Gamified Music Learning System with VR Force Feedback for Rehabilitation

A Major Qualifying Project

Submitted to the Faculty of

Worcester Polytechnic Institute

in partial fulfillment of the requirements for the

Degree in Bachelor of Science

in Robotics Engineering

and Biomedical Engineering

By

Connor Flanigan

William Manning

Elizabeth Martino

Date: April 28, 2016

Project Advisors:

Professor Scott Barton, Advisor

Professor Marko Popovic, Advisor

Abstract

Many conditions cause loss of coordination and motor capabilities in the extremities. One such condition is stroke, which affects approximately 15 million people worldwide each year. [1] Many robotic systems have been developed to assist in the physical and neurological rehabilitation of patients who have suffered a stroke. This project combines an alternate exo-skeletal actuator design with gamification elements to create an interface to be used in future robotic rehabilitation systems as well as address the compliance problem found in rehabilitation.

Acknowledgements

We would like to thank Professor Marko Popovic and Professor Scott Barton for providing guidance and assistance for the duration of our project. We would also like to thank Sean Flanigan for helping us with machining. We really appreciate it!

Table of Contents

Table of Figures

Authorship

Introduction	William Manning
Background	Elizabeth Martino
Project Approach	Elizabeth Martino
Alternate designs	
Mechanical	Connor Flanigan
Electrical	William Manning
PC software design	Elizabeth Martino William Manning
Design Verification	
Hardware	William Manning Connor Flanigan
Software	Elizabeth Martino
Experiments and results	William Manning Connor Flanigan Elizabeth Martino
Discussion	William Manning Connor Flanigan Elizabeth Martino
Conclusions and Future Work	Elizabeth Martino

Chapter 1: Introduction

Many conditions cause loss of coordination and motor capabilities in the extremities. One such disability is stroke. Every year, approximately 15 million people worldwide suffer strokes each year. [1] The nature of this nervous system damaging condition makes it an optimal target for a robotic rehabilitation device. The exercises for these patients focus not on muscle strength but on improving synaptic plasticity and rebuilding neural pathways. This is accomplished through daily repetition of fine motor control centered activities. However, similar to when a student learns to play an instrument, physical therapy patients often lose motivation to do these activities due to their inability to quantify their progress in the program.

This project aims to address the compliance issue in stroke rehabilitation using gamification methods, specifically music teaching. A key element of music teaching is providing motivation for music students to practice proper technique. Many methods have been developed to integrate music teaching and technology to encourage students to learn the necessary skills to play an instrument. Leveraging parallels between physical therapy and music teaching, these aids can be used as rehabilitative tools for students with injuries or disabilities that limit fine motor control, in addition to improving the learning process for music students. By introducing a gamified musical element to their regimen, music teaching methods can be applied to the patient's daily routine to encourage practice and the completion of the exercises. The combination of a robotic assistive device capable of recording performance data and the music therapy elements will allow the patient to track their progress as well as feel a sense of accomplishment having completed a fun and challenging activity.

Previous solutions have been developed to actuate the extremities of stroke patients for the purposes of rehabilitation. These solutions including hard and soft actuation as well as integration with gamification elements. Some such solutions are the *CyberGrasp* by CyberGlove systems [2] and the *InMotionArm* [3]. However, there have been obstacles preventing their widespread use in physical therapy, specifically the cost and weight of these systems. [4] This project aimed to expand on the hardware of these systems by creating a lightweight and low profile system that would be optimal for use by stroke patients with reduced muscular strength and range of motion. With this system, we also aim to demonstrate how a robotic rehabilitation system could be integrated with gamification elements such as virtual reality technology and musical elements to create an engaging experience addressing the issue of compliance in rehabilitation.

There were two designs considered for motivating the patient's hands. The first was Bowden cables: inexpensive, lightweight cables that can be used to apply either a pushing or a pulling force. These cables require a linear actuator for their control. The second option was a pneumatic continuum actuator. This method of actuation can allow for variable stiffness, which is important in modeling human motor functions, as well as actuation in 3 degrees of freedom. However, these pneumatic systems require a compressor and expensive valves and solenoids for their operations. We examined multiple sensors for determining the hand position as well as calculating the forces being applied to the hand. The four kinds of sensors we discussed were IR cameras (Leap Motion), encoders, potentiometers and piezoelectric deflection sensors.

The final design is a Bowden cable driven system. A motor is directly coupled to a 3.5 inch screw drive. The screw drive drives a dual spring force transducer inside a linear bearing sheath to prevent unwanted twisting. The dual spring system is connected to the Bowden cable via a piston, and the Bowden cable attaches directly to the hand. The motor is driven by an H-Bridge motor controller. The hand position is determined in two ways. Firstly, the *Leap Motion* uses high-resolution skeletal tracking to determine both the hand's position in space as well as joint angles. Secondly, a combination of a 10 kΩ slide potentiometer and an encoder determine the relative finger position as well as the position of the system overall. The difference between these positions is used to determine the force being applied to the hand. The system is controlled by an *Arduino Mega 2560* and powered by a 400W PC power supply.

There are many options for creating visual and audio effects. We aimed to integrate a number of these industry standard applications with the control of our system. This integration demonstrates that third-party external applications can be made to establish communication between existing software for the specific purpose of creating rehabilitation applications. We used five pieces of software; *Arduino, Cycling '74 Max 7, Ableton Live 9,* the *Leap Motion* SDK and the *Unity 3D* engine.

We were able to achieve our primary objectives of safety, low weight and decreased cost. The system only weighs 4.8 lbs for the entire hand and the total cost for fabrication was only $2500 which is less than solutions currently on the market. The system was able to output 1.6 lbs of force at the end effector and move at 0.21 inches per seconds. While this is lower than preferred performance, this can be rectified by better materials, stronger Bowden cables and a higher power motor. Future work on this project should involve expanding the model to the entire hand including the thumb, wrist, abduction and adduction. Future work should also include conducting a user study to determine the level of engagement provided by the chosen gamification software as well as how best to use this software to address the compliance issue.

Chapter 2: Background

2.1 The Prevalence of Stroke

Stroke is one of the most prevalent causes of neuromuscular damage. Approximately 15 million people worldwide suffer from a stroke each year. [1] A stroke is caused by a blockage of blood flow to the brain, causing the death of brain cells. (Figure 1) This causes a loss of motor and cognitive functions, usually on one side of the body. Long term disability occurs in many stroke victims with 50% of patients experiencing partial paralysis and 30% of patients requiring assistance in order to walk. [5] A loss of muscle mass is associated with stroke, due to disuse, causing decreased strength and range of motion. One study found that patients can lose up to 4.5% of the muscle mass in their lower limb and up 14.5% muscle mass in the upper thigh, if they are unable to walk 6 months after the stroke. [6]

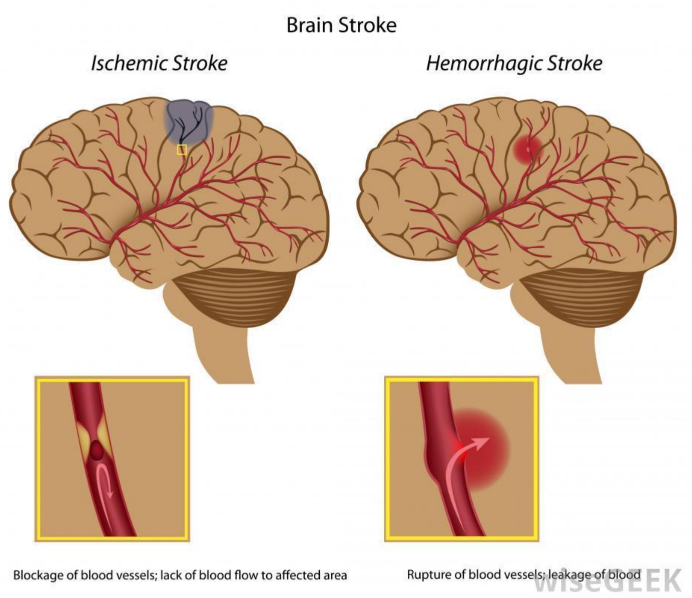

Brain Stroke
Ischemic Stroke
Hemorrhagic Stroke
Blockage of blood vessels; lack of blood flow to affected area
Rupture of blood vessels; leakage of blood
WiseGEEK

Figure 1. Example of Ischemic and Hemorrhagic Stroke

2.2 Physical Therapy for Stroke Victims

2.2.1 Methods of Stroke Therapy

In stroke therapy, exercises assigned to patients are intended to help the patient focus on improving plasticity and rebuild neural pathways lost as a result of the stroke. This drives the types of activities to be less focused on rebuilding muscle strength and more focused on regaining certain motor functions like grasping and reaching, using goal-oriented exercises. [7] There are two common benchmarking tests used in stroke therapy. The first is the Box and Block test. This involves moving small blocks from one box to another The score is calculated as how many blocks the patient is able to move into the box in a 60 second time period. [8] The second is the Nine-Hole Peg Test. This involves the patient placing pegs into small holes on a board. The patient’s score is calculated as how quickly they can place all of the pegs and then remove them and return them to the container. [9] Patient improvement as well as the success of different therapy methods are measured using these types of tests.

Manual physical therapy is also prominent in stroke therapy. Manual therapy involves the therapist physically manipulating the joints and limbs of the patient to complete the exercise. The exercises can be categorized as active or passive. Passive therapy is when there are no forces but those of the patient's own muscles acting to complete the exercise. Active manual therapy is when the therapist will apply external forces, either in the direction of the activity (active assistive therapy) or opposing the patient's movements (active resistive therapy). One example of active resistive therapy specific to stroke, is constraint limited therapy**.** In most cases of stroke, the patient will lose functionality only on one side of the body. By physically constraining the patient's functional side, the therapist can encourage movement in the non-functional side. [7]

2.2.2 Challenges in Stroke Therapy

There are many challenges in stroke therapy that limit the benefits to the patients. The first and most prominent issue is that of compliance. Many patients do not regularly complete the exercises prescribed by their therapists. Patients will often get frustrated by the lack of immediate improvement, or they will find the exercises uninteresting. Providing motivation for physical therapy patients has been a major area of research. [10] Another challenge to stroke rehabilitation is the availability of peripheral equipment. Due to travel or financial limitations, many patients can only receive physical therapy once or twice a week. For some of these patients, the exercises have a greater impact at more intensive intervals. [4] A third challenge in stroke therapy is that without external hardware, the patients and clinicians have no way to measure the patient's progress. [11]

2.2.3 Gamification in Physical Therapy

Gamification is the use of game-like elements to motivate and encourage a user to do non-gaming activities. [12] This motivation comes from rewarding an objective done well and providing goals for improvement. One example of this is called We Day, which is a movement to get youth to work to improve their community and gain leadership skills. This movement uses gamification by using apps for youth to use that track how many hours of community service they do, and how they improve in both the quality and quantity of their community service.. This provides motivation because it allows the youths to see where they were, and how much they improved, to motivate them to improve to do more and to contribute to the world in the ways they learn from other children. The youth are motivated to continue their work by not only being able to visualize the success they already had and how others are doing, but also through the peer recognition of their work. [13]

Gamification is already being used in the field of physical therapy. One such example is the use of the CONTRAST arm rehabilitation program. This program sets up many small games for use on a tablet that a stroke victim can use to stimulate their arm and complete “task oriented activities”. [14] This program was tested and worked well at engaging the user and creating a want to complete a game, thus improving motivation to complete the exercises through diversity of difficulty and also the design of the games, which allows users to work on many different skills they need to regain, like the ability to pinch. It also improved motivation by providing visual representations of improvement in how well they completed the tasks. The program was successful, even improving the use of the arm for the stroke survivor [14]

The use of the haptic feedback has also been shown to be successful in physical therapy. The use of the A Virtual-Reality-Based Telerehabilitation System with Force Feedback is an arm used to help manipulate the user’s arm to redevelop their motor skills. [14] This arm was used to help move the arm of the patient, with the feedback noticeably helping through forced actuation. [14] In our project, we wish to combine the gamification theory with feedback, to make an encompassing physical therapy tool.

2.2.4 Music in Physical Therapy

There are many benefits associated with using music in conjunction with physical therapy. Many parallels exist between music teaching methods and physical therapy, primarily in the area of using motivational tools and the reinforcement of repetitive tasks to teaching technical skills. [15] In addition, listening to and creating music involves a number of cognitive and motor skills that make it a strong choice for use in stroke therapy. Musical training can improve auditory processing, linguistic skills, spatial and mathematical skills, and social skills. Trained musicians show increased plasticity in the critical cognitive periods both in childhood as well as later into adulthood. [15] In addition to the cognitive benefits of music, due to the level of engagement that music provides as well as its rewarding nature, it also serves well as a motivational tool to address the compliance issue inherent to physical therapy. [10] With these attributes of music related activities in mind, musical activities are ideal for use in stroke therapy.

One example of a musically facilitated therapy program is the MusicGlove created by Flint Rehabilitation Devices. (Figure 2) The glove is a wearable sensor that detects hand position and orientation and uses the output data in a music-based video game. [16] Patients that use the MusicGlove to perform, gripping and grasping tasks have shown a faster rate of improvement in motor functions than patients participating in conventional physical therapy. One such test is the Box and Block test, which is when one sets up a box with a partition inside of it. The user must carry a block using their dominant hand across the partition and drop in onto the other side for a minute. They then repeat this with their non-dominant hand. Using the Box and Block test, the patients scores increased from -0.29±2.27 blocks for conventional grip training to 3.21±3.82 blocks using the MusicGlove. Patients also found that using the MusicGlove, they were much more motivated to complete their exercises, demonstrating the potential benefits of music as a solution to the compliance issue. [16]

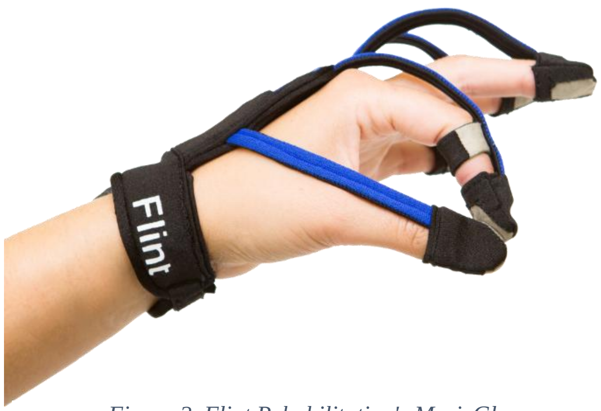


Figure 2. Flint Rehabilitation's MusicGlove

2.5 Robotics as a Means for Rehabilitation

Robotic assistive devices can act as an all-in-one system for the degrees of freedom it is capable of actuating, passively or actively, for both resistive and assistive methods. In the case of stroke rehabilitation, this allows the user to begin by creating muscle memory from assistive movement, potentially by measuring the movement of another limb Robotics have been used in rehabilitation as a tool to add to or augment existing care. Patient rehabilitation using a robotic device allows a physical therapist to create a more personalized treatment plan for each patient taking into account their rate of improvement as well as their specific needs. [4]

Further, systems can be rented to patients to use remotely as an outpatient, either between visits to a therapist, potentially to reduce the frequency of visits, or entirely remotely, reducing the cost associated with complex and laborious medical procedures. The integration of a digital therapy system with a computer allows the therapy device to be used a peripheral controller in a video game. As previously stated, patients frequently do not follow through with all of their exercises, it then follows that adding an incentive in the form of a game would increase compliance. Further, music has been shown to help provide motivation to patients, so it follows that a music game using a therapy device as a controller would increase compliance immensely.

2.5.1 Effective Uses of Robotics in Physical Therapy

Many methods have been developed to use robotics in musculoskeletal rehabilitation. Systems can be wearable or fixed. Wearable systems are supported by the user, and fixed system or supported by a desktop, the floor or some rigid surface that isn't attached to the patient. [17] One example of a fixed system is the *InMotionArm*. (Figure 3) The user places their forearm in a small brace and manipulates a joystick to play video games that correspond with physical therapy exercise. [3]

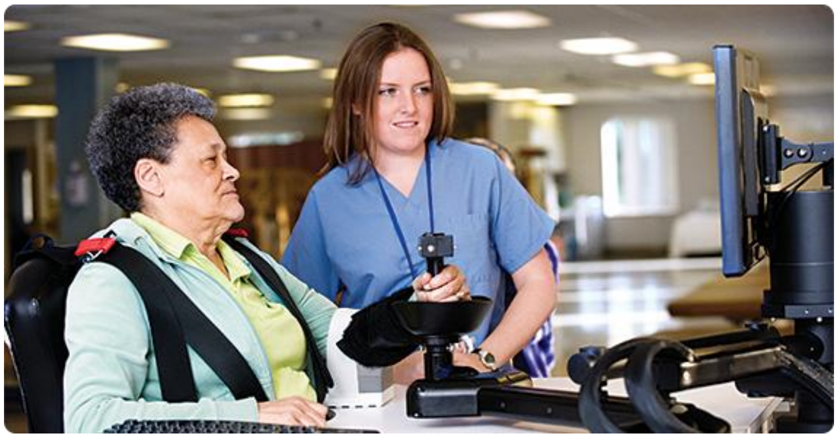


Figure 3. InMotionArm in action [3]

One benefit of a fixed system is that the user doesn't have to lift or move any weight other than what is required to operate the system. [3] However, a wearable system is more likely to be easily transported, which can improve accessibility.

Two categories of rehabilitative systems are those that use "hard" actuators and those that use "soft" actuators. "Hard" actuation consists of a system which is not designed to have an elastic element, for instance, a hydraulic system, which uses an incompressible fluid to (typically) displace a piston. Two types of hard actuators are motors attached to gear trains and linkages. One example of a wearable system using hard actuators is the HEXOSYS II. (Figure 4) The HEXOSYS II uses a series of rigid linkages to actuate the joints of the fingers by applying a displacement. [18]

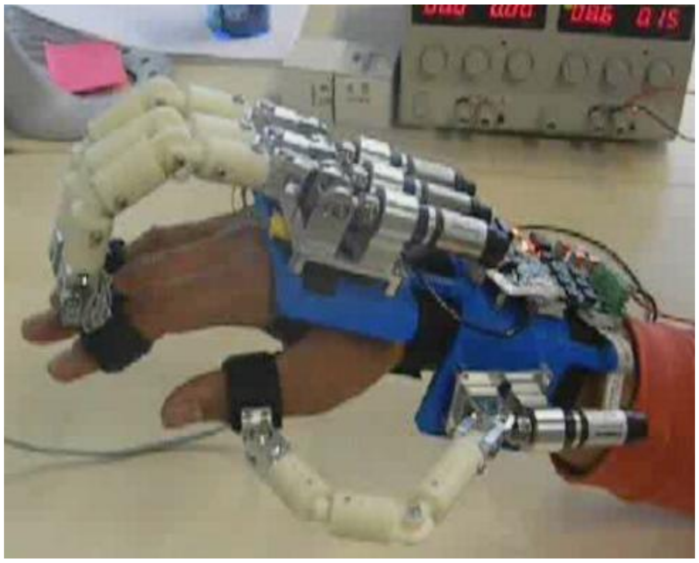


Figure 4. The HEXOSYS II, an assistive device that uses hard linkages

Soft actuation consists of a system which has an elastic element designed in it, such as a pneumatic system, which uses a compressible fluid to (typically) apply a force to a piston or motor. Two types of soft actuators are pneumatically driven systems and motors with elastic components in series. One example of a soft actuated system is the continuum actuator glove. (Figure 5) The continuum actuator fills the elastic tube with fluid, causing it to extend. One plane of the tube is rigid, causing a bending motion the finger. This device showed a mild degree of success. [19]

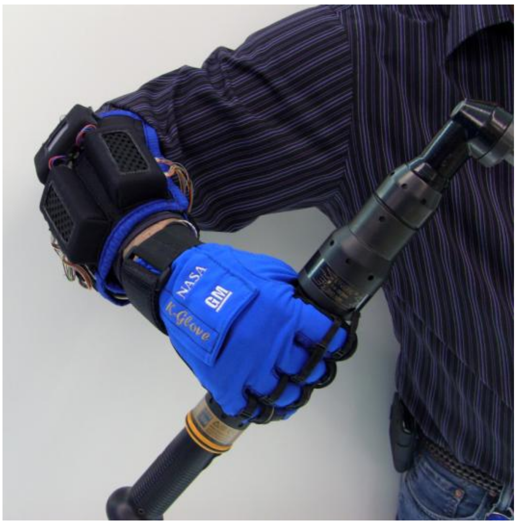


Figure 5. Pneumatic continuum actuator glove

A large variety of sensors, such as accelerometers, visual processing, flexion sensors and various methods of forward kinematics, are used to operate the systems that have been successful in testing and in clinical settings. One method, used in the WPI Bowden Cable Exoskeleton Hand MQP in 2013, (Figure 6) used Electromyogram (EMG) to measure muscle contractions in the forearm. The device was able to actuate the hand using feedback from the EMG to detect hand positions. [20]

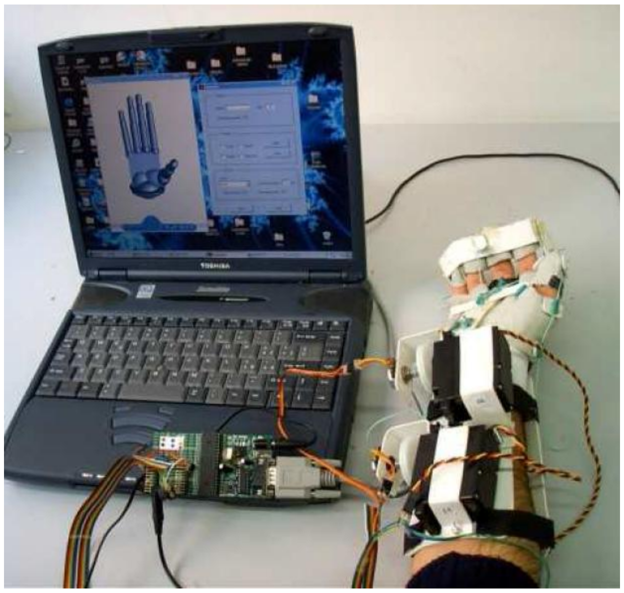


Figure 6. An EMG Controlled Exoskeleton for Hand Rehabilitation [20]

There are also passive systems that can assist in rehabilitation. The company Saebo develops rehabilitative products that aim to assist patients who’ve suffered a loss of motor control in daily activities involving gripping and pinching. The SaeboFlex and the SaeboGlove (Figure 7) have elastic components that assist the user in extending their fingers to release objects. [21] While this system can potentially improve the user’s quality of life as well as their motor capabilities, it lacks the integration with software that could provide valuable diagnostic data to the patient’s physical therapist.

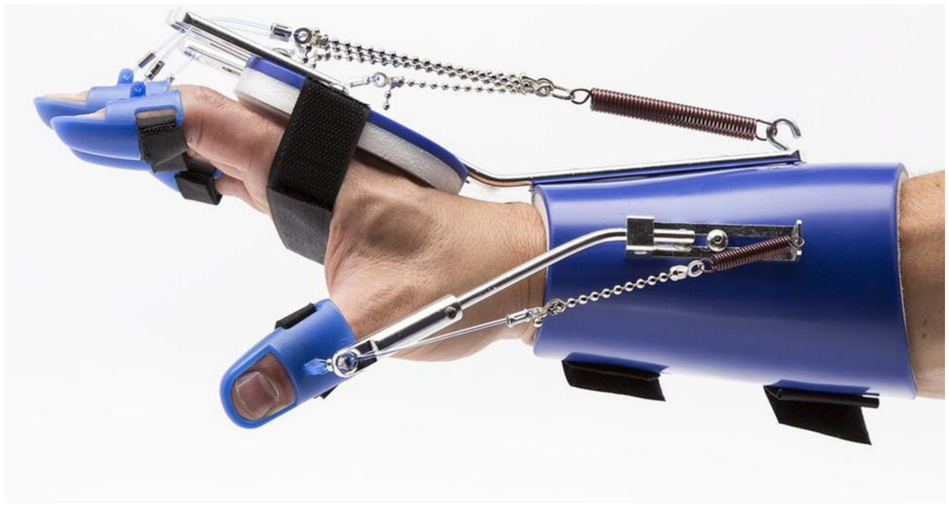


Figure 7. SaeboFlex

One system that is popular in the realm of physical therapy, as well as tactile feedback for other purposes, is the *CyberGrasp* made by CyberGlove systems. (Figure 8) The *CyberGrasp* uses a cable driven system to actuate the hand. The system is lightweight at 0.6 lbs per hand and can apply 2.7 lbs of pinching force at each finger. However, the system has a large profile and is too expensive for widespread use.

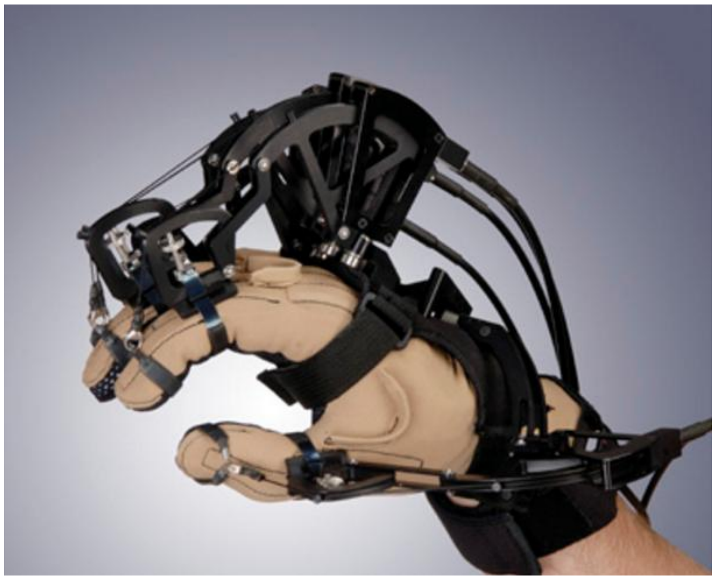


Figure 8. CyberGrasp by CyberGlove Systems [1]

2.5.2 Challenges in Robotics and Physical Therapy

There are many challenges that arise when robotic solutions are applied to stroke therapy. This project will address five of these challenges, outlined below:

Challenge 1: Providing engaging activities and useful feedback to the user

Addressing the compliance issue in physical therapy is a key contribution of robotics in physical therapy. The nature of robotics is to provide an interaction between sensing and actuation. This makes incorporating a gamification element into physical therapy using a robotic system a valid solution to the compliance issue. However, in order for this to succeed, the feedback to the user needs to be useful with respect to the exercise or the overall therapy program. [10]

Challenge 2: Difficulties in modeling the function of the body part being assisted

In order to provide the correct rehabilitative care, the system has to be capable of motivating the body in a way that accurately models how the patient will be functioning in everyday life. For example, in the case of designing a prosthetic leg, the structure of the leg would have to be such that it is capable of support the weight of the patient’s upper body, as well as bend for walking, jumping, etc. [22] This project is focused on the functions of the hands and wrist. The majority of the fingers are actuated by tendons that attach to the joints of the fingers and the bones of the arm. Functions such as abduction and adduction of fingers, movement of the thumb and movement of the wrist are controlled by a large set of muscle groups in similar configurations. [23]

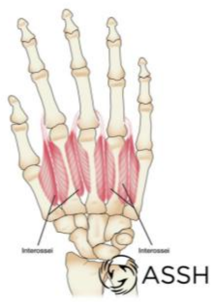


Figure 9. Lumbrical Muscles

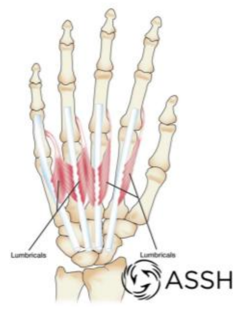


Figure 10. Interosseous Muscles

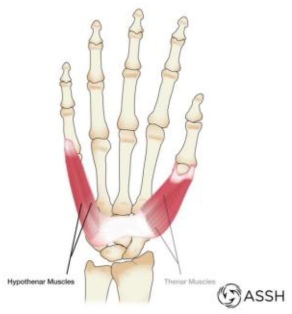


Figure 11. Hypothenar Muscles

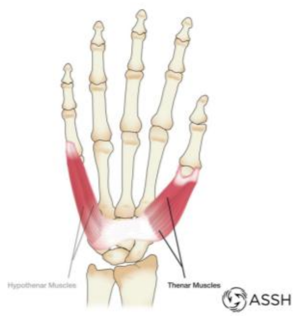


Figure 12. Thenar Muscles

There are four main muscle groups of the hand. The first, in Figure 2.9, is the lumbrical muscles. These muscles are responsible for abduction and adduction of the interphalangeal joints. The second muscle group, the interosseous muscles are responsible for abduction and adduction of the fingers. (Figure 10) The hypothenar and the thenar muscles control the flexion of the little finger and the thumb respectively. (Figures 11 and 12) A table depicting all muscles and tendons responsible for movement in the hand is available in Appendix A.

These elements are elastic, and, in a healthy functioning hand, the muscles and tendons exhibit modulating stiffness to allow for precise control of the displacement and force applied by the different parts of the hand. [23] In order for a rehabilitative device to be a successful method of treatment, it must emulate these characteristics.

Challenge 3: Implementing a near instant response time

Simulation sickness is when a user experiences motion sickness symptoms when there are large latencies between the user's input and the game's output. Simulation sickness can occur in games or game related media if the latency exceeds 48 ms. [24] In order to provide an optimal experience while using the system, the overall latency between the sensor inputs and the actuation should be minimized.

Challenge 4: Degrees of freedom vs. the size and weight of the system

A major obstacle in designing robotic rehabilitation devices, especially in the case of stroke therapy and other disorders causing a loss of motor skills, is the tradeoff between degrees of freedom and the size and weight of the system. When the number of degrees of freedom the system must actuate in increases, the number of actuators required on the system increases, naturally increasing the weight and size of the materials used. [4] To completely actuate the hand, there are approximately 30 degrees of freedom that must be controlled. [25] Muscle mass loss in stroke victims makes heavy wearable systems an invalid option. Currently used solutions to this problem have been large, fixed systems that would be impossible to transport to a patient's home. [26] A successful system would be small and lightweight enough to be easily transported, to accommodate patients with travel restrictions as well as be operate by those with largely decreased muscle strength and range of motion.

Challenge 5: Low availability due to cost

Most of the systems described in section 2.6.1 are only affordable to large hospitals. Oftentimes, hospitals will only be able to purchase a small number of systems making the therapy available to patients less frequent than is optimal for rehabilitation. One example of this problem is the cost of the *CyberGrasp* system, which can cost upward of $10,000. [27] Therapists say that patient's showed improved motor functions after completing more intensive and frequent sessions of therapy using a robotic device. [4] A successful system would be affordable enough that the patient could rent the system from the hospital or some other firm for the duration of their rehabilitation program. This would allow them to have daily access to the system from their own home.

Chapter 3: Project Approach and Goals

3.1 Need Statement

Since this project concept was student generated, we weren't given a client statement. Based on the original concept, the team developed the following statement to describe the outcomes of the project.

> "A key element of music teaching is providing motivation for music students to practice proper technique. Many methods have been developed to integrate music teaching and technology to encourage students to learn the necessary skills to play an instrument. Leveraging parallels between physical therapy and music teaching, these aids can be used as rehabilitative tools for students with injuries or disabilities that limit fine motor control, in addition to improving the learning process for music students. We would like to design a lightweight exoskeleton to facilitate rehabilitation using music teaching. The exoskeleton will interface with a software that provides visual and musical feedback to the user. The system will provide musical exercises for physical therapy patients as well as allow users to play songs using the virtual instrument."

3.2 Project Objectives

From this client statement, we were able to determine the following project objectives. The objectives are presented according to priority.

1. **Safe**

   The most important consideration to make for this design is the safety of the user and any other person that might be involved in the operation of the aid. The system cannot injure any involved persons.

2. **Lightweight**

   The user base of this system will be stroke patients, who in most cases have decreased muscle strength and range of motion. For these patients, lifting heavy actuators will make it nearly impossible to use the aid.

3. Cost Effective

By keeping the costs of the system low, hospitals and other clinics will be able to purchase multiple systems and possibly rent them out for use by individuals. This will increase the availability of the system for the users.

4. Provides useful user feedback

There are many robotic systems that are currently used that manipulate the user’s limbs. However, not all of these manipulators provide feedback to the user during operation. This design will provide force feedback as well as visual and musical feedback to the user which will not only assist in the completion of the exercise but will also provide motivation to the user.

5. Precision sensing

Precision sensing was given the second highest priority among the objectives because it is critical for the function of the system. Precision sensing will be responsible for providing the user with useful feedback on their performance. Also, without precision sensing of the force and position, actuation can’t be precise. Without precise actuation, the device won’t be capable of providing rehabilitative benefits to the user.

6. Adaptable for different exercises

The third most important objective is that the system can be used for a large variety of physical therapy exercises. This will allow the system to be helpful to the user throughout the duration of their physical therapy program. The system should be able to provide assistive and resistive forces.

7. Has integration with existing software

While integration with existing software and creating an API may seem like similar objectives, it is more important for the system to be versatile than it is for it to be packaged with a pre-existing software. Ideally, the system would be coupled with a program as well as having an API, having an API alone is a higher priority.

8. One size fits most

While it is important that the system can be used by users with all hands sizes, it is simpler and more time efficient to demonstrate a proof of concept for an average sized hand, and then allow the manufacturers to create the same system with different dimensions. Therefore, one size fits all was given the lowest priority.

3.3 Revised Need Statement

After determining the goals of the project, we were able to revise the client statement into a more succinct statement of need.

1. Design an alternate robotic assistive device that will expand on current physical therapy methods for fine motor control by developing methods to improve accessibility via decreasing the cost and weight of the system
2. Design a controller that can be adapted to future robotic assistive devices that provide an engaging experience using gamification elements to motivate patients to complete their regular therapy regimen and provide useful feedback on their performance.

3.4 Design Specifications

From the objectives and constraints defined in this chapter, we were able to determine physical specifications that the system must meet in order for the project to be considered successful.

Latency

To avoid simulation sickness, we need to keep below a latency of 48 ms. [24] Taking into consideration the friction of the physical system and our inability to access the most mechanically efficient materials to build our prototype, we set a specification of 100 total ms of latency between when the user inputs a command and when the force is output at the end effectors of the prototype. This will allow for large amounts of friction in the physical system, the latency between the microcontroller and the PC as well as the latency of any other PC software used.

Speed of the physical system

We were able to use the *Leap Motion* skeletal tracking to measure the maximum speed of a finger at 10 m/s. In stroke patients, this speed is greatly reduced. On average, stroke patients will be able to move their full range of motion in approximately 1.9 sec. [28] The full range of motion of our system is 3.5 inches, so the goal speed of the system is 3.5 inches/sec.

Output force at the end effector

The average pinching strength for stroke patients is 1.2 lbs. [29] Since our system is designed to provide motivational forces, instead of actuation forces, we decided to benchmark our force output goal to the pinching capability of the *CyberGrasp* which is 2.7 lbs. This will provide more than enough force to motivate the patient's finger while reducing the risk of physical harm to the patient's hand.

Weight

The average amount of torque that can be supported at the wrist is between 14.1 and 16.7 lb-ft from flexion to extension. [30] On average stroke patients suffer an 18% loss of muscle strength. [6] In order for the system to be usable by patients who have recently suffered a stroke, the weight being supported by the patient should be between 11.6 and 13.7 lb-ft. If we assume the average distance from the wrist to the center of gravity of the supported weight is 6 inches, the weight on the patient's hand should not exceed 5.8 lbs.

Chapter 4: Alternate Designs

4.1 Preliminary Mechanical designs

The first step in the mechanical design is to consider how to motivate the finger. Firstly, Bowden cables, which allow for a large amount of versatility if built so the attachment points are not at fixed lengths from each other provide a low profile, which allows vision sensors to work without interference and does not impede its own motion, as well as not requiring too much mass to attach to the hand. Bowden cables would also allow us to choose which form of actuator to be used on the other end. Figure 13 shows an example of a Bowden cable actuator. Pneumatic continuum actuators are much larger profile, but because of their non-metal construction are much lower weight. The actuator for the system would require a large amount of pneumatic control. Figure 14 shows a pneumatic continuum actuator. The overall tradeoff is for the full 3DoF the continuum actuator is capable of. The last method of actuation considered were linkage systems. The linkages allow for very precise manipulation but require the entire machine to be held on the hand, drastically increasing weight.

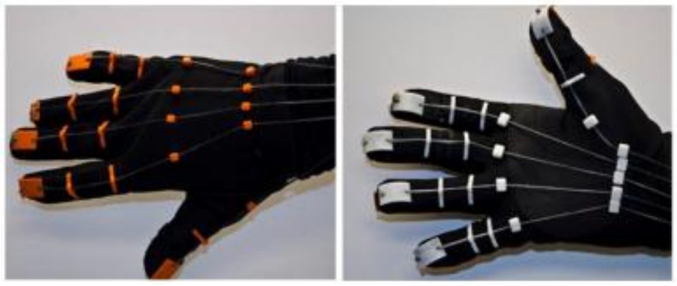


Figure 14. Bowden cable actuated rehabilitative glove [31]

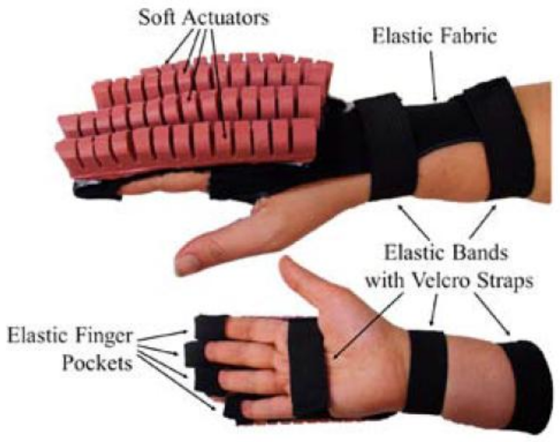


Figure 13. Pneumatic continuum actuated rehabilitative glove [32]

The linear actuation systems considered were DC motors attached to a linear gear system, and electromagnetic linear motor, both of which would require an elastic element or a pneumatic piston system. A motor attached to a linear gear system is fairly simple and cheap to produce, but are relatively high friction, or are more complex low friction systems. Linear motors behave similarly to rotary motors without the need for gear systems but are much more expensive. Both methods require an elastic element which, for rotary motors, could be a friction plate or torque converter, both of which are complex and require a complex system to operate and control. A spring system, using metal springs, magnetic springs, or pneumatic springs would allow simpler control. A pneumatic driver is inherently soft actuation. Both pneumatic systems require pneumatic control, which, in order to compress enough air quickly enough, require a loud compressor and a large system of solenoids, which is very expensive when they need to operate very rapidly.

To start again with the actuation of the finger itself, the linkage systems is far too heavy to be viable if it’s small enough to move. Between pneumatic continuum actuators and Bowden cables, the pneumatic control systems required for a continuum actuator is far too expensive, requiring a high throughput compressor and very fast actuating solenoid valves. For this reason, we eliminated the continuum actuators and pneumatic piston drivers.

Given that we’re using Bowden cable, we needed a linear actuator with an elastic system. Linear motors are far too expensive to be plausible, so a rotational motor was chosen. The option to use a screw drive in series was the simplest and cheapest solution, as it required less linear slides than a rack and pinion, or other linear gear systems, which meant it would be easier to build and maintain. Lastly, the elastic element could be steel springs, pneumatic springs, or magnetic springs. Steel spring systems are cheap, but wear out; periodic maintenance is fairly simple. Pneumatic springs wear out much more slowly, but have more static friction, and may move their neutral position if one chamber leaks into another. Magnetic springs are very difficult to wear down by normal use, but lose strength if any impact hits the system, and can transfer their magnetic fields to and across ferrous materials. The best solution was to use steel springs and design the system so the springs can be easily replaced.

In addition to the mechanism itself, the linear slider needed rails of some sort, so we needed to choose which mechanism to keep everything co-linear. A very simple solution is to take a pair of parallel rails, one with two points of contact, the other with a single point, which keeps the whole mechanism moving smoothly in a line. Fewer points increase instability and more points increase friction. A simple and well-proven solution is to use LM8UU bearings, the same used on many 3D printers, which reduced the price of bearings in small numbers, making this the most viable method.

4.2 Preliminary electrical designs

There are many methods for determining the location and orientation of the hand. Previous projects have used piezoelectric sensors or electromyograms (EMG) to determine the intent of the user to contract a muscle. [33] One objective of this project is that the controller be able to follow a patient through the extent of their rehabilitation. The controller must be versatile enough that a system will be able to provide assistive forces as well as resistive forces in an application agnostic implementation. With this in mind, as well as the decision to use soft actuation, we examined sensors that would provide a high-resolution description of the location and orientation of the hand, as opposed to only detecting the intended movement from the user. We examined four methods of detecting the position and forces of the hands. These methods were piezoelectric flexion sensors, linear variable resistors, pneumatic pressure sensors and visual processing.

Piezoelectric deflection sensors

Deflection sensors are small piezoelectric sensors that would be fixed in parallel with the finger whose resistance changes when the finger is flexed or extended. Using an array of these sensors at the different joint of the fingers and hands would provide a description of the orientation of the fingers relative to the hand. There are a number of issues surrounding the use of flexion sensors. The sensors themselves are very prone to noise. [34] Since there are over 30 degrees of freedom in the fingers and thumb alone, a large number of the sensors would be required to accurately measure the finger orientation, requiring a large amount of circuitry and space. Piezoelectric sensors are also a more expensive option. [35]

Slide Potentiometers

One method that could be used to detect hand position is using a slide potentiometer. Potentiometers are simple to use. The voltage output simply changes with the displacement of the wiper. The response of potentiometers is not exactly linear [36], however for these purposes, it is close enough to linear that it wouldn't affect the position reading.

Differential Pressure Sensor

For detecting the position of the hand in a pneumatically actuated system, we would use digital pressure sensors. These sensors are piezoelectric sensors used in series with the pneumatic cylinder. [37] The pressure in the cylinder can be used to determine the displacement of the cylinder using the equations below relating pressure to cylinder volume. (Equations 1, 2) In order to use pneumatics, we would also require electronic solenoids, which can be expensive to purchase.

$$P_1 * V_1 = P_2 * V_2 \quad (1)$$

$$V = L * \pi * r^2 \quad (2)$$

Where

L = length of the cylinder

r = radius of the piston base

Visual Processing

Visual processing pertaining to the human body is called skeletal tracking. [38] The most common method is using arrays of infrared cameras to determine joint angles and the position in space. One such system is the *Leap Motion*. The *Leap Motion* uses two IR cameras to measure the joint angles specifically in the hands.

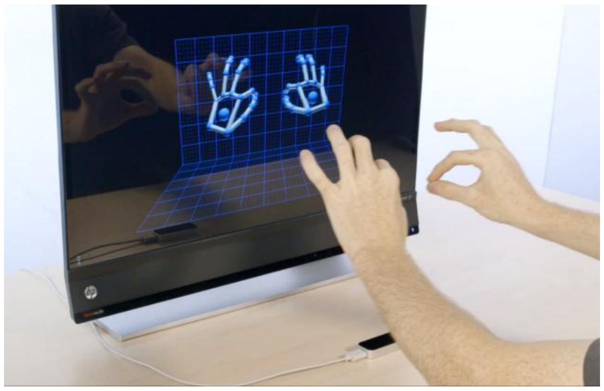


Figure 15. The Leap Motion sensor being used for skeletal tracking [39]

4.3 Preliminary PC software designs

A major goal of this project is to determine methods for providing the user with engaging and useful feedback regarding their performance in rehabilitation activities. There are many ways to accomplish this. This project focuses on industry standard software with adaptable programming interfaces (API), for ease of development. We explored many options for software solutions for both the visual and audio elements of the project.

4.3.1 *Leap Motion* and Gesture Recognition

The choice to use the *Leap Motion* was supported by the compatibility with other useful software as well as the range of functions the skeletal tracking provides. The *Leap Motion* Software Development Kit (SDK) is compatible with both *Unity 3D* and *Unreal* game engines.

The *Leap Motion* has many pre-programmed gestures for use in the SDK. It is also possible to program custom gestures. The four main gestures *Leap Motion* comes with are Swipe, Circle, Key Tap, and Screen Tap. (Figure 16) The Swipe gesture is when the *Leap Motion* detects that a vertical open hand has moved horizontally from one side of its vision to the other, for example, moving your hand from the left to the right of the screen. This gesture can be used for functions such as switching menus, like swiping your finger across a smartphone screen. Another pre-programmed gesture is the Circle Gesture, which is when a finger draws a circle in the air. The next gesture is the Key Tap gesture, which is when a finger moves downward from a horizontal position as if it was pressing a key of a piano or keyboard. The final pre-programmed gesture is the Screen Tap gesture, which is where a finger retracts and extends as if pressing into the screen of the computer. For our purposes, we also need another gesture, the pinch. This is a gesture that detects when a finger and the thumb are pinching and determines which finger is pinching.

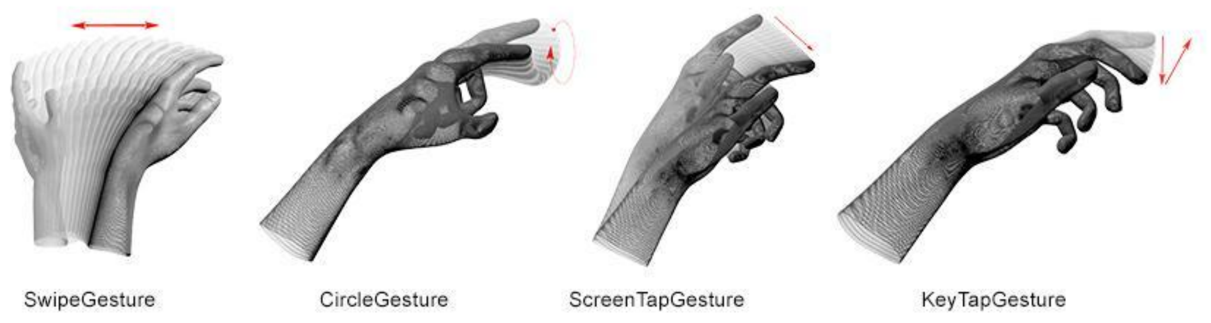


Figure 16. Leap Motion pre-programmed gestures [40]

In order to assist with rehabilitation, we need to be able to conduct rehabilitation exercises in our program. Using gesture recognition, one exercise stroke victims often use is pinching pennies and stacking them one on top of another. In order to be able to simulate this exercises, we need to be able to recognize the user is pinching as well as the strength with which they are pinching. However, we can also combine the gestures, to create more complex exercises if we need to. One example of this is having one hand pinching to add a bass line of a song and have one hand have a finger doing the Key Tap gesture to play percussive sounds. Using the *Leap Motion* gesture recognition, these exercises can become more engaging for the user. Also outside of stroke rehabilitation, these can be used to create different games and could help add a realistic part to many different games, needing the use of one's hands as the controller.

The 4 main gesture come with *Leap Motion* are easily used. One must first enable the gestures in the program using the *Leap Motion* SDK. The next step is to create a listener specific to each gesture. Here it is easy to add filtering. Many of the gestures performance can overlap, so it is wise to only listen for gestures currently being used.

However, we also needed one of our own gestures. In order to do this, we created a pinch gesture. This gesture can be used further for rehabilitation because the motor function of pinching can be quite difficult for some people after a stroke. *Leap motion* already has a pinch strength method, which returns a number between 0 and 1. This tells how close the fingers are together. We further improved the pinch functionality by being able to determine which finger is pinching. The pinch strength method only indicates if a finger is pinching, but not which finger is performing the action. In order to determine which finger is pinching, we created a listener for the frame *the Leap Motion* sees so that when a hand becomes visible, the code will enact those actions. We next check the first hand the *Leap Motion* sees, and see if this hand is currently pinching. If this hand is pinching, it determines if it is the left or right hand, and then it iterates over the fingers to see which finger is pinching. This is done by setting a minimum distance, which is approximately 50 mm. This can easily be configured for each person who uses this program. Next, we compare the distance the finger index finger is from the thumb. If this distance is shorter than 50 mm, this becomes the next minimum distance. We repeat this procedure with each finger on the hand, to find which finger or fingers are the closest. When it is determined which fingers are pinching, we then return an object of type, FingerType, which returns the id of the finger, the name of the finger as a string, such as "index." This is then used as a parameter for a second data type, called HandandPinch, which returns the FingerType, along with if the hand is a left hand or a right hand. This procedure repeats with the other hand, if it is there, to determine which finger(s) is pinching.

4.3.2 Audio software

In order to create an engaging experience for the user of the device, a number of different kinds of software need to be used. We examined many options for providing audio feedback to the user. This includes game engines, visualizers, and MIDI controllers. The first decision we made was regarding the MIDI controller for the audio output.

We chose *Cycling 74 Max 7* to create the musical elements. *Max 7* is a visual programming language used for multimedia applications. *Max 7* also has support for *Ableton Live 9*, a MIDI application, through the *Max for Live* package. It can create and manipulate MIDI effects in the *Live* environment. *Max 7* is also compatible with *Leap Motion* and has support for serial communication with the *Arduino*. This is useful because we were able to use *Max 7* to do more intensive data processing, such as smoothing, external to the *Arduino*.

In order to incorporate skeletal tracking data from the *Leap Motion* into *Max 7*, we used an external called *Leap Modulation*. It imports the *Leap Motion* executable files and formats the data in a way that we could parse it and then performs real-time data processing. The external library provides 4 sets of data; frame data, hands data, finger data and gesture data. We used a combination of skeletal tracking data and the built-in *Leap Motion* gesture recognition to map musical effects to the user's hand movements.

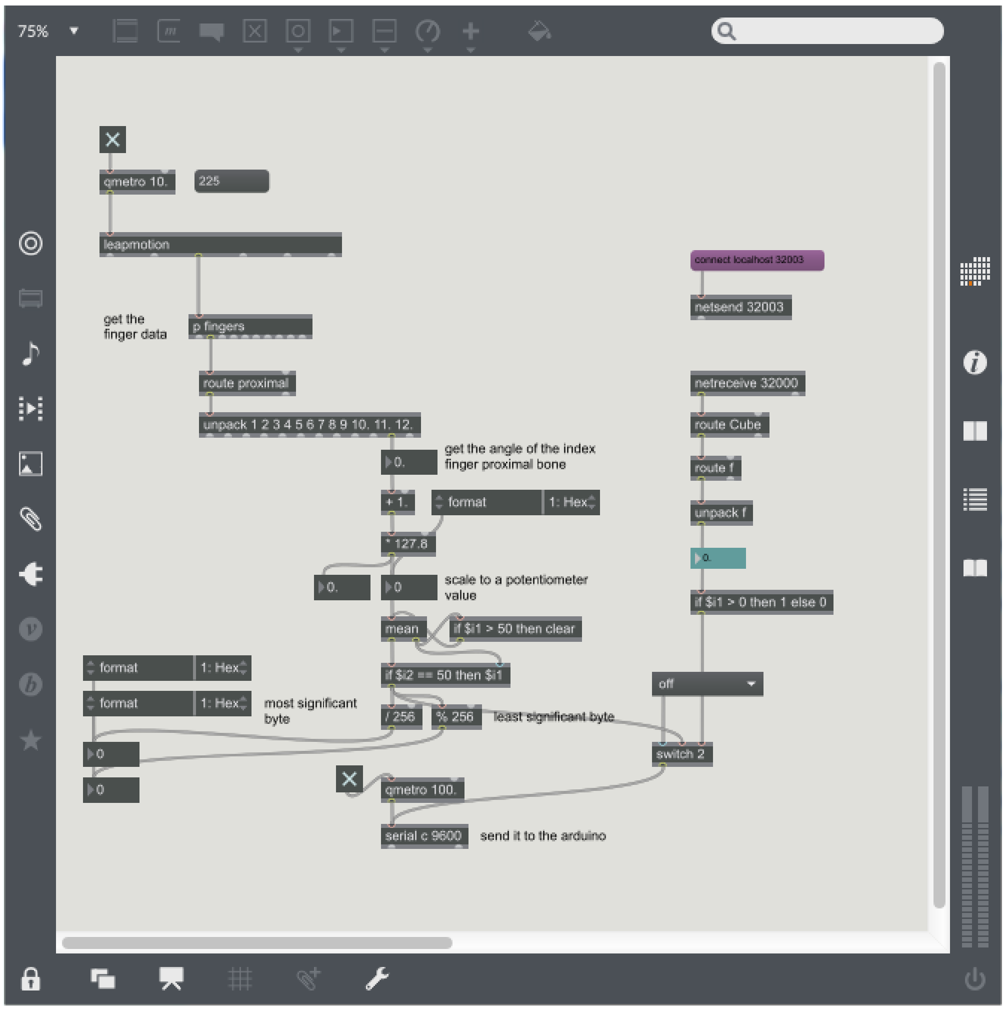


Figure 17. Example Max 7 Patcher. Processes Leap Motion skeletal tracking data and sends a control value to the Arduino

4.3.3 Visual elements

There were two primary choices for creating visual effects for our system. The first option was to design our own graphics in either of the supported game engines, *Unreal* or *Unity 3D*. This would allow us to have complete customization of our game as well as give us access to the embedded physics engines for use with the control of the actuators.

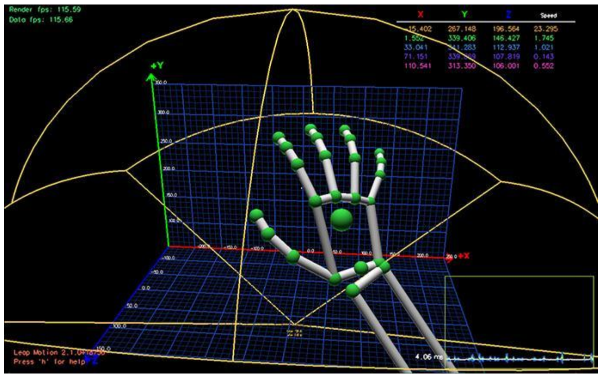


Figure 18. Leap Motion visualizer [40]

The second option was to use a third party visualizer that was made specifically for the *Leap Motion*. Many such apps have been created for musicians to create music using the *Leap Motion* skeletal tracking and provide many options for visual effects that correlate to user movements.

We decided on using the *Unity 3D* engine. This is because while the *Unreal* engine and *the Unity 3D* engine both support the *Leap Motion*, the *Unity 3D* engine has a patcher that already supports the integration with *Max 7*. This integration allows us to easily send the data from the *Unity 3D* engine to *Max 7* to be used to control the motors of our system. We also decided not to use a third party visualizer for the *Leap Motion* because of the difficulty of using the data from the third party software. While the integration with *Leap Motion* is better than the integration with of *Leap Motion* and *Unity*, the ease of use of *Unity* made it a better choice in light of the fact that we did not have sufficient time to develop a custom visualizer.

4.3.4 Conceptual Designs for Demonstrations of Gamification

We had three initial concepts for the design of the visual and audio feedback. The goal of this part of the project was to come up with a way to best demonstrate the potential of the system for use in stroke therapy as well as the capabilities of the system in expanded media applications.

The first conceptual design considered was modeling the game after popular music and rhythm video game such as *Guitar Hero*. [41] The musical element would be based on existing instruments, i.e. a piano or a violin. The user would then perform songs or exercises while wearing our system, and playing on a physical instrument. There would be a visualizer similar to that of *Guitar Hero* games in that the user will be prompted to play a note with the correct timing on scrolling platform. The user would then be provided with statistics on their performance and their progress over time.

Figure 19. Guitaer Hero III Legends of Rock gameplay [41]

The second concept was similar to the first, only instead of being modeled after existing instruments, the user would use a series of gestures to create musical effects or songs. For this concept, a music staff would be displayed and the user would use pinching and gripping gestures to manipulate the notes on the staff, similar to music composition software such as *Sibelius*. [42] This option would demonstrate the gripping and pinching exercises most commonly used in stroke therapy while providing an interactive music learning experience.

The third option was to create an amorphous music creating software that used gesture recognition and skeletal tracking to create music and sound effects. The user will only see basic visualizer effects, changing colors and a model of their hands. Having a simplified interface decreases the amount of time spent making graphical elements while still demonstrating the capabilities of the system.

Chapter 5: Design Verification

5.1 Final design selection

The complete assembly can be divided into two parts for consideration, the actuators, and the hand. The actuators are each driven by an encoded DC motor attached to a screw drive that moves the aluminum body. The body has two main compartments, one is empty space to allow the screw drive to move freely, and the other houses the spring assembly. (Fig 21) The spring assembly is a pair of springs with the base of a piston placed in between the two and spaced such that the springs are always acting on the piston. The shaft of the piston is fed through a spring and exits the aluminum body opposite the screw drive; the piston drives the Bowden cable and the potentiometer. This creates an elastic element for the system. The body has a 3D printed sheath, which holds three linear bearings, and is attached to rails, to allow for low friction translation while preventing any other movement.

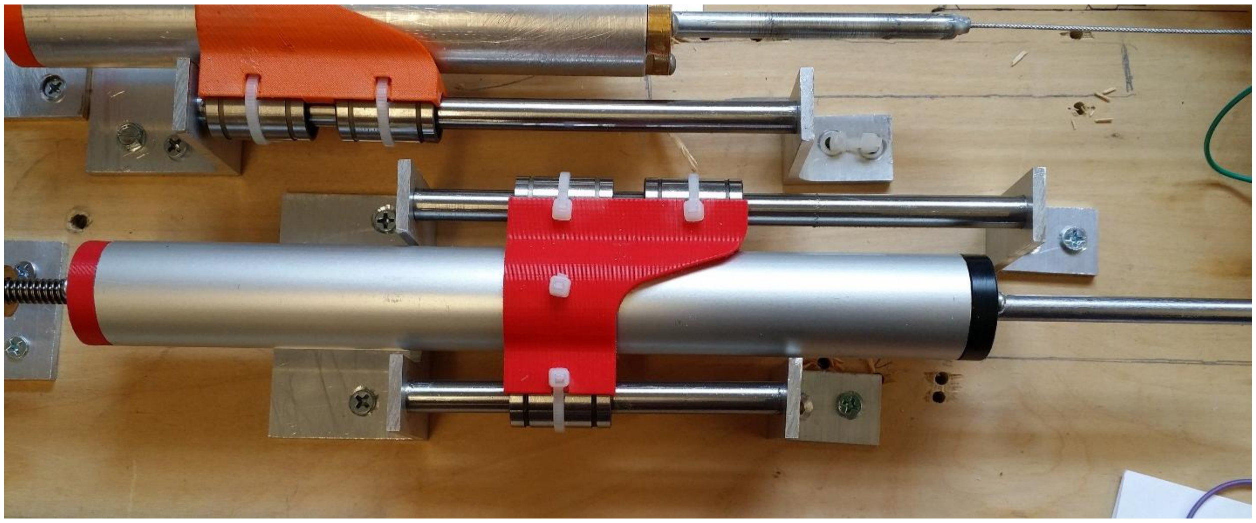


Figure 20. Final actuator design

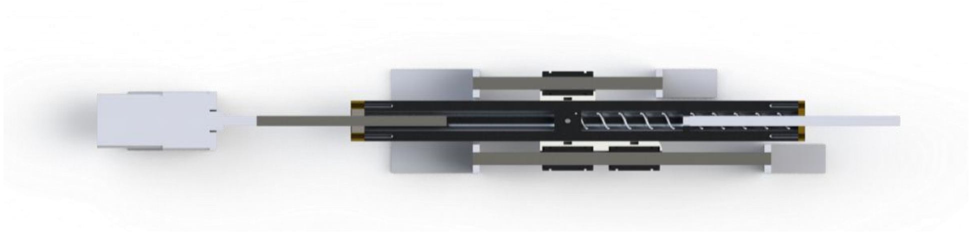


Figure 21. Cross Section of the force transducer. The system is divided into two parts. The first section (left-most) couples the transducer to the motor via threaded rod. The second section (right-most) houses two compression springs around a piston. The piston connects to the Bowden cable.

We decided that to control our motor, a proportional–integral–derivative controller (PID) would be the best choice. A PID is a controller that continually calculates the error between the value of the sensor, and the target value. It then uses the error, the total sum of all errors so far, and the difference between the current error and the previous error in order to determine the speed at which to drive the motor to the set point. We chose a PID because it is a proven and simple way to control the type of actuator we implemented. Using a PID controller allows us to easily control the angle of the finger joint connected to the actuator, which we can do efficiently and safely.

In order to get the data from the potentiometer for the PID, we needed to use SPI with an MC3008 ADC. Using this, we are able to process the data from the potentiometer to the mega, and run our PID from there.

The Leap Motion was chosen as the primary hand position sensor. This system was a good choice for this project because it generates 200 frames per second of hand position data that is accurate within 0.7 mm [43] and has a software development kit (SDK) making data collection and processing simple. We decided to use the *Leap Motion* in conjunction with physical sensors to control the system. The final circuit includes a 10 kΩ potentiometer in series with each Bowden cable and a quadrature encoder on each motor. The sensors were chosen in order to determine the input position of the motor and the output position of the Bowden cable to control the force output of the actuator rather than the displacement. The force being applied to the piston was calculated using the following equations, where k is the spring constant.

$$EncPosition = encCount/\left(\left(\frac{Threads}{inch}\right) * \left(\frac{ticks}{revolution}\right)\right) \tag{3}$$

$$Force = k * (EncPosition - PotPosition) \tag{4}$$

Where

k = 2172.5 kg/s^2

threads/inch = 20

ticks/rev = 4480

For prototyping purposes, the *Arduino* Mega 2560 was selected as the microcontroller. Additional components that were used were the MCP3008 ADC to convert the analog signal from the slide potentiometers and a dual channel motor controller. (See Appendix B for list of parts) Communication between the microcontroller and the ADC will cause only a small latency of 0.11 ms per degree of freedom, which the *Arduino Mega 2560* is fast enough to handle and meets the specification of having an overall latency of less than 100 ms for the number of signals being sent over the serial port. The circuit for one degree of actuation is depicted below. (see Appendix D for complete *Arduino* code)

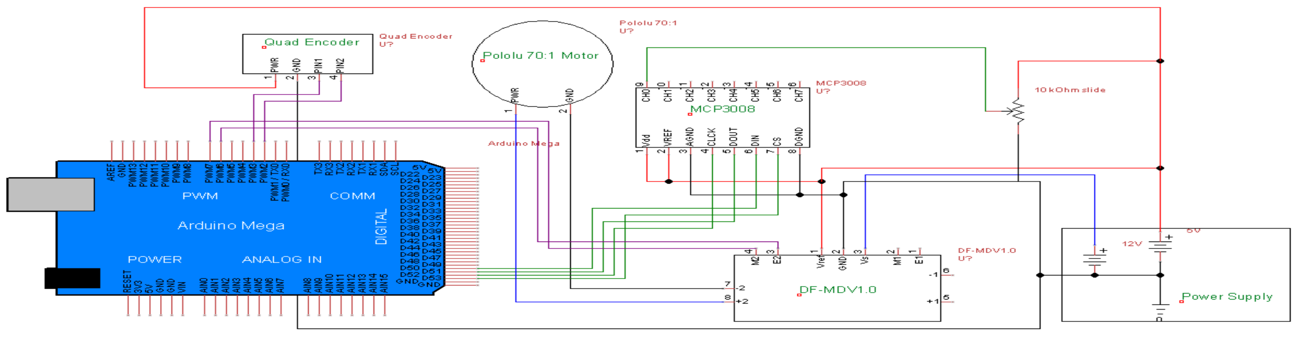


Figure 22. Circuit schematic for 1 DoF

5.2 Software

There were five main pieces of software used for this project. The communication between these pieces of software is depicted in Figure 23 below.

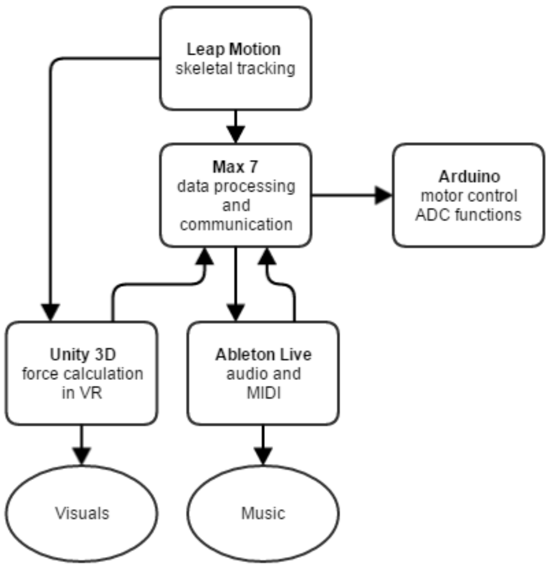


Figure 23. Software architecture diagram

Cycling '74's Max 7 was used as the central data processing point. Skeletal tracking data from the *Leap Motion* SDK was sent to *Max 7*. The position of the joints was then used to calculate the desired set-point for the actuator and this data was sent using serial communication to the *Arduino*. The skeletal tracking data was also sent to the *Unity 3D* engine. This tracking data was used to create 3D models of the hand bones joints in the game environment.

Unity 3D

We decided on using the *Unity 3D* engine for the custom visuals used by our system. We created a "game" in which there were objects of varying sizes. (Figure 24) In this case they were blocks. Using *Unity*'s physics engine, we made the blocks be constructed out of various materials (steel, wood, etc). These different materials have different properties and react differently when interacting with other objects in the game environment.

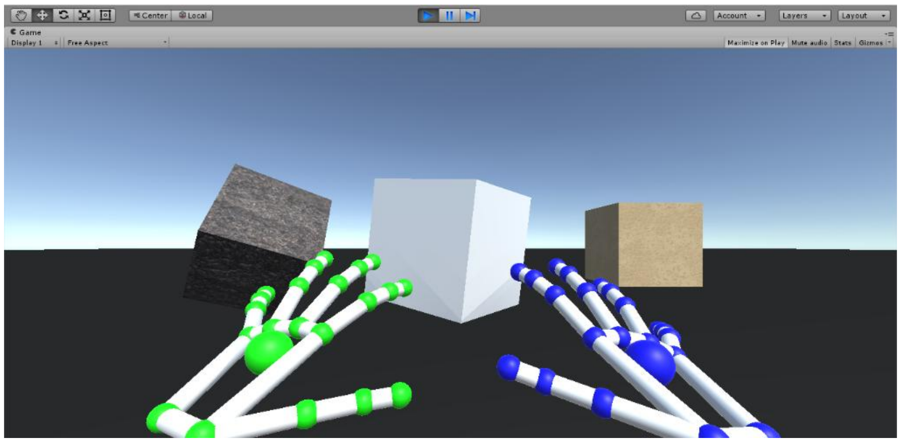


Figure 24. Unity 3D game with Leap Motion hand models. The cubes are made of different materials giving them different properties in the physics engine.

We used this function of the physics engine to determine the forces being exerted on the *Leap Motion* hand model. Each bone of the *Leap Motion* hand model had a collider assigned to it. A collider is the component of the game object that triggers events upon interacting with another object in the environment. We wrote the OnCollision that uses the C# scripts JitMessenger, JitSend and JitReceive to send the impulse created by the collision to *Max 7*. In *Max 7*, the impulse was used to calculate the force exerted on the hand model by the game object using the following equations:

$$I = m * V \tag{5}$$

$$F = m * a \tag{6}$$

$$F = \int I \, dt \tag{7}$$

Where

I = impulse

m = mass

V = velocity

a = acceleration

This force varied depending on the material that the object was made of. This data was then sent over serial communication to the *Arduino* and used to set the neutral force position of the actuator. Although the algorithm to calculate the gripping force exerted by the hand, falls outside the scope of this project, we were able to demonstrate that the forces calculated in the physics engine, could be used to emulate interacting with objects in a game engine.

Ableton Live 9 and *Max for Live*

For the design of our interactive music element, we decided to use gestures and hand position to create amorphous musical effects. To do this we used *Ableton Live 9* alongside a *Max for Live* instrument that accesses the *Leap Motion* skeletal tracking data to add effects to MIDI instruments and audio clips. We used two of the *Leap Motion* pre-programmed gestures (swipe and key tap) as well as our custom pinch gesture.

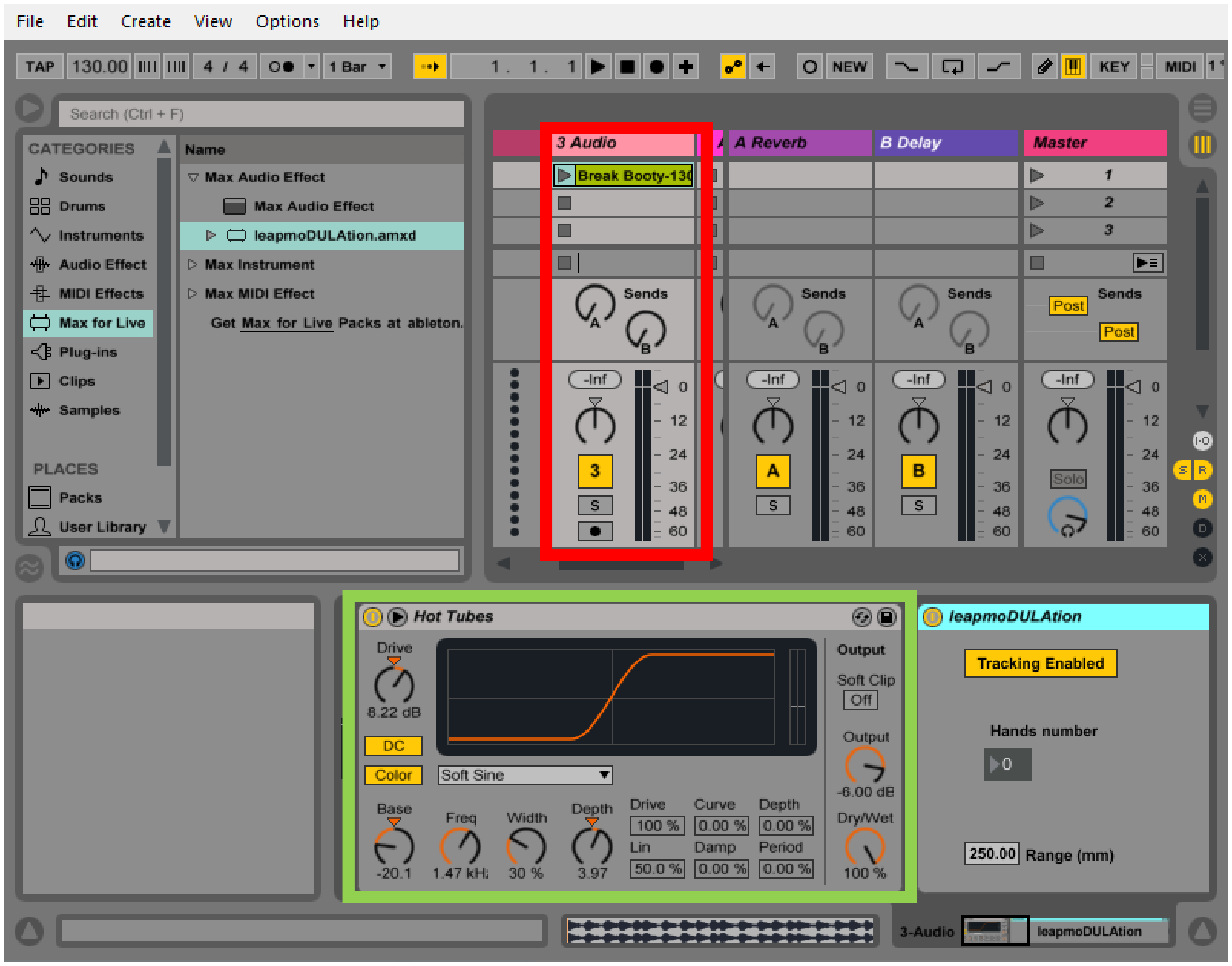


Figure 25. Ableton Live 9 scene. Highlighted in red is an audio clip. Highlighted in green is a saturator audio effect

We created a "scene" in *Live* that had a variety of audio and MIDI clips. (Fig 25 The swipe gesture was used to cycle through different audio clips. The key tap gesture would start and stop the clips depending on which finger was used.

Figure 26. Max for Live Leap Motion instrument

We also applied different effects to the clips. Some examples of the effect we were able to use filters, phasers, loopers, reverb, saturators, etc. (Fig 26) The parameters of these effects include the gain, frequency, hundreds of others. Instead of using the binary gestures to control these parameters, we used the continuous values of palm position, hand roll, and grip strength. The combination of these effects allowed the user to create unique songs in real-time.

5.2 Experiments and Results

5.2.1 System Speed

The speed of the system was measured by using stopwatch to measure how long it took to move the actuator from the minimum position to the maximum position. The load-less speed of the output when the Bowden cable was removed was 0.21 in/s, running at 252 RPM. When the Bowden cable was attached, its speed was reduced to 0.17 in/s, running at 204 RPM. This data was measured by timing the speed of the mechanism to traverse the 3.5-inch workable length at maximum power.

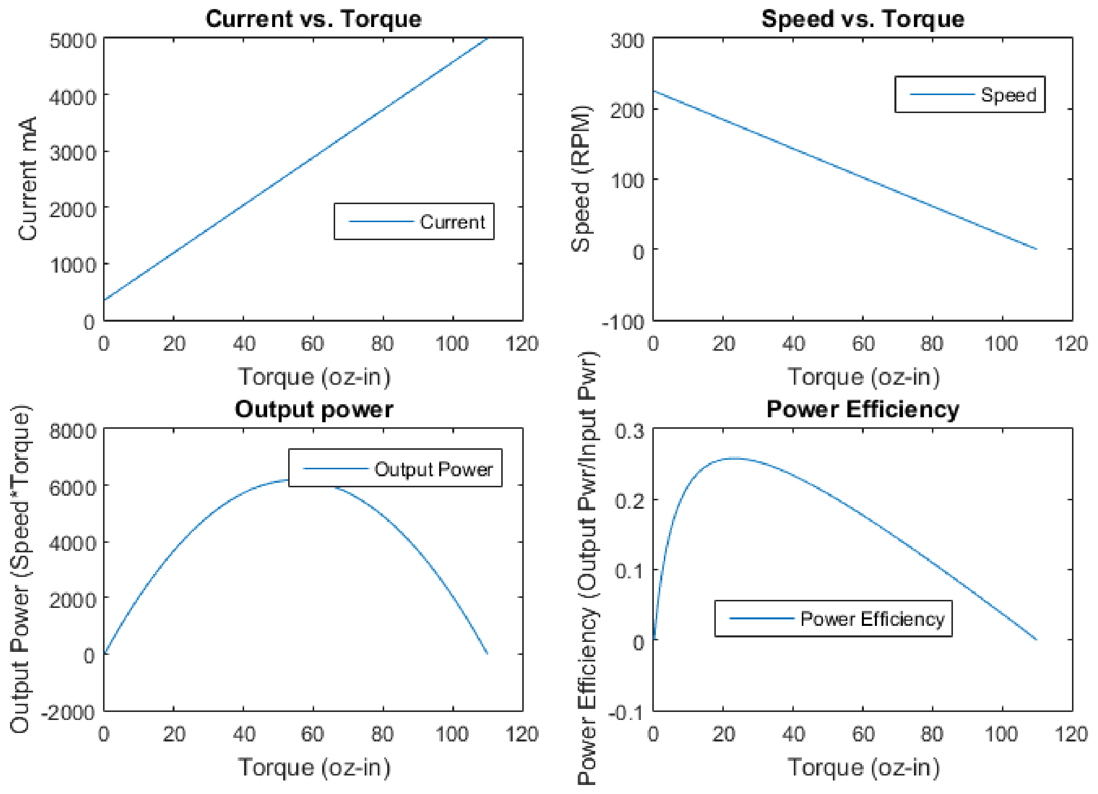


Figure 27. Torque-Speed curves of the actuator

Figure 27 describes the behavior of the motor with the resistance of the mechanism taken into account. Running at 12V, the motor runs with maximum power at 100RPM, or 0.083in/s at the actuator, and maximum efficiency at about 20 RPM, or 0.017in/s.

5.2.2 Output Force

The output force capability of the actuator was measured by setting the set-point of the PID controller to the position at which the demonstration model was flexed 45 degrees. We then added small measured weights onto the demonstration model until the motor stalled. The total weight at the end effector was added to the weight of the model to determine the amount of force being exerted by the actuator. The system was able to lift 1.6 lbs of force at the end effector. Since the weight was being lifted at a 45 degree angle (see Equation 8), the actual output force was 1.13 lbs. This value was potentially reduced by the bowing of the Bowden cable.

$$Force = \cos(45^\circ) * weight \tag{8}$$

5.2.3 Latency

The total software latency from getting the skeletal tracking data from the *Leap Motion*, calculating the ADC set-point value and using the PID controller to set the motor control value is 31.4 ms. We calculated this by setting up a timer in the *Arduino* control loop that starts as soon as a byte is received over serial communication from *Max 7*. This timer incremented a counter once every 100 microseconds; when this counter was printed out after computing the data from *Max 7*, the timer incremented 4 times meaning the *Arduino* latency was about 0.4 ms. We know the latency between *Max 7* and the *Arduino* is about 1 ms, because of the metronome we set that sends data once every 1 ms. The latency of the *Leap Motion* itself is 30 ms, so the total latency of our system is approximately 31.4 ms.

Chapter 6: Discussion

6.1 Evaluating Design Goals

Safety

The highest priority characteristic of this project was that it had to be safe for anyone involved in its use. The actuator has mechanical stops in its design to prevent the system from moving to a position or applying a force that falls outside of the normal operating range of a person's hands. In addition, all of the materials used that come in contact with the user during use are biocompatible.

This actuator falls under the Class II device classification for the FDA. This requires the device to have proper bench testing and emergency stops to be approved. [44] By completing these steps, our system would be usable in medical environments.

Weight

The second goal was for the weight of the system that is to be supported by the patient to be no more than 5 lbs. We achieved this objective, coming in at 4.8 lbs. At this weight, the system is far from ideal for use by stroke patients, but it could be reduced significantly by optimizing the materials used. The plastics used in the cable mounts and conduits are nearly as light as possible, but the cables are made to handle forces much greater than what our system will exert, and so reducing the weight of the cables by using materials that still meet our specifications but weigh less would reduce or solve the issue to make the system usable for physical therapy purposes.

Cost

The third constraint of our project was the cost of the system. As shown in Figure 28 the total price for a single actuator is $105. To actuate all degrees of freedom in the hand would require 20 actuators. Figure 29 shows that with 20 actuators, as well as the control system. The total cost to fabricate the system is $2439. This is significantly lower than the price of a single *CyberGrasp* (> $10,000), even if adding the price of manufacturing, which is estimated to be about $5000.

Item	**Cost**
Bowden Cables	$11.00
Slide potentiometer	$5.00
3D printed parts	$20.00
Stock material	$19.00
Motor	$50.00
Total for one actuator	**$105.00**

Figure 28. Cost to fabricate a single actuator

Item	**Cost**
Leap Motion	$80.00
Power Supply	$30.00
Microprocessor	$50.00
ADC x 3	$9.00
Motor controller x 10	$170.00
Actuator cost x 20	$2100.00
Total for Hand	**$2439.00**

Figure 29. Cost of the system for the entire hand

By reducing the cost of the system, it has the potential to be more accessible to patients. This is significant because a hospital or clinic would be able to purchase many systems and rent them out to patients, instead of requiring use of the system to be in-house. This will simultaneously allow more patients to access the system and allow physical therapists to assist more patients at the same time. This could also allow patients with limited mobility due to location, lack of transportation, or poor financial status to have to travel less to receive care.

System Speed and Output Force

The speed and force the system can output are a function of power of the motor, and so one can't be improved without sacrificing the other with the same motor. Instead one should consider the power of the machine. The overall power was too low, but the issue is resolved by replacing the motor with one more powerful. The exception to this being increasing the efficiency, which will recover force lost in the system. Overall, the speed was significantly slower than our desired speed of 3.5 in/sec, but the speed it ran at was capable of demonstrating the properties of the system to an acceptable degree. Additionally the force it could apply was below what we desired, but similarly was capable of demonstrating the properties we desired.

Latency

The overall latency of the system is about 31.4 ms. Our initial goal was a latency less than 100 ms, to test the system and confirm the concept is valid. The latency of our system falls within our initial goal, but can be easily improved by optimizing the communication between all software components. Since the *Leap Motion* itself has a 30 ms latency, it takes up a significant allotment.

Having a low latency is a significant in addressing the compliance issue. In order to prevent simulation sickness, the latency should be below 48 ms. [24] If the system causes simulation sickness, it would discourage patients from using the system, therefore preventing the system from being a useful physical therapy tool. By not adding significantly to latency of the *Leap Motion*, we've reduced the potential of our system to cause simulation sickness to patients using it.

Precision Sensing and Providing Useful Information to the User

In order to provide data regarding the patient's performance as well as actuate with precision, we needed to be able to precisely sense the user's hand and finger positions. In order to sense where the hand is in space, we use the *Leap Motion* which provides high resolution skeletal tracking of the user's hands. We are able to accurately depict where a user's hand would be in space, and interact with virtual objects using this data. To determine the user's relative finger position, we used the potentiometer readings and the encoder readings. Using the difference between the position of the piston determined by the potentiometer, and the position of the overall system determined by the encoder, we are able to calculate the force being applied by the actuator.

This data can be used by the physical therapist to track the patient's progress as well as create more personalized therapy programs. The system is also able to provide how much force is being exerted by the user, which is helpful in determining the rate at which a patient is improving. Additionally, this data can be presented in a way that depicts progress to the patient, which can be helpful tool in addressing patient compliance.

Adaptability for Different Exercises

In order for the system to be useful in the physical therapy industry, the system should be able to facilitate a variety of exercises. With many previous solutions, one kind of exercise or task was able to be accomplished during their use. For example, the *InMotionArm*, which consists of a joystick apparatus, can only assist in activities related to gripping. It has a limited scope of functions that the user can participate in.

We were able to create two functional modes. The first is designed to emulate assistive therapy. A functional hand is used with the *Leap Motion* skeletal tracking to control the position of the actuator. In this mode, the user can use hand position and gestures such as swiping or tapping to create musical effects in *Ableton Live 9*. The visual effects are provided simply by the *Leap Motion* visualizer. The second mode is designed to model strength building exercises. The *Leap Motion* is used with hand completing the exercises. The hand interacts with objects in the *Unity 3D* environment, and the forces between the hand model and the virtual objects are calculated using *Unity*'s physics engine. This force data is processed in *Max 7* and used in the control loop to apply resistive forces to the user's hand, based on the interaction with the virtual objects.

These two modes demonstrate the potential for using gamification software to create a wide range of physical therapy related activities. Although our demonstration was limited to a 2 DoF model, the software integration provides proof of concept for using existing gamification software to develop a variety of engaging activities for stroke patients, instead of only being used for training in one function.

Integration with existing software

By means described in Chapter 5, this project was able to successfully demonstrate communication between multiple industry standard software packages used for game related applications for the purpose of controlling a robotic rehabilitation system. This illustrates the potential for future developers to ship third party packages that integrate these larger applications for the specific purpose of use with robotic rehabilitation systems.

One size fits most

The design of the mounting points for the hand allows the system to work for anyone whose hands fit roughly into the attachment points. In order to retain the advantages the mechanism has for safety, though, the actuators would need to be calibrated. For instance, if the lengths of one patient's fingers is shorter than the other, then the actuators must be limited so that they cannot move past maximum flexion and extension points. Aside from software, physical fail-safes are built in, so if the maximum range is greater than necessary, then an adjustable mechanical stop can prevent the machine from moving outside the safe range, preventing the user from being hurt by the machine. Additionally, the method of attaching the machine to the fingers of the user is to use Velcro straps, which basically fits any size finger, and in more extreme cases, attachment points could be swapped out to be of a larger or smaller radius for more comfort.

6.2 Challenges in the Project

1. Cables are made of steel and are therefore too heavy

The weight of the mechanism placed on the hand was nearly too high to be considered a success, because the majority of the weight is within steel Bowden cables. The cables we initially used were much heavier, and were rated for much larger forces. The conduit, which is a set of layers of plastic, steel, and more plastic, is created to cope with far more force than we work within. If it were replaced with weaker plastic conduit, the weight could be reduced without sacrificing cost or reducing strength below acceptable levels.

2. Limited access to Suppliers

One challenge we faced while building our prototype is that we were not able to buy materials in bulk. Since we were only making two actuators, we did not need a lot of materials. This meant that we had to buy our supplies in small amounts, and at an inflated price. If someone is going to build many of our systems, they would be able to buy the material in bulk, and get a discount for buying bulk, thus making the system more affordable.

Another challenge we faced was that we did not have representation with big suppliers. While looking for a sufficient amount of Bowden cables, or looking for a specific supplier, we were unsuccessful when we reached out to these suppliers. This could be because we were just students, who were not affiliated with a company, so the companies did not want to sell to us because we would not be very profitable to them. If we were affiliated with a company they would have the connections with these suppliers, and be able to get better cables for better prices, and have better communication with the companies.

3. Budget Limitations

Since the MQP budget is substantially less money than a manufacturer would have, we had less money to test different materials. We did not have enough money to buy many different kinds of Bowden cable to test how they perform, and determine the advantages of each cable. We also did not have the budget to test pneumatics, because it was too cost prohibitive.

Another challenge with the budget was the ability to test different types of motors. We were able to use the motors that we had access to without having to spend a lot of money. However, if we had a larger budget, we could test different motors, in order to drive our system at the speed and the power that we need.

4. Motor power

The power the motor was capable of reaching was not sufficient to move the actuator at the speed with the force we needed. We knew the motor wasn't powerful enough, but we didn't take friction in the rest of the system into account. Simply using a higher power motor, 60-100W or so, would solve this problem.

5. Bowden cables, caused friction, bowing, bending

The cables used were not optimal for our purposes, as the conduits had large amounts of friction in them, which we weren't able to accurately measure, as it changes with the curvature of the line. The friction was reduced slightly with the use of lubricant. The stiffness also caused problems: if too stiff, the cable can't curve enough to bend around the joint it's attempting to move, permanently bending the cable and requiring a large amount of work to deform and increasing friction after it becomes deformed; additionally, if the cable is not stiff enough, the cable will bow, pushing it outside of the conduit and removing any force being applied, and potentially jamming the machine.

6. Hand machining causes friction

Machining the parts caused quality issues with the whole mechanism. Largely, this is because the parts were machined by hand, which added a layer of human error. The lower quality of the parts creates friction, and caused parts to stick or be misaligned. When mounting these parts, they became more misaligned. Further, the 3D printed parts were not submitted to be printed with overshoot taken into account, so the 3D printed parts were slightly thicker than they should have been. The lax tolerances increases the friction in the entire system, reducing efficiency.

7. Optimization

Most of the pieces of software we used are API's for independent developers. In addition to some of the technology being new, projects using this software in conjunction with each other is a relatively new endeavor, so there hasn't been optimization specifically for the applications explored in this project. In many cases, custom scripts had to be written to establish communication between API's, potentially using large amounts of computing power. A solution would be to create optimized scripts that operate the pieces of software from within the *Unity 3D* engine to save computing power.

There are also issues of robustness. For example, the *Leap Motion* can output large quantities of skeletal tracking data; however, if the user interweaves their fingers, the *Leap Motion* will not be able to reconcile which hand is where. This, among other issues of robustness in the software, led to slow or poor operation. If someone were to create a set of external packages that allow these pieces of software to be used together, they would have to be optimized for communication between different pieces of software, as well as between software and hardware.

8. Too many colliders for accurate force calculations

Each finger bone in the *Leap Motion* model has a collider and impulse forces associated with it; we are able to output the force between each collider and the object, as well as the point of contact. Using the force output from the hand colliders and the objects, we were able to adjust the neutral force of the system. While we were able to demonstrate that the forces between the *Leap Motion* hand models and objects in the *Unity 3D* engine could be calculated, the mechanics of the hand are incredibly complicated. Producing an algorithm that determines the contribution of each finger bone and joint to the gripping of an object and using that to determine the force control value for each actuator was too time consuming and complex to fall under the scope of this project.

6.3 Big Picture Impacts

Economic Impact

Physical therapy is expensive, at times enough to prevent a patient from receiving as much help as they need for a full recovery. Further, if a patient can't afford physical therapy for the frequency of their exercises, compliance drops off drastically, impacting quality of patient care. By both reducing the cost of robotic therapy solutions, and the cost of tracking a patient's exercises, the therapist would be able to treat the patient with similar efficacy, with fewer visits, reducing the overall cost of care.

Environmental Impact

Our system is made out of many reusable materials. One such example is the aluminum for the actuator, which can easily be recycled. The motors can also be repurposed if they becomes obsolete. The base of the frame is made out of wood, which is easily recycled.

However, the Bowden cables of our system can be improved heavily. The Bowden cables we are using can be easily deformed. If the cables need to be replaced often, this would be an environmental problem because of the need to constantly make new cables.

Besides the high turnover rate for Bowden cables, the system has a long life cycle. A lot of the system is quite robust, and will not be prone to damage. The aluminum cylinder is strong and would only fail after the rest of the system is destroyed. The rings for the fingers would also be long lasting, as the plastic is not only lightweight, but is also strong enough to be able to resist breaking under the greatest force the system can output.

Societal Impact

Our system aims to improve the availability of rehabilitation, through either making it more accessible for patients who can’t make it to the hospital, or the ability to get more rehabilitation. If a patient has trouble making it to the hospital, or needs more exposure to therapy tools and activities, our system will allow them to complete exercises from their home so that they will not need to return to the hospital for a longer time, or will allow them to use the system on their own time and have more time with therapy overall.

Our system will also allow therapists to help more patients because the therapists do not need to spend as much time hands on with each patient. The therapist can allow the system to help the patient with their therapy as they work with other patients, and will still be able to monitor other patients’ progress. Since our system could track a patient’s progress, the therapists would be able to leave the patient, and periodically check in to make sure that they are completing their exercises.

Ethical Concerns

The most prominent ethical concern surrounding this project is that the system should be distributed and accessed in a fair and just manner. It is possible, even with the reduced cost of our system compared to other solutions on the market that hospitals or physical therapy centers can overcharge for the use of the system, thus treating the patients as a means to make profits.

Manufacturability

Most parts used in this system lend themselves well to larger scale production, as many components are either easy to buy from a supplier, or could be injection molded plastic, rather than 3D printed. The machined parts could also be redesigned to be easier to machine en mass, but, as it is currently, assembly would still need to be manual. The design, however, was intended to be easy to create on a small scale, using simple dimensions and common sizes to reduce machining needs. Further, the design was intended to be assembled easily by hand, as the whole design is able to work effectively with relatively low quality machining and assembly.

Sustainability

Our system is sustainable, but can also be easily improved. Our system does not need much upkeep in order for it to keep working. None of the parts are going to be under enough stress that would cause the system to need to be replaced. However, the Bowden cables do require a fair amount of upkeep. The Bowden cables will need to be lubricated about once every couple weeks, in order to minimize the amount of friction that would be in the system. This can be annoying for the therapists, however, because "many therapists may stop using devices if set-up takes more than 5 minutes." [45]

Chapter 7: Conclusions and Future Work

The device and software developed as a result of this project satisfied two goals. The first was to create a system that could be used in rehabilitation for stroke patients that can improve accessibility by decreasing the cost and weight. The second was to use gamification related software to demonstrate the potential for addressing the compliance issue in stroke rehabilitation by integrating audio and visual effects with robotic systems.

The system costs less than $2,500 dollars per hand to fabricate and weighs less than 5 lbs. Although the speed and output force of the system are low compared to other previously developed solutions, these issues can be rectified by higher power motors, higher quality Bowden cables to reduce bowing and friction and automated machining methods.

The next step for this project would be to expand the 2 DOF model we created to the entire hand, including abduction, adduction, the thumb, and the wrist. Creating this model would require applying a mathematical model of hand mechanics to both determine the forces being applied in the virtual environment and actuate the hand accurately according to the hand's anatomical function

Another recommendation for future work would be to conduct a user study by creating rehabilitation related games with the integrated software. This study could be used to optimize communication between gamification related software as well as determine how best to use this software to address the compliance issue in stroke rehabilitation. The future of this project is aimed to develop useful methods for integration of gamification and control software to improve the quality of treatment for stroke patients.

References

1. American Heart Association, 'http://circ.ahajournals.org/content/125/1/e2.full', 2012. [Online]. Available: http://circ.ahajournals.org/content/125/1/e2.full. [Accessed: 15- Oct- 2015].
2. "CyberGrasp", *CyberGlove Systems LLC,* 2015. [Online]. Available: http://www.cyberglovesystems.com/cybergrasp. [Accessed: 28- Apr- 2016].
3. Interactive-motion.com, 'Interactive Motion Technologies | InMotion ARM™ Interactive Therapy System', 2015. [Online]. Available: http://interactive-motion.com/healthcarereform/upper-extremity-rehabilitiation/inmotion2-arm/. [Accessed: 15- Oct- 2015].
4. P. Lum, D. Reinkensmeyer, R. Mahoney, W. Rymer and C. Burgar, 'Robotic Devices for Movement Therapy After Stroke: Current Status and Challenges to Clinical Acceptance', *Topics in Stroke Rehabilitation*, vol. 8, no. 4, pp. 40-53, 2002.
5. N. Scherbakov and W. Doehner, 'Sarcopenia in stroke-facts and numbers on muscle loss accounting for disability after stroke', *Journal of Cachexia, Sarcopenia and Muscle*, vol. 2, no. 1, pp. 5-8, 2011.
6. C. English, H. McLennan, K. Thoirs, A. Coates and J. Bernhardt, 'Reviews: Loss of skeletal muscle mass after stroke: a systematic review', *International Journal of Stroke*, vol. 5, no. 5, pp. 395-402, 2010.
7. B. Dobkin, 'Strategies for stroke rehabilitation', *The Lancet Neurology*, vol. 3, no. 9, pp. 528-536, 2004.
8. The Rehabilitation Measures Database, 'Rehab Measures - Box and Block Test', 2012. [Online]. Available: http://www.rehabmeasures.org/Lists/RehabMeasures/DispForm.aspx?ID=917. [Accessed: 15- Oct- 2015].
9. The Rehabilitation Measures Database, 'Rehab Measures - Nine-Hole Peg Test', 2014. [Online]. Available: http://www.rehabmeasures.org/lists/rehabmeasures/dispform.aspx?id=925. [Accessed: 15- Oct- 2015].
10. E. Miendlarzewska and W. Trost, 'How musical training affects cognitive development: rhythm, reward and other modulating variables', *Front. Neurosci.*, vol. 7, 2014.
11. M. Sailer, J. Hense, H. Mandl and M. Klevers, 'Psychological Perspectives on Motivation through Gamification', *Interaction Design and Architecture(s)*, vol. 19, pp. 28-37, 2015.
12. B. Burke, 'Forbes Welcome', *Forbes.com*, 2014. [Online]. Available: http://www.forbes.com/sites/gartnergroup/2014/04/10/how-gamification-motivates-the-masses/. [Accessed: 15- Oct- 2015].

13. A. Duffy, "Jonathan Pitre inspires 16,000 at We Day Ottawa", *Ottawa Citizen*, 2015. [Online]. Available: http://ottawacitizen.com/news/local-news/16000-cheer-we-day-in-ottawa. [Accessed: 28- Apr- 2016].
14. V. Popescu, G. Burdea, M. Bouzit and V. Hentz, 'A virtual-reality-based telerehabilitation system with force feedback', *IEEE Transactions on Information Technology in Biomedicine*, vol. 4, no. 1, pp. 45-51, 2000.
15. J. Standley, 'A Meta-Analysis on the Effects of Music as Reinforcement for Education/Therapy Objectives', *Journal of Research in Music Education*, vol. 44, no. 2, p. 105, 1996.
16. N. Friedman, V. Chan, A. Reinkensmeyer, A. Beroukhim, G. Zambrano, M. Bachman and D. Reinkensmeyer, 'Retraining and assessing hand movement after stroke using the MusicGlove: comparison with conventional hand therapy and isometric grip training', *Journal of NeuroEngineering and Rehabilitation*, vol. 11, no. 1, p. 76, 2014.
17. O. Unluhisarcikli, B. Weinberg, M. Sivak, A. Mirelman, P. Bonato and C. Mavroidis, 'A Robotic Hand Rehabilitation System with Interactive Gaming Using Novel Electro-Rheological Fluid Based Actuators', *IEEE International Conference on Robotics and Automation*, 2010.
18. J. Iqbal and K. Baizid, "Stroke rehabilitation using exoskeleton-based robotic exercisers:", *Biomedical Research*, vol. 26, no. 1, pp. 197-201, 2016.
19. P. Polygerinos, Z. Wang, K. Galloway, R. Wood and C. Walsh, 'Soft robotic glove for combined assistance and at-home rehabilitation', *Robotics and Autonomous Systems*, vol. 73, pp. 135-143, 2015.
20. Mulas, M., Folgheraiter, M. and Gini, G., "An EMG controlled exoskeleton for hand rehabilitation," Proc. of the 9th International Conference on Rehabilitation Robotics, pp.371-374, 2005.
21. SaeboFlex, "SaeboFlex | Saebo", *Saebo*, 2016. [Online]. Available: http://www.saebo.com/saeboflex/. [Accessed: 28- Apr- 2016].
22. S. Ito, H. Kawasaki, Y. Ishigure, M. Natsume, T. Mouri and Y. Nishimoto, 'A design of fine motion assist equipment for disabled hand in robotic rehabilitation system', *Journal of the Franklin Institute*, vol. 348, no. 1, pp. 79-89, 2011.
23. Assh.org, 'Hand Anatomy', 2015. [Online]. Available: http://www.assh.org/handcare/hand-arm-anatomy. [Accessed: 16- Oct- 2015].
24. Draper, M.H., Viire, E.S., Furness, T.A., Gawron, V.J. (2001). Effects of image scale and system time delay on simulator sickness with head-coupled virtual environments. Human Factors, 43(1), 129-146.

25. G. ElKoura and K. Singh, 'Handrix: Animating the Human Hand', *Symposium on Computer Animation*, 2003.

26. G. Fazekas, M. Horvath, T. Troznai and A. Toth, 'Robot-mediated upper limb physiotherapy for patients with spastic hemiparesis: A preliminary study', *Acta Derm Venereol*, vol. 39, no. 7, pp. 580-582, 2007.

27. G. Burdea and P. Coiffet, *Virtual reality technology*. Hoboken, N.J: Wiley-Interscience, 2003.

28. American Physiological Society. "How A Stroke Affects Hand Function; Roadmap For Rehabilitation." ScienceDaily. ScienceDaily, 2009. <www.sciencedaily.com/releases/2009/06/090615093927.htm>

29. A. Klaiput and W. Kitisomprayoonkul, "Increased Pinch Strength in Acute and Subacute Stroke Patients After Simultaneous Median and Ulnar Sensory Stimulation", *Neurorehabilitation and Neural Repair*, vol. 23, no. 4, pp. 351-356, 2008.

30. A. Schweizer, O. Frank, P. E. Ochsner and H. A. C. Jacob, "Friction between human finger flexor tendons and pulleys at high loads," Journal of Biomechanics, vol. 36, no. 1, p. 63–71, 2003.

31. Delph II MA, Fischer SA, Gauthier PW, Martinez Luna CH, Clancy EA, Fischer GS, *A Soft Robotic Exomusculature Glove with Integrated sEMG Sensing for Hand Rehabilitation*, 13th International Conference on Rehabilitation Robotics (ICORR), Seattle, WA, June 2013.

32. P. Polygerinos, S. Lyne, Z. Wang, L. Nicolini, B. Mosadegh, G. Whitesides and C. Walsh, "Towards a Soft Pneumatic Glove", *IROS*, 2013.

33. Muhammad Zahak Jamal, *Signal Acquisition Using Surface EMG and Circuit Design Considerations for Robotic Prosthesis*. INTECH Open Access Publisher, 2012.

34. J. Dosch and B. Hynd, "Analysis of Electrical Noise in Piezoelectric Sensors", *PCB*, 2007. [Online]. Available: http://www.sem.org/PDF/Electrical_noise_in_piezoelectric_sensors.pdf. [Accessed: 28- Apr- 2016].

35. M. Dirjish, "What’s The Difference Between Piezoelectric And Piezoresistive Components?", *Electronicdesign.com*, 2016. [Online]. Available: http://electronicdesign.com/components/what-s-difference-between-piezoelectric-and-piezoresistive-components. [Accessed: 28- Apr- 2016].

36. R. Elliot, "Potentiometers (Beginners' Guide to Pots)", *Sound.westhost.com*, 2016. [Online]. Available: http://sound.westhost.com/pots.htm. [Accessed: 28- Apr- 2016].
37. K. Pressure Inc., "pressure sensors - how sensors work", *Sensorland.com*, 2016. [Online]. Available: http://www.sensorland.com/HowPage004.html. [Accessed: 28- Apr- 2016].
38. OptiTrack, "Skeleton Tracking - NaturalPoint Product Documentation", *Wiki.optitrack.com*, 2016. [Online]. Available: http://wiki.optitrack.com/index.php?title=Skeleton_Tracking. [Accessed: 27- Apr- 2016].
39. K. Yeung, "Leap Motion Unveils 3D Sculpting App, Previews New Tracking Software", *The Next Web*, 2013. [Online]. Available: http://thenextweb.com/insider/2013/11/20/leap-motion-unveils-freeform-3d-sculpting-app-previews-new-gesture-tracking-software/#gref. [Accessed: 28- Apr- 2016].
40. A. Davis, "Getting Started with the Leap Motion SDK", *Leap Motion Blog*, 2014. [Online]. Available: http://blog.leapmotion.com/getting-started-leap-motion-sdk/. [Accessed: 28- Apr- 2016].
41. *Guitar Hero III Legends of Rock*, Activision and RedOctane, Oct 28, 2007
42. "Sibelius | Avid", *Avid.com*. [Online]. Available: http://www.avid.com/sibelius. [Accessed: 28- Apr- 2016].
43. "Controller — Leap Motion JavaScript SDK v2.3 documentation", *Developer.leapmotion.com*. [Online]. Available: https://developer.leapmotion.com/documentation/javascript/api/Leap.Controller.html. [Accessed: 28- Apr- 2016].
44. "Product Classification", *Accessdata.fda.gov*, 2016. [Online]. Available: http://www.accessdata.fda.gov/scripts/cdrh/cfdocs/cfPCD/classification.cfm?ID=PHL. [Accessed: 28- Apr- 2016].
45. P. Maciejasz, J. Eschweiler, K. Gerlach-Hahn, A. Jansen-Troy and S. Leonhardt, "A survey on robotic devices for upper limb rehabilitation", *Journal of NeuroEngineering and Rehabilitation*, vol. 11, no. 1, p. 3, 2014.

Appendix A - Muscle Groups of the Hand

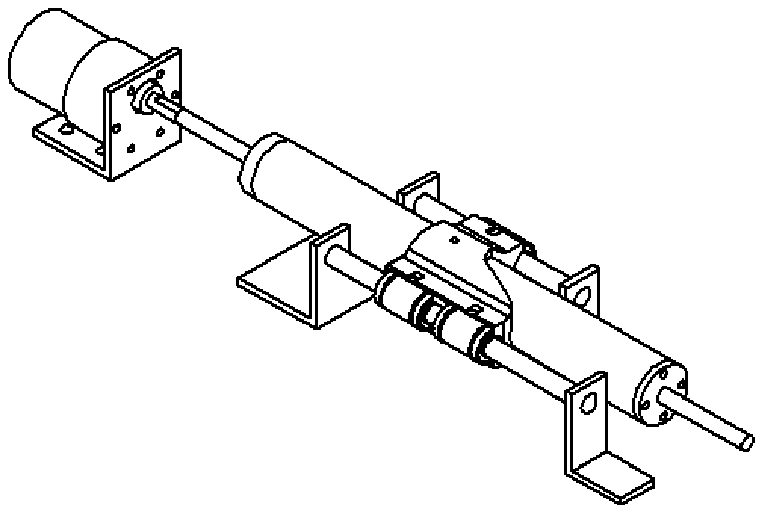
Part	Flex	Extend	Abductor/Adductor
little finger	flexor digitorum superficialis (tip) flexor digitorum profundus (middle) lumbricals (interphalangeal) hypothenar	extensor digiti minima lumbricals (interphalangeal) hypothenar	hypothenar interosseous
ring finger	flexor digitorum superficialis (tip) flexor digitorum profundus (middle) lumbricals (interphalangeal)	extensor digitorum communis lumbricals (interphalangeal)	interosseous
middle finger	flexor digitorum superficialis (tip) flexor digitorum profundus (middle) lumbricals (interphalangeal)	extensor digitorum communis lumbricals (interphalangeal)	interosseous
index finger	flexor digitorum superficialis (tip) flexor digitorum profundus (middle) lumbricals (interphalangeal)	extensor digitorum communis extensor indicus proprius lumbricals (interphalangeal)	interosseous
thumb	flexor pollicis longus thenar	extensor digitorum communis extensor pollicis longus extensor pollicis brevis thenar	abductor pollicis longus thenar interosseous
wrist	flexor carpi radialis flexor carpi ulnaris	extensor carpi radialis brevis extensor carpi radialis longus extensor carpi ulnaris	abductor pollicis longus extensor carpi ulnaris

Appendix B - Part List

Part Name	Part No.	Retailer	Retailer Headquarters	Store Webpage
Pololu 37D Gearmotor	2823	Pololu	920 Pilot Rd. Las Vegas, NV 89119 USA	https://www.pololu.com/product/2823
Right-Hand General Purpose Acme Hex Nut	94815A007	McMaster-Carr	200 New Canton Way Robbinsville, NJ 08691-2343	http://www.mcmaster.com/#catalog/122/3196/=1264a97
General Purpose Acme Fully Threaded Rod, ASTM A193 Grade B7 Steel, 1/4"-16 Acme Size	93410A904	McMaster-Carr	200 New Canton Way Robbinsville, NJ 08691-2343	http://www.mcmaster.com/#catalog/122/3194/=1264aut
Multipurpose O1 Tool Steel, Tight-Tolerance Rod, 8MM Diameter, 3' Long	88625K67	McMaster-Carr	200 New Canton Way Robbinsville, NJ 08691-2343	http://www.mcmaster.com/#catalog/122/3750/=1264b9t
MCP3008 - 8-Channel 10-Bit ADC With SPI Interface	856	Adafruit	150 VARICK ST NEW YORK, NY 10013	https://www.adafruit.com/product/856
3-248 - Heavy Duty Conduit-	3-248	MFG Supply	MFG Supply P.O. Box 208	http://www.mfgsupply.com/3-248.html

.203" OD Per 50'			Medford, WI 54451	
3-246 - Heavy Duty Inner Wire-.072" Per 100'	3-246	MFG Supply	MFG Supply P.O. Box 208 Medford, WI 54451	http://www.mfgsupply.com/3-246.html
POT SLIDE 10K OHM .5W 100MM	PTB0143-2010BPB103-ND	DIGI-KEY	DIGI-KEY 701 BROOKS AVE. SOUTH P.O. BOX 677 THIEF RIVER FALLS MN 56701-0677	http://www.digikey.com/product-search/en/potentiometers-variable-resistors/slide-potentiometers/262243?k=PTB0143-2010BPB103-ND&FV=fff40004%2Cfff80063&mnonly=0&newproducts=0&ColumnSort=0&page=1&quantity=0&ptm=0&fid=0&pageSize=25
DFRobot 4.8-46V, 2A Dual Motor Controller	RB-Dfr-19	Robot Shop	RobotShop Inc. 18005 rue Lapointe Building 305 Mirabel, QC, Canada J7J 0G2	http://www.robotshop.com/en/dfrobot-4-8-46v-2a-dual-motor-controller.html
Insignia™ - 400W ATX Power Supply	NS-PCW4050	Insignia	7601 Penn Avenue South Richfield, MN 55423-3645	http://www.insigniaproducts.com/products/computer-speakers-accessories/NS-PCW4050.html
Shimano Road Shift Cable and Housing Set (Black)	Y60098501	Shimano	Shimano American Corp. 1 Holland Irvine, CA 92618	http://www.amazon.com/Shimano-Shift-Cable-Housing-Black/dp/B004XUQ1UQ/

Appendix C - Parts Drawings



ITEM NO.	PART NUMBER	QTY.
1	Tubular Transducer	1
2	Tubular Cap	1
3	Hexagonal Cap	1
4	Tubular Sheath	1
5	LM8UU	3
6	Smooth Rod - Port	1
7	Smooth Rod - Starboard	1
8	Piston	1
9	acme nut	1
10	Acme Rod	1
11	pololu gear motor	1
12	motor bracket	1
13	wide bracket	1
14	small bracket	2
15	Part1^Tubular Transducer	2

UNLESS OTHERWISE SPECIFIED:
DIMENSIONS ARE IN INCHES
TOLERANCES:
FRACTIONAL ±
ANGULAR: MACH ± BEND ±
TWO PLACE DECIMAL ±
THREE PLACE DECIMAL ±
INTERPRET GEOMETRIC TOLERANCING PER:
MATERIAL
FINISH
DO NOT SCALE DRAWING

	NAME	DATE
DRAWN		
CHECKED		
ENG APPR.		
MFG APPR.		
Q.A.		
COMMENTS:		

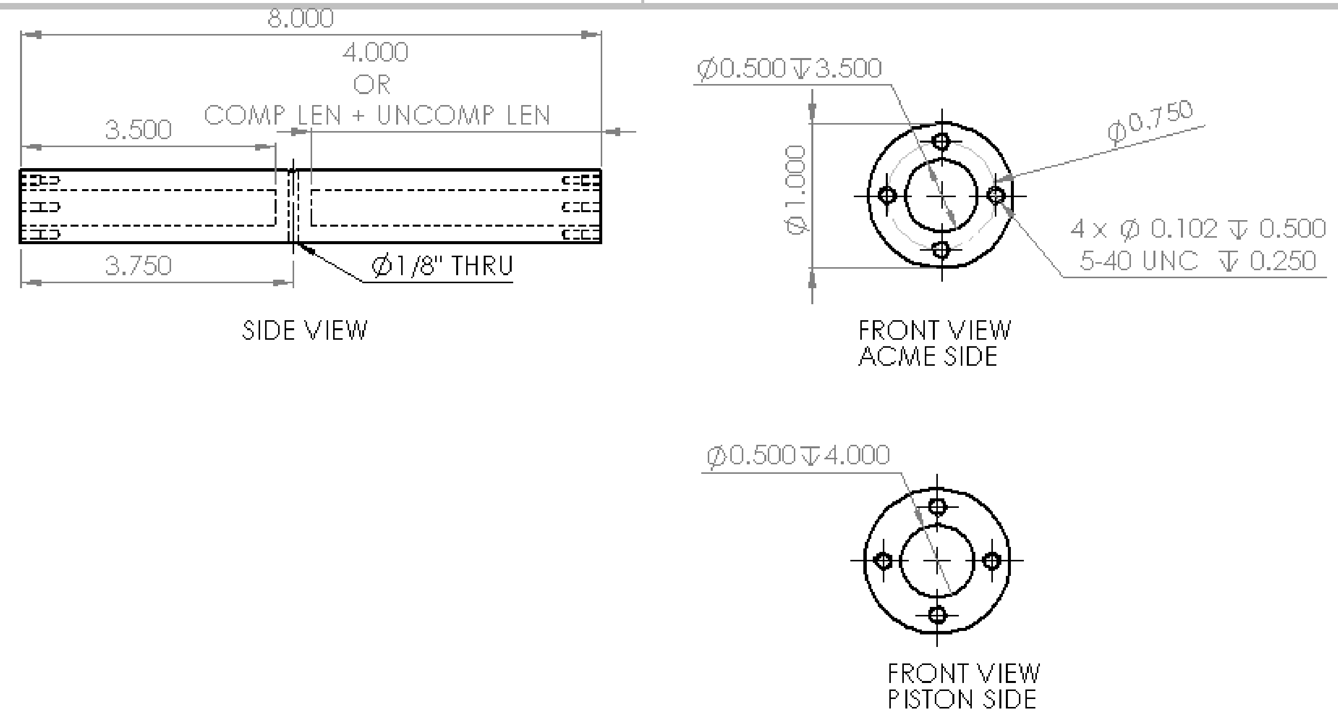
NEXT ASSY | USED ON
APPLICATION

PROPRIETARY AND CONFIDENTIAL
THE INFORMATION CONTAINED IN THIS DRAWING IS THE SOLE PROPERTY OF <INSERT COMPANY NAME HERE>. ANY REPRODUCTION IN PART OR AS A WHOLE WITHOUT THE WRITTEN PERMISSION OF <INSERT COMPANY NAME HERE> IS PROHIBITED.

TITLE:

SIZE A DWG. NO. Tubular Transducer REV

SCALE: 1:2 WEIGHT: SHEET 1 OF 5



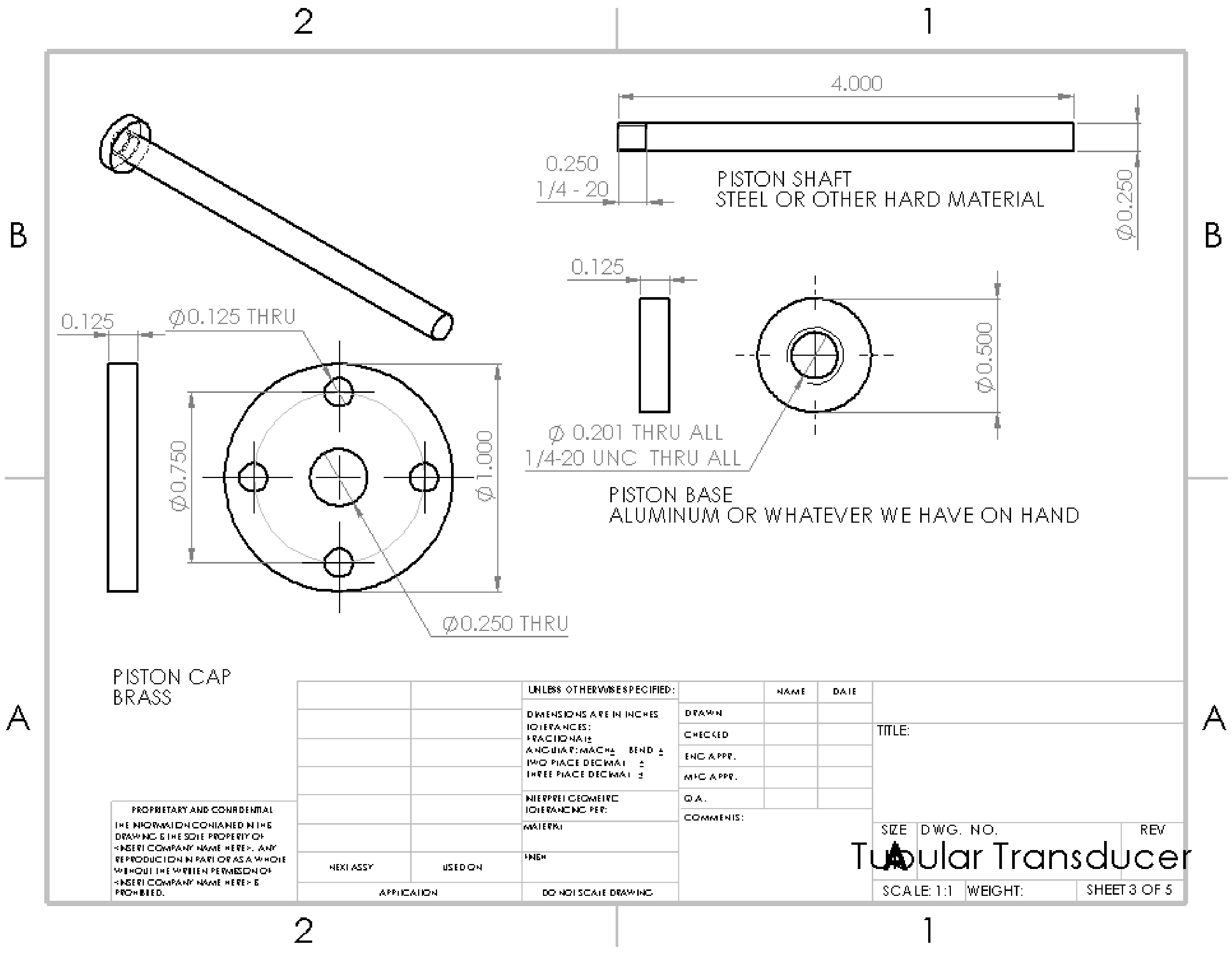
UNLESS OTHERWISE SPECIFIED:		NAME	DATE
DIMENSIONS ARE IN INCHES TOLERANCES: FRACTIONAL± ANGULAR: MACH± BEND ± TWO PLACE DECIMAL ± THREE PLACE DECIMAL ±	DRAWN		
	CHECKED		
	ENG APPR.		
	MFG APPR.		
INTERPRET GEOMETRIC TOLERANCING PER:	Q.A.		
MATERIAL	COMMENTS:		
FINISH			
DO NOT SCALE DRAWING			

NEXT ASSY	USED ON
APPLICATION	

TITLE:

Tubular Transducer

SIZE	DWG. NO.	REV
A		
SCALE: 1:2	WEIGHT:	SHEET 2 OF 5



2
1
B
A
4.000
0.250
1/4 - 20
PISTON SHAFT
STEEL OR OTHER HARD MATERIAL
Ø0.250
0.125
Ø0.500
Ø 0.201 THRU ALL
1/4-20 UNC THRU ALL
PISTON BASE
ALUMINUM OR WHATEVER WE HAVE ON HAND
0.125
Ø0.125 THRU
Ø0.750
Ø1.000
Ø0.250 THRU
PISTON CAP
BRASS
PROPRIETARY AND CONFIDENTIAL
THE INFORMATION CONTAINED IN THIS DRAWING IS THE SOLE PROPERTY OF <INSERT COMPANY NAME HERE>. ANY REPRODUCTION IN PART OR AS A WHOLE WITHOUT THE WRITTEN PERMISSION OF <INSERT COMPANY NAME HERE> IS PROHIBITED.
NEXT ASSY
USED ON
APPLICATION
UNLESS OTHERWISE SPECIFIED:
DIMENSIONS ARE IN INCHES
TOLERANCES:
FRACTIONAL±
ANGULAR: MACH± BEND ±
TWO PLACE DECIMAL ±
THREE PLACE DECIMAL ±
INTERPRET GEOMETRIC TOLERANCING PER:
MATERIAL
FINISH
DO NOT SCALE DRAWING
NAME
DATE
DRAWN
CHECKED
ENG APPR.
MFG APPR.
Q.A.
COMMENTS:
TITLE:
SIZE
A
DWG. NO.
Tubular Transducer
REV
SCALE: 1:1
WEIGHT:
SHEET 3 OF 5

Appendix D - *Arduino* Code

```
//#include <AnalogSmooth.h>
#include <MCP3008.h>
#include <PID_v1.h>
#include <Encoder.h>
/**
   https://github.com/br3ttb/Arduino-PID-
Library/blob/master/examples/PID_Basic/PID_Basic.ino
   https://github.com/MichaelThessel/arduino-analog-smooth
   https://www.pjrc.com/teensy/td_libs_Encoder.html
   https://github.com/nodesign/MCP3008
*/

#define K 2172.5 //in kg/s/s
#define THREADS 20 //20 threads per inch
#define TICKS_PER_REV 4480 //encoder count
#define ROD_LENGTH 3.5 // in inches
#define POT_MIN 181
#define POT_MAX 1017

//Define Variables we'll be connecting to
double setpoint, input, output;

//Specify the links and initial tuning parameters
double Kp = 1, Ki = 0, Kd = .01;
PID myPID(&input, &output, &setpoint, Kp, Ki, Kd, DIRECT);


int state = 0; //variable for state machine

//how many data points have been counted
const int D_THRESH = 30;
int d_count = 0;
int new_set; //variable for smoothing
int serial_val;
//AnalogSmooth aSmooth = AnalogSmooth(10); //define the analog smoother

//motor pins
int E1 = 4;
int M1 = 5;
int ENC1 = 2;
int ENC2 = 3;
int encCount;

Encoder enc1(ENC1, ENC2);

//adc pins
const int clck_pin = 52;
const int miso_pin = 50;
const int mosi_pin = 51;
const int slave_select_pin = 53;

//setup the ADC, uses MCP library
```

```
MCP3008 pot1(clck_pin, mosi_pin, miso_pin, slave_select_pin);

void setup()
{
  Serial.begin(9600); // open the arduino serial port
  pinMode(M1, OUTPUT);
  //setpoint for PID, default to center
  setpoint = 510;
  myPID.SetMode(AUTOMATIC);
  new_set = 0;
  encCount = 0;
}

void loop()
{

  if (Serial.available()) {
    //if it's the first loop, initialize the state machine
    if (state == 0) {
      state = 1;

    }
    serial_val = Serial.read();
  }

  if (state > 0) {

    setpoint = serial_val * 4;
    //get the position
    input = potRead(1);

    if (potRead(1) > 490 && potRead(1) < 1000) {
      //change the direction of the motor
      if (input < setpoint) { //this changes the direction
        digitalWrite(M1, HIGH);
        myPID.SetControllerDirection(DIRECT);
      }
      else {
        digitalWrite(M1, LOW); //speed control
        myPID.SetControllerDirection(REVERSE);
      }


      //       Serial.println(new_set);
      //compute what the motor control value should be
      myPID.Compute();

      //speeeeeeeeed booooostt
      analogWrite(E1, output);
      Serial.println(output);
    }
  }
  else{
      analogWrite(E1, 0);
}
  }
```

```
//calculates the force of the finger on the bowden cable
//@param the number of ticks the encoder has counted
//@param the position of the pot, 0 to 1024
float calculateForce(int encCount, int potRead) {

  //this will give us position in inches from origin
  float encPos = encCount / (TICKS_PER_REV * THREADS);

  //this will give us pot position in inches
  float potPos = ((potRead - POT_MIN) * (ROD_LENGTH)) / (POT_MAX - POT_MIN);

  // F = kx, where k is spring constant and x is displacement
  return K * (encPos - potPos);
}

//reads the ADC
//@param which channel? 0 through 7
int potRead(int chan) {
  return pot1.readADC(chan);
}
```

Appendix E - *Unity 3D* Scripts

JitMessenger

```
// mu (myu) Max-Unity Interoperability Toolkit
// Ivica Ico Bukvic <ico@vt.edu> <http://ico.bukvic.net>
// Ji-Sun Kim <hideaway@vt.edu>
// Keith Wooldridge <kawoold@vt.edu>
// With thanks to Denis Gracanin

// Virginia Tech Department of Music
// DISIS Interactive Sound & Intermedia Studio
// Collaborative for Creative Technologies in the Arts and Design

// Copyright DISIS 2008.
// mu is distributed under the GPL license v3 (http://www.gnu.org/licenses/gpl.html)

using UnityEngine;
using System.Collections;

public class JitMessenger : MonoBehaviour {

public GameObject send;
private JitSend writer;
private bool valid = false;
private bool isCollide = false;
private float impulse;

       // Use this for initialization
       void Start () {
              if (send == null) send = GameObject.Find("Main Camera");
              if (send != null) writer = (JitSend)send.GetComponent("JitSend");
              if (writer != null) valid = true;
        impulse = 5;
       }

       // Update is called once per frame
       // This is where you should forward info as needed
       void Update () {

              if (valid) {

                     //It is recommended to use this script with rigid bodies as that
allows for
                     //easy monitoring of object's activity and consequently its need to
send state data
                     if (!GetComponent<Rigidbody>().IsSleeping()) {
              //  float[] tmp = {transform.eulerAngles.x, transform.eulerAngles.y,
transform.eulerAngles.z};
                float[] tmp2 = { impulse};
                            write('f', tmp2);
               // write('f', tmp2);
            }
              }
        }

        private void write(char method, float[] var) {
                string toWrite = name + " " + method;
                for(int i = 0; i < var.Length; i++){
                        toWrite += " " + var[i];
                }
                toWrite += ";\n";
                writer.write(toWrite);
        }

    public void setImpulse(Collision c)
    {
        impulse = c.impulse.magnitude;
    }

    public void setSend(bool s)
    {
        isCollide = s;
    }
}
```

JitReceive

```
// mu (myu) Max-Unity Interoperability Toolkit
// Ivica Ico Bukvic <ico@vt.edu> <http://ico.bukvic.net>
// Ji-Sun Kim <hideaway@vt.edu>
// Keith Wooldridge <kawoold@vt.edu>
// With thanks to Denis Gracanin

// Virginia Tech Department of Music
// DISIS Interactive Sound & Intermedia Studio
// Collaborative for Creative Technologies in the Arts and Design

// Copyright DISIS 2008.
// mu is distributed under the GPL license v3 (http://www.gnu.org/licenses/gpl.html)

using UnityEngine;
using System.Collections;
using System;
using System.Text;
using System.Net;
using System.Net.Sockets;
using System.IO;

public class JitReceive : MonoBehaviour {

        public int portNo;
        public int maxObjects;

        //struct for pointing to various objects
        private struct objectList {
                public string objName;
                public GameObject objPointer;
                public void set(string s, GameObject gp) {
                        objName = s;
                        objPointer = gp;
                }
        }
        private int numObjects;
        private objectList[] o;

        //custom calls pointer
        JitCustomEvents jitCustom;

        private TcpClient incoming_client;
        private NetworkStream netStream;
        private TcpListener server;
        private bool waiting;

        // Use this for initialization
        void Start () {

                if (portNo == 0) portNo = 32003;
                if (maxObjects == 0) maxObjects = 1024;
                waiting = false;
                server = new TcpListener(IPAddress.Any, portNo);
                server.Start();
                numObjects = 0;
                o = new objectList[maxObjects];
                jitCustom = (JitCustomEvents)GetComponent("JitCustomEvents");
        }

        // Update is called once per frame
        void Update () {

                string s;
                string[] values;

                if (server.Pending()) {
                        incoming_client = server.AcceptTcpClient();
                        netStream = incoming_client.GetStream();

                        waiting = true;
                }
                while (waiting && netStream.DataAvailable) {
                        try {
                                int numread = 0;
                                byte[] tmpbuf = new byte[1024];
                                numread = netStream.Read(tmpbuf, 0, tmpbuf.Length);

                                s = Encoding.ASCII.GetString(tmpbuf, 0, numread);
                                s = s.Replace("\n","");
                                values = s.Split(';');

                                if (values.Length > 1) {
                                        for (int i = 0; i < (values.Length-1); i++) {
                                                Parse(values[i]);
                                        }
                                }
                                else Parse(values[0]);
                        }
```

```
            //Called when netStream fails to read from the stream.
            catch (IOException e) {
                waiting = false;
                netStream.Close();
                incoming_client.Close();
            }
            //Called when netStream has been closed already.
            catch (ObjectDisposedException e) {
                waiting = false;
                incoming_client.Close();
            }
        }
    }

    void Parse(string toParse) {

        GameObject target = null;
        int i;
        bool found = false;

        string[] values = toParse.Split(' ');

        if (numObjects > 0) {
            for (i = 0; i < numObjects && !found; i++) {
                if (values[0].Equals(o[i].objName)) {
                    found = true;
                    target = o[i].objPointer;
                }
            }
        }

        if (numObjects == 0 || !found) {
            target = GameObject.Find(values[0]);
            if (target) {
                o[numObjects].set(values[0], target);
                numObjects++;
            }
        }

        if (!target) {
            print("Requested object " + values[0] + " not found.");
        }
        else {
            switch (values[1]) {
                case "m":
                    if (values.Length == 5) {
                        move(target, values[2], values[3], values[4]);
                    }
                    break;
                case "M":
                    if (values.Length == 5) {
                        reposition(target, values[2], values[3],
values[4]);
                    }
                    break;
                case "r":
                    if (values.Length == 5) {
                        rotate(target, values[2], values[3], values[4]);
```

```
                                        }
                                        break;
                                case "R":
                                        if (values.Length == 5) {
                                                absoluteRotate(target, values[2], values[3],
values[4]);
                                        }
                                        break;
                                case "s":
                                        if (values.Length == 5) {
                                                scale(target, values[2], values[3], values[4]);
                                        }
                                        break;
                                case "S":
                                        if (values.Length == 5) {
                                                absoluteScale(target, values[2], values[3],
values[4]);
                                        }
                                        break;
                                case "c":
                                        if (jitCustom) {
                                                string[] val = new string[values.Length - 3];
                                                for (int j = 0; j < values.Length - 3; j++) {
                                                        val[j] = values[j + 3];
                                                }
                                                custom(target, values[2], val);
                                        }
                                        else {
                                                Debug.Log("Custom script not found.");
                                        }
                                        break;
                        }
                }
        }

        void custom(GameObject tgt, string method, string[] val) {
                int sz = val.Length;
                float[] param = new float[sz];
                for (int i = 0; i < val.Length; i++) {
                        param[i] = (float)System.Convert.ToDouble(val[i]);
                }
                jitCustom.run (tgt, System.Convert.ToInt32(method), param);
        }

        void scale(GameObject tgt, string xVal, string yVal, string zVal) {
                Vector3 newScale = new Vector3((float)System.Convert.ToDouble(xVal),
                        (float)System.Convert.ToDouble(yVal),
(float)System.Convert.ToDouble(zVal));
                tgt.transform.localScale += newScale;
        }

        void absoluteScale(GameObject tgt, string xVal, string yVal, string zVal) {
                Vector3 newScale = new Vector3((float)System.Convert.ToDouble(xVal),
                        (float)System.Convert.ToDouble(yVal),
(float)System.Convert.ToDouble(zVal));
                tgt.transform.localScale = newScale;
        }
```

```
	void reposition(GameObject tgt, string xLoc, string yLoc, string zLoc) {
		Vector3 newLoc = new Vector3((float)System.Convert.ToDouble(xLoc),
			(float)System.Convert.ToDouble(yLoc), -
(float)System.Convert.ToDouble(zLoc));
		tgt.transform.position = newLoc;
	}

	void move(GameObject tgt, string xVal, string yVal, string zVal) {
		tgt.transform.Translate((float)System.Convert.ToDouble(xVal),
			(float)System.Convert.ToDouble(yVal), -
(float)System.Convert.ToDouble(zVal));
	}

	void rotate(GameObject tgt, string xVal, string yVal, string zVal) {

		tgt.transform.Rotate((float)System.Convert.ToDouble(xVal),
			(float)System.Convert.ToDouble(yVal),
(float)System.Convert.ToDouble(zVal));
	}

	void absoluteRotate(GameObject tgt, string xVal, string yVal, string zVal) {
		float toX = (float)System.Convert.ToDouble(xVal);
		float toY = (float)System.Convert.ToDouble(yVal);
		float toZ = (float)System.Convert.ToDouble(zVal);
		Quaternion rot = Quaternion.identity;
		rot.eulerAngles = new Vector3(toX, 180-toY, toZ);
		tgt.transform.rotation = rot;
	}
}
```

JitSend

```
// mu (myu) Max-Unity Interoperability Toolkit
// Ivica Ico Bukvic <ico@vt.edu> <http://ico.bukvic.net>
// Ji-Sun Kim <hideaway@vt.edu>
// Keith Wooldridge <kawoold@vt.edu>
// With thanks to Denis Gracanin

// Virginia Tech Department of Music
// DISIS Interactive Sound & Intermedia Studio
// Collaborative for Creative Technologies in the Arts and Design

// Copyright DISIS 2008.
// mu is distributed under the GPL license (http://www.gnu.org/licenses/gpl.html)

using UnityEngine;
using System.Collections;
using System;
using System.Text;
using System.Net;
using System.Net.Sockets;
using System.IO;
```

```
public class JitSend : MonoBehaviour {

public int portNo = 32000;
public string addr = "127.0.0.1";
public int packetTreshold = 100;

private TcpClient client;
private NetworkStream netStream;
private Queue updateSend;

private int msgCounter;
private float connectionAttempt;
private bool connection;

	void Update() {
		if (updateSend.Count != 0) {
			try {
				if (!connection) {
					connectionAttempt += Time.deltaTime;
					if (connectionAttempt > 1.0f) {
						connectionAttempt = 0.0f;
						try {
							client.Connect(addr, portNo);
							connection = true;
						}
						catch(Exception e) {
							connection = false;
						}
					}
					if (connection) {
						netStream = client.GetStream();
					}
					else if (msgCounter > packetTreshold) {
						updateSend.Clear();
						msgCounter = 0;
					}
				}
				if (connection) {
					string toWrite = (string)updateSend.Peek();
					byte[] output;
					output = Encoding.ASCII.GetBytes(toWrite);
					netStream.Write(output, 0, toWrite.Length);
					updateSend.Dequeue();
				}
			}
			//Called when netStream has been closed.
			catch (Exception e) {
				netStream.Close();
				client.Close();
				client = new TcpClient();
				connection = false;
				try {
					client.Connect(addr, portNo);
					connection = true;
				}
				catch(Exception se) {
					connection = false;
				}
```

```
                    if (connection) netStream = client.GetStream();
                }
            }
    }

    // Use this for initialization
    void Start () {
            client = new TcpClient();
            updateSend = new Queue();
            msgCounter = 0;
            connectionAttempt = 0;
            connection = false;
            if (packetTreshold == 0) packetTreshold = 1;
    }

    // Externally called from jitMessenger script associated with various objects
    public void write (string toWrite) {
            updateSend.Enqueue(toWrite);
            msgCounter++;
    }
}
```

OnCollision

```
using UnityEngine;
using System.Collections;

public class OnCollision : MonoBehaviour
{

    private JitMessenger mess;

    void OnCollisionEnter(Collision col)
    {
            if (col.gameObject.name == "palm")
            {
                mess = (JitMessenger)
GameObject.Find("Cube").GetComponent("JitMessenger");
                mess.setSend(true);
            }
    }
    void OnCollisionStay(Collision col)
    {
        if (col.gameObject.name == "palm")
        {
            mess = (JitMessenger)GameObject.Find("Cube").GetComponent("JitMessenger");
            mess.setImpulse(col);
        }
    }

    void OnCollisionExit(Collision col)
    {
        if (col.gameObject.name == "palm")
        {
            mess = (JitMessenger)GameObject.Find("Cube").GetComponent("JitMessenger");
            mess.setSend(false);
        }
    }
}
```