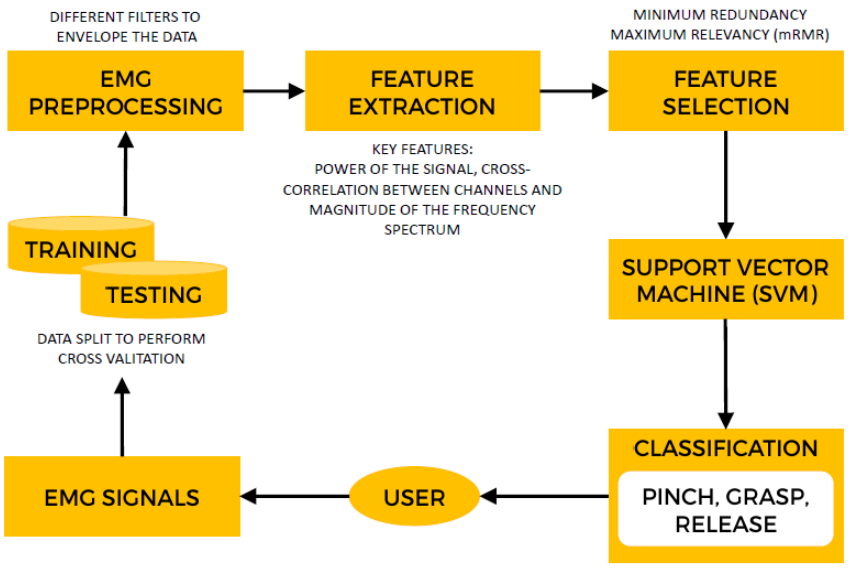

Figure 11: Gesture Classification

Preprocessing the Data

The first step in preprocessing the data was to ensure that the correct outputs were given to the EMG samples for training. The data was recorded in a controlled environment where the position of the arm, the timing of the motions, and the pressure on the object to pinch or grasp was recorded to ensure similar performance on every trial. We recorded data of natural pinch and grasp motions and the trials were split into 100 sample windows. This ensures that the classifier is able to classify with a delay of approximately 0.5 seconds in order to provide pseudo-real-time performance.

However, when recording the data for training, some problems were encountered with the synchronization of the Myo clock to the MATLAB counter resulting in a mismatch of data points between trials. Due to this problem and the fact that there is varied delay in reaction time between trials, it was essential to look through the enveloped data and determine when the flexion onset and release occurred to ensure that the correct class output was given for each of the 100 sample time windows. This improvement alone increased the classification rate by about ten percent.

Furthermore, additional preprocessing of the data was required before the feature extraction process could begin. This involved enveloping the data by taking the absolute value of the raw data and low-pass filtering the data using different cut-off frequencies and different order filters. The resulting data for the trials looks like the data shown in Figure 10 above. The filters used had a cut-off of 2Hz and 5Hz, and these two filters seemed to provide the most amount of information after feature extraction was performed. These filters were applied to each of the 100 sample time windows to produce the same results as the real-time implementation.

Feature Extraction

In machine learning, feature extraction is the process of deriving features or values from an initial dataset that is intended to provide useful information for facilitating the learning and generalization steps. After testing several different features extracted from the enveloped and raw data of the EMG signals, the key features that we found are highlighted below.

The first feature that we examined was the power of the signal. This consisted of calculations of the variance, standard deviation, and mean absolute value of the windowed signal. The overall values for all the channels were found as well as the values for each of the eight electrodes. This allowed for a comparison between the overall power of the signal between different motions as well as the subtle differences between the power of the channels that could help to distinguish pinch and grasp gestures.

The correlation coefficients were also determined from the dataset as a set of features obtained from the preprocessed data. The correlation coefficients are a measure of the linear independence of the EMG channels. This feature provided good indication of whether a pinch or grasp gesture is being performed. This may be due to the fact that the grasp motion sometimes involves extension to grasp larger objects which would be evident by the correlation between the different channels as some channels detect more of the extension movement than others.

The frequency domain of the signals was also examined by obtaining the magnitude spectrum of the fast Fourier Transform (FFT) of the raw signal. The mean of the magnitude spectrum was taken over 20Hz intervals.

Further testing on taking the cross-correlation with delay and taking the FFT of the cross-correlation was unable to be conducted due to limitations in memory encountered while performing feature selection.

Feature Selection

Feature selection is the process of selecting a subset of relevant features, which are variables or predictors, for use in model construction using the SVM. The benefits of using a feature selection algorithm include simplification of the models for interpretation purposes, shorter training times, and enhanced generalization by reducing overfitting.

The feature selection algorithm that we used was the minimum Redundancy Maximum Relevance (mRMR) algorithm. Maximum relevance is characterized by mutual information as a measure to define the dependency of variables. This algorithm is able to calculate the best features of the dataset given the desired number of features from the total number of features in the set. We found that the number of desired features plays an important role in determining the classification rate. The optimal number of features was obtained by running the training process in a loop to determine the number of features that produced the minimum error rate. The mRMR algorithm is available in two versions based of the calculation of mutual information difference (MID) and mutual information quotient (MIQ). Both were tested and the MIQ feature selection algorithm was used in the final implementation because the classification rate was slightly better than that of MID. The most relevant features according to the algorithm were the correlation coefficients as well as some of the FFT and power features.

Support Vector Machine

The machine learning tool that was used in our project was the Support Vector Machine (SVM). SVM is a type of supervised machine learning algorithm, meaning that the correct outputs of the trials must be given in the training phase. SVM is able to perform binary classification by constructing a hyperplane that separates the two classes in the higher dimensional space produced by the feature set. The goal is to choose the hyperplane that maximizes the distance between the points in the two classes; hence SVM is also referred to as a Large Margin Classifier.

The SVM training function on MATLAB involves some parameters such as the choice of the kernel function. Two different kernel functions were tested, a linear kernel and a Gaussian kernel. The kernel used in our final implementation was the Radial Basis Function (Gaussian kernel) due to its improved performance. The optimization method that was used for solving the quadratic programming problem was the sequential minimal optimization (SMO) algorithm. While SVM is meant to distinguish between two different classes, SVM can be modified for multi-class classification by creating several models using a one-vs-all method.

Classification with State Machine

In the initial implementation of the classifier, we attempted to classify between all our classes at once using multi-class SVM classification. At first we attempted to classify between 6 different classes: nothing, release, pinch, grasp, pinch onset, and grasp onset. However, we found that trying to differentiate the onsets of the motions was particularly difficult in real-time due to their short time intervals. Therefore, we included the onsets with their respective pinch/grasp class and our multi-class SVM was able to accurately differentiate between all 4 classes.

However, we were able to achieve higher classification rates by separating the classes into two classifiers. The idea was to split the classes into two classifiers that would be able to perform binary classification between two classes rather than four or six. The resulting classifiers would be put into a state machine that performs logical operations to check which state the hand is in at the current time to determine which classifier will be used next. Different implementations were tested but the final implementation that produced the best results is shown in the state machine diagram in Figure 12.

In the real-time implementation of the classification, the state machine is initialised to state zero, meaning that it assumes the hand is in resting position. In state zero, it will use the flexion/extension classifier to determine whether the hand is still at rest (class = nothing) or whether a flexion of the hand has occurred (class = grasp or pinch). Once flexion has been detected, it immediately sends the same signal to the grasp/pinch classifier which determines whether the flexion that occurred was a grasp or a pinch and then sets the state to state one. Once in state one, the state machine assumes you are still grasping or pinching the object and the flexion/extension classifier is again used to determine if the hand is still flexed or if an extension has occurred (class = release). Once extension has occurred the state machine is set back to state zero where the hand is at rest. The appropriate signal output is sent to the Arduino device to control the motors once the half-second signal is classified.

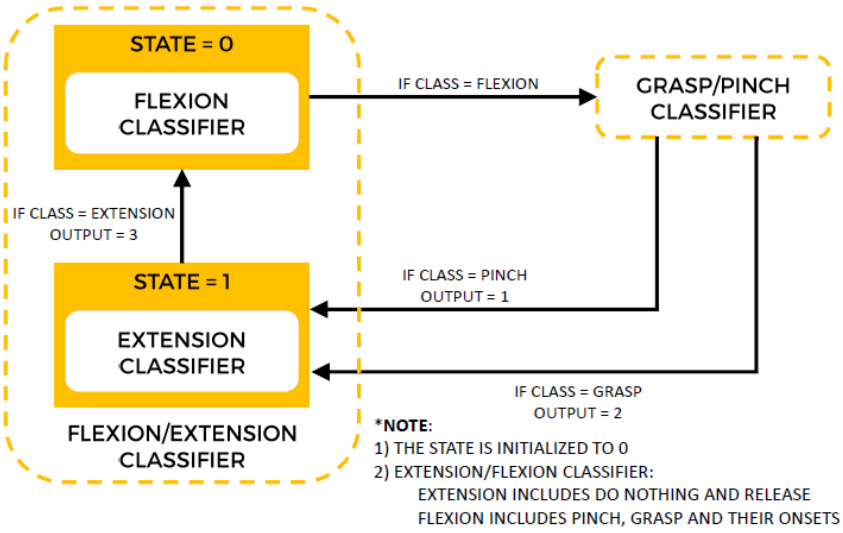

Figure 12: State Machine

In the training phase of these classifiers, the output of the training data was modified to fit the new sets of classes. The flexion/extension classifier used the entire dataset to distinguish between flexion, which included the pinch and grasp classes, and extension, which included the nothing and release classes. The grasp/pinch classifier only used a subset of the data pertaining to the grasp and pinch classes only to distinguish between the two different flexion classes.

This version of the state machine classifies one half-second signal at a time and sends the output to the Arduino rather than continuously collecting data. Classifying one 100 sample window may cause the misclassification of false positives due to errors in the output. We attempted to classify several sample windows at a time and checking whether they agree. However, when it was tested we did not find many problems associated with errors while checking several windows caused issues with missing the gestures altogether and a longer delay between the performed action and the motor response.

Results

The classification rate used to determine the accuracy of the classifiers was obtained using 10-fold cross validation. The use of separate classifiers in a state machine produced the highest classification rates, as explained the previous section. The following table displays the classification accuracies for the test data set.

SINGLE CLASSIFIER	
TYPE	**ACCURACY**
NOTHING VS GRASP VS PINCH VS RELEASE	86.0%
MULTIPLE CLASSIFIERS	
TYPE	**ACCURACY**
EXTENSION VS FLEXION	91.1%
GRASP VS PINCH	99.4%

Table 3: Classification Rates for Test Data Sets

Discussion

The fingers move in two main ways: flexion and extension. Meanwhile the thumb has two joints that flex and extend, however, it also has an additional joint that allows for several unique movements such as circumduction, abduction, adduction, and retropulsion. Thus, the thumb must be treated separately. Moreover, another difficulty encountered lies within the wrist. It must be stabilized depending on the motor placement. Since the motors will be fixed on top of the wrist, the thumb and the wrist will be fixed by a thumb stabilizer in order to simplify the design.

Mechanical design can effectively produce the desired gestures however limitations in the motor torque does not provide sufficient force. The design consists of three rings, one located on each phalange of the finger. These rings along with the cable and ring linkage system help control the DIP and PIP joints to get the right finger movement. However, with this design, the MCP joint required to attain the desired knuckle movement cannot be controlled. Thus future improvements do include a band around the hand acting as a 'fourth ring'. Furthermore, there is a trade-off between the size and the power of the motor. In this model, we chose motors of an ideal size. In future models of this device, a stronger motor is desired to produce ideal pressure when holding objects. Refining the type of cable used can also help improve the desired force of the hand gestures. In the current design, the fishing line connection loosens after continuous use, decreasing the amount of pressure the gestures produce. Finding an alternative to fishing line that produces a seamless connection without the need for tying off the cable would help solve this problem.

In terms of the signal processing, the changes that caused the largest improvement in classification rate were ensuring the correct output for the training data, using feature selection algorithms, and separating the classifiers by using a state machine. After suitable feature extraction was performed, determining the subset of the most relevant features helped to produce the best performance for each classifier. Using the separate classifiers in the state machine provided the lowest error rate that was able to successfully distinguish between the different classes in the real-time implementation. The classification rate is inherently limited by the specifications of the Myo armband. The sample rate is relatively low for EMG applications and the large electrodes allow for crosstalk. Improved real-time performance could be achieved by implementing classification and training with an adequate microcontroller to avoid problems incurred with MATLAB due to time delays or memory allocation problems. Real-time performance can also be achieved by retraining the data with the ExoAD device on instead of training with natural pinching and grasping motions without the device. Future improvements include the addition of force feedback by using pressure sensors to control the amount of pressure output at the fingertips if motors that produce large torques are to be used.

Sustainability

Producing a sustainable product was also one of our design goals in the development of the ExoAD. Some of the key points regarding social, economic, and environmental impact are discussed below.

- **Social**
  - The patient will experience the greatest positive effect, being able to complete many day-to-day tasks without assistance from others due to the mobility of the device.
  - Patients will also be able to perform rehabilitation exercises in the comfort of their own home. This will allow them to do these exercises more frequently in order to strengthen their hand motion, decreasing recovery time.
  - The device will help decrease the strain on the health care system by allowing patients to perform rehabilitation exercises any time, without the need of supervised physiotherapy. It will free up time for healthcare providers to focus on other patients.
- **Economic**
  - The device is cost effective due to 3D printed materials and simplistic design, which allows the device to be more accessible to patients.
  - The cost of the product allows it to be affordable for patients in any income bracket.
  - Decrease in rehabilitation recovery time and the ability to perform tasks without assistance will allow patients to be able to work and gain income, which can contribute to the economy.
- **Environmental**
  - Biodegradable 3D printing material (for mass production).
  - Glove and thumb splint can be reused for different purposes.
  - Electrical components do use valuable natural resources, but they can be recycled at certain facilities or returned to be refurbished and reused.
  - Rechargeable power supply (for mass production).

Appendix A - Software

Matlab Code

Machine Learning Script

```
% SVM Two Classifiers Train Test
clear all; close all; clc;
% NOTE:
% CLENCH = GRASP
% NF = Extension/Flexion classifier
% CP = Grasp/Pinch classifier

load XrawData
load YData

[YDataNF, YDataCP, idxCP] = TwoClassifiers3(YData);
XrawDataCP = XrawData(idxCP);

%% Features of all data for cvpartition
XFeaturesData1 = featureSelectionWin2(XrawData); %2350 x 204
XFeaturesData2 = featureSelectionWin2(XrawDataCP); %1166 x 204

%% Partition into training and testing
% Get the index for testing/training cross validation
CVO1 = cvpartition(YDataNF,'Kfold',10);
err1 = zeros(CVO1.NumTestSets,1);
CVO2 = cvpartition(YDataCP,'Kfold',10);
err2 = zeros(CVO2.NumTestSets,1);

%% NF Classifier
display('MIQ');
display('NF Classifier:');
% for n1 = 5:20
%     display(' ');
%     display(['Number of mRMR features: ', num2str(n1)]);
for i = 1:CVO1.NumTestSets
    trainIdx1 = CVO1.training(i);
    testIdx1 = CVO1.test(i);
    % NF
    Xtrain1 = XFeaturesData1(trainIdx1,:);
    Xtest1 = XFeaturesData1(testIdx1,:);
    Ytrain1 = YDataNF(trainIdx1,:);
    Ytest1 = YDataNF(testIdx1,:);

    nFeatures1 = 15;

    mRMRfeatures1 = mrmr_miq_d(Xtrain1, Ytrain1, nFeatures1);
%     save mRMRfeatures1NF mRMRfeatures1

    Xfeaturestrain_mrmr1 = zeros(size(Xtrain1,1),nFeatures1);
    Xfeaturestest_mrmr1 = zeros(size(Xtest1,1),nFeatures1);

    for j = 1:nFeatures1
        Xfeaturestrain_mrmr1(:,j) = Xtrain1(:,mRMRfeatures1(j));
        Xfeaturestest_mrmr1(:,j) = Xtest1(:,mRMRfeatures1(j));
    end

    %% SVM Train
    tic;
    models1NF = svmtrain(Xfeaturestrain_mrmr1, Ytrain1,'kernel_function','rbf');
```

```
%     save models1NF models1NF
    toc;

    test1 = svmclassify(models1NF,Xfeaturestest_mrmr1);
    err1(i) = sum(Ytest1 ~= test1); %mis-classification rate
    display(['The error rate 1 is ', num2str(100*(err1(i)/length(Ytest1))), '%']);

    % 'kernel_function' = 'rbf'
    % 'rbf_sigma' - scaling factor in the gaussian radial basis function
    % default = 1
    % 'method' - method used to find the hyperplane
    % default = 'SMO' - Sequential minimal optimization (L1 soft margin SVM)
    %           'LS' - least squares (L2 soft-margin SVM)
    % 'autoscale'

end

cverr1 = 100*(sum(err1)/sum(CVO1.TestSize));
display(['>>> The total error rate 1 is ', num2str(cverr1), '%']);

% end

%% CP Classifier
clear i j;

display('-------------------------------------------');
display('MIQ');
display('CP Classifier:');
% for n2 = 5:20
%     display(' ');
%     display(['Number of mRMR features: ', num2str(n2)]);
for i = 1:CVO2.NumTestSets
    trainIdx2 = CVO2.training(i);
    testIdx2 = CVO2.test(i);
    % CP
    Xtrain2 = XFeaturesData2(trainIdx2,:);
    Xtest2 = XFeaturesData2(testIdx2,:);
    Ytrain2 = YDataCP(trainIdx2,:);
    Ytest2 = YDataCP(testIdx2,:);

    nFeatures2 = 10;

    mRMRfeatures2 = mrmr_miq_d(Xtrain2, Ytrain2, nFeatures2);
%     save mRMRfeatures2CP mRMRfeatures2

    Xfeaturestrain_mrmr2 = zeros(size(Xtrain2,1),nFeatures2);
    Xfeaturestest_mrmr2 = zeros(size(Xtest2,1),nFeatures2);

    for j = 1:nFeatures2
        Xfeaturestrain_mrmr2(:,j) = Xtrain2(:,mRMRfeatures2(j));
        Xfeaturestest_mrmr2(:,j) = Xtest2(:,mRMRfeatures2(j));
    end

    %% SVM Train
    tic;
    models2CP = svmtrain(Xfeaturestrain_mrmr2, Ytrain2,'kernel_function','rbf');
%     save models2CP models2CP
    toc;

    test2 = svmclassify(models2CP,Xfeaturestest_mrmr2);
    err2(i) = sum(Ytest2 ~= test2); %mis-classification rate
    display(['The error rate 2 is ', num2str(100*(err2(i)/length(Ytest2))), '%']);
```

```
end

cverr2 = 100*(sum(err2)/sum(CVO2.TestSize));
display(['>>> The total error rate 2 is ', num2str(cverr2), '%']);

% end
```

Feature extraction function

```
function [Xfeatures, Xenv] = featureSelectionWin2(data)
% Filter
Fs = 200; % Sample rate of the EMG data

% First butterworth fc = 2, n = 3
N1 = 3;
Fc1 = 2;
[B1,A1] = butter(N1,(Fc1/(Fs/2)), 'low');

% Second butterworth fc = 5, n = 10
N2 = 10;
Fc2 = 5;
[B2,A2] = butter(N2,(Fc2/(Fs/2)), 'low');

trials = length(data);
channels = 8;

% Enveloping
for i = 1:trials
    rectify_clench{i,1} = abs(data{i,1});
    rectify_clench{i,2} = abs(data{i,1});
%     rectify_clench{i,3} = abs(data{i,1});
    envelope_clench{i,1} = filter(B1,A1,rectify_clench{i,1});
    envelope_clench{i,2} = filter(B2,A2,rectify_clench{i,2});
%     envelope_clench{i,3} = filter(B3,A3,rectify_clench{i,3});
    Xenv{i,1} = envelope_clench{i,1};
    Xenv{i,2} = envelope_clench{i,2};
%     Xenv{i,3} = envelope_clench{i,3};
end

% Features:
% variance, standard deviation, mean absolute value
% taken over 100 sample data for ALL CHANNELS
% and taken over each channel
Xfeatures = [];

for i = 1:trials
    var_channels = zeros(1,16);
    std_channels = zeros(1,16);
    mav_channels = zeros(1,16);
    for k = 1:2
        for j = 1:channels
            var_channels((k-1)*channels+j) = var(Xenv{i,k}(:,j));
            std_channels((k-1)*channels+j) = std(Xenv{i,k}(:,j));
            mav_channels((k-1)*channels+j) = mean(Xenv{i,k}(:,j));
        end
    end

    % slope for third filter only? 1:25? on data?
    var_slope = var(Xenv{i,2}(76:100,:)) - var(Xenv{i,2}(51:75,:));
    std_slope = std(Xenv{i,2}(76:100,:)) - std(Xenv{i,2}(51:75,:));
    mav_slope = mean(Xenv{i,2}(76:100,:)) - mean(Xenv{i,2}(51:75,:));
```

```
    max_slope = max(Xenv{i,2}(76:100,:)) - min(Xenv{i,2}(76:100,:));

    cc_raw = corrcoef(data{i,1});
    cc_env1 = corrcoef(Xenv{i,1});
    cc_env2 = corrcoef(Xenv{i,2});
%     cc_env3 = corrcoef(Xenv{i,3});
    corrcoef_raw = (cc_raw(logical(triu(true(size(cc_raw)))-eye(8))))';
    corrcoef_env1 = (cc_env1(logical(triu(true(size(cc_env1)))-eye(8))))';
    corrcoef_env2 = (cc_env2(logical(triu(true(size(cc_env2)))-eye(8))))';
%     corrcoef_env3 = (cc_env3(logical(triu(true(size(cc_env3)))-eye(8))))';

    feature_fft = zeros(1,40);
    for j = 1:channels
        Xfeaturefft{i,1}(:,j) = fftshift(abs(fft(data{i,1}(:,j), 200)));
    end
    feature_fft(1:8) = mean(abs(Xfeaturefft{i,1}(1:20,:)));
    feature_fft(9:16) = mean(abs(Xfeaturefft{i,1}(21:40,:)));
    feature_fft(17:24) = mean(abs(Xfeaturefft{i,1}(41:60,:)));
    feature_fft(25:32) = mean(abs(Xfeaturefft{i,1}(61:80,:)));
    feature_fft(33:40) = mean(abs(Xfeaturefft{i,1}(81:100,:)));

%     xcorr_raw = xcorr(data{i,1});
%     xcorr_env1 = xcorr(Xenv{i,1});
%     xcorr_env2 = xcorr(Xenv{i,2});
%     xcorr_feature = [...
%         xcorr_raw(100)...
%         xcorr_raw(120,:) xcorr_raw(140,:)...
%         xcorr_env1(100,:) xcorr_env1(120,:) xcorr_env1(140,:)...
%         xcorr_env2(100,:)...
%         xcorr_env2(120,:) xcorr_env2(140,:)...
%         ];
%
%     xcorr_fft_raw = zeros(1,320);
%     xcorr_fft_env1 = zeros(1,320);
%     xcorr_fft_env2 = zeros(1,320);
%     for j = 1:(channels*channels)
%         Xfftxcorr{i,1}(:,j) = fftshift(abs(fft(xcorr_raw(:,j))));
%         Xfftxcorr{i,2}(:,j) = fftshift(abs(fft(xcorr_env1(:,j))));
%         Xfftxcorr{i,3}(:,j) = fftshift(abs(fft(xcorr_env2(:,j))));
%     end
%     xcorr_fft_raw(1:64) = mean(abs(Xfftxcorr{i,1}(1:20,:)));
%     xcorr_fft_raw(65:128) = mean(abs(Xfftxcorr{i,1}(21:40,:)));
%     xcorr_fft_raw(129:192) = mean(abs(Xfftxcorr{i,1}(41:60,:)));
%     xcorr_fft_raw(193:256) = mean(abs(Xfftxcorr{i,1}(61:80,:)));
%     xcorr_fft_raw(257:320) = mean(abs(Xfftxcorr{i,1}(81:100,:)));
%     xcorr_fft_env1(1:64) = mean(abs(Xfftxcorr{i,1}(1:20,:)));
%     xcorr_fft_env1(65:128) = mean(abs(Xfftxcorr{i,1}(21:40,:)));
%     xcorr_fft_env1(129:192) = mean(abs(Xfftxcorr{i,1}(41:60,:)));
%     xcorr_fft_env1(193:256) = mean(abs(Xfftxcorr{i,1}(61:80,:)));
%     xcorr_fft_env1(257:320) = mean(abs(Xfftxcorr{i,1}(81:100,:)));
%     xcorr_fft_env2(1:64) = mean(abs(Xfftxcorr{i,1}(1:20,:)));
%     xcorr_fft_env2(65:128) = mean(abs(Xfftxcorr{i,1}(21:40,:)));
%     xcorr_fft_env2(129:192) = mean(abs(Xfftxcorr{i,1}(41:60,:)));
%     xcorr_fft_env2(193:256) = mean(abs(Xfftxcorr{i,1}(61:80,:)));
%     xcorr_fft_env2(257:320) = mean(abs(Xfftxcorr{i,1}(81:100,:)));
%
    Xfeatures = [Xfeatures;...
        var_channels std_channels mav_channels...
        var_slope std_slope mav_slope max_slope...
        corrcoef_raw corrcoef_env1 corrcoef_env2...
        feature_fft...
%         xcorr_feature...
%         xcorr_fft_raw...
```

```
        ];
end

end
```

Flexion/Extension classifier

```
function class1 = NFclassifier(data)

load models1NF
load mRMRfeatures1NF

Xraw{1,1} = data{1,1}(1:100,:);

XFeaturesData1 = featureSelectionWin2(Xraw);

n1 = size(mRMRfeatures1,2);

XFeatures_mrmr1 = zeros(1,n1);
for j = 1:n1 %mRMR nFeatures
    XFeatures_mrmr1(1,j) = XFeaturesData1(1,mRMRfeatures1(1,j));
end

class1 = svmclassify(models1NF, XFeatures_mrmr1);

end
```

Grasp/Pinch classifier

```
function class2 = CPclassifier(data)

load models2CP
load mRMRfeatures2CP

Xraw{1,1} = data{1,1}(1:100,:);

XFeaturesData2 = featureSelectionWin2(Xraw);

n2 = size(mRMRfeatures2,2);

XFeatures_mrmr2 = zeros(1,n2);
for j = 1:n2 %mRMR nFeatures
    XFeatures_mrmr2(1,j) = XFeaturesData2(1,mRMRfeatures2(1,j));
end

class2 = svmclassify(models2CP, XFeatures_mrmr2);

end
```

C Code

```
%  clear
%  clc

 %%
% [~, ind]=unique(f.aCounter);
% f.accData =f.accData(ind, :);
%
% [~, ind]=unique(f.aCounter);
% f.gyroData =f.gyroData(ind, :); %  save fist_1

% Cell Array

% trial = trial + 1;

%% State Machine

% State:
% 0 = Nothing
% 1 = Pinch
% 2 = Clench
% 3 = Release

% Classes:
% 1 = Nothing
% 2 = Clench Onset
% 3 = Pinch Onset
% 4 = Clench
% 5 = Pinch
% 6 = Release

figure;
state = 0; % Initialize state ONCE to nothing
% make sure to save state and load for next run
count = 0;

while (count<=7)

f=myo_2c_sfunc;
[~, ind]=unique(f.eCounter);
f.emgData =f.emgData(ind, :);

EMG_data={}; % initialize before you start
trial = 1; % initialize trial counter
EMG_data{1,trial}=f.emgData;

%%%% CHANGE COUNTER TO < 1000 FOR ~100 SAMPLES

if(state == 0)
    classNF = NFclassifier(EMG_data);
       if (classNF==2)
         classCP = CPclassifier(EMG_data);
         if (classCP==1)
             output = 2;
             state = 1;
             display('GRASP');
             close;
```

```
            figure; imshow(pic_grasp);
        else %(classCP==2)
            output = 1;
            state = 1;
            display('PINCH');
            close;
            figure; imshow(pic_pinch);
        end
    else %(classNF==1)
        output = 0;
        state = 0;
       display('NOTHING STATE 0');
    end
elseif (state == 1)
    classNF = NFclassifier(EMG_data);
       if (classNF==1)
             output = 3;
             state = 0;
             display('RELEASE');
        close;
        figure; imshow(pic_extend);
       else %(classNF==2)
             output = 0;
             state = 1;
             display('NOTHING STATE 1');
    end
else
    display('Error: Incorrect State');
end

display(['>>> state =', num2str(state)]);

fprintf(com,'%i',output);
% pinch=1, grasp=2, extend=3

count = count +1;
clear EMG_data;

end
```

Arduino Code

```
#include <SoftwareSerial.h> //include the serial library
#include <Servo.h> // include the servo library

//Defining the TX-11 and RX-10 (instead of 0 and 1)
SoftwareSerial BTSerial(10, 11); // HC-05 object (Rx, Tx)
Servo LServo;  // create a servo object for right side
Servo RServo;  // create a servo object for right side
int BluetoothData; // the binary data given from comp
char gesture;

void setup()
    {
    //Activate digital pin outputs connected to motors
    LServo.attach(8); // attaches the servo on pin 8 to the L servo object
    RServo.attach(9); // attaches the servo on pin 9 to the R servo object
    // The default baud rate for communication is 9600
    Serial.begin(9600); // Arduino to comp
    BTSerial.begin(9600); // comp to Arduino
    BTSerial.println("Bluetooth on...");
    // initialize motors
    //    LServo.write(170);     // extension
    //    RServo.write(0);   // extension
    //    gesture='E';
    }

void loop()
    {
  // READ from HC-05 and WRITE to Arduino Serial Monitor
      if (BTSerial.available()) {
        BluetoothData = BTSerial.read();
        if (BluetoothData==48){ // nothing

        }
        else if (BluetoothData==49){ // pinch
          LServo.write(0);   // flexion
          RServo.write(0);   // extension
          gesture='P';
        }
        else if (BluetoothData==50){ // grasp
          LServo.write(0);   // flexion
          RServo.write(170);     // flexion
          gesture='G';
        }
        else if (BluetoothData==51){ //extension
          LServo.write(170);     // extension
          RServo.write(0);   // extension
          gesture='E';
        }
        Serial.write(BluetoothData); // write back to comp
        Serial.print(" | Gesture: ");
        Serial.print(gesture);
        Serial.print(" | Bluetooth output: ");
        Serial.println(BluetoothData,DEC);
      }

  // READ Arduino Serial Monitor and WRITE to HC-05
      if (Serial.available()) {
        BTSerial.write(Serial.read()); // write back to Arduino
      }

 }
```

Appendix B - Hardware

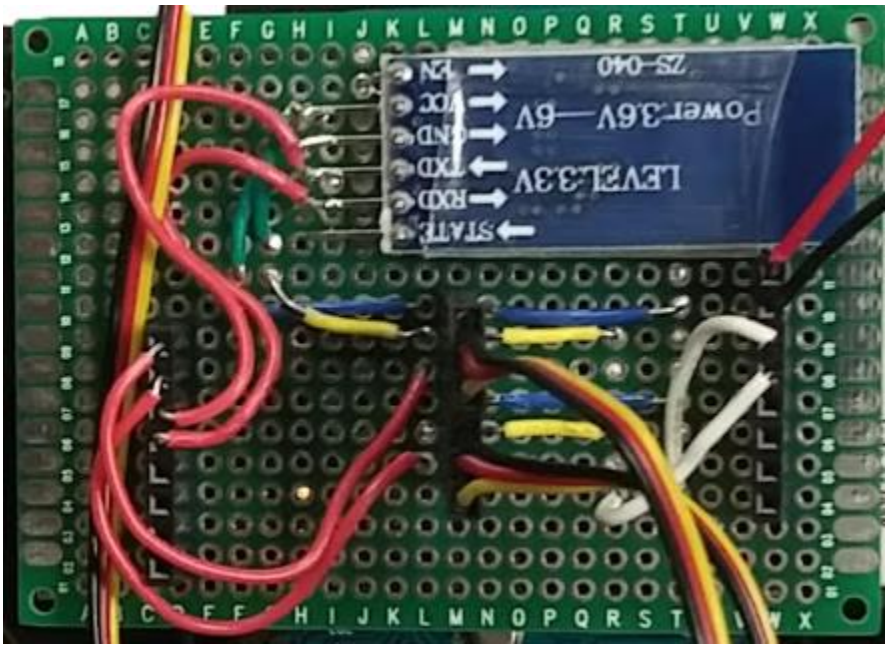

Figure 13: PCB Strip Matrix Board Circuit

Appendix C – 3D Models of Printed Parts

Figure 14: Rings for index and middle finger

Figure15: Rings for ring and pinky finger

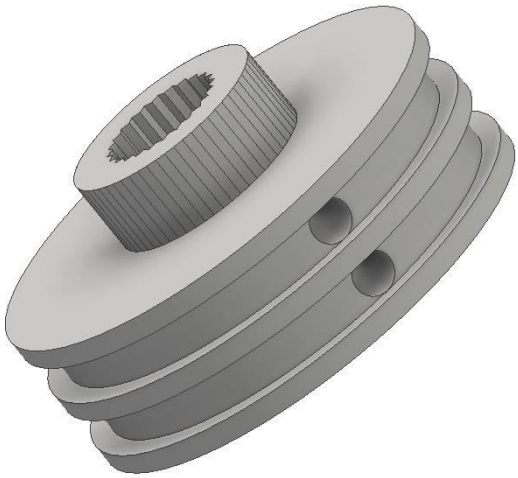

Figure16: Motor horns

Bibliography

[1] Marcello Mulas, Michele Folgheraiter and Giuseppina Gini, "An EMG-controlled Exoskeleton for Hand Rehabilitation," IEEE, Chicago, 2005.

[2] Lenny Lucas, Matthew DiCicco, and Yoky Matsuoka, "An EMG-Controlled Hand Exoskeleton for Natural Pinching," NSF, Pittsburgh, 2004.

[3] Matthew DiCicco, Lenny Lucas, Yoky Matsuoka, "Comparison of Control Strategies for an EMG Controlled Orthotic Exoskeleton for the Hand," IEEE, Pittsburgh, 2004.

[4] H. Peng, "mRMR (minimum Redundancy Maximum Relevance Feature Selection)," 2005. [Online]. Available: http://penglab.janelia.org/proj/mRMR/. [Accessed 2016].

[5] TM Manini, SL Hong, BC Clark, "Aging and muscle: a neuron's perspective," Ncbi, 2013.

[6] Christopher N Schabowsky, Sasha B Godfrey, Rahsann J Holley and Peter S Lum, "Development and pilot testing of HEXORR: Hand Exoskeleton Rehabilitation Robot," *Journal of NeuroEngineering and Rehabilitation,* 2010.

[7] M. C. Staff, "Electromyography (EMG)," Mayo Clinic, 25 October 2012. [Online]. Available: http://www.mayoclinic.org/tests-procedures/electroconvulsive-therapy/basics/definition/prc-20014183.

[8] Hiroshi Yamada, Morihiko Okada, Toshiaki Oda, Satoko Nemoto, Tomomi Shiozaki, Tomohiro Kizuka, Shinya Kuno, and Tadashi Masuda, "Effects of Aging on EMG Variables During Fatiguing Isometric Contractions," J. Human Ergol., Japan, 2000.

[9] Nisim Benjuya, Steven B.Kenney, "Myoelectric Hand Orthosis," *American Academy of Orthotists & Prosthetists,* vol. 2, pp. 149-154, 1990.

[10] Mahdi Khezri and Mehran Jahed, "Real-time intelligent pattern recognition algorithm for surface EMG signals," Biomedical Central Ltd, 2007.

[11] Pilwon Heo Gwang Min Gu, Soo-jin Lee, Jywhan Rhee and Jung Kim, "Current Hand Exoskeleton Technologies for Rehabilitation and Assistive Engineering".

[12] Yasuhia Hasegawa, Yasuyuki Mikami, Kosuke Watanabe and Yoshiyuki Sankai, "Five-Fingered Assistive Hand with Mechanical Compliance of Human Finger," IEEE, Pasadena, 2008.