



Major Qualifying Project
Accurate Prosthetic Hand

Students: Mervyn A. Larrier, Jr.

Elina Saint-Elme

Casey Kracinovich

Dylan Renshaw

Advisor: Marko Popovic

Co-Advisor: Karen Troy

Abstract

The purpose of this project, Accurate Prosthetic Hand, is to explore a method by which to improve the dexterity of artificial hands by closely mimicking the biomechanics of a human hand. The biomechanics of each human finger was analyzed to determine the kinematics and finger trajectories that the experimental hand should be able to mimic. In addition to the kinematics and finger trajectories, the biomechanics analysis was used to determine what materials were needed to simulate tendons and musculature of the human hand. The mechanical system of the experimental hand is actuated using several stepper motors controlled by electroencephalogram (EEG) signals and electromyograph (EMG) signals. The majority of the experimental hand's motions are controlled using EEG, with three distinct thoughts executing three distinct grips: pinch, hook, and point. EMG signals are used for finer motor control, such as controlling the strength of each grip pattern. The completion of this project resulted in a prosthetic hand capable of nine degrees of freedom as well as the creation of a control system that relies on sensory input from the mind and body; all while preserving the important biomechanical data that allows the human hand its unique dexterity.

Acknowledgements

Thanks to our advisors Marko Popovic and Karen Troy. Also, thanks to all of the members of Popovic Labs, particularly Yunda Li, Michael Pickett, Matthew Bowers, and Chinmay Harmalkar.

Executive Summary

I. INTRODUCTION

A common problem amongst modern upper-limb prostheses is their level of functionality. Body-powered prostheses like the quintessential hook prosthesis are limited in their dexterity because they often have no biomedical mechanisms, instead opting for lower cost, simple gripping mechanisms. Higher function biomechanically inspired prostheses, such as the Deka arm are often capable only of gripping and have convoluted controls. Even expensive hand currently on the market, such as the Bebionic arm which has multiple grip patterns, have fingers and a thumb that are locked in place at fixed distances from each other (though the Bebionic thumb may be manually adjusted between one of two positions). All of these hands translate have translated the human hand into a purely mechanical structure that sacrifices the dexterity and versatility of the limb they are meant to replace by ignoring the important biological features of the human hand. Also, many of the more affordable hands do not have any input from biological signals (higher end models carry EMG sensors, but they tend to be prohibitively expensive) and the ones that do still rely on buttons for many functions.

This project, inspired by the University of Washington paper *Design of a Highly Biomimetic Anthropomorphic Robotic Hand towards Artificial Limb Regeneration*, sought to provide an alternate method of design for an upper-limb prosthesis that would preserve the important biological characteristics of the human hand with improved functionality.

I. METHODOLOGY

Device Design:

To create the device, cadaveric scans of the hand distal to the radiocarpal joint were converted into stereolithography (.stl) files and modified using SolidWorks. The modifications included joint caps on the major joints distal to the carpals to simulate the effect of articular cartilage, threading holes, screw holes, and changing the geometry of the distal phalanges. Also, angled cuts were made to the metacarpals proximal to the carpals. This ensured that the hand back at a slight angle. The bones were then 3D printed with PLA filament. The soft tissue replacing musculotendon simulant (MTS) was made from EPDM rubber and was used to replicate the extensor expansion and tendon sheaths found in the hand. This allowed for curling and flexion of the fingers. Ligaments and the flexor and extensor tendons were simulated with braided fishing line. The lines replacing the extensor and flexor tendons were connected to a bank of stepper motors (seven at the minimum) which allowed for active actuation of the hand.

The MTS was attached to the bones via M1.4 screws and the fishing line to the stepper motors by spools. Further, a forearm casing was designed in which to hold the stepper motors and a secondary device was created in order to allow the fingers to achieve lateral motion. Finally, there are two features of the device that do not align

biologically: the carpometacarpal (CMC) joint of the first digit and the wrist. The CMC joint of the first digit was changed from a saddle joint to a ball and socket joint and then constrained in order to ease manufacturability. The wrist was changed to a universal joint because it would allow for the same range of motion as a biologically accurate wrist while still being manufacturable. Lines connected to an extra stepper motor in the forearm allow for lateral deviation (radial and ulnar) as well as for extension and flexion of the wrist.

Controls Design:

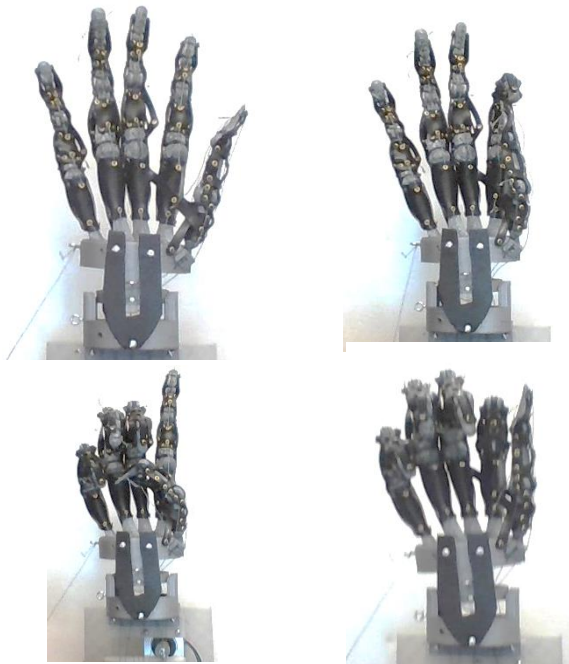
To control the device, a combination of EMG and EEG signals as well as pressure sensors were used. The pressure sensors served the purpose of providing the hand with some pressure sensitivity. The EEG signals were used for broad motions, i.e. the three distinct grips. EMG signals were used to facilitate finer motor control, such as controlling the strength of the grip. The EMG sensors were placed in specific areas on the arm to ensure somewhat consistent data, and the EEG signals were collected via a cheaply purchased, single electrode Bluetooth headset capable of collecting the brain's beta waves.

II. PRELIMINARY RESULTS

Mechanical Testing

The purpose of the mechanical testing is to ensure that our project meets specific design deliverables; one of them being that the prosthetic hand is capable of consistently gripping with 10% of the gripping force of a human hand, which is the equivalent of 1 lb.

Before completing the testing for design deliverables, preliminary mechanical testing had to be completed. The hand must achieve three grip patterns: pinching, pointing and a fist.



Controls Testing

Beyond the development of a complex prosthetic produced for a relatively cheap price, this project ultimately resulted in a platform that is dependent on manufactured EMG and EEG sensors. The EMG sensor was provided by Myoware, and the EEG sensor was provided by Neurosky. As part of the preliminary testing, students tested the control scheme on about 30 participants at the Cambridge Science Festival. The results yielded that the control scheme is reliable, but the threshold for the voltage difference that's acquired from the signal processing equipment needs to be adjusted for each individual. In the formal testing completed for this project, only the students in the group were tested on. The final results yielded that the EMG can be used to actuate the hand with an 80% success rate, while EEG allows a 68% success rate.

III. DISCUSSION

Beyond the development of a complex prosthetic produced for a relatively cheap price, this project ultimately resulted in a platform upon which further development towards a new kind of prosthesis can be made. Further work on the hand will include adding a more robust EEG data collection system, refining the lateral motion of the fingers, adding more distinct grips, and bolstering the pressure sensitivity of the hand. It will also include more human testing, in order to gather more data on how the device and controls work for different people. Future work may also include the addition of a simple artificial neural network to ease calibration of the device and increase the number of possible grips it can perform. Thermal sensors may also be added in order to prevent the hand from interacting with anything that may damage it. Ideally the hand would also be able to be produced via injection molding, in order to reduce cost and manufacturing time. The EPDM used to make the MTS may also be replaced with another more wear resistant rubber however, neither of these changes are currently a primary concern.

Table of Contents

Abstract.....	2
Acknowledgements.....	3
Executive Summary	4
I. INTRODUCTION	4
I. METHODOLOGY	4
Device Design:	4
Controls Design:.....	5
II. PRELIMINARY RESULTS.....	5
Mechanical Testing	5
Controls Testing	6
III. DISCUSSION.....	6
Table of Contents.....	7
List of Figures	9
List of Tables.....	9
Chapter 1: Introduction	10
Chapter 2: Background Research.....	11
2.1 Existing Prosthetic Hands	11
2.2 Biomechanics of a Human Hand.....	15
2.2.1 Biological Manipulation of a human hand	16
2.3 Control Mechanisms for Prosthetic Hands	19
Chapter 3: Proposed Project Design	21
3.1 Project Goal.....	21
3.1.1 Project Overview.....	21
3.2 Design Specifications	24
3.3 Project Management	26
3.4 Design Decisions.....	26

Chapter 4: Implementation.....	28
4.1 Mechanical Design	29
4.1.1 Biomechanics	29
4.1.4 Tendon Sheathing	32
4.2 Control Theory for Hand.....	34
Chapter 5: Analysis and Testing Results	35
5.1 Mechanical Grips and Force Testing.....	35
5.3 Signal Testing.....	36
5.3.1 EMG Signal Analysis	36
5.3.2 EEG Signal Analysis.....	38
5.3.3 Signals Testing Results	39
Chapter 6: Conclusion and Future Work.....	41
Bibliography	42
Appendix	43
Appendix A: Bill off Materials.....	43
Appendix B: Diagram of ECDU edupack.....	44
Appendix C: Programming	44
Mechanical Test: Fist.....	44
Mechanical Test: Pinch.....	51
Mechanical Test Point	59
EEG: Thought 1	78
EEG: Thought 2	96
EEG: Thought 3.....	106
Appendix D: CAD FILES	116
Progression of Muslutendon Simulant.....	116
Thumb Design	117

List of Figures

Figure 1: Hosmer Hook - http://hosmer.com/products/hooks/	12
Figure 2: The Open Hand Project Prosthetic - http://www.openhandproject.org/	13
Figure 3- The bebionic hand - http://bebionic.com/	13
Figure 4: Johns Hopkins Univerity Applied Physics Lab Prosthetic Arm - http://www.jhuapl.edu/newscenter/pressreleases/2014/141216.asp	15
Figure 5: From Beasely’s Surgery of the Hand: The bones of the hand and carpus & Beasely’s: the ligaments of the fingers.....	18
Figure 6: From: https://depts.washington.edu/msatlas/haexthood.html extensor hood and central slip.....	19
Figure 7: Index Finger Point - https://pixabay.com/en/photos/index%20finger/	23
Figure 8: Gripping Hand - http://www.fotosearch.com/LIF136/ga238003/	23
Figure 9: Pinching Finger - https://hackaday.io/project/6273-1w-true-mppt	24
Figure 10: Bones Diagram	29
Figure 11: Dynamics of Bone Structure	29
Figure 12: Bond Diagram.....	31
Figure 13: Solidworks model of 3d printed bones.....	32
Figure 14: Various tested tendon sheathing methods, from left to right: straws, eye screws, 1/16 inch EPDM with grommets thread holes.....	32
Figure 15: Control Flow Chart for Prosthetic Hand	34
Figure 16: Execution of Grip Patterns	35
Figure 17: Prototype of EMG sensor holder	36
Figure 18: Fist - Differentiating Finger Signals.....	37
Figure 19: EMG - Fist.....	38
Figure 20: EEG 3 distinct thoughts	39

List of Tables

Table 1: Decision Matrix for Material Selection.....	28
Table 2: Force required for full articulation with various tendon sheathing methods (all units in lbs).	33
Table 3: Force Testing Results	36
Table 4: EMG Testing Results	39
Table 5: EEG Testing Results.....	40

Chapter 1: Introduction

Worcester Polytechnic Institute is an engineering university located in Worcester Massachusetts. As part of the graduating requirements for the school, students are expected to complete a final engineering project, each student following the requirements needed for each department. For this particular project, students created a prosthetic hand, combining their knowledge of biomedical engineering and mechanical engineering, while completing further research on the electrical engineering and computer programming needed to make this project a success.

The inspiration behind this project was first realized when students completed some preliminary background research. It was realized that existing prosthetic hands sacrifice dexterity for cost and vice versa. This means, that advanced prosthetic hands that look human like and can complete many degrees of freedom are expensive, in many cases reaching prices as high as \$11,000.00. Students believed that by creating a hand that is inspired by the human body, and controlled from sensory input, an inexpensive prosthetic hand that is dexterous could be achieved: hence providing advanced control for little money.

To achieve this goal, the hand completed for this project had three primary deliverables: for the hand to be biomimetic, controlled by electroencephalograph (EEG) signals, and controlled by electromyograph (EMG) signals. Biomimetic meaning that the hand physically looks like a human hand.

EMG signal acquisition is a method of acquiring electrical impulses from human muscles. EEG signal acquisition is a method of acquiring electrical impulses from the mind.

The remainder of this report explains the process in which students were able to achieve their goals. There is a Background section explaining preliminary research, an overview of project goals, an overview of the design of the project, and lastly the results of the project.

Chapter 2: Background Research

Prior to completing the prototype of the final design, students completed background research to understand exactly what components were necessary to design a hand. In doing so, students had to acquire knowledge about existing products in the market, the biomechanics of the human hand and the existing control mechanisms used in prosthetic hands today.

2.1 Existing Prosthetic Hands

There are a plethora of prosthetic hands that patients can purchase. Hands range in simplicity, some being as simple as a mechanical hook. Other hands are advanced, not only in mechanical design, but also in the control mechanisms for the hand. For example, there are existing hands that are invasive in nature, and require surgery to install into the human or surgery to change the nerve endings on the location of impact. This section entitled “existing prosthetic hands,” focuses on a brief explanation of select hands that explain the varying types of hand on the market today. The hands are Hosmer Hook, the Open Hand Project, the Bebionic Hand and a Johns Hopkins neuroprosthetic hand.

While there are many hands that are simple, students found that the Hosmer Hook, was as simple as a prosthetic hand could be. As shown in Figure 1, the Hosmer Hook allows for a wide variety of configurations, allowing the practitioner to clinically match the patient needs and performance goals to achieve maximum independence. The company was founded in 1912, by the Hosmer-Dorrance Corporation and continues the tradition of quality, durability, and performance. This hand sells for \$450.00, and remains to be the most affordable hand on the market, even with other contributions in technology since 1912. (1)



Figure 1: Hosmer Hook - <http://hosmer.com/products/hooks/>

With the creation of the 3D printer, this technology can be used to manufacture prosthetic hands for less money. The Open Hand Project is a company based on open-source technology. The company provides material such as manufacturing method, Computed Aided Design downloads, and electronic material for patients, doctors, or engineers to contribute to the advancement of prosthetic hands. The company is still a start-up, but provides much potential for the communities around the world to contribute to the effort of creating a functional, and aesthetically pleasing prosthetic hand. An image of one of their initial products can be found in Image 2. This prosthetic sells for \$1,633.68. (2)



Figure 2: The Open Hand Project Prosthetic - <http://www.openhandproject.org/>

The company Bebionic advertises as one of the most dexterous prosthetic hands available on the market today. Their website advertises their latest product, the Bebionic small as winning the New Mechanical Product of the Year Award in 2015. An image of the Hand can be found below in Figure 3.



Figure 3- The bebionic hand - <http://bebionic.com/>

While this hand is aesthetically pleasing and dexterous, there are still many shortcomings for the Bebionic products. While the company website does not provide prices for their products, the Bebionic small is advertised as costing \$11,000.00. The hand relies on the patient to mechanically push buttons to change different hand positions. The company does sell a hand that is controlled by EMG signals, however as shown in the videos on the website, the hand still

requires a mechanical release to transition between grips. However, despite these shortcomings, the company allows patients to program whatever grips are appropriate for them based on their day-to-day use. Unfortunately, the baseline price for the cheapest hand is 11,000.00 and the other hands sold by the company increase in price. (3)

In addition to prosthetic hands that are sold on the market, there are research initiatives that are aspiring to make prosthetic hands more reliable and appealing. The Johns Hopkins Applied Physics Laboratory has a Neuroprosthetics lab whose purpose is to integrate Brain Machine Interface with prosthetic limbs. As part of the research, the Applied Physics Laboratory has explored the use of EEG, and invasive methods of control that require surgery for the prosthetic hand to be controlled by the patient, as if the prosthetic were their own biological hand. (4)

Johns Hopkins University has had much success with their initiative. In Figure 4, Les Baugh is shown with two prosthetic arms. He became the first bilateral shoulder-level amputee to wear and simultaneously control two of the Laboratory's Modular Prosthetic Limbs. Most importantly, Les Baugh, was able to operate the system by simply thinking about moving his limbs. (5)



Figure 4: Johns Hopkins University Applied Physics Lab Prosthetic Arm - <http://www.jhuapl.edu/newscenter/pressreleases/2014/141216.asp>

Again, this prosthetic hand is not available for sale on the market yet. However according to the US Department of Veterans affairs, Les Baugh's hand is estimated to cost 6 million dollars. (6) Also, according to the US Department of Veterans Affairs, the average lifetime cost for prosthetics and medical care for loss of a single arm for a veteran of the Iraq or Afghanistan wars will be \$823,299. (7)

2.2 Biomechanics of a Human Hand

Human hands are incredibly dexterous and complex in terms of possible executable motions. This dexterity is what allows humans to manipulate a wide variety of objects and interact fully with their environment. Many robotic hands and prosthetics attempt to mimic this dexterity in order to increase their versatility, however, they often run into issues.

The most pressing issue is the conversion of all of the joints in the hand into revolute joints for simplicity of design and manufacture. While understandable and accurate to a degree, this robs the fingers of the hand of much of their natural dexterity and in the process, makes them less versatile.

In each finger, there are three joints responsible for allowing them to move as they do: the distal interphalangeal joint (DIP), the proximal interphalangeal joint (PIP), and the

metacarpophalangeal joint (MCP). As previously stated, these joints are often all modeled and translated as pin joints for the sake of simplicity, but this does not accurately describe their motion or function.

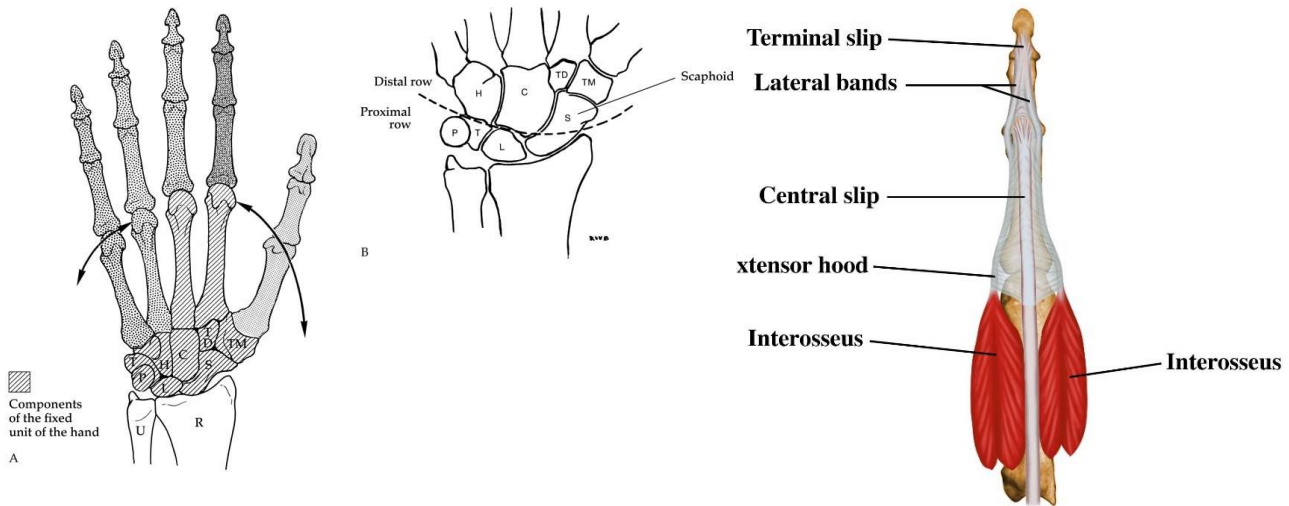
The DIP and PIP joints can both be modeled as constrained revolute joints, restricted to approximately 90° of rotation in palmar flexion and anywhere between 0° and 45° for dorsal extension. The MCP joint, unlike the others, is not a revolute joint at all. Rather, it's a spherical joint. As a result, its conversion to a revolute joint in each finger is where most of the dexterity of the hand is lost. The MCP joint of each of the fingers, being a spherical joint, has 3 degrees of freedom. It is capable of approximately 90° of flexion and extension (assisted). The medial adduction of each MCP can reach a maximum of approximately 45°, but most fingers are not able to achieve this because they are blocked by the other fingers of the hand. The same is true of lateral abduction. This feature of the MCP is crucially important to the biomechanics of the hand because it is the key feature that gives hands their many degrees of freedom and their versatility. Conversion of this joint into a revolute removes two degrees of freedom in each finger.

Beyond the translation of the MCP joint into a revolute joint, there also exists the problem of the inflexible wrist. Many prosthetic wrists are incapable of flexion and extension, as well as radioulnar deviation. They sacrifice these features in favor of supination and pronation. While a very useful function of the wrist, biologically supination and pronation occur at the superior radioulnar joint; not the inferior radioulnar joint, which is where radioulnar deviation, flexion, and extension occur.

These changes limit the dexterity and therefore the usability of modern prosthetic hands because they ignore important biological data. In the following sections, we will further investigate this biological data.

2.2.1 Biological Manipulation of a human hand

The human hand is made up of three basic but very important parts: the bones, the connective tissue, and the muscles. In order to understand how the hand moves the way it does, it is prudent to start with the basic kinematic structure of the hand.



The bones create the basic mechanical structure of the hand and allow for most of the motion the hand is able to produce. The superior phalanges are the bones that make up each finger. At the ends of each phalanx, sans the most distal end of the distal phalanx, is articular cartilage that allows for low friction motion of each phalanx relative to each other. The phalanges connect together via revolute joints to create a cantilever 3-bar linkage (save for the thumb which can be modeled as a 2-bar linkage) with a connection to a spherical joint at the grounded end of the mechanism. This spherical joint, the MCP joint, connects the superior phalanges to the metacarpals. The metacarpals experience very little motion relative to each other and can therefore be described as a structure, or a mechanism with no degrees of freedom. The metacarpals further connect to the carpals, the bones that make up the proximal section of the palm and the distal section of the wrist. The eight bones of the carpal are what allow for flexion and extension of the wrist. Because of their nature, as well as the way they connect to the

radioulnar joint, the wrist can be described as an ellipsoid joint. Therefore, the wrist has three degrees of freedom.

For all of the joints in the hand to work, they need to be held together. This job belongs to the connective tissue within the hand. The ligaments of the hand connect most features within it. The DIP, PIP, and MCP joints are all connected medially and laterally by collateral ligaments that span the gap in each joint. On the palmar surface of each of these joints is a thick band of ligament known as the volar plate. The volar plate is what restricts the revolute joints of the fingers in dorsal extension. The absence of such an analog on the dorsal surface of the fingers is what allows the

Figure 5: From Beasley's Surgery of the Hand: The bones of the hand and carpus & Beasley's: the ligaments of the fingers

grasping motion of the hand to be possible. Actuating this grasping motion is possible due to the flexor tendons, which are joined to multiple muscles in the forearm. The flexor tendons -divided into the flexor tendon sublimis and profundus – can be found on the palmar surface of each finger, and pass through a series of elastic sheaths known as the flexor tendon sheaths. These sheaths act as a linear pulley mechanism in each finger that translate the sliding motion of the flexor tendon and allow the hand to make grasping motions while still being able to hold objects. On the dorsal surface of each finger is what's known as the dorsal expansion. This can be subdivided into the extensor tendon and the tendon sheath. The extensor tendon (which can be further divided into the lateral bands and the central slip) is responsible for recovery of the fingers once they've flexed. That is, opening the hand once closed. The bone of the carpus (the wrist) are all held together by a web of interconnecting ligaments. (8)

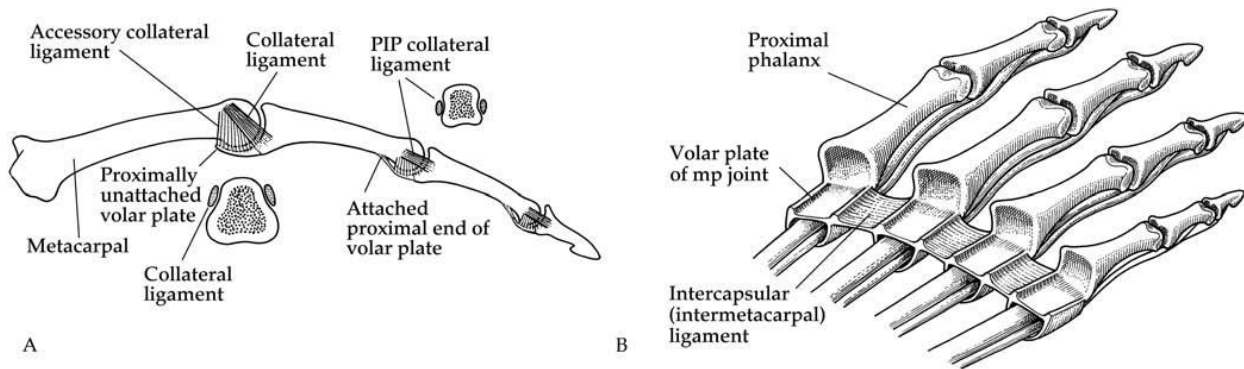


Figure 6: From: <https://depts.washington.edu/msatlas/haexthood.html>..... extensor hood and central slip

To control this very complex system requires the cooperation of the central nervous system (CNS) and the efferent nerves of the peripheral nervous system (PNS). Simplified, the thought to create motor function passes from the limbic system through the association cortex. The association cortex is what allows for recognition, selections, and integration of sensory motion to allow proper control of the hand. Movement has two ways of being transported from the brain to the muscles. The pyramidal tract transmits information that arises from the cerebral cortex. This tract is responsible primarily for fine motor control. The extrapyramidal tract transmits neural information that arises in the brainstem and is responsible for postural control and muscle control of flexion and extension of the hands and fingers. For the purposes of this project, the extrapyramidal tract is of chief importance.

2.3 Control Mechanisms for Prosthetic Hands

As mentioned previously in the Introduction section, two of the deliverables for the project is for the hand to be controlled by electroencephalograph (EEG) signals, and controlled by electromyograph (EMG) signals. EMG signal acquisition is a method of acquiring electrical impulses from human muscles. EEG signal acquisition is a method of acquiring electrical impulses from the mind. EEG and EMG signal acquisition is not new technology.

As early as 1875, EEG and identification of brain activity was discovered by a Russian physician named Richard Carlton. In 1911, Franklin Offner, a professor of biophysics at Northwestern University developed a prototype of the EEG that incorporated a piezoelectric inkwriter called a Crystograph to record brain activity. Since this time, medical communities have used improved methods of Offner's research to record brain activity to help diagnose medical ailments. (9) As for EMG, Cram and Stegar introduced a clinical method for scanning a variety of muscles using an EMG sensing device in 1980. (10)

Since this the purpose of this project is to find creative ways to use EEG and EMG practically for a prosthetic hand, students found a variety of research articles to help create a base line for their project. The West Pomeranian University of Technology, Szczecin wrote a paper that presents how to record and process the raw EEG signal from the MindWave, an inexpensive EEG headset sold to children. This paper then show how Fast Fourier Transform (FFT) is used to find the frequency components in a time domain signal. Based on a sample signal and spectral analysis, the basic EEG waves are presented. (11) In the International Journal of Computer Applications (0975 – 8887), there is another article explaining how to use the same headset to control a simple robotic car. (12)

EMG signal acquisition is a simple task, that involves putting electrodes on the arm, and purchasing an inexpensive sensor to pick up the signal. Students were originally interested to know if it was possible to use the EMG signal to differentiate between the different muscles that would normally move the fingers. An IEEE, article presented during the 1999, International Conference on Intelligent Robots and Systems provides a method of discriminating 10 motions using EMG signal acquisition by using a variety of mathematical transformations to filter the signal. The 10 motions are used for a prosthetic arm in the article.(13)

Chapter 3: Proposed Project Design

After completing the preliminary background research, the students developed a series of design specifications to create the prototype of this initial hand. In addition to this, students also had to create a project management plan to accomplish their tasks. Lastly, students worked together to identify what materials would be necessary to actuate a biomimetic hand.

3.1 Project Goal

The purpose of this project is to develop an improved control system for prosthetic hands through the use of EMG and EEG signal acquisition as well as a biomimetic design. As a result, the project is divided into two primary components: the biosignal control system and the physical prosthetic hand. The end goal for this project is to build a platform based upon the biosignals and physical control system that can be built upon and further developed. Ultimately, our main project goals can be listed as follows:

- To create a control system for a prosthetic hand which utilizes both EMG and EEG.
- To build a biomimetic prosthetic hand with which to test the control system.
- To test our control system's ability to perform a series of simple tasks.

3.1.1 Project Overview

This section explains a brief overview of how the hand will be actuated and controlled. Students considered how many degrees of freedom are necessary, which actuators would be useful, and how the basic control of the hand should operate.

3.1.1.A Actuators and Degrees of Freedom

Considerations for actuation and degrees of freedom include control and cost.

The considerations made for this project was whether or not to use an underactuated design. Regardless of performance, an underactuated design will be less expensive than a fully actuated design. However, it is likely that some basic motions will be intractable to accomplish with an underactuated hand, as every motion produced by the hand carries a significant degree of complexity.

The basic design uses artificial muscles and tendons driven by stepper motors to control the hand. Stepper motors were chosen because of their ability to hold a configuration as well as their relatively low cost. Servos were also considered, but ultimately rejected because while they are faster than stepper motors, they have far less holding torque and are much more expensive. The Stepper motor purchased for this project can be found in Appendix A, Bill of Materials.

In order to produce the most naturalistic motions with our design, we selected a biomimetic muscletendon-based design with the following actuator count:

- 2 actuator for the pinkie and ring finger, which are slaved to each other
- 2 actuators for the index and middle finger, each
- 1 actuator for all lateral finger motion
- 3 actuators x 1 thumb
- 2 wrist actuators

This totals to 12 actuators, all of which will be stepper motors.

3.1.1.B User Control & Operation & Hand Motions

For the control system of this project to be considered an improvement over existing inexpensive prosthetic hands, students needed the hand to offer the user the ability to control the hand naturally. Since students are using EEG and EMG, the basic control is for the user to initiate an action via a thought, and release the grip with EMG.

As mentioned previously in the project goals, the prosthetic arm should be capable of performing a series of simple tasks. Those tasks will be particular grip patterns that the user uses in everyday life.

By the end of this project, students would like to complete a minimum of 3 motions that are initiated through the signal acquisition of EMG and EEG. Those grip patterns are listed as followed: the active index form, the hook grip, and the precision pinch grip

Active Index Grip



Figure 7: Index Finger Point - <https://pixabay.com/en/photos/index%20finger/>

Pointing the index finger is a common hand motion used in everyday life. This single hand motion can be to touch a touch screen monitor, attract attention for questions, or point to areas of interest. As a result, completing the active index grip, is one primarily goal.

Hook Grip

Often times when a person is handling an object, such as picking up a bag, they are using this hook grip motion. The hook grip is perhaps the most common variation of the power grasp. This grip provides a lot of maneuverability because the same motion can be used for objects of varying size, such as a metal bar or a glass of water. Determining when to stop gripping will be determined by the control loop for the tactile sensing.



Figure 8: Gripping Hand - <http://www.fotosearch.com/LIF136/ga238003/>

Precision Tip Pinch Grip

One of the hand's most dexterous abilities is the ability to pinch. Therefore, part of this project will be exploring how to create high precision control by pinching small objects.



Figure 9: Pinching Finger - <https://hackaday.io/project/6273-1w-true-mppt>

3.2 Design Specifications

Project specifications:

The final design specifications are listed below.

- The hand will be biomimetic
 - o The articulations will be based upon the natural musculoskeletal features of the human hand
- The hand will be able to perform a precision pinch grip, a hook (power) grip, and a simple point.
- Have at least 4 functional DOFs to the hand- ideally, we want at least 6, and possibly up to 12. Below are the required DOFs, and a couple additional ones we will try to add.
 - o Extension/flexion of 4 fingers, slaved together (required)
 - o Extension/flexion of thumb (required)
 - o Bend wrist in two directions (required)
 - o Lateral motion for 4 fingers, slaved together

- 2 additional DOFs for thumb- lateral motion + additional extension/flexion
 - 2 DOFs for pinky and ring finger, which are slaved together
 - 2 DOFs for index and middle finger, which are slaved together
- The entire system (forearm section included) will weigh no more than the average weight of the male forearm/hand combination
 - Male because the cadaveric hand sample upon which the system will be based is that of a male
 - Entire system should be less than 5lbs
- The entire system should be less than or equal to 89% of the length the average forearm & hand to allow for attachment distal to the elbow
 - <16 inches long
- In the event that the EEG or EMG cannot be made to provide the input for the device, voice recognitions will be implemented instead
 - Configured to the individual user
 - Preset hand positions, with the ability to add more
 - Activation by a phrase: "Hand: [action to be completed]" for example
- Able to interpret the EMG signals to initiate actuators
 - 20 % success rate = 20% of the time, the arm is able to acquire the EMG signal and correlate it to a hand movement
- Able to interpret the EEG signal
 - 10 % success rate = 10% of the time, the arm is able to acquire the EEG signal and correlate it to a hand movement (i.e.: a thought initiates motor actuation)
- Able to consistently grip within 10% of the gripping force
 - 10% success rate = 10% the time, the measured force matches the ideal force
 - Testing
 - Ideal grip = 1 lb of force
 - The digital grip strength meter is tested with a trial & error process run
- Success and Failure conditions:
 - If a particular input is received but it does not yield the proper output. For example, if we have **A** and we're supposed to get **B**, we have a success. But if we have **A** and we get **C** instead, that's a *failure*.
 - If a particular input is received and it yields the proper output, but the output is not completed or half executed, that is a failure. For example, if the hand is supposed to perform a power grip but can only bring the fingers halfway in without performing the full grasp, that is a *failure*.
 - If a particular input is received and yields the proper output but cannot maintain the action, this is a failure. For example, if the hand is to point, but after completing the action immediately goes back to a neutral position, it is a *failure*.
 - If a particular input is received but no output is yielded, that is a *failure*.
 - If a particular input is received and it causes the system to spasm, that is a *failure*.
 - Success conditions, therefore, are:
 - The hand can properly match an input to an output 20% of the time.
 - The hand can properly and fully complete a particular action given a particular input.
 - The hand can maintain an action once given a particular input until otherwise prompted to complete another action.

3.3 Project Management

To manage the project, the MQP group was subdivided into two groups: prototyping and signals acquisition. The signals acquisition subgroup was responsible for the electronics and programming of the hand. They developed the code we used to translate specific thoughts and muscle movements into executable actions by the experimental hand. The prototyping subgroup was responsible for the development of the hand. This included translating the cadaveric bone scans into a mechanically functional hand as well as designing a wrist and lateral motion mechanism for the experimental hand. The two subgroups would meet regularly in order to facilitate clear communication as well as to discuss dependent design considerations. Clear communication was further facilitated via the use of Slack for group communication when members were unable to meet in person. Trello was used earlier in the project to keep tasks up to date and assigned to individual members, but as the lines became blurred between the subgroups later in the project and meetings occurred more frequently, this tool was abandoned for simplicity.

3.4 Design Decisions

As mentioned previously, the mechanical design of the project is inspired by the body. The design as referenced in the background research section of the report required for the muscular tendons to have elastic material properties. Students selected the materials by following a specific procedure.

To select a viable material for prototyping, a list of mechanical property requirements were determined based on the estimated size, range of motion, and loads that the material would have to handle. These criteria were then used to make a decision matrix to select viable

materials. The first set of criteria for preliminary consideration were the ductility, yield strength, and modulus. After this initial set of criteria were met, materials were then checked for price, fatigue strength, ability to be laser cut, and ease of purchase.

The metacarpophalangeal joints (MCP joint) range of motion creates the largest expected strain on the material at around 100% strain. If the material were to yield due to exceeding its strain at yield, it would occur at the MCP joint and permanently damage the material, which would change its modulus, and affect the motion of the hand. When selecting for the initial materials, a minimum ductility requirement of 120% strain was selected to account for the tolerance in material processing, and make sure the material does not stretch to the point of yield/failure. If the ductility is higher than 120%, the material will still be fit for purpose, so there was no maximum value applied to this criteria.

The maximum expected force on any section of the MTS was expected not to exceed seven pounds. This is due to the maximum force that the selected stepper motors can produce. These motors are the largest and strongest that would be expected to be feasible, so this seven pound force is a maximum for the current system, and likely the maximum for future systems. Using this force, and an estimated cross sectional area of 5mm^2 based on the geometry of the native tendon, a minimum yield strength can be determined. The material cannot yield under strain or stress, and as such, the yield strength of seven pounds over 5mm^2 , or about 1100 psi with a healthy tolerance, was selected. The material can have a higher yield strength, but this minimum in combination with the yield strain will prevent the material from tearing or failing under any circumstance.

Assuming the material would reach the minimum 120% strain at the minimum yield strength, a maximum modulus can be determined. The material needs to be able to achieve its full range of motion (the expected strain at the MCP joint) before the stepper motors exceed their maximum output. If the modulus is too high, it does not matter if the material won't fail because the material will prevent the fingers from curling, like trying to use a steel bar to hold

them in place. A maximum of 916 psi was set such that the force of the motors running at full power would have the capacity to create a full range of motion around the MCP joint, based on the original tolerances. A lower modulus would allow the same 120% strain to be achieved at lower stresses so a minimum modulus was not set.

Using these initial criteria and a piece of software called CES EduPack, a list of materials (see Appendix B) was generated. The program looked at the criteria, and any material in its database that matched that criteria was listed. These materials were then reviewed for cost, fatigue life, similarity to native tissue, and the ability to be laser cut and readily purchased. The final decision matrix used to select the material can be found in the table below.

Table 1: Decision Matrix for Material Selection

Material	Modulus	Yield strength	Yield strain	Fatigue strength	Laser cut?	Price
(CES EDUPack)	(psi)	(ksi)	(%strain)	(ksi)		(USD/lb)
EPDM	876	1.86	465	0.742	yes	1.64
Natural rubber	240	3.56	690	1.42	yes	0.976
Reinforced rubber	763	3.41	538	1.37	yes	1.46
Ethylene vinyl acetate	914	2.9	310	1.16	yes	2.57
Chlorosulfonated polyethylene	936	3.78	475	1.51	NO	2.7

Chapter 4: Implementation

The final design for the project used a series of 3d printed parts, stepper motors, screws, and the EPDM material. The assembly of the hand itself does not require complex manufacturing methods such as CNC to create. Instead, soft methods such as sewing the thread through the bones and EPDM, and screwing in pieces with a Philips screwdriver were the methods needed to assemble. The simplicity with manufacturing helps reduce the cost of the prototype. The Bill of Materials for the Hand can be found in Appendix A.

4.1 Mechanical Design

The Mechanical design of the hand was completely inspired by human anatomy. This section explains how the mechanical designs of the hand is analogous to the parts of the hand such as bones and joint caps, muscular tendon simulant, the thumb and the wrist.

4.1.1 Biomechanics

As previously mentioned, each finger of the hand, save for the thumb, can be modeled as a cantilever 3-bar linkage with the grounded end connected to a spherical joint. As detailed by the paper “*An analytical expression for the D.I.P.–P.I.P. flexion interdependence in human fingers*”, there is a relationship between the angle of the DIP joint and the PIP joint in flexion that can be seen below:

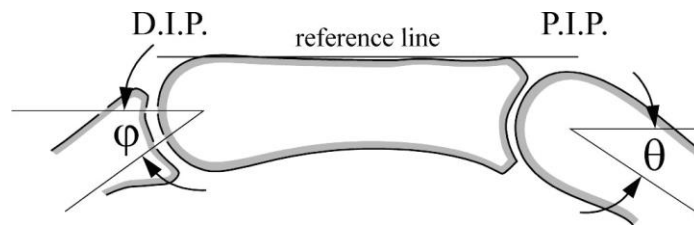


Figure 10: Bones Diagram

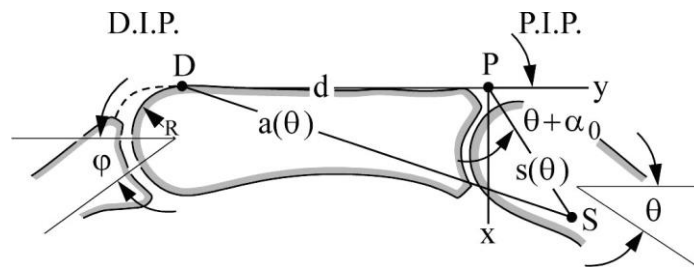


Figure 11: Dynamics of Bone Structure

from the reference paper

$$\varphi(\theta) = \frac{\sqrt{a_x(0)^2 + a_y(0)^2 + s(0)} - \sqrt{a_x(\theta)^2 + a_y(\theta)^2 - s(\theta)}}{R(\theta)}$$

Where a_x and a_y are connected to interphalangeal distance d and the length of the most distal spiral fiber (part of the lateral bands) s . R refers to the radius of curvature of the middle phalanx. All other parameters besides d are dependent on the independent variable θ

This relationship can help to properly model the kinematic outline of the fingers, particularly for determining interphalangeal angles. As the metacarpals act together as a structure, no intense modeling need be done for them. Also, as all the fingers sans the thumb have the same geometry, it is enough to model one finger symbolically.

To replicate the constraints the volar plate places on each finger, a simulant for the volar plate will be created instead of adding a mechanical stop to the bone. This will increase bio-fidelity as well as ease reparability of each finger. This will be done using ligamenture in the form of fishing line. The collateral ligaments will also be replicated using fishing line, and joint caps will be added to the connecting ends of the phalanges and the head of the metacarpals in order to replicate articular cartilage.

The extensor expansion as well as the tendon sheaths will be replicated using a musculotendon simulant. Because the resources do not currently exist to replicate the drastically different material properties of the muscles and tendons, an elastomer with a decently high tensile strength was used as a compromise. For this iteration of the hand, EPDM rubber was used because not only is it possible to laser cut this material, but it is also cheap and wear resistant. For the purposes of modeling, this material can be modeled as a spring.

The extensor expansion and the tendon sheaths were kept to ensure bio-fidelity as well as to make certain that the fingers of the hand worked properly while maintaining their dexterity. The resulting kinematic model appears as so:

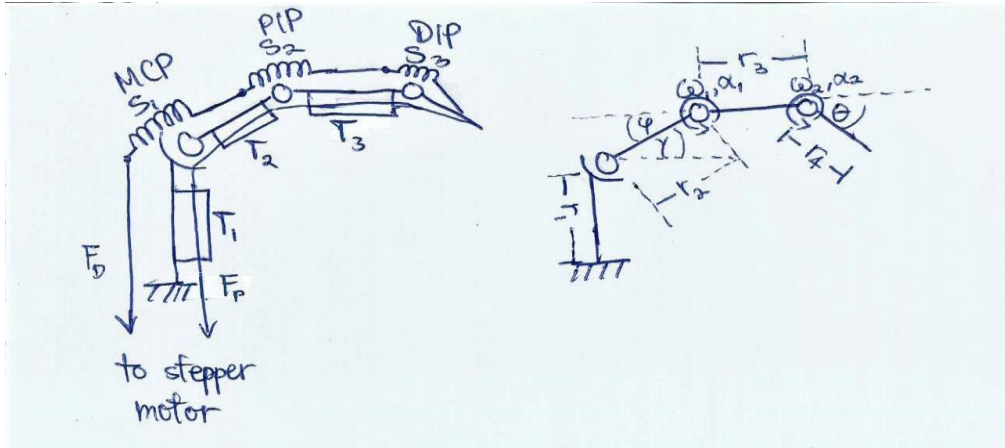


Figure 12: Bond Diagram

Because the majority of the work done by the flexor tendon is done by the flexor tendon profundus, it is acceptable to reduce the flexor tendon to the flexor tendon profundus alone. The flexor tendon, as well as the extensor tendon will both be replicated using fishing line.

Therefore the design of the hand, is modeled off of the bond diagram in Figure 12. The SolidWorks example can be found in the image below. Each finger has this general shape. Since the math provided for this section is theoretical, students still had to make various iterations of the finger until the final bones were complete. The right most image shows the final finger. In addition to this more CAD files can be found in Appendix B of the report.

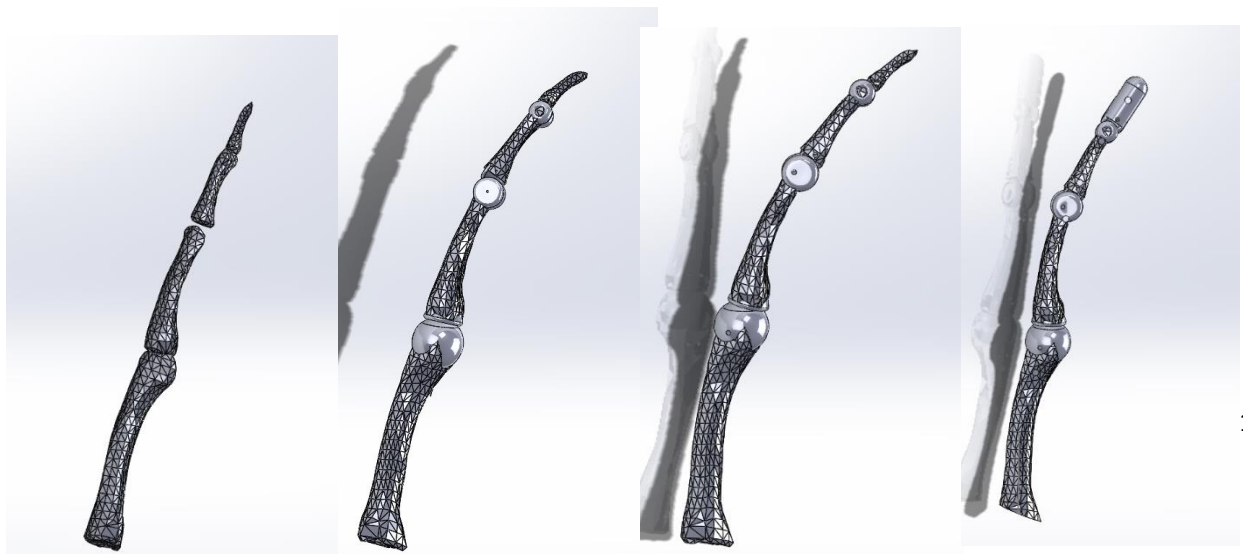


Figure 13: Solidworks model of 3d printed bones

4.1.4 Tendon Sheathing

While developing the fingers, students realized that the tendon sheaths through which the palmar tendons are run are one of the most important factors in deciding the required force for articulation. Students found that in order to ensure smooth articulation, it was necessary to keep the required articulation force significantly below the stepper motors' maximum effective holding force (7 lbs, divided by 2 because the 4th and 5th fingers are slaved together).

Originally, students planned to simply run the tendons beneath strips of EPDM which had been fastened to the bones, but found that the usual 1/16 inch thick EPDM resulted in the 4th and 5th fingers jittering, rather than articulating smoothly. In response, the articulation force requirements of various sheathing methods were tested, as seen below in Table 2. This test was performed by building a single test finger, and repeatedly changing its sheathing, using a crane scale to test the required force to fully articulate



Figure 14: Various tested tendon sheathing methods, from left to right: straws, eye screws, 1/16 inch EPDM with grommets thread holes

Table 2: Force required for full articulation with various tendon sheathing methods (all units in lbs).

Sheathing Method	Test 1	Test 2	Test 3	Average
eye screws	1.2	1.3	0.9	1.133333
Straws	0.9	1.2	1	1.033333
1/32 inch EPDM	1.3	1.3	1.2	1.266667
1/32 inch EPDM with grommets thread holes	0.8	1	1	0.933333
1/16 inch EPDM	2.2	2.1	1.9	2.066667
1/16 inch EPDM with grommets thread holes	1.8	2	1.7	1.833333

Beyond the force requirements, we had a handful of additional considerations:

- Durability- there were concerns that the friction of the tendon against the thinner EPDM would cause wear and eventually breakage. Breakage actually occurred shortly after we tested the 1/16 inch EPDM without grommets holes.
- Grasping ability- the eye screws being affixed to the palmar surface of the hand might interfere with the hand's ability to grasp objects.
- Operability- the straws, because they tended to contact each other during articulation appeared to reduce the hand's ability to fully articulate.

Though all of the tested sheathing methods were below the required articulation force of 3.5 lbs, it was nonetheless necessary to minimize the load on the stepper motors, as the previously seen jittering occurred when using 1/16 inch EPDM. Therefore, we chose the 1/32 inch EPDM with grommets thread holes, as the grommets holes would prevent the durability problems presented by the 1/32 inch EPDM alone.

4.2 Control Theory for Hand

The control schematic for the hand remains relatively simple, as shown in the figure below. The electronics used for the material can be found in Appendix A. The programming for the control can be found in Appendix C.

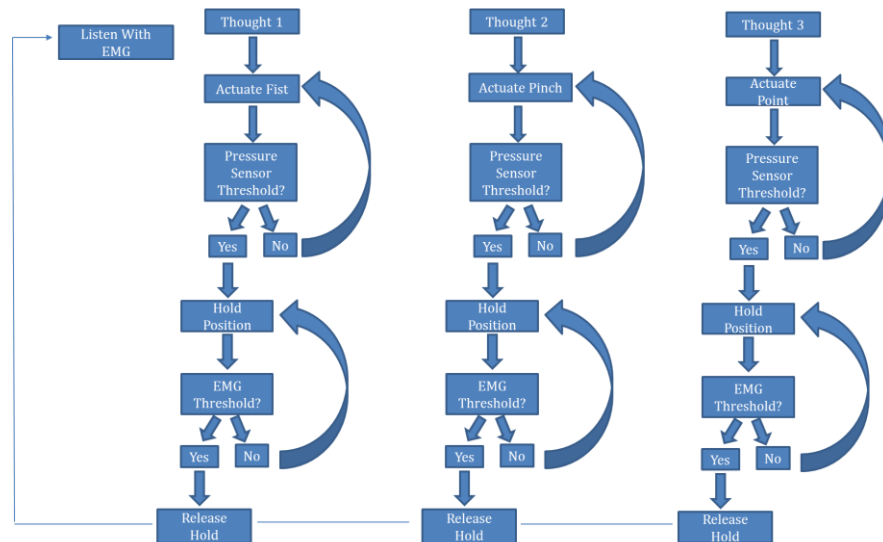


Figure 15: Control Flow Chart for Prosthetic Hand

Essentially, the EEG component of the hand is controlled by the Mindwave Neurosky Bluetooth headset. The hand is initiated by a thought. If three distinct thoughts are too be acquired from the headset, the thoughts then corresponds to the three particular grip patterns mentioned previously: pinch, point and fist. Once the grip patterns in attained, the hand waits to receive feedback from the environment from a pressure sensor. Feedback from the environment was not a design specification for the project, but it is needed to explain the flow control of the hand. The pressure sensor used for this hand can still be found in the Appendix A, in bill of materials. Once feedback is acquired from the pressure sensor, the hand will hold that position until, EMG provides input to relax the hand. The sensor used was the myoware EMG sensor. In summary, the person thinks a thought, it moves the hand, and to release the grip the patient flexes.

Chapter 5: Analysis and Testing Results

When the hand was completed, students completed force testing, and signal testing for EMG and EEG, based on the design specifications provided in Chapter 3. Students only completed trial runs on the total of 4 students who completed the project. In Chapter 6, students will explain the testing can be built upon for a future project.

5.1 Mechanical Grips and Force Testing

As determined by the design specifications for this project, the hand must be capable of completing three distinct grip patterns: pinch, point and fist. Please see the Figure below, to see the execution of such grip patterns. Pinching is the second photo, index finger point is the third photo and fist is the fourth photo.



Figure 16: Execution of Grip Patterns

To complete the force testing, students used the grip position, as shown in photograph 4, in Figure 10. The grip position was attached to a metal rod that was attached to a Newton Balance scale. The zero value of the scale, was recorded with the rod attached. The procedure was completed five times. Then value was recorded when the hand gripped the rod. The results for the forces can be found below. As shown in the Difference section of the report, 4 out of the 5 trials revealed that the hand is capable of gripping within 1 pound.

Table 3: Force Testing Results

Trial	Starting Force (lbs)	Ending Force	Difference
1	1.1	2	0.9
2	0.9	1.8	0.9
3	0.7	1.3	0.6
4	0.9	1.5	0.6
5	0.9	2.2	1.3

5.3 Signal Testing

To test the signal acquisition for the prototype of this hand, it was simply a matter of determining how reliably the hand can be actuated. Students decided to create a binary method of determining whether the signals can be actuated. However this section of the report also explains the signals acquired and why they produced reliable results.

5.3.1 EMG Signal Analysis

As mentioned previously, the EMG signal acquisition was dependent on an inexpensive Myoware EEG sensor. The sensor provides the ability to collect raw data or use the embedded filtering. For this project, students used the generic filtering provided by the sensor.

A total of 4 sensors were used. A prototype of a wrist band was created, and is shown in the photograph below.

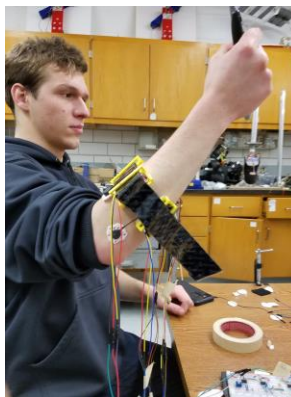


Figure 17: Prototype of EMG sensor holder

Four Myoware sensors were used to determine whether each individual finger could be differentiated. Essentially, one sensor was used for the thumb, index, and pinky. A fourth sensor was used with the middle and ring finger being slaved. Students found that the different fingers could be differentiated with their Myoware arm band. The graph below, shows the varying signals as a team member points his finger.

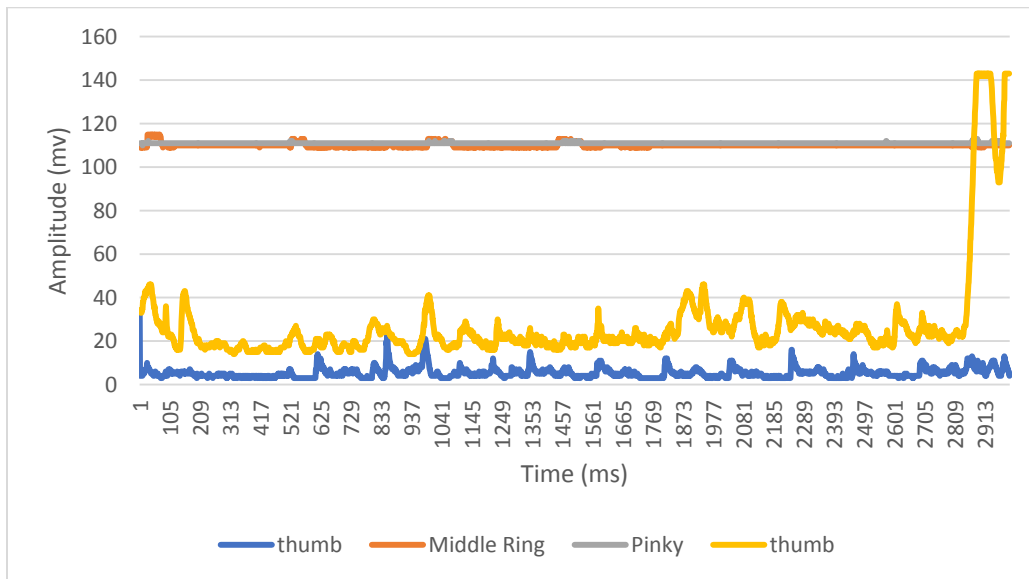


Figure 18: Fist - Differentiating Finger Signals

Unfortunately, students did not have the opportunity to use this information in the control flow of the project. Instead, students only used the EMG signal to relax the grip patterns being used. The graph below, shows a sample of flexing continuously, essentially to show that the sensor can be used reliably for simple actuation.

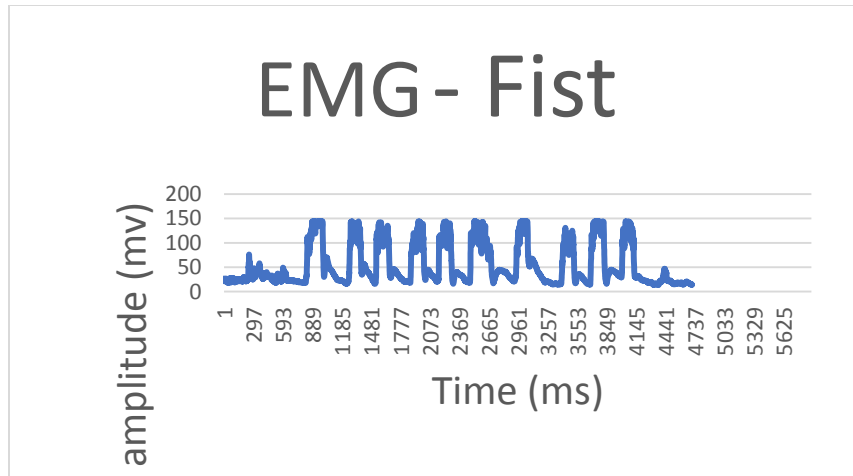


Figure 19: EMG - Fist

5.3.2 EEG Signal Analysis

The EEG actuation was completely dependent on the analog data acquired from the Neurosky Mindwave headset. The mindwave headset only has one electrode, and remainder of the components filter the signal. The students did not do additional work or research on filtering, as the headset provided enough filtering to prove the proof of concept; can EEG with one electrode be used in the application of actuating a prosthetic hand and can different thoughts be differentiated.

After completing the testing among the four students, the headset can be distinguished between thoughts, as shown in the Figure below. However, with just one electrode there are over lapping signals between the distinct thoughts. Therefore while EEG can actuate the 3 distinct thoughts independently, the headset cannot be relied upon to work continuously. More electrodes, as well as additional filtering will be required so that this information can be used in one complete flow, instead of

independently. The program can be found in appendix C.

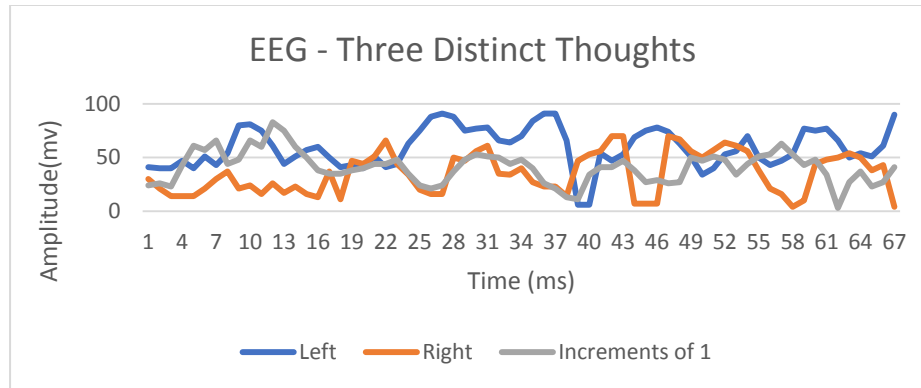


Figure 20: EEG 3 distinct thoughts

5.3.3 Signals Testing Results

As mentioned previously, the testing for the signals was binary. For example, did EMG actuate the hand, yes or no? Did EEG actuate the correct grip pattern, yes or No. The testing was completed among the 4 students in the group 8 times. The results can be found in the tables below.

For EMG, testing, the results are less than 90% most likely because during the testing the sticky electrodes used, came off. However, EMG exceeded the design deliverable of 20% success rate for actuating the prosthetic arm. The results are shown below. By averaging the results together, the final success rate was 80.25%.

Table 4: EMG Testing Results

Student 1			Student 2			Student 3			Student 4		
Trial	Actuated Hand	Did not Actuate Hand	Trial	Actuated Hand	Did not Actuate Hand	Trial	Actuated Hand	Did not Actuate Hand	Trial	Actuated H	Did not Actuate Hand
1	x		1		x	1	x		1	x	
2	x		2		x	2		x	2	x	
3	x		3			3	x		3	x	
4	x		4	x		4	x		4		x
5	x		5	x		5	x		5	x	
6	x		6	x		6	x		6	x	
7		x	7	x		7	x		7	x	
8	x		8	x		8	x		8	x	
total	7/8 = 82%		total	5/8 = 62%		total	7/8 = 82 %		total	7/8 = 82%	

For EEG, the results are listed below. The variation in results is attributed to the fact that more electrodes and additional filtering must be used to further differentiate the information pulled from the headset.

Table 5: EEG Testing Results

Student 1								
thought 1			thought 2			thought 3		
Trial	Actuated Hand	Did not Actuate Hand	Trial	Actuated Hand	Did not Actuate Hand	Trial	Actuated Hand	Did not Actuate Hand
1		x	1		x	1	x	
2			2		x	2	x	
3	x		3		x	3		x
4	x		4	x		4	x	
5		x	5	x		5	x	
6	x		6	x		6	x	
7	x		7	x		7	x	
8	x		8	x		8	x	
total	5/8 = 62%		total	5/8 = 62%		total	7/8 = 82 %	
								average = 68
Student 2								
thought 1			thought 2			thought 3		
Trial	Actuated Hand	Did not Actuate Hand	Trial	Actuated Hand	Did not Actuate Hand	Trial	Actuated Hand	Did not Actuate Hand
1	x		1	x		1		x
2	x		2	x		2	x	
3	x		3	x		3	x	
4	x		4	x		4	x	
5	x		5	x		5	x	
6		x	6		x	6		x
7		x	7		x	7		x
8		x	8		x	8		x
total	5/8 = 62%		total	7/8 = 82%		total	5/8 = 62 %	
								average = 68
Student 3								
thought 1			thought 2			thought 3		
Trial	Actuated Hand	Did not Actuate Hand	Trial	Actuated Hand	Did not Actuate Hand	Trial	Actuated Hand	Did not Actuate Hand
1	x		1	x		1	x	
2	x		2	x		2	x	
3	x		3	x		3		x
4	x		4	x		4		x
5	x		5	x		5		x
6	x		6		x	6		x
7	x		7		x	7	x	
8	x		8		x	8	x	
total	8/8 = 100%		total	5/8 = 62%		total	4/8 = 50 %	
								average = 70
Student 4								
thought 1			thought 2			thought 3		
Trial	Actuated Hand	Did not Actuate Hand	Trial	Actuated Hand	Did not Actuate Hand	Trial	Actuated Hand	Did not Actuate Hand
1		x	1	x		1		x
2			2	x		2	x	
3	x		3	x		3	x	
4	x		4	x		4	x	
5	x		5		x	5	x	
6	x		6		x	6	x	
7	x		7	x		7	x	
8	x		8	x		8		x
total	7/8 = 82%		total	6/8 = 62%		total	6/8 = 62 %	
								average = 68

Chapter 6: Conclusion and Future Work

Once the project was completed, students realized that they had met all of the design specifications provided for the project. The hand weighs 4.95 pounds. The length of the prosthetic is 15 inches. The EEG and EMG, both exceed 10% and 20% success rate respectively. The hand is biomimetic. All 9 degrees of freedom are capable with the hand. In addition to this, the total manufacturing cost of the hand is currently at \$350.00. However, there are many components to the project that are left undone. It is the hope that the next students who will work on this project will be able to complete the following.

For EMG signal acquisition, it is the hope that students who are majoring in electrical engineering can complete additional filtering methods to make the differentiation between fingers more accurate. In addition to this, students should use this differentiation in the control flow of the future project. For EEG, it is heavily suggested that students purchase the OpenBCI headset, which has more electrodes. This will result in ability to differentiate between the thoughts reliably and continuously, instead of independently. The addition of electrodes also allows the opportunity for machine learning, and hence more than three grip patterns for the patient.

In addition to this, students should consider a feedback loop, that allows the patient to know when the hand is interacting with an object. For additional electronics, students should look into sensors for pressure and temperature control, so the patient knows when the hand is in danger.

Mechanically, the joints on each finger are slaved together. To achieve more degrees of freedom, there should be a stepper motor at each joint, instead of each finger. This may increase the weight and size of the prosthetic, therefore students should also complete additional research on different types of stepper motors, even if they are more expensive.

If the students who continue with next iteration of this project use these recommendations in, it is the hope that the prosthetic will have advanced control for minimal cost. The contribution to the field of prosthetics, is that this could be the first non-invasive EEG controlled prosthetic that is biomimetic. If this end goal can be achieved for a fraction of the cost to the primary competitors, such as Bebionic, there are many people who would benefit from this design.

Bibliography

- (1) Hosmer Hooks. N.p., n.d. Web. 27 Apr. 2017. <http://hosmer.com/products/hooks/>
- (2) Gibbard, Joel. "Open Hand Project." The Open Hand Project - Home. N.p., n.d. Web. 27 Apr. 2017. <http://www.openhandproject.org/>
- (3) "Features." Bebionic. N.p., n.d. Web. 27 Apr. 2017. http://bebionic.com/the_hand/features
- (4) Revolutionizing Prosthetics. N.p., n.d. Web. 27 Apr. 2017. <http://www.jhuapl.edu/prosthetics/>
- (5) Amputee Makes History with APL's Modular Prosthetic Limb. N.p., n.d. Web. 27 Apr. 2017. <http://www.jhuapl.edu/newscenter/pressreleases/2014/141216.asp>
- (6) David K. Blough, PhD; Sharon Hubbard, MS; Lynne V. McFarland, PhD; Douglas G. Smith, MD; Jeffrey M. Gabel, MD, MPH, MSW; Gayle E. Reiber, PhD, MPH. "Journal of Rehabilitation Research & Development (JRRD)." Prosthetic cost projections for servicemembers with major limb loss from. N.p., n.d. Web. 27 Apr. 2017. <http://www.rehab.research.va.gov/jour/10/474/Blough.html>
- (7) "Features." Bebionic. N.p., n.d. Web. 27 Apr. 2017. <http://health.costhelper.com/prosthetic-arms.html>
- (8) K. van Zwieten, K. Schmidt, G. Jan Bex, P. Lippens and W. Duyvendak, "An analytical expression for the D.I.P.–P.I.P. flexion interdependence in human fingers", *Acta of Bioengineering and Biomechanics*, vol. 17, no. 1, 2015.
- (9) Swartz, Barbara E. (1998). "The advantages of digital over analog recording techniques". *Electroencephalography and Clinical Neurophysiology*. 106 (2): 113–7. doi:10.1016/S0013-4694(97)00113-2. PMID 9741771.
- (10) Cram, JR.; Steger, JC. (Jun 1983). "EMG scanning in the diagnosis of chronic pain.". *Biofeedback Self Regul.* 8 (2): 229–41. PMID 6227339.
- (11) "Processing and spectral analysis of the raw EEG signal ..." N.p., n.d. Web. 27 Apr. 2017. https://www.researchgate.net/profile/Wojciech_Salabun/publication/260036874_Processing_and_spectral_analysis_of_the_raw_EEG_signal_from_the_MindWave/links/0046352f29268c29dc000000.pdf
- (12) "Brain Computer Interface System for Mind Controlled Robot using Bluetooth" Siliveru

Ramesh.N.p.,n.d.Wed.27Apr.2017<https://pdfs.semanticscholar.org/61ed/0d85fb69232267d7358bc4f9a68eb397227e.pdf>

(13) "EMG Prosthetic Hand Controller Discriminating Ten Motions using Real-time Learning Method." Daisuke Nishikawa N.p.,n.d.Wed.27Apr.2017. (14) <http://heim.ifi.uio.no/~mes/inf1400/COOL/Robot%20Prosjekt/Arm/Links/PAPER/00811706.pdf>

(14) Zhe Xu and E. Todorov, "Design of a highly biomimetic anthropomorphic robotic hand towards artificial limb regeneration," *2016 IEEE International Conference on Robotics and Automation (ICRA)*, Stockholm, 2016, pp. 3485-3492.

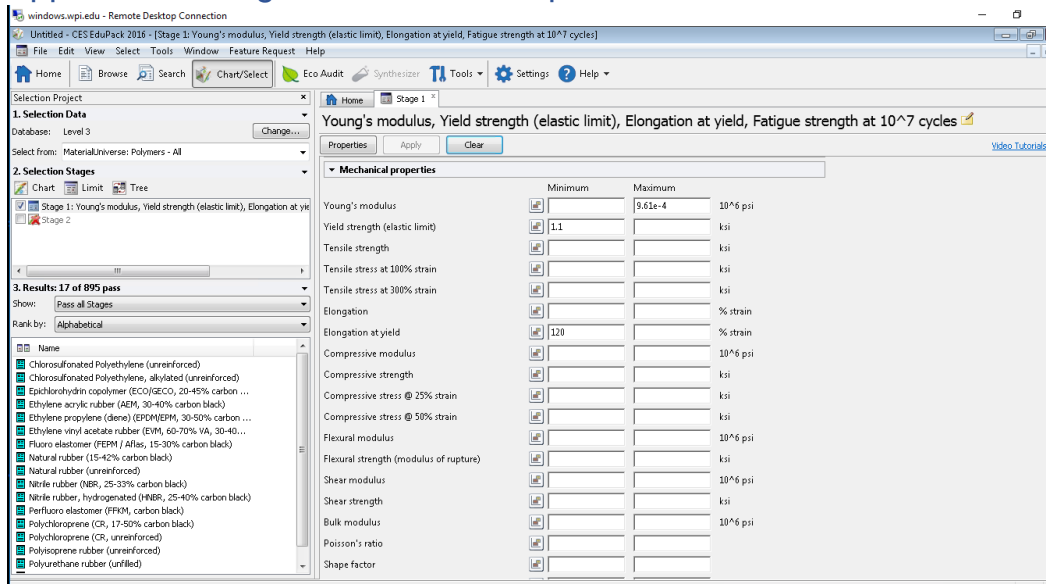
(15) E. Guizzo, "Dean Kamen's "Luke Arm" Prosthesis Receives FDA Approval", *IEEE Spectrum: Technology, Engineering, and Science News*, 2014. [Online]. Available: <http://spectrum.ieee.org/automaton/biomedical/bionics/dean-kamen-luke-arm-prosthesis-receives-fda-approval>. [Accessed: 13- May- 2014].

Appendix

Appendix A: Bill off Materials

Item No.	Description	Quantity	Units
1	1.4M screws	160	-
2	aluminum-PLA composite 3D printer filament	1/4	Lbs
3	Stepper Motor: Bipolar, 200 Steps/Rev, 35x28mm, 10V, 0.5 A/Phase	9	-
4	A4988 Stepper Motor Driver Carrier	9	-
5	1/16 inch EPDM rubber sheet	12x36	in ²
6	1/32 inch EPDM rubber sheet	12x36	in ²
7	25 lb thin braided fishing line	25	yds
8	1/8 inch acrylic sheet	1	ft ²
9	1/8 inch diameter brass dowel	1	ft
10	1/16 inch brass grommets	180	-
11	Arduino Mega	1	-
12	Neurosky Mindwave Headset	1	-
13	Myoware EMG sensor	4	-
14	Pressure sensor from Adafruit	5	-

Appendix B: Diagram of ECDU edupack



Appendix C: Programming

Mechanical Test: Fist

/*

This program actuates a fist that curls forwards and backs. just for video taping

*/

```
// defines pins numbers
```

```
//CURLING FINGERS
```

```
//Middle Finger
```

```
const int stepPin1= 2;
```

```
const int dirPin1= 3 ;
```

```
// Ring
```

```
const int stepPin2 = 4;
```

```
const int dirPin2 = 5;
```

```
// index finger
```

```
const int stepPin3 = 6;
```

```
const int dirPin3 = 7;
```

```
// the thumb forwards CURL
```

```
const int stepPin4 = 8;
```

```
const int dirPin4 = 9;
```

```
//the piny curl
```

```
const int stepPin5 = 10;
```

```
const int dirPin5 = 11;
```

```
//moving fingers backwards
```

```
/*
```

```
//we may not use this... since the recovery on the hand is pretty decent... but if we do, these are the pins.
```

```
//Middle Finger Backwards
```

```
const int stepPin6= 52;
```

```
const int dirPin6= 50 ;
```

```
// Pinky and Ring back
```

```
const int stepPin7 = 48;
```

```
const int dirPin7 = 46;
```

```
// index finger back
const int stepPin8 = 44;
const int dirPin8 = 42;

// the thumb back
const int stepPin9 = 40;
const int dirPin9 = 38;*/

void setup() {
  // Sets the two pins as Outputs
  Serial.begin(9600);

  //Middle Finger Curl
  pinMode(stepPin1,OUTPUT) ;
  pinMode(dirPin1,OUTPUT);

  //Ring Curl

  pinMode(stepPin2,OUTPUT) ;
  pinMode(dirPin2,OUTPUT);

  //index curl
  pinMode(stepPin3,OUTPUT) ;
  pinMode(dirPin3,OUTPUT);

  //thumb forward curl
  pinMode(stepPin4,OUTPUT) ;
  pinMode(dirPin4,OUTPUT);
```

```

    //pinky curl
    pinMode(stepPin5,OUTPUT) ;
    pinMode(dirPin5,OUTPUT);

/*
    //middle back
    pinMode(stepPin6,OUTPUT) ;
    pinMode(dirPin6,OUTPUT);

//pinky and ring back
    pinMode(stepPin7,OUTPUT) ;
    pinMode(dirPin7,OUTPUT);

//index back
    pinMode(stepPin8,OUTPUT) ;
    pinMode(dirPin8,OUTPUT);

    //thumb back
    pinMode(stepPin9,OUTPUT) ;
    pinMode(dirPin9,OUTPUT);
*/

}

void loop()
{
    // Middle Finger curl
    //ring pinky

```

```

//index
//thumb
//pinky

delay(2000);

{

digitalWrite(dirPin1,LOW);
digitalWrite(dirPin2,LOW);
digitalWrite(dirPin3,LOW);
digitalWrite(dirPin4,LOW);
digitalWrite(dirPin5,LOW);

// Enables the motor to move in a particular direction
// Makes (x number) pulses for making one full cycle rotation ``
for(int x = 0; x < 700; x++)

{
    //middle finger
    digitalWrite(stepPin1,HIGH);
    delayMicroseconds(1000);
    digitalWrite(stepPin1,LOW);
    delayMicroseconds(1000);

    //ring

    digitalWrite(stepPin2,HIGH);
    delayMicroseconds(700);

```



```
digitalWrite(stepPin2,LOW);  
delayMicroseconds(700);
```

```
//index
```

```
digitalWrite(stepPin3,HIGH);  
delayMicroseconds(1000);  
digitalWrite(stepPin3,LOW);  
delayMicroseconds(1000);
```

```
//thumb forwards curl
```

```
digitalWrite(stepPin4,HIGH);  
delayMicroseconds(1000);  
digitalWrite(stepPin4,LOW);  
delayMicroseconds(1000);
```

```
//pinky
```

```
digitalWrite(stepPin5,HIGH);  
delayMicroseconds(700);  
digitalWrite(stepPin5,LOW);  
delayMicroseconds(700);
```

```
}
```

```
delay(2000); // One second delay
```

```
}
```

```
//BACKWARDS
```

```

{
digitalWrite(dirPin1,HIGH); //middle
digitalWrite(dirPin2,HIGH); //ring
digitalWrite(dirPin3,HIGH); //index
digitalWrite(dirPin4,HIGH); //thumb
digitalWrite(dirPin5,HIGH); //pinky
// Enables the motor to move in a particular direction
// Makes (x number) pulses for making one full cycle rotation ``
for(int x = 0; x < 700; x++)
{

    //middle finger

    digitalWrite(stepPin1,HIGH);
    delayMicroseconds(1000);
    digitalWrite(stepPin1,LOW) ;
    delayMicroseconds(1000);

    //ring
    digitalWrite(stepPin2,HIGH);
    delayMicroseconds(700);
    digitalWrite(stepPin2,LOW) ;
    delayMicroseconds(700);

    //index
    digitalWrite(stepPin3,HIGH);
    delayMicroseconds(1000);
    digitalWrite(stepPin3,LOW) ;

```

```

delayMicroseconds(1000);

//thumb
digitalWrite(stepPin4,HIGH);
delayMicroseconds(1000);
digitalWrite(stepPin4,LOW) ;
delayMicroseconds(1000);

//pinky
digitalWrite(stepPin5,HIGH);
delayMicroseconds(700);
digitalWrite(stepPin5,LOW) ;
delayMicroseconds(700);

}

// delay(1000); // One second delay

}

}

Mechanical Test: Pinch
/*
This program actuates a pinch that curls forwards and backs. just for video taping
*/

```

```
// defines pins numbers
```

```
//CURLING FINGERS
```

```
//Middle Finger
```

```
const int stepPin1= 2;
```

```
const int dirPin1= 3 ;
```

```
// Ring
```

```
const int stepPin2 = 4;
```

```
const int dirPin2 = 5;
```

```
// index finger
```

```
const int stepPin3 = 6;
```

```
const int dirPin3 = 7;
```

```
// the thumb forwards CURL
```

```
const int stepPin4 = 8;
```

```
const int dirPin4 = 9;
```

```
//the piny curl
```

```
const int stepPin5 = 10;
```

```
const int dirPin5 = 11;
```

```
//moving fingers backwards
```

```
/*
```

//we may not use this... since the recovery on the hand is pretty decent... but if we do, these are the pins.

//Middle Finger Backwards

const int stepPin6= 52;

const int dirPin6= 50 ;

// Pinky and Ring back

const int stepPin7 = 48;

const int dirPin7 = 46;

// index finger back

const int stepPin8 = 44;

const int dirPin8 = 42;

// the thumb back

const int stepPin9 = 40;

const int dirPin9 = 38;*/

void setup() {

 // Sets the two pins as Outputs

 Serial.begin(9600);

 //Middle Finger Curl

 pinMode(stepPin1,OUTPUT) ;

 pinMode(dirPin1,OUTPUT);

 //Ring Curl

```
pinMode(stepPin2,OUTPUT) ;
pinMode(dirPin2,OUTPUT);

//index curl
pinMode(stepPin3,OUTPUT) ;
pinMode(dirPin3,OUTPUT);

//thumb forward curl
pinMode(stepPin4,OUTPUT) ;
pinMode(dirPin4,OUTPUT);

//pinky curl
pinMode(stepPin5,OUTPUT) ;
pinMode(dirPin5,OUTPUT);

/*
//middle back
pinMode(stepPin6,OUTPUT) ;
pinMode(dirPin6,OUTPUT);

//pinky and ring back
pinMode(stepPin7,OUTPUT) ;
pinMode(dirPin7,OUTPUT);

//index back
pinMode(stepPin8,OUTPUT) ;
pinMode(dirPin8,OUTPUT);

//thumb back
```

```

pinMode(stepPin9,OUTPUT) ;
pinMode(dirPin9,OUTPUT);
*/
}

void loop()
{
// Middle Finger curl
//ring pinky
//index
//thumb
//pinky

delay(2000);

{

// digitalWrite(dirPin1,LOW);
// digitalWrite(dirPin2,LOW);
digitalWrite(dirPin3,LOW);
digitalWrite(dirPin4,LOW);
// digitalWrite(dirPin5,LOW);

// Enables the motor to move in a particular direction
// Makes (x number) pulses for making one full cycle rotation ``
for(int x = 0; x < 700; x++)

{

```

```
//middle finger
/* digitalWrite(stepPin1,HIGH);
delayMicroseconds(1000);
digitalWrite(stepPin1,LOW);
delayMicroseconds(1000);

//ring

digitalWrite(stepPin2,HIGH);
delayMicroseconds(700);
digitalWrite(stepPin2,LOW);
delayMicroseconds(700); */

//index
digitalWrite(stepPin3,HIGH);
delayMicroseconds(700);
digitalWrite(stepPin3,LOW);
delayMicroseconds(700);

//thumb forwards curl
/*digitalWrite(stepPin4,HIGH);
delayMicroseconds(1000);
digitalWrite(stepPin4,LOW);
delayMicroseconds(1000);*/

//pinky
/* digitalWrite(stepPin5,HIGH);
delayMicroseconds(700);
digitalWrite(stepPin5,LOW);
delayMicroseconds(700);*/
```



```

}

        delay(2000); // One second delay

}

//BACKWARDS

{

// digitalWrite(dirPin1,HIGH); //middle
// digitalWrite(dirPin2,HIGH); //ring
digitalWrite(dirPin3,HIGH); //index
digitalWrite(dirPin4,HIGH); //thumb
// digitalWrite(dirPin5,HIGH); //pinky
// Enables the motor to move in a particular direction
// Makes (x number) pulses for making one full cycle rotation ``
for(int x = 0; x < 700; x++)
{

        //middle finger

        /* digitalWrite(stepPin1,HIGH);

```

```
delayMicroseconds(1000);
digitalWrite(stepPin1,LOW) ;
delayMicroseconds(1000);*/

//ring
/* digitalWrite(stepPin2,HIGH);
delayMicroseconds(700);
digitalWrite(stepPin2,LOW) ;
delayMicroseconds(700);*/

//index
digitalWrite(stepPin3,HIGH);
delayMicroseconds(700);
digitalWrite(stepPin3,LOW) ;
delayMicroseconds(700);

//thumb
/* digitalWrite(stepPin4,HIGH);
delayMicroseconds(1000);
digitalWrite(stepPin4,LOW) ;
delayMicroseconds(1000);*/

//pinky
/* digitalWrite(stepPin5,HIGH);
delayMicroseconds(700);
digitalWrite(stepPin5,LOW) ;
delayMicroseconds(700);*/
```

```
}  
  
    // delay(1000); // One second delay  
  
}  
  
}
```

Mechanical Test Point

```
/*  
This program actuates an index point that curls forwards and backs. just for video taping  
*/
```

```
// defines pins numbers
```

```
//CURLING FINGERS
```

```
//Middle Finger
```

```
const int stepPin1= 2;
```

```
const int dirPin1= 3 ;
```

```
// Ring
```

```
const int stepPin2 = 4;
```

```
const int dirPin2 = 5;
```

```
// index finger
```

```
const int stepPin3 = 6;
```

```
const int dirPin3 = 7;
```

```
// the thumb forwards CURL
```

```
const int stepPin4 = 8;
```

```
const int dirPin4 = 9;
```

```
//the piny curl
```

```
const int stepPin5 = 10;
```

```
const int dirPin5 = 11;
```

```
//moving fingers backwards
```

```
/*
```

```
//we may not use this... since the recovery on the hand is pretty decent... but if we do, these are the pins.
```

```
//Middle Finger Backwards
```

```
const int stepPin6= 52;
```

```
const int dirPin6= 50 ;
```

```
// Pinky and Ring back
```

```
const int stepPin7 = 48;
```

```
const int dirPin7 = 46;
```

```
// index finger back
```

```
const int stepPin8 = 44;
```

```
const int dirPin8 = 42;
```

```
// the thumb back
const int stepPin9 = 40;
const int dirPin9 = 38;*/

void setup() {
  // Sets the two pins as Outputs
  Serial.begin(9600);

  //Middle Finger Curl
  pinMode(stepPin1,OUTPUT) ;
  pinMode(dirPin1,OUTPUT);

  //Ring Curl

  pinMode(stepPin2,OUTPUT) ;
  pinMode(dirPin2,OUTPUT);

  //index curl
  pinMode(stepPin3,OUTPUT) ;
  pinMode(dirPin3,OUTPUT);

  //thumb forward curl
  pinMode(stepPin4,OUTPUT) ;
  pinMode(dirPin4,OUTPUT);

  //pinky curl
  pinMode(stepPin5,OUTPUT) ;
  pinMode(dirPin5,OUTPUT);
```

```

/*
  //middle back
  pinMode(stepPin6,OUTPUT) ;
  pinMode(dirPin6,OUTPUT);

  //pinky and ring back
  pinMode(stepPin7,OUTPUT) ;
  pinMode(dirPin7,OUTPUT);

  //index back
  pinMode(stepPin8,OUTPUT) ;
  pinMode(dirPin8,OUTPUT);

  //thumb back
  pinMode(stepPin9,OUTPUT) ;
  pinMode(dirPin9,OUTPUT);
*/
}

void loop()
{
  // Middle Finger curl
  //ring pinky
  //index
  //thumb
  //pinky

  delay(2000);

```

```

{

digitalWrite(dirPin1,LOW);
digitalWrite(dirPin2,LOW);
// digitalWrite(dirPin3,LOW);
digitalWrite(dirPin4,LOW);
// digitalWrite(dirPin5,LOW);

// Enables the motor to move in a particular direction
// Makes (x number) pulses for making one full cycle rotation ``
for(int x = 0; x < 700; x++)

{
    //middle finger
    digitalWrite(stepPin1,HIGH);
    delayMicroseconds(1000);
    digitalWrite(stepPin1,LOW);
    delayMicroseconds(1000);

    //ring

    digitalWrite(stepPin2,HIGH);
    delayMicroseconds(700);
    digitalWrite(stepPin2,LOW);
    delayMicroseconds(700);

    //index
    /* digitalWrite(stepPin3,HIGH);

```

```
delayMicroseconds(1000);  
digitalWrite(stepPin3,LOW);  
delayMicroseconds(1000); */
```

```
//thumb forwards curl  
digitalWrite(stepPin4,HIGH);  
delayMicroseconds(1000);  
digitalWrite(stepPin4,LOW);  
delayMicroseconds(1000);
```

```
//pinky  
digitalWrite(stepPin5,HIGH);  
delayMicroseconds(700);  
digitalWrite(stepPin5,LOW);  
delayMicroseconds(700);
```

```
}
```

```
delay(2000); // One second delay
```

```
}
```

```
//BACKWARDS
```

```
{
```



```

digitalWrite(dirPin1,HIGH); //middle
digitalWrite(dirPin2,HIGH); //ring
digitalWrite(dirPin3,HIGH); //index
digitalWrite(dirPin4,HIGH); //thumb
digitalWrite(dirPin5,HIGH); //pinky
// Enables the motor to move in a particular direction
// Makes (x number) pulses for making one full cycle rotation ``
for(int x = 0; x < 700; x++)
{

    //middle finger

    digitalWrite(stepPin1,HIGH);
    delayMicroseconds(1000);
    digitalWrite(stepPin1,LOW) ;
    delayMicroseconds(1000);

    //ring
    digitalWrite(stepPin2,HIGH);
    delayMicroseconds(700);
    digitalWrite(stepPin2,LOW) ;
    delayMicroseconds(700);

    //index
    /* digitalWrite(stepPin3,HIGH);
    delayMicroseconds(1000);
    digitalWrite(stepPin3,LOW) ;

```

```

delayMicroseconds(1000);*/

//thumb
digitalWrite(stepPin4,HIGH);
delayMicroseconds(1000);
digitalWrite(stepPin4,LOW) ;
delayMicroseconds(1000);

//pinky
digitalWrite(stepPin5,HIGH);
delayMicroseconds(700);
digitalWrite(stepPin5,LOW) ;
delayMicroseconds(700);

}

// delay(1000); // One second delay

}

}

EMG code
#include <Servo.h>

Servo IndexFinger;
Servo MiddleFingerRingFinger;
Servo Pinky;

```

Servo Thumb;

//pulse for stepper motor

const int pulse = 400;

// pins for stepper motor

//Index Finger Initiate

const int stepPin1= 2;

const int dirPin1= 3;

//Index Finger Reverse

const int stepPin2 = 4;

const int dirPin2 = 5;

// MiddleFingerRingFinger Initiate

const int stepPin3 = 6;

const int dirPin3 = 7;

// MiddleFingerRingFinger Reverse

const int stepPin4 = 8;

const int dirPin4 = 9;

//Pinky Finger initiate

const int stepPin5 = 11;

const int dirPin5 = 12;

//Pinky Finger Reverse

```
const int stepPin6 = 13;
```

```
const int dirPin6 = 14;
```

```
//Thumb Initiate
```

```
const int stepPin7 = 13;
```

```
const int dirPin7 = 14;
```

```
//Thumb Reverse
```

```
const int stepPin8 = 13;
```

```
const int dirPin8 = 14;
```

```
//NOTE ADD IN THREE STEPPER MOTORS:
```

```
//NOTE CHANGE TO NUMBERS
```

```
//threshold values for the sensor
```

```
const int IndexFingerThresh = 500;
```

```
const int MiddleFingerRingFingerThresh = 500;
```

```
const int PinkyThresh = 500;
```

```
const int ThumbThresh = 500;
```

```
//analog inputpins. these are attached to the sensors
```

```
const int IndexFingerAnalogIn = A1;
```

```
const int MiddleFingerRingFingerAnalogIn = A2;
```

```
const int PinkyAnalogIn = A3;
```

```
const int ThumbAnalogIn = A4;
```

```
//analog output pins. these are attached to the stepper motor
```

```
const int IndexFingerAnalogOut = 9;
```

```
const int MiddleFingerRingFingerAnalogOut = 8;
```

```

const int PinkyAnalogOut = 7;
const int ThumbAnalogOut = 6;

//1. values read from the pot/sensor
//2. values output to the PWM, (analogout)

int sensorValueIndexFinger = 0;
int outputValueIndexFinger = 0;

int sensorValueMiddleFingerRingFinger = 0;
int outputValueMiddleFingerRingFinger = 0;

int sensorValuePinky = 0;
int outputValuePinky = 0;

int sensorValueThumb = 0;
int outputValueThumb = 0;

void setup() {
  // put your setup code here, to run once:
  Serial.begin(9600);
  IndexFinger.attach(9);
  MiddleFingerRingFinger.attach(8);
  Pinky.attach(7);
  Thumb.attach(6);
  // Sets the two pins as Outputs

  //finger one
  pinMode(stepPin1,OUTPUT) ;

```

```

pinMode(dirPin1,OUTPUT);

//finger two

pinMode(stepPin2,OUTPUT) ;
pinMode(dirPin2,OUTPUT);

//finger two
pinMode(stepPin3,OUTPUT) ;
pinMode(dirPin3,OUTPUT);

//finger two
pinMode(stepPin4,OUTPUT) ;
pinMode(dirPin4,OUTPUT);

//finger two
pinMode(stepPin5,OUTPUT) ;
pinMode(dirPin5,OUTPUT);

}

void loop() {
// put your main code here, to run repeatedly:
//index Finger
{
int valueIndexFinger = analogRead(A1);

if (valueIndexFinger >= IndexFingerThresh-50 && valueIndexFinger <=
IndexFingerThresh+50)
{ //iniating motor
digitalWrite(dirPin1,HIGH); // Enables the motor to move in a particular direction

```

```

// Makes (x number) pulses for making one full cycle rotation ``
for(int x = 0; x < pulse; x++) {
digitalWrite(stepPin1,HIGH);
delayMicroseconds(500);
digitalWrite(stepPin1,LOW);
delayMicroseconds(500);
}
delay(1000); // One second delay
}
else
//turn motor in oppostie directuon for some slack
//fingerone backward
digitalWrite(dirPin1,LOW); //Changes the rotations direction
// Makes 400 pulses for making two full cycle rotation
for(int x = 0; x < pulse; x++) {
digitalWrite(stepPin1,HIGH);
delayMicroseconds(500);
digitalWrite(stepPin1,LOW) ;
delayMicroseconds(500);
}

delay(500);

// iniatite other motor dedicated for pulling
digitalWrite(dirPin2,HIGH); // Enables the motor to move in a particular direction
// Makes (x number) pulses for making one full cycle rotation ``
for(int x = 0; x < pulse; x++) {
digitalWrite(stepPin2,HIGH);
delayMicroseconds(500);
digitalWrite(stepPin2,LOW);
}

```

```

        delayMicroseconds(500);
    }
    delay(1000); // One second delay
}

//MiddleFingerRingFinger
{
    int valueMiddleFingerRingFinger = analogRead(A2);
    if (valueMiddleFingerRingFinger >= MiddleFingerRingFingerThresh-50 &&
valueMiddleFingerRingFinger <=MiddleFingerRingFingerThresh+50)
    {
        { //iniating motor
            digitalWrite(dirPin3,HIGH); // Enables the motor to move in a particular direction
            // Makes (x number) pulses for making one full cycle rotation ``
            for(int x = 0; x < pulse; x++) {
                digitalWrite(stepPin3,HIGH);
                delayMicroseconds(500);
                digitalWrite(stepPin3,LOW);
                delayMicroseconds(500);
            }
            delay(1000); // One second delay
        }
    }

else {
    //turn motor in oppostie directuon for some slack
    //fingerone backward
    digitalWrite(dirPin1,LOW); //Changes the rotations direction
    // Makes 400 pulses for making two full cycle rotation

```



```

for(int x = 0; x < pulse; x++) {
    digitalWrite(stepPin3,HIGH);
    delayMicroseconds(500);
    digitalWrite(stepPin3,LOW) ;
    delayMicroseconds(500);
}

delay(500);

// initiate other motor dedicated for pulling
digitalWrite(dirPin2,HIGH); // Enables the motor to move in a particular direction
// Makes (x number) pulses for making one full cycle rotation ``
for(int x = 0; x < pulse; x++) {
    digitalWrite(stepPin4,HIGH);
    delayMicroseconds(500);
    digitalWrite(stepPin4,LOW);
    delayMicroseconds(500);
}
delay(1000); // One second delay
}

}

//Pinky

{
    int valuePinky = analogRead(A3);
    if (valuePinky >= PinkyThresh-50 && valuePinky <= PinkyThresh+50)
    {

```

```

{ //iniating motor
    digitalWrite(dirPin5,HIGH); // Enables the motor to move in a particular direction
    // Makes (x number) pulses for making one full cycle rotation ``
    for(int x = 0; x < pulse; x++) {
        digitalWrite(stepPin5,HIGH);
        delayMicroseconds(500);
        digitalWrite(stepPin5,LOW);
        delayMicroseconds(500);
    }
    delay(1000); // One second delay
}}

```

```

else {
    //turn motor in oppostie directuon for some slack
    //fingerone backward
    digitalWrite(dirPin1,LOW); //Changes the rotations direction
    // Makes 400 pulses for making two full cycle rotation
    for(int x = 0; x < pulse; x++) {
        digitalWrite(stepPin5,HIGH);
        delayMicroseconds(500);
        digitalWrite(stepPin5,LOW) ;
        delayMicroseconds(500);
    }

    delay(500);

    // iniatite other motor dedicated for pulling
    digitalWrite(dirPin2,HIGH); // Enables the motor to move in a particular direction
    // Makes (x number) pulses for making one full cycle rotation ``

```

```

        for(int x = 0; x < pulse; x++) {
            digitalWrite(stepPin6,HIGH);
            delayMicroseconds(500);
            digitalWrite(stepPin6,LOW);
            delayMicroseconds(500);
        }
        delay(1000); // One second delay

    }

}

//Thumb
{
    int valueThumb = analogRead(A4);
    if (valueThumb >= ThumbThresh-50 && valueThumb <= ThumbThresh+50)
    {
        { //iniating motor
            digitalWrite(dirPin7,HIGH); // Enables the motor to move in a particular direction
            // Makes (x number) pulses for making one full cycle rotation ``
            for(int x = 0; x < pulse; x++) {
                digitalWrite(stepPin7,HIGH);
                delayMicroseconds(500);
                digitalWrite(stepPin7,LOW);
                delayMicroseconds(500);
            }
            delay(1000); // One second delay
        }
    }

else {

```

```

//turn motor in oppostie directuon for some slack
//fingerone backward
digitalWrite(dirPin7,LOW); //Changes the rotations direction
// Makes 400 pulses for making two full cycle rotation
for(int x = 0; x < pulse; x++) {
    digitalWrite(stepPin7,HIGH);
    delayMicroseconds(500);
    digitalWrite(stepPin7,LOW) ;
    delayMicroseconds(500);
}

delay(500);

// iniatite other motor dedicated for pulling
digitalWrite(dirPin2,HIGH); // Enables the motor to move in a particular direction
// Makes (x number) pulses for making one full cycle rotation ``
for(int x = 0; x < pulse; x++) {
    digitalWrite(stepPin8,HIGH);
    delayMicroseconds(500);
    digitalWrite(stepPin8,LOW);
    delayMicroseconds(500);
}
delay(1000); // One second delay

}

}

```

```

int sensorValueIndexFinger = analogRead(IndexFingerAnalogIn);
int sensorValueMiddleFingerRingFinger = analogRead(MiddleFingerRingFingerAnalogIn);
int sensorValuePinky = analogRead(PinkyAnalogIn);
int sensorValueThumb = analogRead(ThumbAnalogIn);

outputValueIndexFinger = map(sensorValueIndexFinger, 0, 1023, 0, 255);
outputValueMiddleFingerRingFinger = map(sensorValueMiddleFingerRingFinger, 0, 1023, 0,
255);
outputValuePinky = map(sensorValuePinky, 0, 1023, 0, 255);
outputValueThumb = map(sensorValueThumb, 0, 1023, 0, 255);

analogWrite(IndexFingerAnalogOut,outputValueIndexFinger);
analogWrite(MiddleFingerRingFingerAnalogOut,outputValueMiddleFingerRingFinger);
analogWrite(PinkyAnalogOut,outputValuePinky);
analogWrite(ThumbAnalogOut,outputValueThumb);

//index finger
//Serial.print("Index Input = ");
//Serial.print(sensorValueIndexFinger);
//Serial.print(",");
//Serial.print("\t Index Output = ");
Serial.print(outputValueIndexFinger);
Serial.print(",");

//middle finger ring finger
//Serial.print("MiddleRing Input = ");
//Serial.print(sensorValueMiddleFingerRingFinger);
//Serial.print(",");

```

```
//Serial.print("\t MiddleRing Output = ");  
Serial.print(outputValueMiddleFingerRingFinger);  
Serial.print(",");
```

```
//pinky
```

```
//Serial.print("Pinky Input = ");  
//Serial.print(sensorValuePinky);  
//Serial.print(",");  
// Serial.print("\t Pinky Output = ");  
Serial.print(outputValuePinky);  
Serial.print(",");
```

```
//thumb
```

```
//Serial.print("Thumb Input = ");  
//Serial.print(sensorValueThumb);  
//Serial.print(",");  
//Serial.print("\t Thumb Output = ");  
Serial.print( outputValueThumb);  
Serial.println(",");  
delay(2);
```

```
}
```

EEG: Thought 1

```
#define LED 13  
#define BAUDRATE 57600  
#define DEBUGOUTPUT 0
```

```
// CASES 3 4 5 6, are when im counting by 1:00
```

```
//case
```

```
//all white is green
```

```
#define GREENLED1 3 //RED
```

```
#define GREENLED2 4
```

```
#define GREENLED3 5
```

```
#define YELLOWLED1 6 //YELLOW
```

```
#define YELLOWLED2 7
```

```
#define YELLOWLED3 8
```

```
#define YELLOWLED4 9
```

```
#define REDLED1 10 //BLUE
```

```
#define REDLED2 11
```

```
#define REDLED3 13
```

```
#define powercontrol 10
```

```
// checksum variables
```

```
byte generatedChecksum = 0;
```

```
byte checksum = 0;
```

```
int payloadLength = 0;
```

```
byte payloadData[64] = {0};
```

```
byte poorQuality = 0;
```

```
byte attention = 0;
```

```
byte meditation = 0;
```

```
// system variables
```

```
long lastReceivedPacket = 0;
```

```
boolean bigPacket = false;
```

```
// defines pins numbers
```

```
//CURLING FINGERS
```

```
//Middle Finger
```

```
const int stepPin1= 2;
```

```
const int dirPin1= 3 ;
```

```
// Pinky and Ring
```

```
const int stepPin2 = 4;
```

```
const int dirPin2 = 5;
```

```
// index finger
```

```
const int stepPin3 = 6;
```

```
const int dirPin3 = 7;
```

```
// the thumb sideways CURL
```

```
const int stepPin4 = 8;
```

```
const int dirPin4 = 9;
```

```
//the thumb forwards curl
```

```
const int stepPin5 = 10;
```

```
const int dirPin5 = 11;
```

```
//moving fingers backwards
```


//we may not use this... since the recovery on the hand is pretty decent... but if we do, these are the pins.

//Middle Finger Backwards

const int stepPin6= 52;

const int dirPin6= 50 ;

// Pinky and Ring back

const int stepPin7 = 48;

const int dirPin7 = 46;

// index finger back

const int stepPin8 = 44;

const int dirPin8 = 42;

// the thumb back

const int stepPin9 = 40;

const int dirPin9 = 38;

////////////////////////////////////

// Microprocessor Setup //

////////////////////////////////////

void setup()

{

/* pinMode(GREENLED1, OUTPUT);

pinMode(GREENLED2, OUTPUT);

```
pinMode(GREENLED3, OUTPUT);
pinMode(YELLOWLED1, OUTPUT);
pinMode(YELLOWLED2, OUTPUT);
pinMode(YELLOWLED3, OUTPUT);
pinMode(YELLOWLED4, OUTPUT);
pinMode(REDLED1, OUTPUT);
pinMode(REDLED2, OUTPUT);
pinMode(REDLED3, OUTPUT);

pinMode(LED, OUTPUT);*/

Serial.begin(BAUDRATE);      // USB

//Middle Finger Curl
pinMode(stepPin1,OUTPUT) ;
pinMode(dirPin1,OUTPUT);

//Pinky and Ring Curl

pinMode(stepPin2,OUTPUT) ;
pinMode(dirPin2,OUTPUT);

//index curl
pinMode(stepPin3,OUTPUT) ;
pinMode(dirPin3,OUTPUT);

//thumb forward curl
pinMode(stepPin4,OUTPUT) ;
pinMode(dirPin4,OUTPUT);
```

```

//thumb sideways curl
pinMode(stepPin5,OUTPUT) ;
pinMode(dirPin5,OUTPUT);

//middle back
pinMode(stepPin6,OUTPUT) ;
pinMode(dirPin6,OUTPUT);

//pinky and ring back
pinMode(stepPin7,OUTPUT) ;
pinMode(dirPin7,OUTPUT);

//index back
pinMode(stepPin8,OUTPUT) ;
pinMode(dirPin8,OUTPUT);

//thumb back
pinMode(stepPin9,OUTPUT) ;
pinMode(dirPin9,OUTPUT);

}

////////////////////////////////////
// Read data from Serial UART //
////////////////////////////////////
byte ReadOneByte()

{

```

```

int ByteRead;
while(!Serial.available());
ByteRead = Serial.read();

#if DEBUGOUTPUT
  Serial.print((char)ByteRead); // echo the same byte out the USB serial (for debug purposes)
#endif

return ByteRead;
}

//////////
//MAIN LOOP//
//////////
void loop()

{
  // Look for sync bytes
  if(ReadOneByte() == 170)
  {
    if(ReadOneByte() == 170)
    {
      payloadLength = ReadOneByte();

      if(payloadLength > 169) //Payload length can not be greater than 169
        return;
      generatedChecksum = 0;
      for(int i = 0; i < payloadLength; i++)
      {
        payloadData[i] = ReadOneByte(); //Read payload into memory

```

```

generatedChecksum += payloadData[i];
}

checksum = ReadOneByte();           //Read checksum byte from stream
generatedChecksum = 255 - generatedChecksum; //Take one's compliment of generated
checksum

if(checksum == generatedChecksum)
{
    poorQuality = 200;
    attention = 0;
    meditation = 0;

    for(int i = 0; i < payloadLength; i++)
    {
        // Parse the payload
        switch (payloadData[i])
        {
        case 2:
            i++;
            poorQuality = payloadData[i];
            bigPacket = true;
            break;
        case 4:
            i++;
            attention = payloadData[i];
            break;
        case 5:
            i++;
            meditation = payloadData[i];
            break;

```

```
case 0x80:
    i = i + 3;
    break;
case 0x83:
    i = i + 25;
    break;
default:
    break;
} // switch
} // for loop
```

```
#if !DEBUGOUTPUT
```

```
    // *** Add your code here ***

    if(bigPacket)
    {
        if(poorQuality == 0)
            digitalWrite(LED, HIGH);
        else
            digitalWrite(LED, LOW);

        Serial.print("PoorQuality: ");
        Serial.print(poorQuality, DEC);
        Serial.print(" Attention: ");
        Serial.print(attention, DEC);
        Serial.print(" Time since last packet: ");
        Serial.print(millis() - lastReceivedPacket, DEC);
        lastReceivedPacket = millis();
        Serial.print("\n");
    }
#endif
```

```
switch(attention / 10)
{

case 0:

/* digitalWrite(GREENLED1, HIGH);
digitalWrite(GREENLED2, LOW);
digitalWrite(GREENLED3, LOW);
digitalWrite(YELLOWLED1, LOW);
digitalWrite(YELLOWLED2, LOW);
digitalWrite(YELLOWLED3, LOW);
digitalWrite(YELLOWLED4, LOW);
digitalWrite(REDLED1, LOW);
digitalWrite(REDLED2, LOW);
digitalWrite(REDLED3, LOW); */

break;

case 1:

/*digitalWrite(GREENLED1, HIGH);
digitalWrite(GREENLED2, HIGH);
digitalWrite(GREENLED3, LOW);
digitalWrite(YELLOWLED1, LOW);
```

```
digitalWrite(YELLOWLED2, LOW);  
digitalWrite(YELLOWLED3, LOW);  
digitalWrite(YELLOWLED4, LOW);  
digitalWrite(REDLED1, LOW);  
digitalWrite(REDLED2, LOW);  
digitalWrite(REDLED3, LOW);*/
```

```
break;
```

```
case 2:
```

```
/* digitalWrite(GREENLED1, HIGH);  
digitalWrite(GREENLED2, HIGH);  
digitalWrite(GREENLED3, HIGH);  
digitalWrite(YELLOWLED1, LOW);  
digitalWrite(YELLOWLED2, LOW);  
digitalWrite(YELLOWLED3, LOW);  
digitalWrite(YELLOWLED4, LOW);  
digitalWrite(REDLED1, LOW);  
digitalWrite(REDLED2, LOW);  
digitalWrite(REDLED3, LOW);*/
```

```
break;
```

```
case 3:
```

```
/*digitalWrite(GREENLED1, HIGH);  
digitalWrite(GREENLED2, HIGH);
```



```
digitalWrite(GREENLED3, HIGH);  
digitalWrite(YELLOWLED1, HIGH);  
digitalWrite(YELLOWLED2, LOW);  
digitalWrite(YELLOWLED3, LOW);  
digitalWrite(YELLOWLED4, LOW);  
digitalWrite(REDLED1, LOW);  
digitalWrite(REDLED2, LOW);  
digitalWrite(REDLED3, LOW);*/
```

```
//delay(1000); // One second delay  
break;
```

```
case 4:
```

```
/*digitalWrite(GREENLED1, HIGH);  
digitalWrite(GREENLED2, HIGH);  
digitalWrite(GREENLED3, HIGH);  
digitalWrite(YELLOWLED1, HIGH);  
digitalWrite(YELLOWLED2, HIGH);  
digitalWrite(YELLOWLED3, LOW);  
digitalWrite(YELLOWLED4, LOW);  
digitalWrite(REDLED1, LOW);  
digitalWrite(REDLED2, LOW);  
digitalWrite(REDLED3, LOW); */
```

```
//delay(1000); // One second delay
```

```
break;
case 5:

/* digitalWrite(GREENLED1, HIGH);
digitalWrite(GREENLED2, HIGH);
digitalWrite(GREENLED3, HIGH);
digitalWrite(YELLOWLED1, HIGH);
digitalWrite(YELLOWLED2, HIGH);
digitalWrite(YELLOWLED3, HIGH);
digitalWrite(YELLOWLED4, LOW);
digitalWrite(REDLED1, LOW);
digitalWrite(REDLED2, LOW);
digitalWrite(REDLED3, LOW);*/
```

```
break;
```

```
case 6:
/*digitalWrite(GREENLED1, HIGH);
digitalWrite(GREENLED2, HIGH);
digitalWrite(GREENLED3, HIGH);
digitalWrite(YELLOWLED1, HIGH);
digitalWrite(YELLOWLED2, HIGH);
digitalWrite(YELLOWLED3, HIGH);
digitalWrite(YELLOWLED4, HIGH);
digitalWrite(REDLED1, LOW);
digitalWrite(REDLED2, LOW);
digitalWrite(REDLED3, LOW);*/
```

```

break;
case 7:
    /*digitalWrite(GREENLED1, HIGH);
    digitalWrite(GREENLED2, HIGH);
    digitalWrite(GREENLED3, HIGH);
    digitalWrite(YELLOWLED1, HIGH);
    digitalWrite(YELLOWLED2, HIGH);
    digitalWrite(YELLOWLED3, HIGH);
    digitalWrite(YELLOWLED4, HIGH);
    digitalWrite(REDLED1, HIGH);
    digitalWrite(REDLED2, LOW);
    digitalWrite(REDLED3, LOW);*/

    {
    digitalWrite(dirPin1,LOW); // Enables the motor to move in a particular direction
    // Makes (x number) pulses for making one full cycle rotation ``
    for(int x = 0; x < 100; x++)
    {
        digitalWrite(stepPin4,HIGH);
        delayMicroseconds(400);
        digitalWrite(stepPin1,LOW);
        delayMicroseconds(400);

        digitalWrite(stepPin2,HIGH);
        delayMicroseconds(400);
        digitalWrite(stepPin2,LOW);
        delayMicroseconds(400);
    }
}

```

```

// delay(10000); // One second delay

}

break;

case 8:
/* digitalWrite(GREENLED1, HIGH);
digitalWrite(GREENLED2, HIGH);
digitalWrite(GREENLED3, HIGH);
digitalWrite(YELLOWLED1, HIGH);
digitalWrite(YELLOWLED2, HIGH);
digitalWrite(YELLOWLED3, HIGH);
digitalWrite(YELLOWLED4, HIGH);
digitalWrite(REDLED1, HIGH);
digitalWrite(REDLED2, HIGH);
digitalWrite(REDLED3, LOW);*/

// delay(1000); // One second delay

{
digitalWrite(dirPin1,LOW); // Enables the motor to move in a particular direction
// Makes (x number) pulses for making one full cycle rotation ``
for(int x = 0; x < 400; x++)
{

```

```

        digitalWrite(stepPin1,HIGH);
        delayMicroseconds(400);
        digitalWrite(stepPin1,LOW);
        delayMicroseconds(400);

        digitalWrite(stepPin2,HIGH);
        delayMicroseconds(400);
        digitalWrite(stepPin2,LOW);
        delayMicroseconds(400);

    }

    // delay(10000); // One second delay

}

break;

case 9:
/* digitalWrite(GREENLED1, HIGH);
digitalWrite(GREENLED2, HIGH);
digitalWrite(GREENLED3, HIGH);
digitalWrite(YELLOWLED1, HIGH);
digitalWrite(YELLOWLED2, HIGH);
digitalWrite(YELLOWLED3, HIGH);
digitalWrite(YELLOWLED4, HIGH);
digitalWrite(REDLED1, HIGH);
digitalWrite(REDLED2, HIGH);
digitalWrite(REDLED3, HIGH);*/

```

```

{
digitalWrite(dirPin1,LOW); // Enables the motor to move in a particular direction
// Makes (x number) pulses for making one full cycle rotation ``
for(int x = 0; x < 400; x++)
{
    digitalWrite(stepPin1,HIGH);
    delayMicroseconds(400);
    digitalWrite(stepPin1,LOW);
    delayMicroseconds(400);

    digitalWrite(stepPin2,HIGH);
    delayMicroseconds(400);
    digitalWrite(stepPin2,LOW);
    delayMicroseconds(400);

}

// delay(10000); // One second delay

}

```

```
break;
```

```
delay(1000); // One second delay
```

case 10:

```
/*digitalWrite(GREENLED1, HIGH);
digitalWrite(GREENLED2, HIGH);
digitalWrite(GREENLED3, HIGH);
digitalWrite(YELLOWLED1, HIGH);
digitalWrite(YELLOWLED2, HIGH);
digitalWrite(YELLOWLED3, HIGH);
digitalWrite(YELLOWLED4, HIGH);
digitalWrite(REDLED1, HIGH);
digitalWrite(REDLED2, HIGH);
digitalWrite(REDLED3, HIGH);*/

{
digitalWrite(dirPin1,LOW); // Enables the motor to move in a particular direction
// Makes (x number) pulses for making one full cycle rotation ``
for(int x = 0; x < 400; x++)
{
digitalWrite(stepPin1,HIGH);
delayMicroseconds(400);
digitalWrite(stepPin1,LOW);
delayMicroseconds(400);

digitalWrite(stepPin2,HIGH);
delayMicroseconds(400);
digitalWrite(stepPin2,LOW);
delayMicroseconds(400);

}

// delay(10000); // One second delay
```

```
    }

    break;

}

}

#endif

    bigPacket = false;
}
else {
    // Checksum Error
} // end if else for checksum
} // end if read 0xAA byte
} // end if read 0xAA byte
}
```

EEG: Thought 2

```
#define LED 13
#define BAUDRATE 57600
#define DEBUGOUTPUT 0

//all white is green
```



```
#define GREENLED1 3
#define GREENLED2 4
#define GREENLED3 5
#define YELLOWLED1 6
#define YELLOWLED2 7
#define YELLOWLED3 8
#define YELLOWLED4 9
#define REDLED1 10
#define REDLED2 11
#define REDLED3 12

#define powercontrol 10

// checksum variables
byte generatedChecksum = 0;
byte checksum = 0;
int payloadLength = 0;
byte payloadData[64] = {0};
byte poorQuality = 0;
byte attention = 0;
byte meditation = 0;

// system variables
long lastReceivedPacket = 0;
boolean bigPacket = false;

////////////////////////////////////
// Microprocessor Setup //
////////////////////////////////////
void setup()
```

```

{
  pinMode(GREENLED1, OUTPUT);
  pinMode(GREENLED2, OUTPUT);
  pinMode(GREENLED3, OUTPUT);
  pinMode(YELLOWLED1, OUTPUT);
  pinMode(YELLOWLED2, OUTPUT);
  pinMode(YELLOWLED3, OUTPUT);
  pinMode(YELLOWLED4, OUTPUT);
  pinMode(REDLED1, OUTPUT);
  pinMode(REDLED2, OUTPUT);
  pinMode(REDLED3, OUTPUT);

  pinMode(LED, OUTPUT);
  Serial.begin(BAUDRATE);      // USB
}

```

```

////////////////////

```

```

// Read data from Serial UART //

```

```

////////////////////

```

```

byte ReadOneByte()

```

```

{
  int ByteRead;
  while(!Serial.available());
  ByteRead = Serial.read();

```

```

#if DEBUGOUTPUT

```

```

  Serial.print((char)ByteRead); // echo the same byte out the USB serial (for debug purposes)

```

```

#endif

```

```

return ByteRead;
}

//////////
//MAIN LOOP//
//////////

void loop()

{
// Look for sync bytes
if(ReadOneByte() == 170)
{
if(ReadOneByte() == 170)
{
payloadLength = ReadOneByte();

if(payloadLength > 169)           //Payload length can not be greater than 169
return;
generatedChecksum = 0;
for(int i = 0; i < payloadLength; i++)
{
payloadData[i] = ReadOneByte();      //Read payload into memory
generatedChecksum += payloadData[i];
}

checksum = ReadOneByte();           //Read checksum byte from stream
generatedChecksum = 255 - generatedChecksum; //Take one's compliment of generated
checksum
}
}
}
}

```

```
if(checksum == generatedChecksum)
{
    poorQuality = 200;
    attention = 0;
    meditation = 0;

    for(int i = 0; i < payloadLength; i++)
    {
        // Parse the payload
        switch (payloadData[i])
        {
            case 2:
                i++;
                poorQuality = payloadData[i];
                bigPacket = true;
                break;
            case 4:
                i++;
                attention = payloadData[i];
                break;
            case 5:
                i++;
                meditation = payloadData[i];
                break;
            case 0x80:
                i = i + 3;
                break;
            case 0x83:
                i = i + 25;
                break;
            default:
```

```
        break;
    } // switch
} // for loop
```

```
#if !DEBUGOUTPUT
```

```
    // *** Add your code here ***
```

```
    if(bigPacket)
    {
        if(poorQuality == 0)
            digitalWrite(LED, HIGH);
        else
            digitalWrite(LED, LOW);

        Serial.print("PoorQuality: ");
        Serial.print(poorQuality, DEC);
        Serial.print(" Attention: ");
        Serial.print(attention, DEC);
        Serial.print(" Time since last packet: ");
        Serial.print(millis() - lastReceivedPacket, DEC);
        lastReceivedPacket = millis();
        Serial.print("\n");
```

```
    switch(attention / 10)
    {
        case 0:
            /* digitalWrite(GREENLED1, HIGH);
            digitalWrite(GREENLED2, LOW);
            digitalWrite(GREENLED3, LOW);
```

```

digitalWrite(YELLOWLED1, LOW);
digitalWrite(YELLOWLED2, LOW);
digitalWrite(YELLOWLED3, LOW);
digitalWrite(YELLOWLED4, LOW);
digitalWrite(REDLED1, LOW);
digitalWrite(REDLED2, LOW);
digitalWrite(REDLED3, LOW);    */
break;
case 1:
/* digitalWrite(GREENLED1, HIGH);
digitalWrite(GREENLED2, HIGH);
digitalWrite(GREENLED3, LOW);
digitalWrite(YELLOWLED1, LOW);
digitalWrite(YELLOWLED2, LOW);
digitalWrite(YELLOWLED3, LOW);
digitalWrite(YELLOWLED4, LOW);
digitalWrite(REDLED1, LOW);
digitalWrite(REDLED2, LOW);
digitalWrite(REDLED3, LOW);*/
break;
case 2:
digitalWrite(GREENLED1, HIGH);
digitalWrite(GREENLED2, HIGH);
digitalWrite(GREENLED3, HIGH);
digitalWrite(YELLOWLED1, LOW);
digitalWrite(YELLOWLED2, LOW);
digitalWrite(YELLOWLED3, LOW);
digitalWrite(YELLOWLED4, LOW);
digitalWrite(REDLED1, LOW);
digitalWrite(REDLED2, LOW);

```

```
    digitalWrite(REDLED3, LOW);  
    break;  
case 3:  
/* digitalWrite(GREENLED1, HIGH);  
   digitalWrite(GREENLED2, HIGH);  
   digitalWrite(GREENLED3, HIGH);  
   digitalWrite(YELLOWLED1, HIGH);  
   digitalWrite(YELLOWLED2, LOW);  
   digitalWrite(YELLOWLED3, LOW);  
   digitalWrite(YELLOWLED4, LOW);  
   digitalWrite(REDLED1, LOW);  
   digitalWrite(REDLED2, LOW);  
   digitalWrite(REDLED3, LOW); */  
    break;  
case 4:  
    digitalWrite(GREENLED1, HIGH);  
    digitalWrite(GREENLED2, HIGH);  
    digitalWrite(GREENLED3, HIGH);  
    digitalWrite(YELLOWLED1, HIGH);  
    digitalWrite(YELLOWLED2, HIGH);  
    digitalWrite(YELLOWLED3, LOW);  
    digitalWrite(YELLOWLED4, LOW);  
    digitalWrite(REDLED1, LOW);  
    digitalWrite(REDLED2, LOW);  
    digitalWrite(REDLED3, LOW);  
    break;  
case 5:  
    digitalWrite(GREENLED1, HIGH);  
    digitalWrite(GREENLED2, HIGH);  
    digitalWrite(GREENLED3, HIGH);
```

```
digitalWrite(YELLOWLED1, HIGH);
digitalWrite(YELLOWLED2, HIGH);
digitalWrite(YELLOWLED3, HIGH);
digitalWrite(YELLOWLED4, LOW);
digitalWrite(REDLED1, LOW);
digitalWrite(REDLED2, LOW);
digitalWrite(REDLED3, LOW);
break;
case 6:
    /*digitalWrite(GREENLED1, HIGH);
    digitalWrite(GREENLED2, HIGH);
    digitalWrite(GREENLED3, HIGH);
    digitalWrite(YELLOWLED1, HIGH);
    digitalWrite(YELLOWLED2, HIGH);
    digitalWrite(YELLOWLED3, HIGH);
    digitalWrite(YELLOWLED4, HIGH);
    digitalWrite(REDLED1, LOW);
    digitalWrite(REDLED2, LOW);
    digitalWrite(REDLED3, LOW); */
    break;
case 7:
    /* digitalWrite(GREENLED1, HIGH);
    digitalWrite(GREENLED2, HIGH);
    digitalWrite(GREENLED3, HIGH);
    digitalWrite(YELLOWLED1, HIGH);
    digitalWrite(YELLOWLED2, HIGH);
    digitalWrite(YELLOWLED3, HIGH);
    digitalWrite(YELLOWLED4, HIGH);
    digitalWrite(REDLED1, HIGH);
    digitalWrite(REDLED2, LOW);
```



```

digitalWrite(REDLED3, LOW);  */
break;
case 8:
/* digitalWrite(GREENLED1, HIGH);
digitalWrite(GREENLED2, HIGH);
digitalWrite(GREENLED3, HIGH);
digitalWrite(YELLOWLED1, HIGH);
digitalWrite(YELLOWLED2, HIGH);
digitalWrite(YELLOWLED3, HIGH);
digitalWrite(YELLOWLED4, HIGH);
digitalWrite(REDLED1, HIGH);
digitalWrite(REDLED2, HIGH);
digitalWrite(REDLED3, LOW);*/
break;
case 9:
/*digitalWrite(GREENLED1, HIGH);
digitalWrite(GREENLED2, HIGH);
digitalWrite(GREENLED3, HIGH);
digitalWrite(YELLOWLED1, HIGH);
digitalWrite(YELLOWLED2, HIGH);
digitalWrite(YELLOWLED3, HIGH);
digitalWrite(YELLOWLED4, HIGH);
digitalWrite(REDLED1, HIGH);
digitalWrite(REDLED2, HIGH);
digitalWrite(REDLED3, HIGH);*/
break;
case 10:
/*digitalWrite(GREENLED1, HIGH);
digitalWrite(GREENLED2, HIGH);
digitalWrite(GREENLED3, HIGH);

```

```

        digitalWrite(YELLOWLED1, HIGH);
        digitalWrite(YELLOWLED2, HIGH);
        digitalWrite(YELLOWLED3, HIGH);
        digitalWrite(YELLOWLED4, HIGH);
        digitalWrite(REDLED1, HIGH);
        digitalWrite(REDLED2, HIGH);
        digitalWrite(REDLED3, HIGH);*/
        break;
    }
}
#endif

    bigPacket = false;
}
else {
    // Checksum Error
} // end if else for checksum
} // end if read 0xAA byte
} // end if read 0xAA byte
}

```

EEG: Thought 3

```

#define LED 13
#define BAUDRATE 57600
#define DEBUGOUTPUT 0

//all white is green
#define GREENLED1 3
#define GREENLED2 4
#define GREENLED3 5

```

```

#define YELLOWLED1 6
#define YELLOWLED2 7
#define YELLOWLED3 8
#define YELLOWLED4 9
#define REDLED1 10
#define REDLED2 11
#define REDLED3 12

#define powercontrol 10

// checksum variables
byte generatedChecksum = 0;
byte checksum = 0;
int payloadLength = 0;
byte payloadData[64] = {0};
byte poorQuality = 0;
byte attention = 0;
byte meditation = 0;

// system variables
long lastReceivedPacket = 0;
boolean bigPacket = false;

////////////////////
// Microprocessor Setup //
////////////////////
void setup()

{
  pinMode(GREENLED1, OUTPUT);

```

```

pinMode(GREENLED2, OUTPUT);
pinMode(GREENLED3, OUTPUT);
pinMode(YELLOWLED1, OUTPUT);
pinMode(YELLOWLED2, OUTPUT);
pinMode(YELLOWLED3, OUTPUT);
pinMode(YELLOWLED4, OUTPUT);
pinMode(REDLED1, OUTPUT);
pinMode(REDLED2, OUTPUT);
pinMode(REDLED3, OUTPUT);

pinMode(LED, OUTPUT);
Serial.begin(BAUDRATE);    // USB
}

////////////////////////////////////
// Read data from Serial UART //
////////////////////////////////////
byte ReadOneByte()

{
  int ByteRead;
  while(!Serial.available());
  ByteRead = Serial.read();

#ifdef DEBUGOUTPUT
  Serial.print((char)ByteRead); // echo the same byte out the USB serial (for debug purposes)
#endif

  return ByteRead;
}

```

```

//////////
//MAIN LOOP//
//////////
void loop()

{
  // Look for sync bytes
  if(ReadOneByte() == 170)
  {
    if(ReadOneByte() == 170)
    {
      payloadLength = ReadOneByte();

      if(payloadLength > 169)           //Payload length can not be greater than 169
      return;
      generatedChecksum = 0;
      for(int i = 0; i < payloadLength; i++)
      {
        payloadData[i] = ReadOneByte();      //Read payload into memory
        generatedChecksum += payloadData[i];
      }

      checksum = ReadOneByte();           //Read checksum byte from stream
      generatedChecksum = 255 - generatedChecksum; //Take one's compliment of generated
checksum

      if(checksum == generatedChecksum)
      {
        poorQuality = 200;

```

```
attention = 0;
meditation = 0;

for(int i = 0; i < payloadLength; i++)
{
    // Parse the payload
    switch (payloadData[i])
    {
    case 2:
        i++;
        poorQuality = payloadData[i];
        bigPacket = true;
        break;
    case 4:
        i++;
        attention = payloadData[i];
        break;
    case 5:
        i++;
        meditation = payloadData[i];
        break;
    case 0x80:
        i = i + 3;
        break;
    case 0x83:
        i = i + 25;
        break;
    default:
        break;
    } // switch
} // for loop
```

```
#if !DEBUGOUTPUT
```

```
// *** Add your code here ***
```

```
if(bigPacket)
```

```
{
```

```
  if(poorQuality == 0)
```

```
    digitalWrite(LED, HIGH);
```

```
  else
```

```
    digitalWrite(LED, LOW);
```

```
  Serial.print("PoorQuality: ");
```

```
  Serial.print(poorQuality, DEC);
```

```
  Serial.print(" Attention: ");
```

```
  Serial.print(attention, DEC);
```

```
  Serial.print(" Time since last packet: ");
```

```
  Serial.print(millis() - lastReceivedPacket, DEC);
```

```
  lastReceivedPacket = millis();
```

```
  Serial.print("\n");
```

```
switch(attention / 10)
```

```
{
```

```
case 0:
```

```
  digitalWrite(GREENLED1, HIGH);
```

```
  digitalWrite(GREENLED2, LOW);
```

```
  digitalWrite(GREENLED3, LOW);
```

```
  digitalWrite(YELLOWLED1, LOW);
```

```
  digitalWrite(YELLOWLED2, LOW);
```

```
  digitalWrite(YELLOWLED3, LOW);
```

```
digitalWrite(YELLOWLED4, LOW);  
digitalWrite(REDLED1, LOW);  
digitalWrite(REDLED2, LOW);  
digitalWrite(REDLED3, LOW);  
break;
```

case 1:

```
digitalWrite(GREENLED1, HIGH);  
digitalWrite(GREENLED2, HIGH);  
digitalWrite(GREENLED3, LOW);  
digitalWrite(YELLOWLED1, LOW);  
digitalWrite(YELLOWLED2, LOW);  
digitalWrite(YELLOWLED3, LOW);  
digitalWrite(YELLOWLED4, LOW);  
digitalWrite(REDLED1, LOW);  
digitalWrite(REDLED2, LOW);  
digitalWrite(REDLED3, LOW);  
break;
```

case 2:

```
digitalWrite(GREENLED1, HIGH);  
digitalWrite(GREENLED2, HIGH);  
digitalWrite(GREENLED3, HIGH);  
digitalWrite(YELLOWLED1, LOW);  
digitalWrite(YELLOWLED2, LOW);  
digitalWrite(YELLOWLED3, LOW);  
digitalWrite(YELLOWLED4, LOW);  
digitalWrite(REDLED1, LOW);  
digitalWrite(REDLED2, LOW);  
digitalWrite(REDLED3, LOW);  
break;
```

case 3:


```

digitalWrite(GREENLED1, HIGH);
digitalWrite(GREENLED2, HIGH);
digitalWrite(GREENLED3, HIGH);
digitalWrite(YELLOWLED1, HIGH);
digitalWrite(YELLOWLED2, LOW);
digitalWrite(YELLOWLED3, LOW);
digitalWrite(YELLOWLED4, LOW);
digitalWrite(REDLED1, LOW);
digitalWrite(REDLED2, LOW);
digitalWrite(REDLED3, LOW);

break;

case 4:
/* digitalWrite(GREENLED1, HIGH);
digitalWrite(GREENLED2, HIGH);
digitalWrite(GREENLED3, HIGH);
digitalWrite(YELLOWLED1, HIGH);
digitalWrite(YELLOWLED2, HIGH);
digitalWrite(YELLOWLED3, LOW);
digitalWrite(YELLOWLED4, LOW);
digitalWrite(REDLED1, LOW);
digitalWrite(REDLED2, LOW);
digitalWrite(REDLED3, LOW);    */
break;

case 5:
/* digitalWrite(GREENLED1, HIGH);
digitalWrite(GREENLED2, HIGH);
digitalWrite(GREENLED3, HIGH);
digitalWrite(YELLOWLED1, HIGH);
digitalWrite(YELLOWLED2, HIGH);
digitalWrite(YELLOWLED3, HIGH);

```

```

digitalWrite(YELLOWLED4, LOW);
digitalWrite(REDLED1, LOW);
digitalWrite(REDLED2, LOW);
digitalWrite(REDLED3, LOW); */
break;
case 6:
    /*digitalWrite(GREENLED1, HIGH);
digitalWrite(GREENLED2, HIGH);
digitalWrite(GREENLED3, HIGH);
digitalWrite(YELLOWLED1, HIGH);
digitalWrite(YELLOWLED2, HIGH);
digitalWrite(YELLOWLED3, HIGH);
digitalWrite(YELLOWLED4, HIGH);
digitalWrite(REDLED1, LOW);
digitalWrite(REDLED2, LOW);
digitalWrite(REDLED3, LOW); */
break;
case 7:
    /*digitalWrite(GREENLED1, HIGH);
digitalWrite(GREENLED2, HIGH);
digitalWrite(GREENLED3, HIGH);
digitalWrite(YELLOWLED1, HIGH);
digitalWrite(YELLOWLED2, HIGH);
digitalWrite(YELLOWLED3, HIGH);
digitalWrite(YELLOWLED4, HIGH);
digitalWrite(REDLED1, HIGH);
digitalWrite(REDLED2, LOW);
digitalWrite(REDLED3, LOW); */
break;
case 8:

```

```
/* digitalWrite(GREENLED1, HIGH);  
digitalWrite(GREENLED2, HIGH);  
digitalWrite(GREENLED3, HIGH);  
digitalWrite(YELLOWLED1, HIGH);  
digitalWrite(YELLOWLED2, HIGH);  
digitalWrite(YELLOWLED3, HIGH);  
digitalWrite(YELLOWLED4, HIGH);  
digitalWrite(REDLED1, HIGH);  
digitalWrite(REDLED2, HIGH);  
digitalWrite(REDLED3, LOW);*/  
break;
```

case 9:

```
/*digitalWrite(GREENLED1, HIGH);  
digitalWrite(GREENLED2, HIGH);  
digitalWrite(GREENLED3, HIGH);  
digitalWrite(YELLOWLED1, HIGH);  
digitalWrite(YELLOWLED2, HIGH);  
digitalWrite(YELLOWLED3, HIGH);  
digitalWrite(YELLOWLED4, HIGH);  
digitalWrite(REDLED1, HIGH);  
digitalWrite(REDLED2, HIGH);  
digitalWrite(REDLED3, HIGH);*/  
break;
```

case 10:

```
/* digitalWrite(GREENLED1, HIGH);  
digitalWrite(GREENLED2, HIGH);  
digitalWrite(GREENLED3, HIGH);  
digitalWrite(YELLOWLED1, HIGH);  
digitalWrite(YELLOWLED2, HIGH);  
digitalWrite(YELLOWLED3, HIGH);
```

```

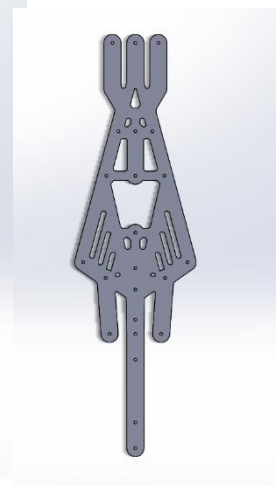
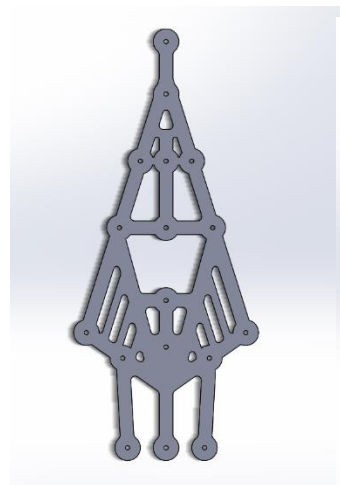
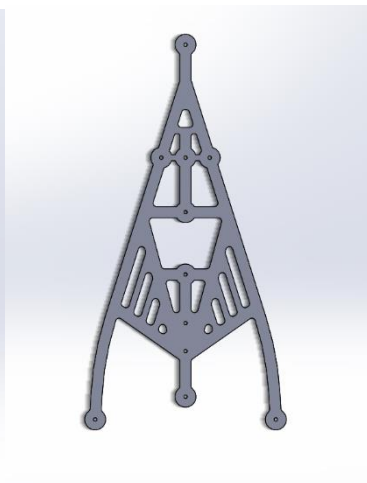
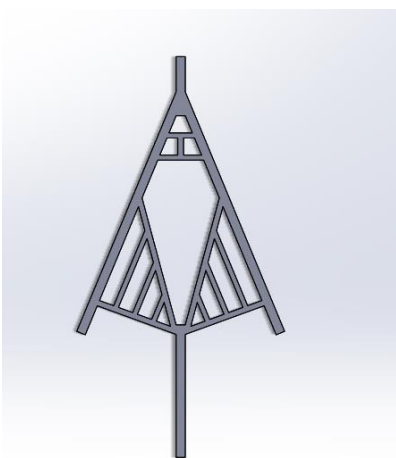
digitalWrite(YELLOWLED4, HIGH);
digitalWrite(REDLED1, HIGH);
digitalWrite(REDLED2, HIGH);
digitalWrite(REDLED3, HIGH);*/
break;
}
}
#endif
bigPacket = false;
}
else {
// Checksum Error
} // end if else for checksum
} // end if read 0xAA byte
} // end if read 0xAA byte

```

Appendix D: CAD FILES

Progression of Muslutendon Simulant.

- The right most image is the final design
- Design based on extensor hood in human finger
- Laser cut from EDPM rubber
- Also used for tendon sheathing



Thumb Design

- Human thumb has a saddle joint
- Ours has a ball-and-socket
- Has additional webbing and specialized MTS

