HYDRO MUSCLE CONTROL SYSTEM

April, 2015

Authors

Andrew Barth
Jonathan Sorrells
Seiichiro Ueda
Jonathan Wu

Advisor

Marko Popovic, Ph.D.
Professor of Physics, Biomedical Engineering, and Robotics Engineering

Co-Advisors

Fred J. Looft, Ph.D.
Professor of Electrical and Computer, and Systems Engineering

Dmitry Berenson, Ph.D.
Professor of Robotics Engineering and Computer Science

A Major Qualifying Project Report Submitted to the Faculty of

WORCESTER POLYTECHNIC INSTITUTE

In partial fulfillment of the requirements for the Degree of Bachelor of Science.

Table of Contents

Table of Figures

Abstract

Hydro-muscles are hydraulically powered actuators that share similar properties with biological muscles. The action of a hydro-muscle can be controlled by outfitting it with sensors to measure length and internal pressure. The data from the sensors is used to implement proportional integral derivative (PID) control of a flow control valve for precise length and force control. Different sensor configurations are investigated and compared to determine what produces the sufficiently accurate readings while still being low cost. It was found that position and force control of hydro-muscles is feasible with sensors attached to the muscles. An application of controlled hydro-muscles is implemented in the form of a teleoperable elbow brace. The performance of the system is discussed, and future recommendations for hydro-muscles are presented.

Acknowledgements

We would like to thank Professor Marko B. Popovic for guiding us through this project and giving us space to work in Popovic Labs. We thank Professor Fred J Looft for providing his insight, feedback on this paper, and for purchasing important components for the project.

We thank Nicholas Deisadze for machining necessary parts and helping with the mechanical portion of our project. We thank Thane R. Hunt for providing general assistance on various parts of the project, and for being awesome.

We also thank Robert Boisse for providing various circuit components and attaching the pressure sensors onto the endcaps. We would also like to thank Joseph St. Germain for letting us borrow tools and providing some insight.

1 Introduction

1.1 Introduction

This project extends the work done for the Hydro Artificial Muscle Exo-Musculature project (Corso et al., 2015), which focused on the development and testing of hydraulically actuated elastic muscles. These muscles consist of stretchable tubes encased in an inelastic sheathing. The muscles apply force by extending and contracting linearly based on the pressure of the water fed into the muscles. Building off of their design, our project was focused on the creation of a control system that makes the muscles usable in a variety of robotic applications. Specifically, each muscle is outfitted with a number of sensors used to measure change in muscle length and the force exerted by the muscle. These readings are then used in conjunction with a hydraulic flow control system to limit the flow rate of water to and from the muscle. This allows for more precise control of length and force, making the muscles usable in practical applications.

To demonstrate the control system, an arm brace was built with several muscles attached to it. The muscles expand and contract, which causes the arm brace to rotate about the elbow joint. The brace can be used in physical therapy, to provide some guidance and force on a patient's arm if the patient needs help moving it. Teleoperation of the brace would allow for someone else to control the brace while the patient wears it. This way, a doctor could wear one brace and remotely guide the patient's arm, eliminating the need for them to leave their homes. The motions can also be recorded and played back, so the doctor can record the motions that their patient should go through, and then the patient can do their exercises at any time. The results from the patient's exercises can be recorded and reviewed at a later time, to track progress. The arm brace may also have applications in virtual reality, by applying force to the arm based on interactions with virtual objects, however, our MQP did not have enough time to examine that application further.

1.2 Problem Statement

The purpose of this project was twofold; first to refine the design of an existing hydro-muscle to incorporate pressure and length sensors and, second, to develop a control system to allow for the accurate and precise manipulation of the hydro-muscles.

Length and force control tests were run on the hydro-muscles, to test the accuracy of the control. Then, to demonstrate the control system's functionality the muscles were attached to an arm brace and used to actuate the brace. As both the muscles and the brace have one degree of freedom, it was a simple way of identifying if the control system was properly functioning. The operator was able to send a desired condition to the muscle and have it respond to match that condition. In the case of an arm brace, a desired extension of the muscle was sent to the control system to achieve an angle of motion on the arm brace and the control system actuated the muscle to reach the desired extension.

1.3 Summary

Extending on the work of the Hydro Artificial Muscle Exo-Musculature project, a control system was integrated into the hydraulic muscles. These newly developed muscles are capable of communicating their current condition and can be precisely controlled. As a demonstration of the control system, an arm brace is actuated using the artificial muscles.

2 Background

2.1 Introduction

This section discusses hydro-muscles, as well as several actuators and systems that are similar to hydro-muscles. First, the similarities with biological muscles are discussed. Next, McKibben muscles, another type of artificial muscle powered by fluid, are discussed. Then, extendable hoses are discussed. Previous hydro-muscle development is discussed, and then control theory as applicable to the hydro-muscles is discussed.

2.2 Background Materials

2.2.1 Biological Muscles

Recent years have seen a growing interest in bionics: the application of biological inspired solutions to mechanical problems. (ESA, 2015) This transfer of technology between life forms and mechanical systems is desirable since evolutionary pressure typically forces biological systems to become highly optimized and efficient. By simplifying and enhancing these systems to solve problems in engineering design, engineers can create novel and effective solutions to many of today's challenges (Chakrabarti & Shu, 2010).

One common area of study in bionics is the creation of artificial muscles: devices or materials that contract, expand, or rotate given external stimulus. Artificial muscles are classified based on their actuation mechanism, the most common categories being electric field, ion, pneumatic, and thermal based. Due to their high flexibility, versatility, and power-to-weight ratio compared with traditional rigid actuators, artificial muscles have the potential to be a highly disruptive emerging technology. (ESA, 2015)

2.2.2 Pneumatic Artificial Muscles

Pneumatic Artificial Muscles (PAMs) are actuators operated by applying pressure to a pneumatic bladder. They are similar to organic muscles in that they contract and exert force when actuated. First developed under the name of McKibben Artificial Muscles in the 1950's, these actuators are some of the earliest bio-inspired actuators to be developed. Pneumatic Artificial Muscles offer a number of advantages over traditional hard-bodied actuators. (Klute et al., 1999) The design of PAM's also makes them compliant without the need for a special controller. This makes them suitable for delicate applications or for human interaction. Additionally, the force exerted by PAMs is not dependent only on pressure but also on their state of inflation, resulting in a non-linear model that allows for more precise control at higher pressures, and also better mirrors the length to tension relationship found in biological muscles. (Klute et al., 1999) As shown in Figure 1, the main element in a PAM is a thin membrane, which makes them much more flexible and lightweight than traditional actuators. This allows for them to be connected directly to the elements that they are powering without the need for special adapters, and also

makes replacing defective or worn parts much easier. McKibben muscles do, however, have the disadvantage that they expand radially when actuated, making them difficult to attach to the surfaces of robotic frames.

Figure 1: McKibben muscle ("Pneumatic Artificial Muscles")

2.2.3 XHOSE Extendible Garden Hose

The Hydro-Muscles were in part inspired by the design of XHOSE's extendible garden hoses shown in Figure 2. These extendible garden hoses consisted of an elastic inner tube combined with a folded outer layer made from nylon webbing. Pressurizing the hose with water caused the hose to expand longitudinally until it reached the limits of the non-elastic outer webbing. Depressurizing the hose shrinks the hose back down to the original size. The hose is lightweight with a 50 foot section weighing only 2 pounds when not filled with water (XHose, 2015).

Figure 2: XHOSE, expandable hose ("Dap")

2.2.4 Hydro-Muscles

Hydro Artificial Exo-Muscles (HAMs) are hydraulically actuated artificial muscles that share many of the same characteristics as biological and Pneumatic Artificial Muscles, differing in that HAMs contract radially and longitudinally at the same time, rather than expanding radially while contracting longitudinally.

HAMs consist of an inner layer of stretchable tubing, and a non-stretchable outer layer. Figure 3 shows the construction of the muscles. The outer layer is wrinkled while the muscle is contracted, and allows for the muscle to expand longitudinally, while only allowing a small amount of radial expansion. When the muscle is pressurized, the muscle expands longitudinally. When the muscle is depressurized, it will contract. The expansion force of the muscle is caused by the pressure inside the muscle, minus the spring force of the inner tubing. Thus, when contracting, the maximum force the muscle can provide is determined by the tubing used. The maximum force that McKibben muscles can exert is determined by the air compressor used. Hydro-muscles, on the other hand, are able to store energy from a slow compressor using the elastic tubing, and release the energy quickly.

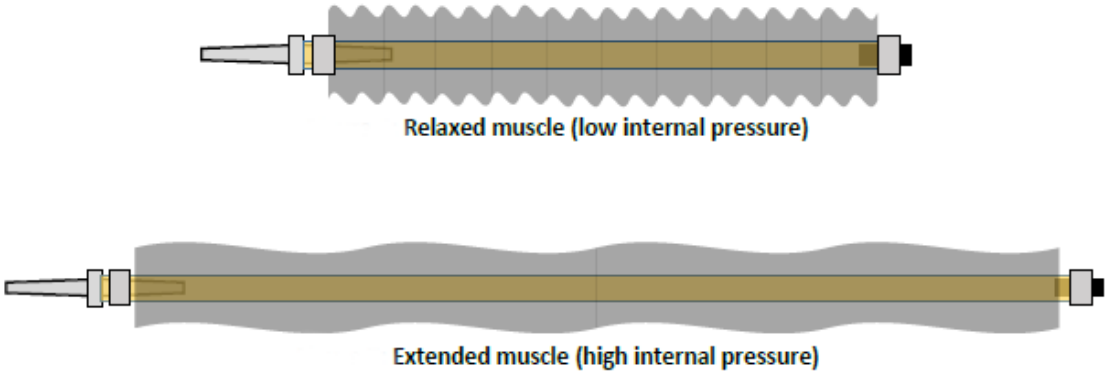

Figure 3: Hydro-muscles, figure from Hydro Artificial Muscle Exo-Musculature report (Corso et al., 2015).

Previously, the hydro-muscles had only very simple control systems associated with them since muscles were controlled manually by opening or closing solenoid valves (Corso et al., 2015). A control system that could have the muscles go to specific lengths or forces would allow these muscles to be used in many more applications. For a control system to be feasible, the muscles would also need to have sensors embedded in them, to determine the current length of the muscle.

2.2.5 Control Theory

When controlling a system, there are two main categories of control: open loop and closed loop, shown in Figure 4. In open loop systems, there is no feedback from state sensors. The controller tries to make the plant reach the desired output based on knowledge of how the system responds to different inputs.

This type of control does not work well for systems that may have outside interference, because the controller does not know about the interference, and therefore cannot compensate for it. A closed loop controller, on the other hand, senses the output state of the system, and takes that into account when controlling it. This allows for the controller to compensate for outside interference to the system ("A Basic Guide").

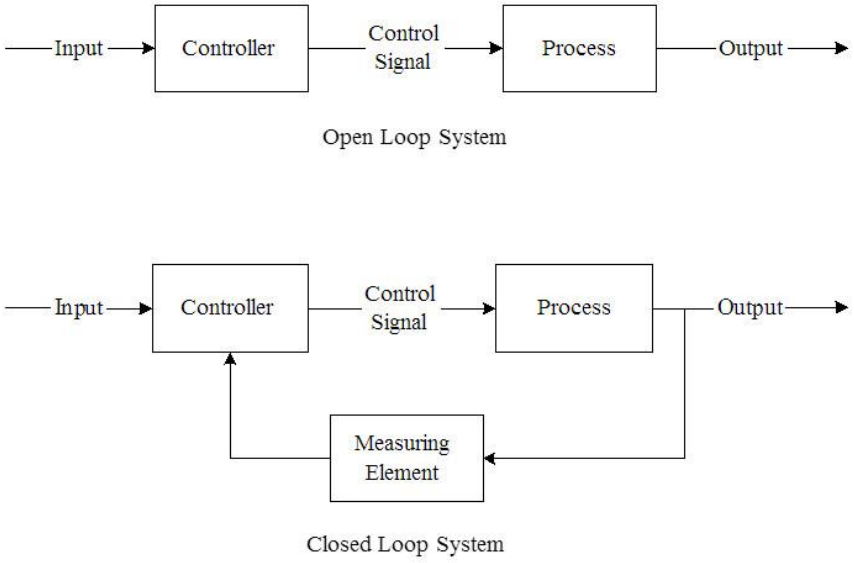

Figure 4: Open vs Closed Loop Controllers, figure from Micromo ("A Basic Guide")

A common type of closed loop controller is a Proportional Integral Derivative controller. This controls the system based on the current error (proportional), past error (integral), and the rate of change of the error (derivative). A PID controller must be tuned to the system that it is controlling, by adjusting the gains and observing the effect on the system. There are different ways of tuning controllers, but one popular method is the Ziegler-Nichols method (Nichols & Ziegler, 1942). This involves increasing the proportional gain until the system starts oscillating, and then setting the P, I, and D gains based on that value (Co, 2004). The system can also be manually tuned by changing one gain at a time, to change the system response to be more desirable. Depending on the system, different characteristics may be more or less preferable, such as overshoot or steady state error. Increasing the P gain, for example, will reduce the time it takes to reach the set point but may cause overshoot if the P gain is set too high, as depicted by the underdamped line in Figure 5. Increasing I will decrease the steady state error, but potentially increase overshoot. Setting D too high will over damp the motion, and setting it too low will result in an underdamped system. The Ziegler-Nichols method may be a good place to start tuning gains, but manual tuning will often be needed to fit the system's specific needs (Zhong, 2006).

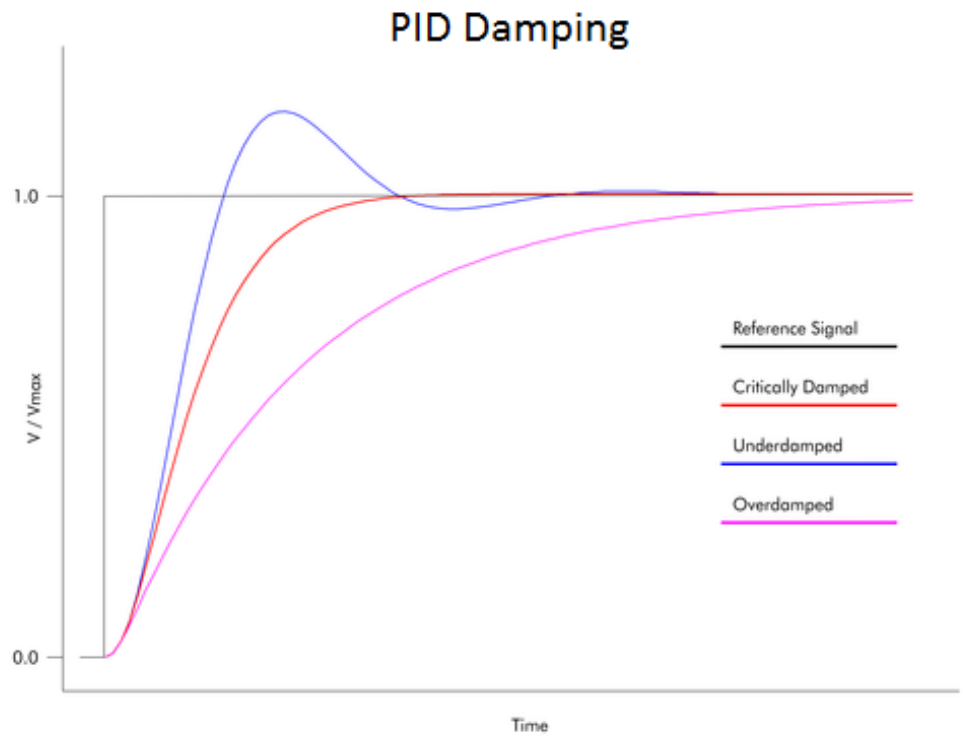

Figure 5 PID Dampening

Another important factor in controlling a system is filtering of the inputs. In an ideal system, the sensor readings would not have any noise, but in real systems there may be substantial noise. Noise in sensor readings can cause the system to jitter back and forth, as the controller responds to perceived deviations from the set point. Filtering the sensor output can lead to a smoother response, and less steady state error, but also increases the settling time, due to the delay introduced by the filter. The set point of the system can also be filtered, to prevent the controller from responding too aggressively to changes in the set point, while still allowing quick response to error caused by external factors (Cooper, 2006).

2.3 Summary

The hydro-muscles are actuators inspired by other artificial muscles and extendable garden hoses. However, unlike the McKibben Muscle, the hydro-muscles longitudinally extend rather than contract when pressurized. The muscles are also equipped with sensors to allow for fine control.

3 Problem Statement

3.1 Introduction

This section outlines the specific project goals, and the requirements related to building and controlling new hydro-muscles.

3.2 Problem Statement

The purpose of this project was to create a control system for the hydro-muscles, which required the attachment of sensors to the muscles. To demonstrate a precision control case, we proposed the development of an instrumented arm brace that, when coupled with our proposed control system, would then control the length or force of a hydro-muscle so that it would match the length or force of the arm brace. From a basic needs perspective, our team used the following statements to drive the design of our system;

1. The sensors should be integrated into the muscle, so that other applications can use the same sensors connected to the muscles.
2. The arm brace should be controlled based on sensor data from muscles connected to a second brace.
3. The arm brace should provide force feedback to the controlling brace, i.e. a master and slave setup.
4. The system should be designed so that it is possible to record and playback motions or forces on the brace, to allow operation of a single brace.
5. The muscles should provide enough force to help guide one's arm in the desired direction, but should not be so strong that the muscles cannot be overpowered by the user.
6. The hydro-muscles should be robust and durable; they should be able to withstand being handled and be able to operate for many hours without breaking.
7. The control system should be designed to include proportional, integrative and derivative (PID) state control with adjustable gains for each state channel.

3.3 Summary

This project improves upon the previously designed muscles by adding a precise control system. Length and force sensors are integrated into the muscle to provide data which can be used for accurate control. Two arm braces are outfitted with these muscles, which can then be used with teleoperation.

4 System Requirements and Design

4.1 Introduction

This section discusses the design of the hydro-muscle arm brace system and the various electronics, controllers, and other components that are included in the system. Each part has unique requirements that define how they must operate as well as conditions and restraints for how they interact with each other.

4.2 System Requirements

The following objectives were decided on to ensure all goals are met for this project.

Requirement 1:	The flow rate of water through a muscle shall be adjustable between zero flow and maximum flow. The flow shall be able to vary from maximum to minimum within 0.5 seconds.
Requirement 2:	The length of a muscle shall be able to be controlled to within $1/8^{th}$ of an inch.
Requirement 3:	The net force applied by a muscle shall be controllable to within 1 Newton.
Requirement 4:	A muscle shall reach its desired force or length within 1 second.
Requirement 5:	A muscle shall be able to be teleoperated using measurements from another muscle. Teleoperated muscles shall be constrained to all aforementioned requirements.
Requirement 6:	A medium for demonstrating the successful implementation of all requirements shall be created.
Requirement 7:	The total cost of a muscle shall be under $10.00.
Requirement 8:	The muscles shall be durable. They need to not break after extensive use.
Requirement 9:	The muscles shall be modular. Parts of the muscle should be interchangeable; one part should be replaceable without needing to replace the entire muscle.

Table 1: System Requirements

4.3 System Needs

This section expands on the needs outlined in the project statement above (section 3.2).

4.3.1 Muscle

The artificial muscle needs to be a robust system that is easily replaceable. If a part of the muscle fails, only that specific part should have to be replaced rather than the whole muscle; this leads to a modular design for the system. The muscle itself should be under $10.00 to construct and the sensors should be exchangeable to fit the user's need, whether they need different sensors or don't need any.

4.3.2 Length Sensor

The length sensor needs to be able to provide accurate measurements of the muscle's length without affecting its performance. This sensor only needs to be able to measure the longitudinal extension of the muscle, but needs to be able to detect changes in length due to outside sources, i.e. someone pulling on

the muscle. The muscles can stretch upwards of 100% strain, so the length sensor should also be able to stretch the full range of motion of the muscles.

4.3.3 Pressure Sensor

The pressure sensor needs to accurately measure the pressure within the muscle. It needs to be water resistance and able to operate in the range of pressures provided by the water pump. The pressure reading from the sensor will be used to calculate the force being applied by the muscle. As with all components of the muscle, it should be easily removable for modularity. The water pump used in this system is capable of producing 100 psi, so the pressure sensor must be able to accurately measure up to at least that pressure.

4.3.4 Signal Conditioning

The signals from the sensors contain undesired noise that needs to be cleaned up. Filters should be designed and built in hardware or implemented in software to provide a clean input that will not confuse the control system. Readings from sensors should be scaled to a range of 0V to approximately 5V to maximize the number of detectable data points when read by the Arduino. Additionally, a transfer function for each circuit should be known so that voltage readings can be easily converted to the proper measurement. All circuits should be neatly placed on a single printed circuit board (PCB).

4.3.5 Controls

The control system needs to be able to bring the muscle to a desired force or length. Since only one variable of the system is being controlled, pressure, it is not possible to control both the force and length of a given muscle at the same time. However, the system should be able to control the length of one muscle and the force of another muscle simultaneously.

The control system has the pressure inside the muscle and the length of the muscle as inputs. The control system is able to output to on/off solenoid valves, and also to a flow control valve.

For teleoperation, the goal is to have two arm braces; a master brace and slave brace. The position of the master brace is measured, and then the slave brace is moved to that position. At the same time, the force on the slave brace is measured, and force feedback is provided to the master brace.

The control system should also be able to record motions from one muscle, and then play back those motions. These recording files are saved on a computer, and can easily be sent from a doctor to a patient, for example. In addition to recordings, it should be possible to input several lengths or forces and times, and have the system calculate a smooth curve that goes to all the given lengths or forces at the given times.

4.4 System Design

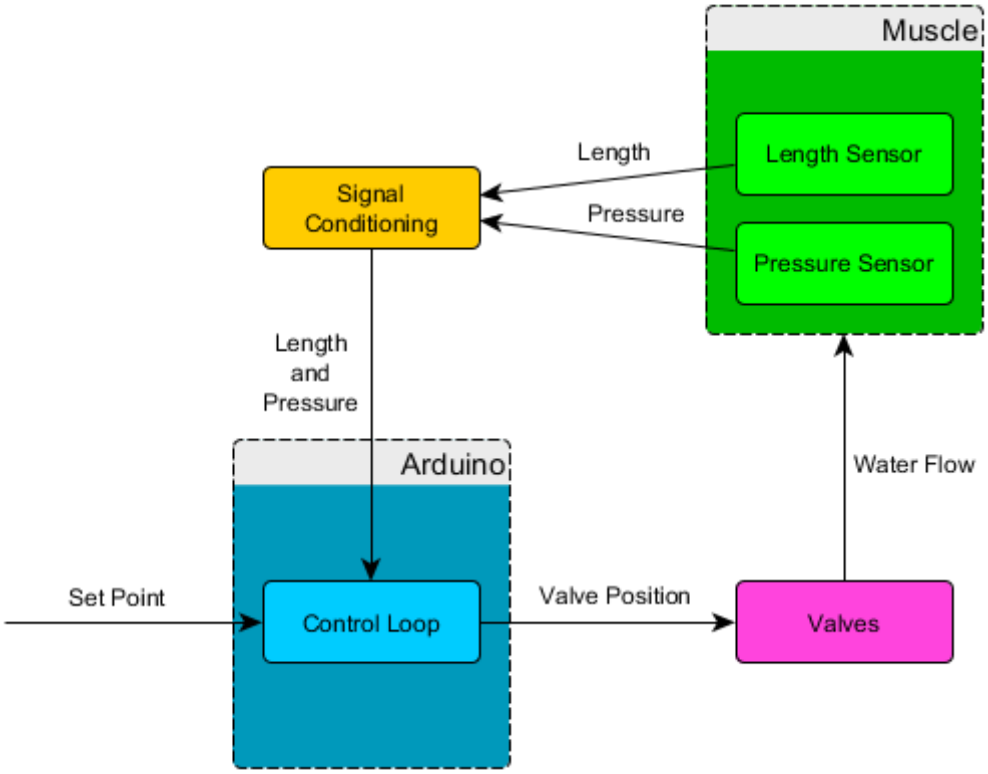

Figure 6: System Block Diagram

Figure 6 summarizes the overall design of the system. All blocks interact such that the muscle matches the desired length or force value sent to the Arduino. The Arduino calculates the difference between the muscle's actual and desired state and outputs control signals to the valves. Signals sent to the valves indicate both the direction of water flow required to reduce the difference between actual and desired states, and the rate at which water should flow through the muscle.

Water flows in and out of the muscle as the valves open and close, changing the internal water pressure. Increasing and decreasing the pressure inside the muscle causes the inner tubing of the muscle to expand and contract, respectively, until the elastic return force of the latex is at equilibrium with the pressure inside the latex tubing. Length and pressure sensors connected to the muscle are used to measure the changes in length and pressure. The output from each sensor is passed through signal conditioning circuitry designed to process each signal and output a signal which can be read by the Arduino.

The Arduino constantly checks the length and pressure readings and adjusts the control signal sent to the valves accordingly. If the muscle is in the desired state, the Arduino will close the valves, prohibiting further flow of water. If a new set point is received, or the muscle deviates from the desired state due to outside influence, the system will react to move the muscle to the desired state.

4.5 Summary

Sensors need to be integrated into the muscle, to measure the length and pressure. The muscle must be durable and inexpensive. The system should be able to control the length or force of the system precisely. The control loop will run on an Arduino, and control the positions of valves to regulate the flow of water into the muscle. The sensors on the muscle will be used to provide feedback to the control system, allowing for accurate control.

5 Results

5.1 Introduction

This section will discuss the choices made to construct the muscles, and the results of experiments that were run to test the muscles and control system.

5.2 Muscle Construction

The muscles were constructed by inserting a section of latex tubing into a length of nylon tubular webbing constraining both the radius and length of the latex during expansion. A plastic hose-barb to thread adapter was then inserted into each end of the tubing and secured with an adjustable hose tie. Both silicone and latex were tested for the muscles, however the silicone suffered from hysteresis and so latex proved to be the better choice to construct the muscles. A sample muscle is shown in Figure 7.

Stretch and pressure sensors were added in order to monitor the elongation and internal pressure of the muscle in order to satisfy Requirement 2 and Requirement 3. There have been several variants of muscles using different sensors. The pressure sensors were inserted through a small hole drilled through the endcap and secured using epoxy, shown in Figure 10. Stretch sensors consisted of a strip of conductive rubber that ran in parallel to the muscle between the rubber tubing and the sheath, and were attached at each end using the hose tie. Another variant of the length sensor, shown in Figure 8, had wires inside both ends of the muscles to measure the resistivity of the water, and the last variant, shown in Figure 9 consisted of a retractable reel connected at one end of the muscle, and an encoder attached to the reel to measure the rotation of the reel, which is proportional to the length of the muscle.

Figure 7: Muscle built using latex, brass fittings, nylon webbing, and conductive rubber

Figure 8: Muscle with wire running out of latex tube for measuring water resistivity

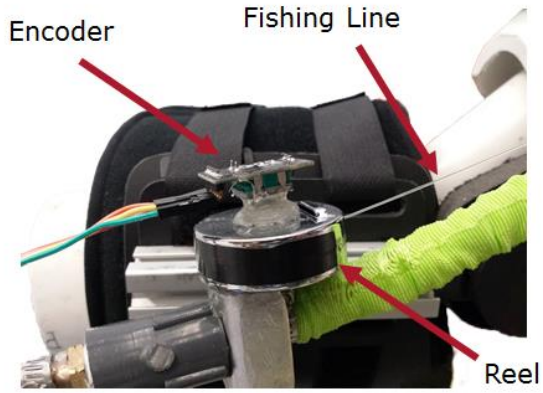

Figure 9: Encoder attached to retractable reel

Figure 10: End cap with pressure sensor attached

The final muscles that we built for our system were 6.5 in long in their resting state, with a maximum length of 12 in when fully expanded and at their maximum pressure. Figure 11 shows the components to make a single muscle. The muscles consisted of 6.5 in of latex tubing, 12 in of nylon tubular webbing, 48in^2 of nylon parachute material, 2 plastic barbed fittings, 2 hose clamps, 1 retractable reel, 1 encoder, 1 end cap, and 1 pressure sensor (see Table 6). The total cost of a muscle plus sensors is $32.89. The majority of this cost is the sensors though. A muscle without sensors costs only $5.57, which satisfies Requirement 7. If the muscle breaks, it can be replaced without needing to replace the sensors, saving cost, and solving the modularity set by Requirement 9. Also, if multiple muscles are used in parallel for increased force, only one of the muscles needs to have sensors in it. The list of materials used to construct the muscles and the prices of each individual material can be found in Appendix B: Materials.

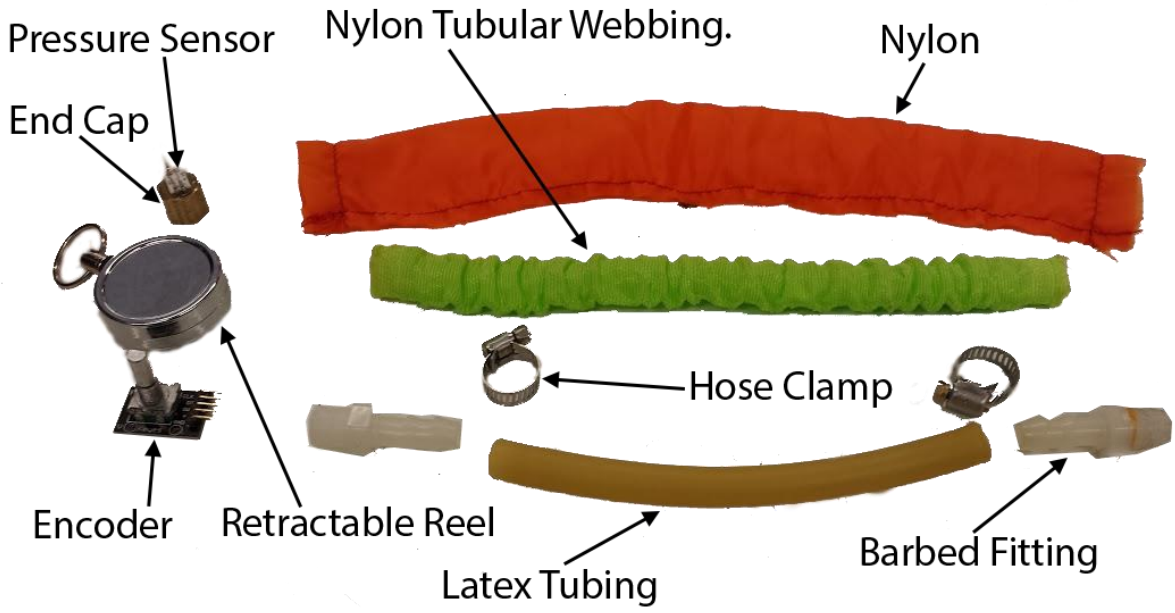

Figure 11: Components of Muscle

5.3 Arm Brace Construction

In order to satisfy Requirement 6, an arm brace actuated using the hydro-muscles was constructed. The arm exo-musculature utilizes two hydro-muscles to control the angle of an elbow brace. The muscles run parallel along both sides of a hinged elbow brace and connect to 80/20 extrusions screwed into the brace at the bicep and forearm, as shown in Figure 12. In the muscles' relaxed state, the arm brace will be curled. As the muscles are pressurized, the brace will rotate to its maximum angle of 150 degrees. This angle was chosen because the torque the muscles apply to the brace decreases at higher angles, and 150 degrees was found to be low enough that the muscles could still move the brace. Two braces are built using this design to work as a master brace and a slave brace. Muscles attached to the slave brace will extend or contract to match the length of the muscles on the master brace. Simultaneously, muscles on the master brace will pressurize or depressurize in order to apply the same force that the muscles on the slave brace are applying. This method will provide force feedback to the master brace, allowing the wearer to feel the forces being applied to the slave brace.

Figure 12: Arm brace

5.4 Muscle Sheathing

In order to save time when constructing the muscles, a pocket hose (variant of the XHOSE) was purchased. The outer nylon webbing was removed from the hose and used as the sheathing to constrain the latex tubing of the muscle. The webbing removed from the hose is shown in Figure 13.

Figure 13: Nylon sheathing removed from hose

The sheathing worked extremely well until the length and pressure sensors needed to be integrated into the system. It was decided that the system would be neater and easier to handle is all the tubes and wires were concentrated onto one end of the muscle. However, because the muscle needs to be connected to the water source from one end of the muscle, the pressure sensor naturally went to the opposite end creating the need to run connections from one end of the muscle to the opposite end. Having wires would cause them to dangle when the muscle was contracted and could snag on something. Conductive thread was tested as an alternative to wires.

5.4.1 Conductive Thread

Conductive thread was sewn along the nylon sheathing and had header pins attached at both ends to connect the pressure sensor to wires on the other side of the muscle, shown in Figure 14. The conductive

thread was effective as a low-profile connection across the muscle. However, a major issue with this modified sheathing is the time it takes to create them. The nylon sheathing was already in a tubular shape so the conductive thread could not be sewn using a sewing machine without sewing the webbing shut, so the sewing had to be done by hand.

Figure 14: Nylon webbing with conductive thread

5.4.2 Handmade Sheathing

Parachute material was also tested as sheathing for the muscle. The parachute material sheathing was made by hand to speed up the process of making sheaths with conductive threads. By making the entire sheath by hand, the conductive thread could be sewn in by the sewing machine before sewing the sheath into a tube. Parachute material was used to make the sheaths because a large sample of it was available. This sheathing with conductive thread sewn horizontally is shown in Figure 15. A second outer sheathing was also made to cover the conductive thread so that they would not short by having someone grab them.

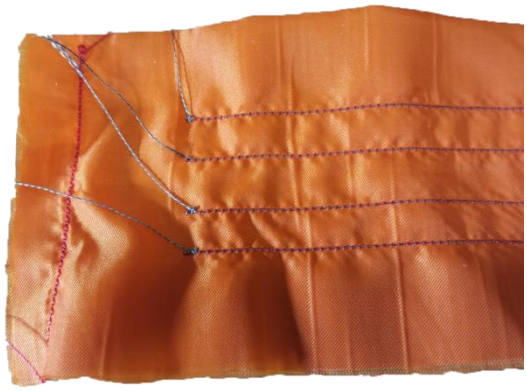

Figure 15: Parachute material with conductive thread

The new sheathing was very promising; however, a few issues arose. The parachute material was very soft and would occasionally twist and connect two lines of conductive thread, shorting them together. The parachute material was rejected because it was not durable enough to meet Requirement 8. The parachute material ripped when it was sewn with more durable stitching, and during testing we observed a parachute sheathing rip when the muscle was pressurized to approximately 60 psi. These issues could

potentially be solved by building the sheaths with a stronger fabric, but there wasn't enough time to acquire and test different materials.

5.4.3 Compromise

Due to a time restraint the sheathing were switched back to the nylon tubular webbing, at this time the outer sheathing from the previous iteration was only used to constrain the wires and prevent them from freely dangling. This method proved to be effective; the only undesired trait was that some length of wire sticks out of one side of the muscle, as shown in Figure 16, which was still much better than the wires hanging.

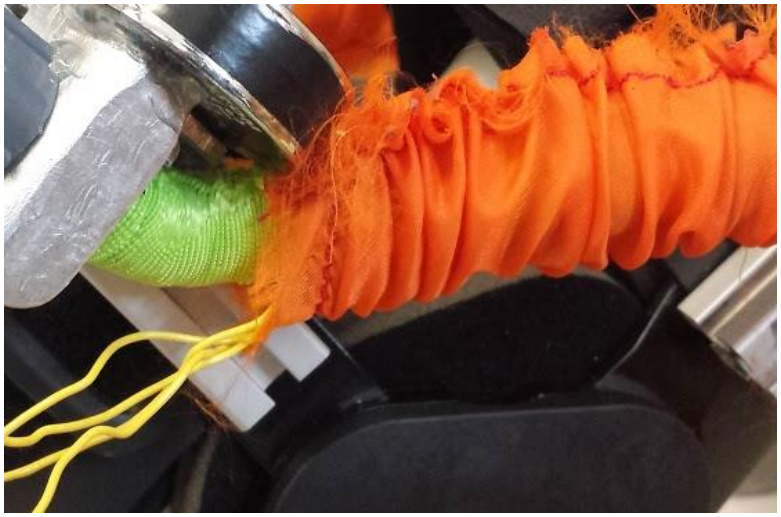

Figure 16: Wires sticking out of outer sheathing

5.5 Length Sensors

A sensor needed to be integrated into the Hydro-muscle to measure its length without inhibiting its motion. The sensor needed to be low-cost and made from easily obtainable parts and materials. The sensor's form factor also had to be low-profile enough so that it wouldn't become obtrusive. Finally, the sensor also had to perform well enough to give consistent and accurate readings quickly enough for the control system to meet Requirement 4.

5.5.1 Conductive Elastic

This method attempted to measure the length of the muscle using the Conductive Elastic Cord from Adafruit. The conductive elastic material was run parallel to the muscle, between the sheath and the latex tubing, and anchored to each end of the muscle by hose clamps. The resistance of the elastic increases linearly as it is stretched so the length of the muscle can be calculated by connecting it in a voltage divider and measuring the change in voltage across the material.

Unlike the latex tubing, conductive elastic is only able to extend to approximately 1.5 times its length (50% strain), limiting the range of motion of the muscle. When placed in a voltage divider circuit, the output voltage across the conductive rubber is described by (1).

$$V = \frac{V_s R_s}{R + R_s} \tag{1}$$

Where V_s is the supply voltage and R_s is the resistance of the elastic. The circuit is given 5V supply and R is chosen to be equal to the resistance of the unstretched elastic. As the elastic stretches, the resistance will vary such that the ratio of stretched length to relaxed length is equal to the ratio of the stretched resistance to the relaxed resistance. Therefore, when the elastic is stretched to its maximum length of 1.5 times its initial length, the resistance will be at 1.5 times its initial resistance. Knowing this, the range of the output voltage across the conductive elastic can be computed using (1), giving the range:

$$2.5V \leq V \leq 3V \tag{2}$$

This means that fully stretching the elastic will only result in a change of 0.5V. Passing this signal into the Arduino's analog to digital converter (ADC) without further conditioning would result in poor precision, due to the small amount of detectable data points in this range ("AnalogRead", 2015). To increase the signal to cover a range of 5V (the range of the ADC) it is passed through an operational amplifier. The output of an operational amplifier is characterized by (3):

$$V_{out} = A(V_+ - V_-) \tag{3}$$

V was passed to the positive input (V_+) and the negative input (V_-) was given a fixed 2.5V. This, along with a gain, A, of 10 gives an output of 5V which efficiently utilizes the full range of the ADC.

Unfortunately, the conductive elastic had large hysteresis and creep, and was not able to provide a linear output voltage over a linear change in length. Alternative stretch sensors to the Adafruit conductive elastic were also researched, but none within budget were found to be suited for the hydro muscles. It is expected that within one or two years stretch sensors will become commercially available and economical for usage in the hydro muscle.

5.5.2 Water Resistivity

The resistivity of the water between the two ends of the muscle was also tested to determine length. The length was calculated using (4):

$$L = \frac{RA}{\rho} \tag{4}$$

However, through testing, it was found that this method was limited by a number of factors preventing accurate readings. One of the factors was that the measured resistivity was found to vary when anything came into contact with both metal fittings on the ends of the muscle. Replacing the endcaps and fittings with plastic parts greatly reduced these inconsistencies, but the muscles were still prone to outside noise. Also, the change in resistance is non-linear when the muscle is expanding radially.

5.5.3 Encoder

The latest length sensing method implemented the use of encoders. The KY-040 Rotary Encoder was used to test this idea. The small rotary encoder was attached to one end of the muscle, fishing line was looped around the shaft of the encoder, threaded through the muscle's sheathing, and attached to the other end. When the muscle extended and contracted the line would spin the encoder. The spinning of the encoder shaft outputs two square waves that are 90 degrees out of phase. The rising or falling of each square wave is referred to as an encoder tick. The encoder has a certain number of ticks per revolution so they can be used to determine position and direction of rotation based on the tick count and the output of the two waves. These encoder readings were used to calculate the position of the muscle. This method was found to be the most reliable as it is an entirely mechanical solution and not subject to hysteresis or creep. Attaching the encoder and the line, however, may become obtrusive in more streamlined applications.

To address this, the fishing line pulley system was replaced with a spring loaded reel to which the encoder was directly attached. This system allowed most of the mechanical components to be covered by a casing which improved the durability and robustness of the design. The reel also only required the fishing line to be threaded through the muscle a single time, reducing the amount of potential interference. One difficulty encountered with this method was that the torsion springs require a lot of tuning to provide enough force to retract the fishing line but also not cause the muscles to buckle. This was mostly resolved by using weaker torsion springs which were taken from a windup toy car, as shown in Figure 17. The weaker springs were used to replace the reel's original spring, causing the pulling force to be less.

Figure 17: Spring replacement. Left: original spring from reel. Right: spring from a toy car.

Like all mechanical switches, the ones inside the encoder are subject to bouncing. Each output of the encoder is debounced using a small RC filter. The filter was designed such that it wouldn't eliminate small ticks that occurred when the muscle was changing length rapidly. The output of the debouncing filter was then input to an inverting Schmitt Trigger, changing the signal back into a square wave and

phase shifting it 180 degrees. The phase shift makes it look like the encoder is rotating in the opposite direction, which is compensated for in the code.

5.6 Pressure Sensor

There are many methods of detecting force in a system. The ideal sensor for detecting force exerted by the muscle would have to be small enough to be mounted onto the muscle without hindering the operation of a muscle, i.e. not limiting its flexibility, elongation, or water flow while maintaining the muscle's durability. Additionally, in order to satisfy the requirement of modularity, the sensor should be easily removed from the muscle.

The force model of a muscle must also be considered. The return force applied by the elastic component of the muscle is equivalent to the change in length multiplied by the spring constant for the material. Opposing the elastic return force is the pressure of the internal fluid. The fluid pushes on the end of the muscle, creating a force proportional to the area of the muscle and the pressure. The sum of these two forces results in the net force exerted at the tip of the muscle. If the elastic force and the radius of the muscle are known, then the overall force can be calculated given knowledge of the internal pressure and length.

Measuring pressure was a simpler task than measuring length, a few Honeywell pressure sensors were tested to find the final sensor used. The first sensor tested was the Honeywells NBPDANN150PAUNV absolute pressure sensor. The muscles are connected to a 100 psi water pump so a sensor capable of operating at 150 psi would give a margin of safety if anything were to go wrong. Unfortunately the sensors would break after some use; something in the water/system may have affected the sensor as they were not meant to work with ionic or corrosive mediums (Basic Board Mount Pressure Sensors, 2014). To remedy this issue the sensors were switched to TBPDPNS100PGUCV gauge pressure sensor. The TBPDPNS100PGUCV has a gel coating applied over the electronic components that protects them from ionic and corrosive mediums. This sensor is rated to operate up to 100 psi and is still compatible with the water pump (Basic Board Mount Pressure Sensors, 2014).

The output from the pressure sensor is given in two voltages, V_+ and V_-, the difference of which relates to the gauge pressure. The exact pressure can be found using (5):

$$P = \frac{100\text{psi} * V}{V_s CF} \tag{5}$$

where V_s is the supply voltage given to the sensor, V is the difference between V_+ and V_-, P is the pressure, and CF is the calibration factor found on the device's datasheet under full scale span ("How To Measure Pressure with Pressure Sensors", 2012). Using a supply voltage of 5V and given a calibration factor of 6.1mV/V, the equation becomes:

$$P = 3.28 * V \quad (6)$$

V is on an order of millivolts, which will need to be amplified in order to increase precision in the ADC. The outputs of the pressure sensor are sent into an operational amplifier, which subtracts the two voltages and multiplies the difference by a gain of approximately 150.

5.7 Valves

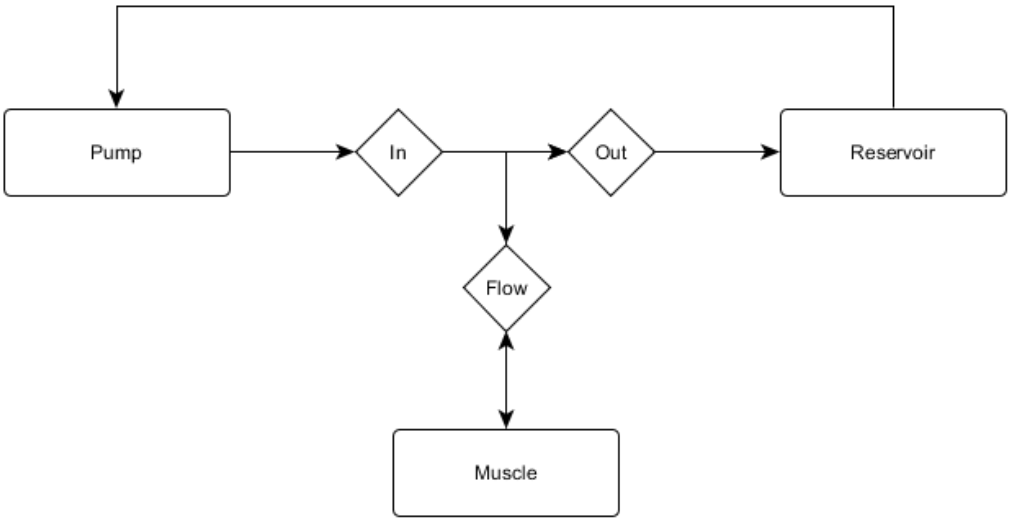

Figure 18: Water flow diagram

Each muscle is controlled by the opening and closing of a solenoid valve shown in Figure 18 as "In" and "Out", however, this only allows for on/off water flow. There will be a small delay between sending a signal to open the valves and when the valve actually opens. Once the valve opens, water will flow through the system as quickly as possible. When closing the valve there will be similar delays between sending the signal and the valve's response, causing additional water to flow through the system. A variable flow control valve, "Flow" in Figure 18, allows for control over the flow rate, which can reduce the error in the system and allow for more precise control.

The flow control valve was constructed from a ball valve with a servo connected to it. The servo and ball valve were secured together, so that they would not rotate relative to each other, and the shaft of the servo was connected to the handle of the ball valve. The servo is capable of turning the ball valve from fully closed to fully open in less than half a second. The response time of our valve system is about 100 ms. This is due to the time it takes for the solenoid valves to open, and the time it takes for the water in the tubing to start flowing. This delay still allows the system to fulfill the one second goal in Requirement 4.

5.8 Signal Conditioning

5.8.1 Sensor Circuits

5.8.1.1 Conductive Rubber

The conductive rubber measures length by changing resistance proportional to length. Resistance of the elastic was measured by connecting the elastic to a 5V power supply and another resistor as shown in Figure 19.

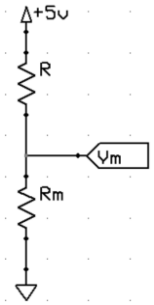

Figure 19: Conductive Elastic Voltage Divider

Rm and Vm represent the resistance of the elastic and voltage across the elastic respectively.

As mentioned in section 5.5.1, the resistance of R is approximately equal to the relaxed resistance of Rm and Vm varies over a range of 0.5V. In order to maximize the precision of data points the Arduino can detect as the elastic stretches, the voltage Vm is sent to the positive input of the operational amplifier circuit shown in Figure 20.

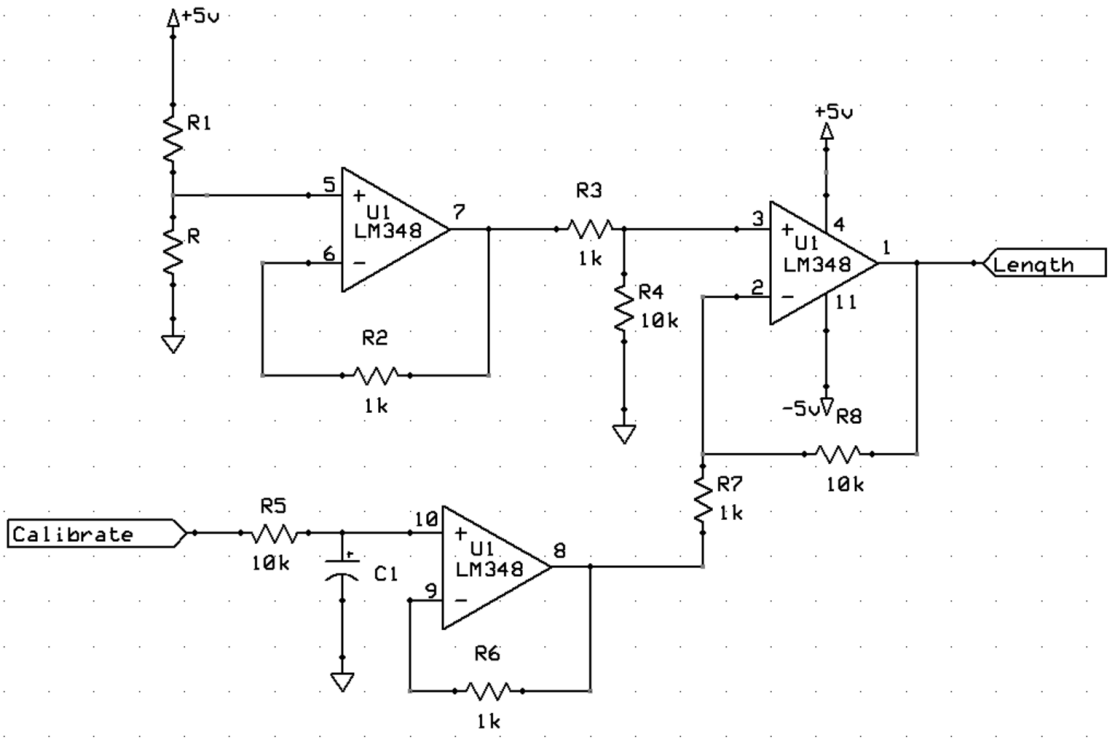

Figure 20: Conductive Elastic Measurement Circuit

An LM348 chip is used to construct the implementation amplifier, which maps the voltage across the muscle to a range of 0V to approximately 4.5V. Despite being powered with 5V, the LM348 chip saturates at a slightly lower voltage due to the internal circuitry of the chip. An analog calibration value was sent from the Arduino to the negative input of the circuit. In order to output an analog voltage, the Arduino outputs a pulse width modulation (PWM) signal, which is then filtered by the first order lowpass filter created by C1 and R5. The voltage across the muscle in its resting state after being expanded and contracted once is used as the calibration voltage. Doing this ensures that the voltage output of the circuit will be approximately 0V when the muscle in its resting state and was necessary due to the conductive elastic's hysteresis.

The voltage across the muscle and the calibration voltage from the filter are sent through unity gain buffers, which output the voltage passed to the positive terminal, but with approximately the same impedance. Outputs from the buffers are sent into the third operational amplifier which outputs a voltage V_O proportional to the length of the conductive elastic. Derivation of the equation relating output voltage, V_O to the voltage across the muscle, $V_{m,}$ and the calibration voltage, V_c, is shown below.

$$0 = V_m\left(\frac{R3}{R3+R4}\right) - \left(V_c\left(\frac{R8}{R7+R8}\right) + V_O\left(\frac{R7}{R7+R8}\right)\right) \tag{7}$$

Solving (7) for V_O gives:

$$V_O = \left(\frac{R7 + R8}{R7}\right)\left(V_m\left(\frac{R3}{R3 + R4}\right) - V_c\left(\frac{R8}{R7 + R8}\right)\right) \quad (8)$$

Plugging the values of R3, R4, R7, and R8 and simplifying yields the final equation:

$$V_O = 10(V_m - V_c) \quad (9)$$

The output voltage V_O was read on an analog input pin of the Arduino and used to calculate the stretched length of the muscle. Plugging (1) into (9) puts V_O in terms of the resistance of the conductive elastic:

$$V_O = 10\left(V_s\left(\frac{R_s}{R_s + R}\right) - V_c\right) \quad (10)$$

Solving for the stretched resistance, R_S, gives:

$$R_s = R\left(\frac{0.1V_O + V_c}{V_s - (0.1V_O + V_c)}\right) \quad (11)$$

The stretched resistance, R_S, now needs to be related to the stretched length of the conductive elastic. Ideally, the ratio of stretched to relaxed resistance should be the same as the ratio of stretched to relaxed length. Manipulating that ratio to solve for the stretched resistance gives:

$$R_s = \frac{RL_s}{L} \quad (12)$$

R represents the relaxed resistance of the elastic and L_S and L are the stretched and relaxed length of the elastic respectively. Plugging these values into (11) and solving for L_S relates the stretched length of the conductive elastic to the voltage output by the instrumentation amplifier.

$$L_s = L\left(\frac{0.1V_O + V_c}{V_s - (0.1V_O + V_c)}\right) \quad (13)$$

Conductive elastic was connected to the muscle such that changes to the length of the muscle were the same as changes in length of the elastic. This allows for calculation of the muscle's length when using the initial length of the muscle for the value L. (13) was used by the Arduino to calculate the muscle's current length.

5.8.1.2 Pressure Sensor

Output from the pressure sensor was conditioned similarly to the conductive elastic; two inputs were sent to an instrumentation amplifier and the difference between the two was amplified.

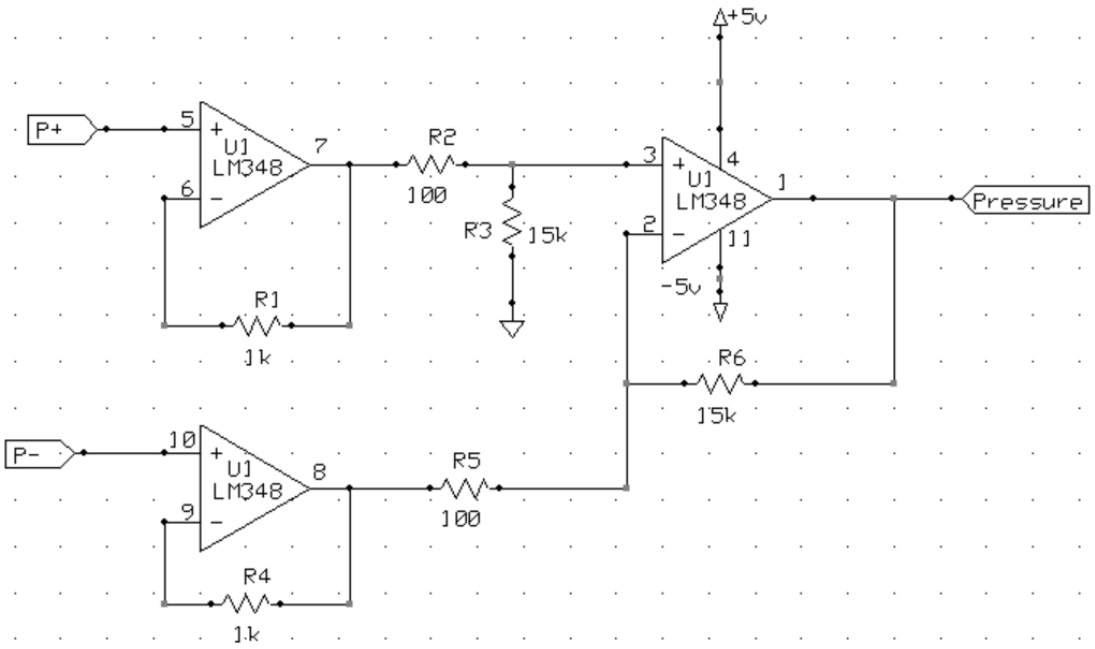

Figure 21: Pressure Sensor Conditioning Circuit

Figure 21 shows the instrumentation amplifier constructed using the LM348 chip. Two outputs from the pressure sensor, P+ and P-, are sent into the two inputs of the amplifier. The required gain of the circuit varied depending on the specific pressure sensor being used. Figure 21 shows the circuit connected with a gain of 150 for the final pressure sensor used.

Using (6), it was calculated that the pressure sensor would output a voltage between 0mV and 30.5mV across the P+ and P- pins. By using 30.5mV as the value of (V_+ - V_-) and 5V as the value of V_{out} in **Error! Reference source not found.**, it was calculated that a maximum gain of 164 was required. Due to he limitations on available resistors, a gain of 150 was used, limiting the maximum output voltage to 4.575V.

The equation relating the output of the amplifier to the pressure being measured is derived below using the gain and (6) solved for the input voltages.

$$(V_+ - V_-) = 0.305P \tag{14}$$

Plugging this and the gain into (3) yields:

$$V_{out} = 45.7P \tag{15}$$

Here, V_{out} is expressed in millivolts and P represents the internal pressure in the muscle measured by the sensor. Scaling V_{out} to be represented in volts and solving for P gives the final equation:

$$P = 21.9V_{out} \tag{16}$$

(16) is used by the Arduino to calculate the pressure based on reading from the ADC.

5.8.1.3 Encoder

In order to accurately read the length from the quadrature encoder, each output signal needed to be debounced and filtered for noise. Output signals from the CLK and DIR pins were each passed through an RC lowpass filter and an inverting Schmitt trigger to ensure a clean signal would be sent to the Arduino for processing.

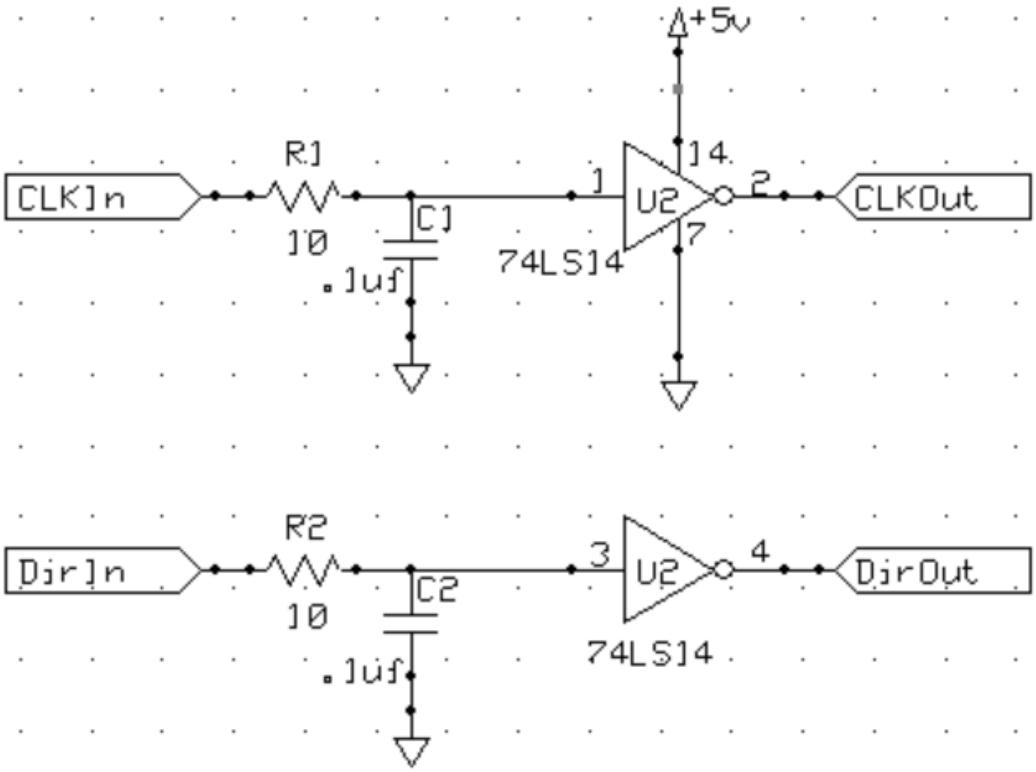

Figure 22: Encoder Signal Conditioning Circuit

Figure 22 shows the circuit used to filter and reconstruct the digital signal output by the encoder. Resistors R1 and R2 are connected in series with C1 and C2 respectively to form an RC lowpass filter. The filter was designed to debounce the switches inside the encoder by reducing the amplitude of the high frequency noise created when opening or closing switches. Each filter has a cutoff frequency of 160KHz. This frequency needed to be relatively high to ensure that no encoder ticks were filtered out when the muscle's length was changing rapidly. Despite inputting a digital signal into the filter, the output voltage measured across the capacitor will be an analog signal. This signal needs to be converted back into a digital signal so that the Arduino can easily determine a logic high from a logic low.

In order to recreate a digital signal, the filtered signal measured across the capacitor is sent to a Schmitt trigger. The Schmitt trigger uses positive feedback to force the output voltage to its maximum or minimum values. The circuit in Figure 22 uses the 74LS14 inverting Schmitt trigger. The 74LS14 has a

positive-going threshold voltage of 1.7V, a negative-going threshold voltage of 0.8V and a maximum output voltage of 4V at 5V supply voltage ("Schmitt Triggers Dual Gate/Hex Inverter"). When the input to the Schmitt trigger rises past the positive threshold, the output is forced low and when the input drops below the negative threshold the output is forced high. Output of each Schmitt trigger is read and processed by the Arduino. Due to the inversion of the signal by the Schmitt trigger, however, the encoder appears to be rotating in the opposite direction to the Arduino. This issue is fixed in the code.

5.8.2 Valve Control

The system uses two solenoid valves to permit the direction of the flow of water through the muscle and a flow control valve to regulate the flow rate. The flow control valve is controlled by a PWM signal sent from the Arduino, which can be sent directly to the servo motor that controls the valve. The solenoid valves are also controlled by signals from the Arduino, however, they require 12V excitation to open. The Arduino is only capable of outputting 5V.

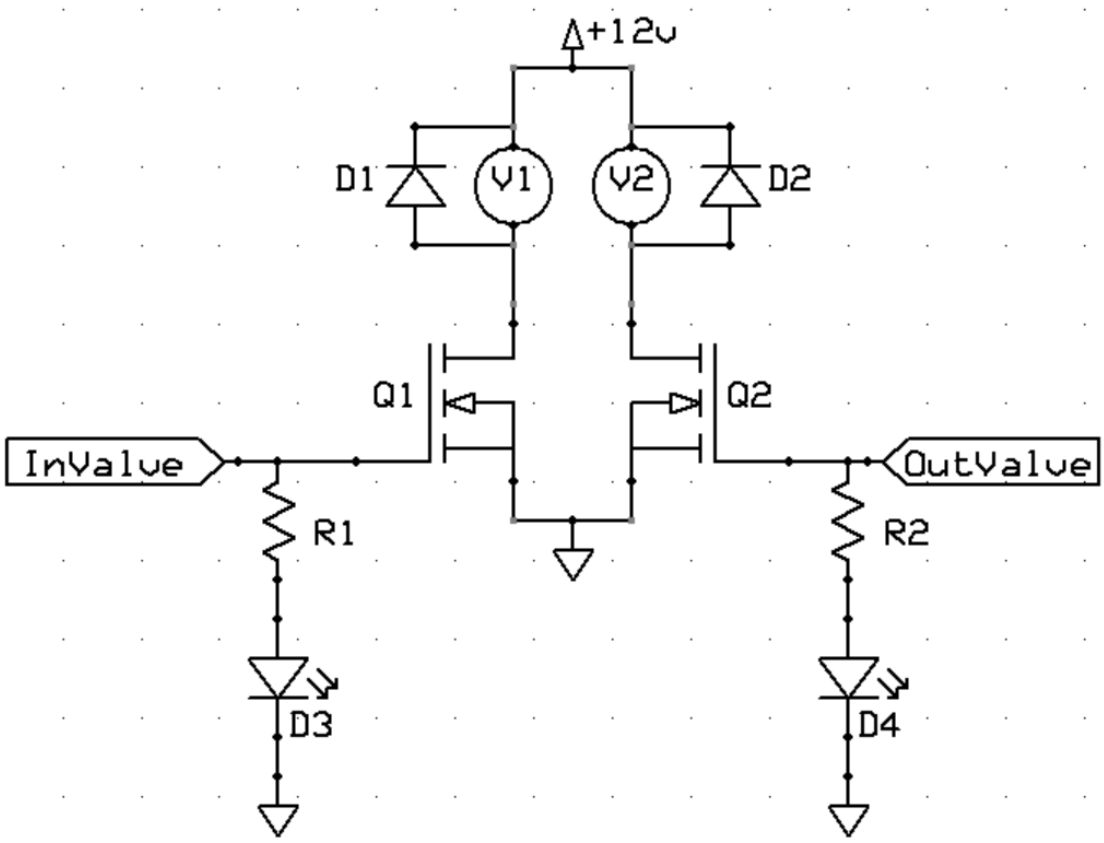

Figure 23: Valve Control Circuitry

Figure 23 shows the circuit used to control the two solenoid valves. V1 and V2 represent the valves, Q1 and Q2 are IRF520 n-channel MOSFETS, and the ports labeled "InValve" and "OutValve" are the signals sent from the Arduino. Each valve is connected to the drain of its corresponding MOSFET, the sources of the MOSFET are connected to ground and control signals are sent to the each MOSFET's gate. When the

Arduino sends 0V to the MOSFET, there will be no current flowing from the drain to the source, acting as an open circuit. In this state there will be a voltage drop of 12V across the MOSFET and the valve will be closed. When a 5V signal is sent to the gate, the MOSFET will allow current to flow from drain to source, opening the valve.

While the solenoid is energized it will be accumulating energy which will need someplace to go after the valve has closed. Diodes D1 and D2 are being used as flyback diodes, which can dissipate the energy stored in the solenoids without damaging other components in the circuit. When the MOSFET turns on the diode will be reverse biased, prohibiting the flow of current through it, however, when the MOSFET turns off, the valve and power supply will be disconnected from ground. Now current will flow in a loop through the diode and the valve until the excess energy has dissipated.

Resistors R1 and R2 and LEDs D3 and D4 were added to provide visual indication when a 5V signal was being sent to the MOSFET. D3 emits green light to indicate that water is flowing into a muscle and D4 emits red light to indicate that water is flowing out of the muscle. R1 and R2 are 10KΩ resistors to limit the current passing through the LEDs to prevent them from burning out.

5.8.3 Printed Circuit Boards

During the testing of various iterations of muscle design, all of the circuitry was wired on a breadboard. Once all circuit designs were finalized, a printed circuit board was designed so that all of the circuits needed to control and measure all aspects of a single hydro muscle were contained in one location. Express PCB's schematic and PCB layout software were downloaded and used to design a PCB for the muscle.

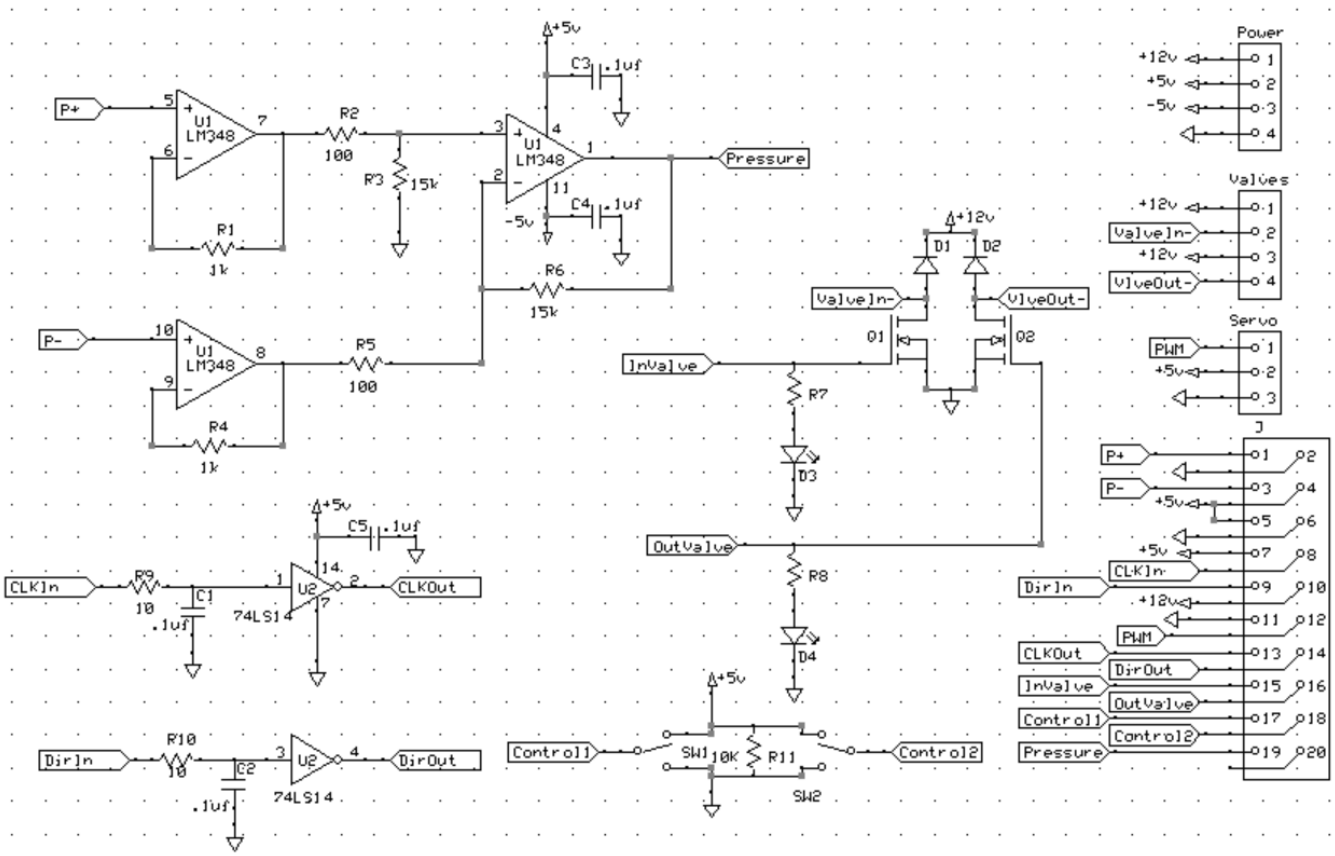

Figure 24: PCB Circuit Schematic Diagram

Figure 24 shows the circuit diagram used to design the PCB. The diagram contains a circuit for length sensing, pressure sensing, valve control, and control method selection. A 0.1µF capacitor was placed at the connection at the power pin of each chip used to protect the chip from power surges. Two switches were added to act as control method selection. Changing the state of the one switch tells the Arduino whether to control muscle length or force and changing the other tells the Arduino whether to record positions or play them back. Finally, connectors were added to allow for external components such as the valves, power supply, sensors, etc. to be connected to the PCB.

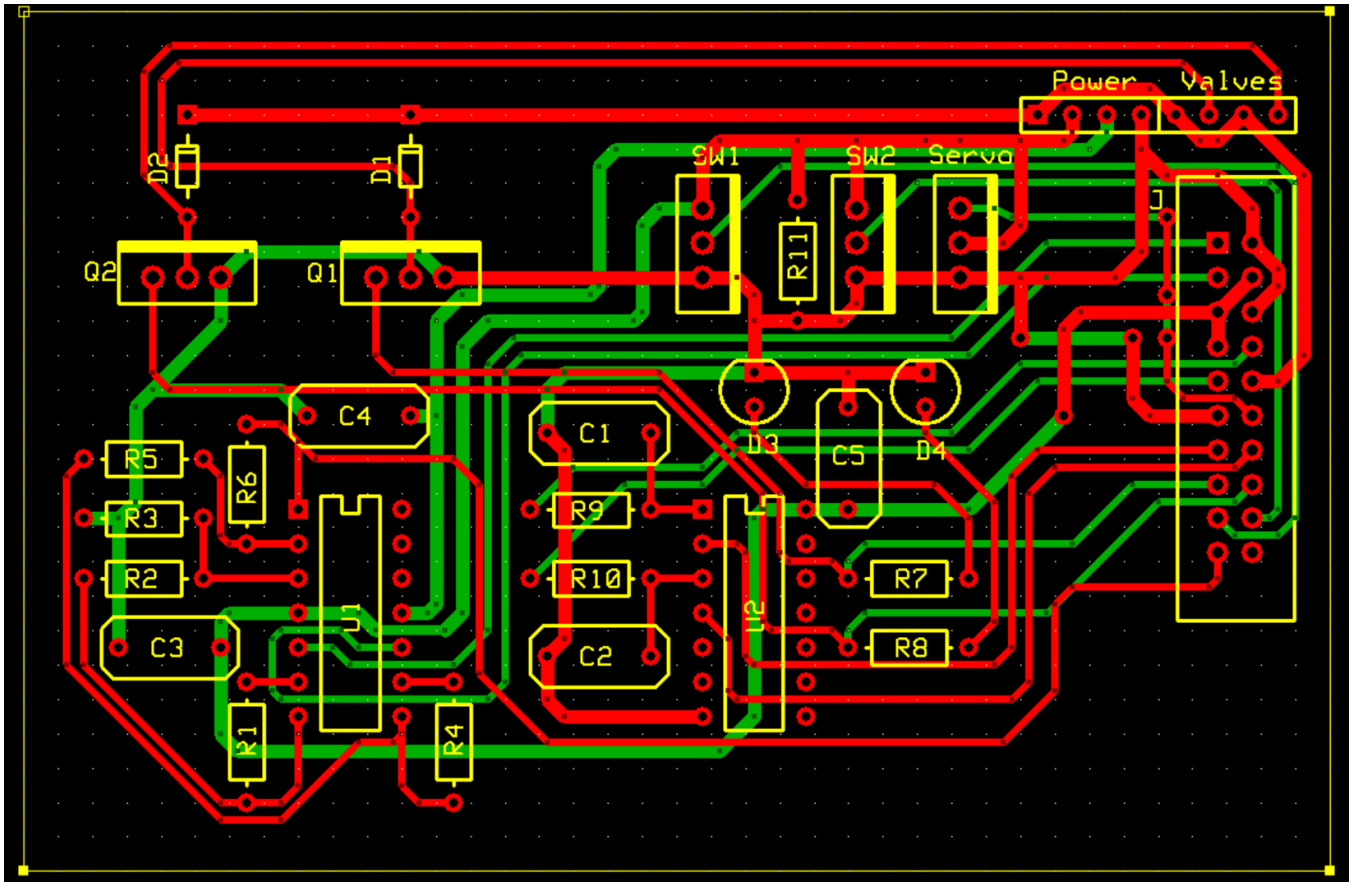

Figure 25: Final PCB Layout

Figure 25 shows the final layout of the PCB with all components outlined in yellow. Red traces are printed on the top layer of the board and green on the bottom. Power and ground connections were drawn to be 0.04 inches wide and all other connections were drawn to be 0.02 inches.

5.9 Controls

The platform for the control system needed to be able to read analog signals from the sensors and output voltages to control the valves. For this, an Arduino Uno was chosen. The Arduino has a 10bit ADC on it, to read the analog signals. The Arduino can output voltages, however it cannot output enough current to run the valves, so MOSFETs were used to transform the signal from the Arduino and send more current to the valves. The code to determine the current length and force based on sensor input runs on the Arduino. A block diagram showing the control algorithm running on the Arduino is shown in Figure 26. The Arduino is used to calculate the valve positions using PID control. The set point is sent to the Arduino over the serial port. Originally, the system was required to have an accuracy of 0.125 in, from Requirement 2. Due to the resolution of the encoder, 0.0633 in, it was decided to increase the tolerance to 0.15 in, because a 0.125 in tolerance only allows for the system to be off by 1 encoder tick, which would be difficult to achieve within the 1 second time allowed by Requirement 4.

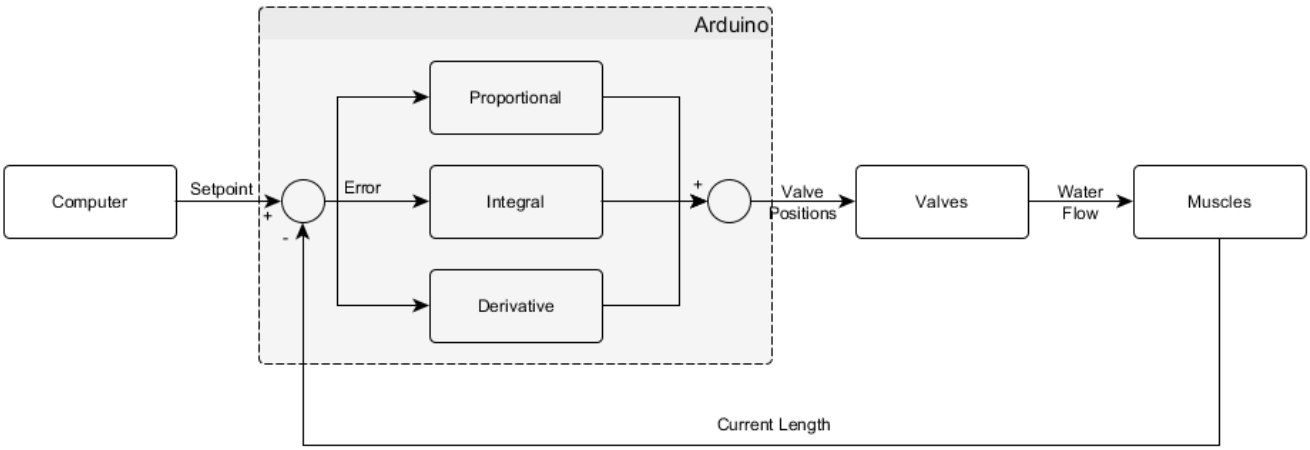

Figure 26: PID block diagram

5.9.1 PID Tuning

The PID loop on the Arduino runs every 11 ms. The PID control of the system was manually tuned. The optimal set of gains was found to be P = 200, I = 0, and D = 120. Converting from values in the code to real units, P = 120 deg/in, I = 0 deg*in/s, and D = 0.82 deg*s/in. This allowed for the system to respond quickly enough (within 1 second, Requirement 4) to set point changes, have small enough steady state error (less than 0.15 in), and not oscillate around the set point. Selected combinations of gains and their step responses are shown in Figure 27 below. The optimal set of gains is shown in the bottom right. Using a higher P gain resulted in a much longer settling time (middle right), and a lower P resulted in increased rise time (lower left). Using a higher D gain resulted in increased rise time (middle left), and a lower D resulted in increased settling time (upper right). Using an I gain resulted in much more overshoot and an increase in settling time (upper left).

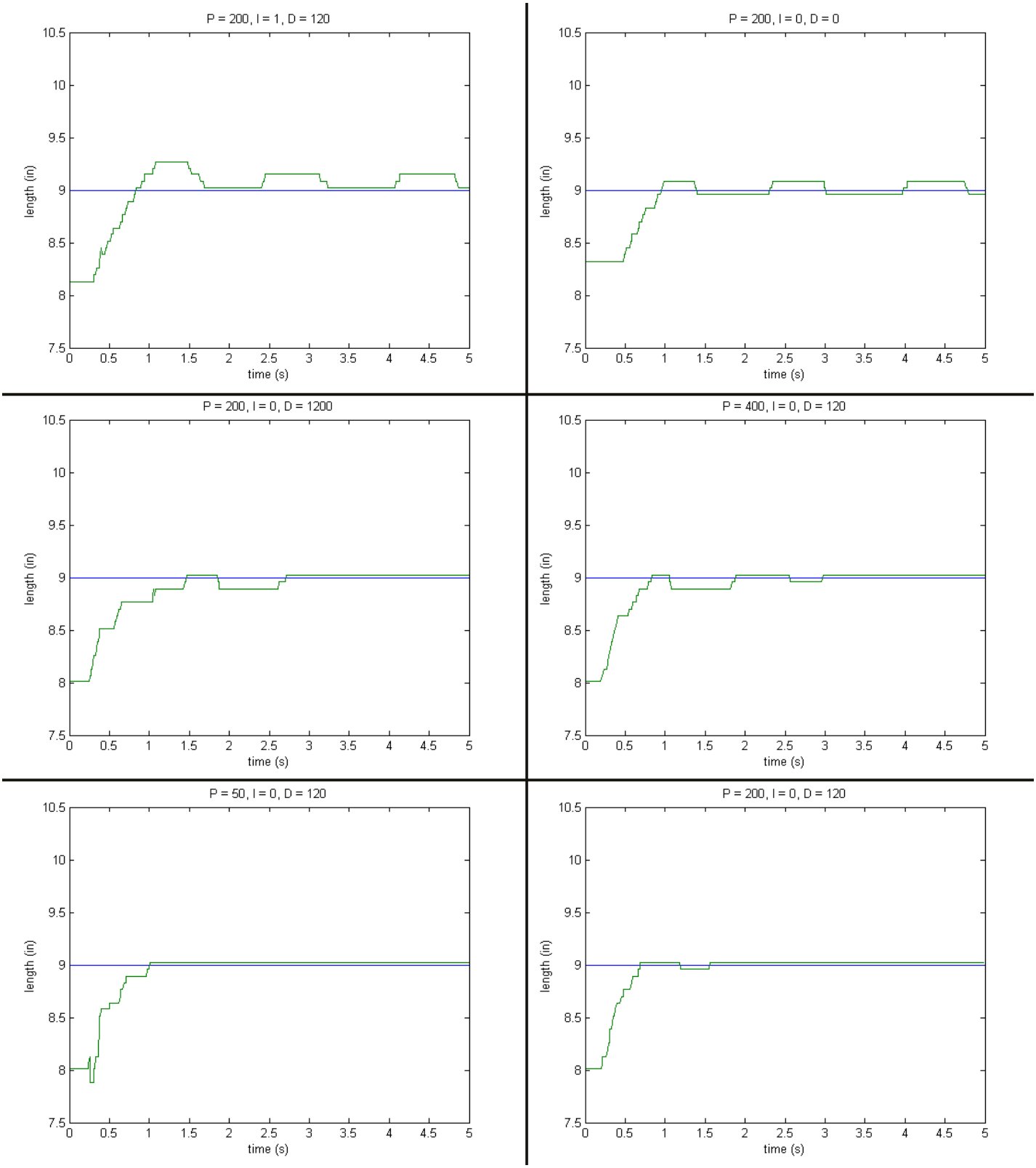

Figure 27: Responses of the system when the goal was changed from 8 in to 9 in, with different PID values.

The PID control system was able to meet the requirements of moving the muscle to the set point with an accuracy of at least 0.15 inch in less than 1 second.

5.9.2 Length Control

The system used PD control to control the position of the muscle. The output of the PD control was the angle to turn the ball valve to, with 0 degrees being fully closed, and 90 degrees being fully open. Positive values indicate that water should flow into the muscle, and negative values indicate that water should flow out of the muscle. The direction of flow was controlled by opening either the inlet or outlet valve, and the rate of flow was controlled by adjusting the amount the ball valve was open.

The system closes both the inlet and outlet valve when the muscle is close to the goal location. This prevents the muscle from jittering back and forth around the set point. The system also reduces the flow rate if it is repeatedly crossing the set point. This helps to reduce the amplitude of the overshoot if the system is underdamped. The flow rate is decreased by 17.7% each time it crossed the same set point again, resulting in much lower overshoot. This decrease is reset each time the set point changed, to allow the system to quickly reach the new set point, even if it was oscillating around the previous one.

5.9.3 Force Control

For force control, the system again used PD control, but with different values of P and D than when doing length control. The system calculates the force the muscle is outputting based on the length and pressure readings.

5.9.4 Record and Playback

To record, the Arduino will output the current length or force over the serial port. The recording is saved to a computer connected to the Arduino. This makes it easy for users to store and share the recording files. The computer is also used when manually inputting desired positions and times. The computer calculates a smooth curve going through all the given points, and sends the incremental set points to the Arduino.

During playback, the computer reads the selected file, and sends the set points to the Arduino over the serial port. The Arduino can be connected either to another Arduino, or to a computer. A Matlab program was written so that the user can enter desired lengths and times (or forces and times), and the program will generate a smooth curve of set points and send those to the Arduino over a USB connection. It can also record the position (or force) of the muscle and save the results to a file on the computer. This file can then be played back to make the muscle repeat the same motion.

5.9.5 Matlab Simulation

A Matlab simulation of the arm brace was created to provide a visual of the real-time data received from the sensors. A picture of the simulation is shown in Figure 28 below. The simulation shows the position of the arm brace (black), and of the muscle (green). The length and pressure of the muscle are displayed below the drawing.

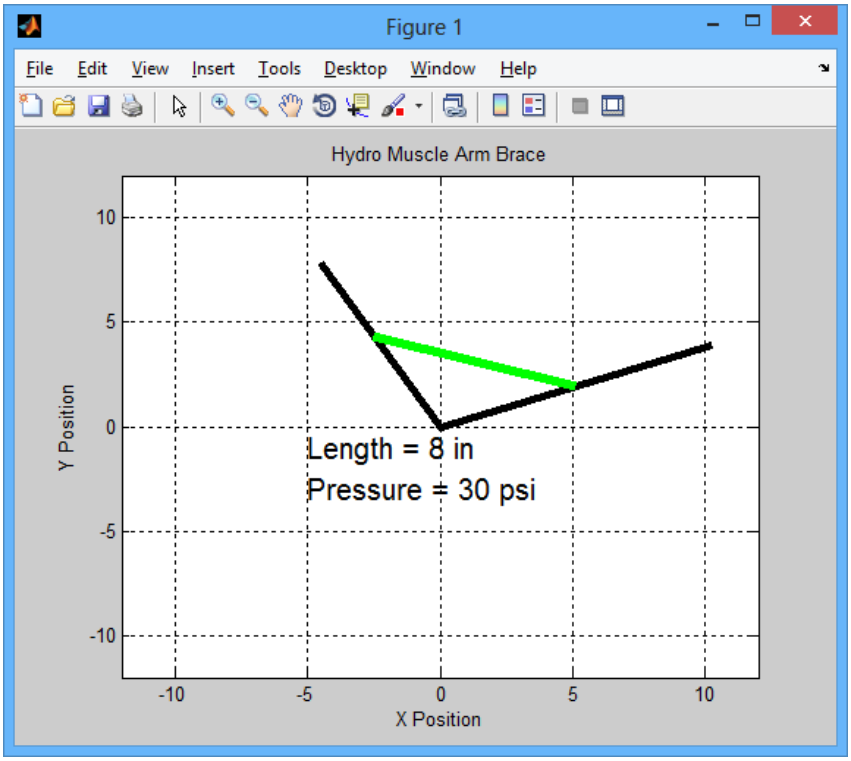

Figure 28: Arm Brace Simulation

5.10 Experiments

When designing a control system, it is important to balance both accuracy and efficiency. While a system may be infinitely accurate, it would not be practical if it takes too long to obtain that accuracy. Similarly, a system that reacts quickly may not be ideal if it must sacrifice accuracy. The control system for the muscles must be tested to ensure that it can operate accurately and in a time efficient manner.

5.10.1 Position Control Experiment

The muscle was tested to ensure that it could accurately and quickly extend to a desired length. To do this, one end of the muscle was secured to the table. The other end was affixed to a slide, constraining the muscle to only extend linearly. Fishing wire was wound once around the shaft of a KY-040 encoder and affixed to the table at either ends of the muscle. Before giving the muscle a desired position the input valve was manually opened, allowing the muscle to pressurize and stretch from its initial length of 16.51cm (6.5in) to 20.32cm (8in). After reaching this position the muscle was given a desired position of 22.86cm (9in). As the muscle extended, its length readings were recorded and plotted versus time.

For length control, a P value of 17.7 deg/cm and a D value of 0.11 deg*s/cm were used. The muscle would stop moving when it got within 0.38 cm of the target position.

The position control of the muscle was successful, and the muscle was able to reach the goal in about half a second. This shows that fine position control of hydro-muscles is possible with simple and cost effective sensors.

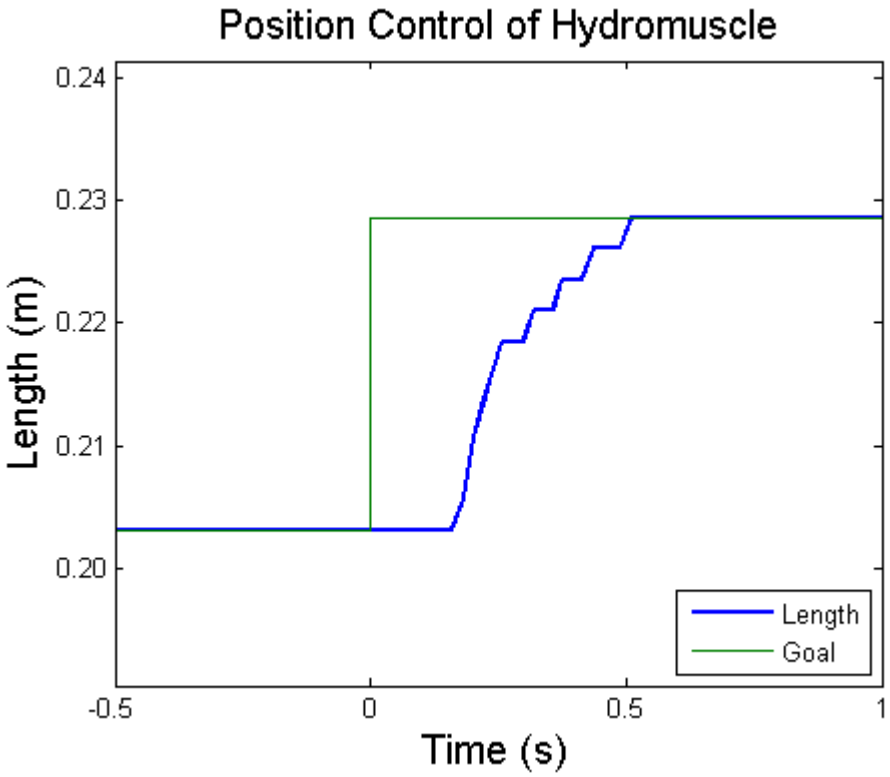

Figure 29: Length vs Time results. The steps are due to the resolution of the encoder that was used (0.1 in resolution).

5.10.2 Force Control Experiment

In order to properly implement force feedback, the force characteristics of the muscle must be known. To test the muscle's ability to match forces the muscle was stretched to 38.5% strain (22.86cm or 9in) and affixed to the table in order to prohibit further changes in length. Keeping the muscle at a fixed length ensures that the elastic force provided by the latex tubing would remain constant while the internal pressure changes, varying the net pulling force applied by the muscle. The force was recorded using a Vernier Dual Range Force Sensor. The muscle was given a goal force, and the force the muscle applied over time while trying to reach the goal was recorded.

For force control, a P value of 4.5 deg/N and a D value of 0.84 deg*s/N were used. The muscle would stop pressurizing when it got within 0.25 N of the target force. When measuring the force, our readings were noisy, due to shockwaves running through the water when the valves were opened or closed. To compensate for this, the signal was passed through a low pass filter with a cutoff frequency of approximately 1.5 Hz.

The force control test was also successful. The muscle took about half a second to get within 1 Newton of the goal force.

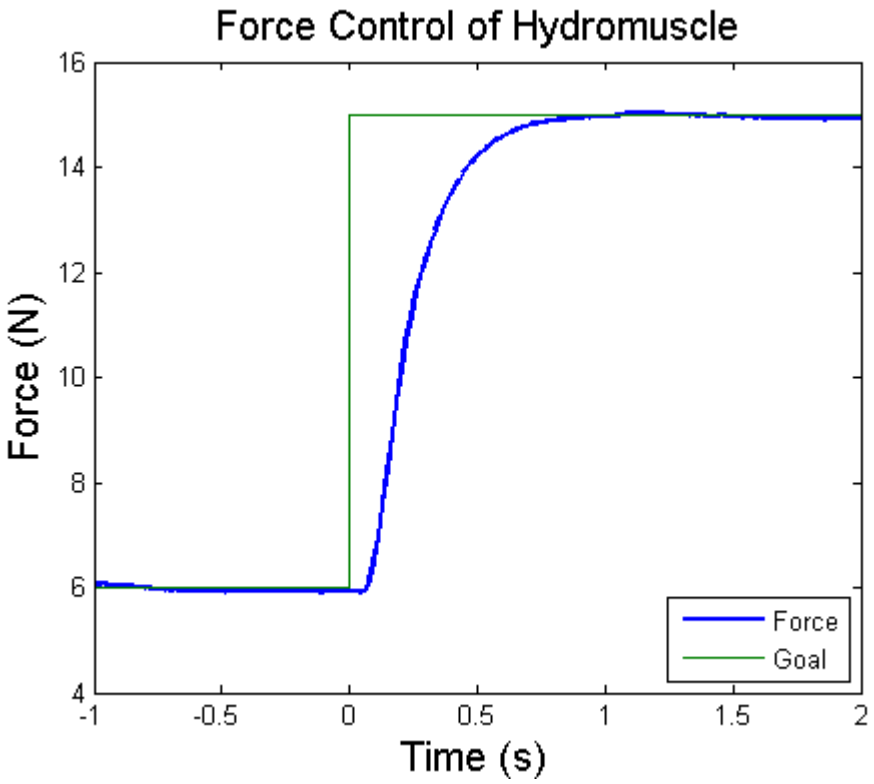

Figure 30: Force vs Time results

5.10.3 Conductive Rubber Testing

To determine if the conductive rubber is accurate enough for our needs, tests were run on it to test for hysteresis and repeatability.

5.10.3.1 Repeatability

The resistance of an 8 in long piece of conductive rubber was measured while at rest. The rubber was then stretched to its maximum length of 12 in (50% strain), and then brought back to a resting position. The resistance of the rubber was recorded, and then the rubber was stretched again. This was repeated 10 times. The resistances are shown in Table 2 below.

trial	resistance (Ω)
resting	634
1	996
2	1012
3	1030
4	1038
5	1020
6	1011
7	1097
8	982
9	1033
10	1003

Table 2: Resistance of Rubber at 8 inches

The highest resistance seen, 1097 Ω, was 115 Ω higher than the lowest resistance seen, 982 Ω. The conductive rubber only changes by about 400 Ω when the muscle is fully extended, so a difference of 115 Ω is very significant. The standard deviation of the resistances measured was 31.5 Ω. These results showed that the conductive rubber was not very repeatable, since the readings were so different for different trials.

5.10.3.2 Hysteresis

To test the conductive rubber for hysteresis, an 8 in long piece of conductive rubber was stretched from resting to maximum length of 12 in, and then brought back to its resting length. The rubber was stretched at a rate of 0.5 in per second, taking a total time of 16 seconds to stretch the rubber and then bring it back. A graph of the resistances vs length is shown in Figure 31 below.

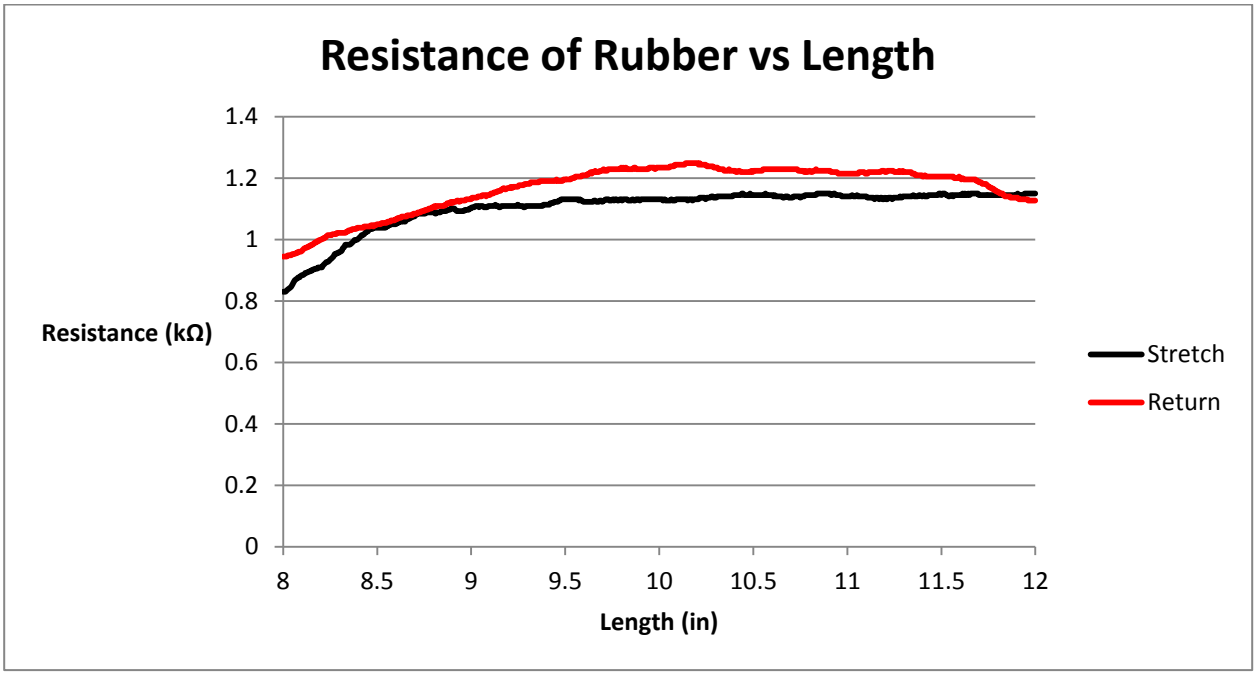

Figure 31: hysteresis of conductive rubber. The black line is while extending, the red line is while contracting

As seen in Figure 31, the conductive rubber has major hysteresis. In order to accurately determine position with this much hysteresis, the controller would need to implement much more complex algorithms, such as an extended Kalman filter, to be able to determine where the muscle is with any amount of accuracy.

5.10.3.3 Settling time

To test the settling time of the conductive rubber, an 8 in piece of conductive rubber was stretched from 11 in to 9 in, and then the resistance was measured over time. The result is show in Figure 32 below.

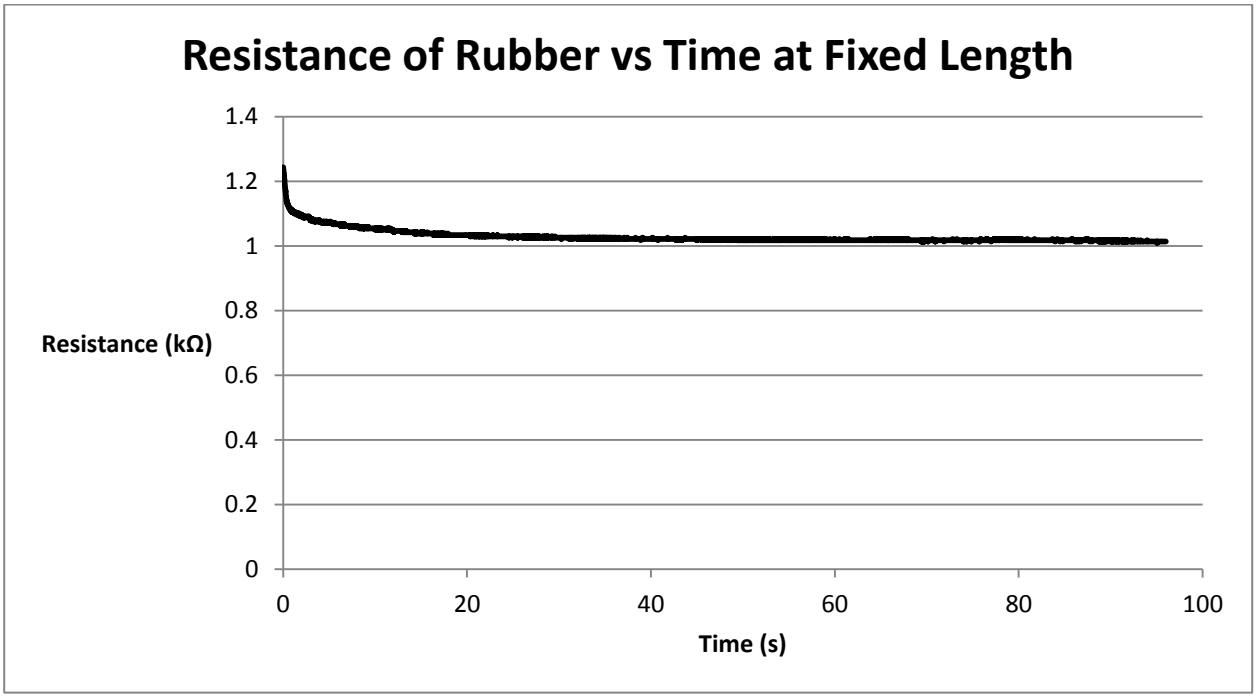

Figure 32: resistance of rubber over time at constant length (9 in)

The rubber took 6.5 seconds to settle to within 5% of the final value. This is far too long, as our control needs to reach its set point within 1 second to satisfy Requirement 4.

5.10.4 Water Resistivity Testing

To determine if the water resistance is accurate enough for our needs, tests were run on it to test for hysteresis, repeatability, and settling time.

5.10.4.1 Repeatability

The resistance of the water in an 8 in long muscle was measured while at rest. The muscle was then inflated to its maximum length of 12 in (50% strain), and then brought back to a resting position. The resistance of the water was recorded, and then the muscle was stretched again. This was repeated 10 times. The resistances are shown in Table 3 below.

trial	resistance (kΩ)
resting	360
1	100
2	280
3	370
4	600
5	620
6	632
7	598
8	502
9	586
10	585

Table 3: Resistance of Water in 8 in muscle

The highest resistance seen, 632 Ω, was significantly more than the lowest resistance seen, 100 Ω. The standard deviation of the resistances measured was 179 Ω. These results showed that the resistance of the water was not very repeatable, since the readings were so different for different trials.

5.10.4.2 Hysteresis

To test the water for hysteresis, an 8 in long muscle was stretched from resting to maximum length, and then brought back to its resting length. The muscle was stretched at a rate of 4 in per second, taking a total time of 2 seconds to stretch the muscle and then bring it back. A graph of the resistances vs length is shown in Figure 33 below.

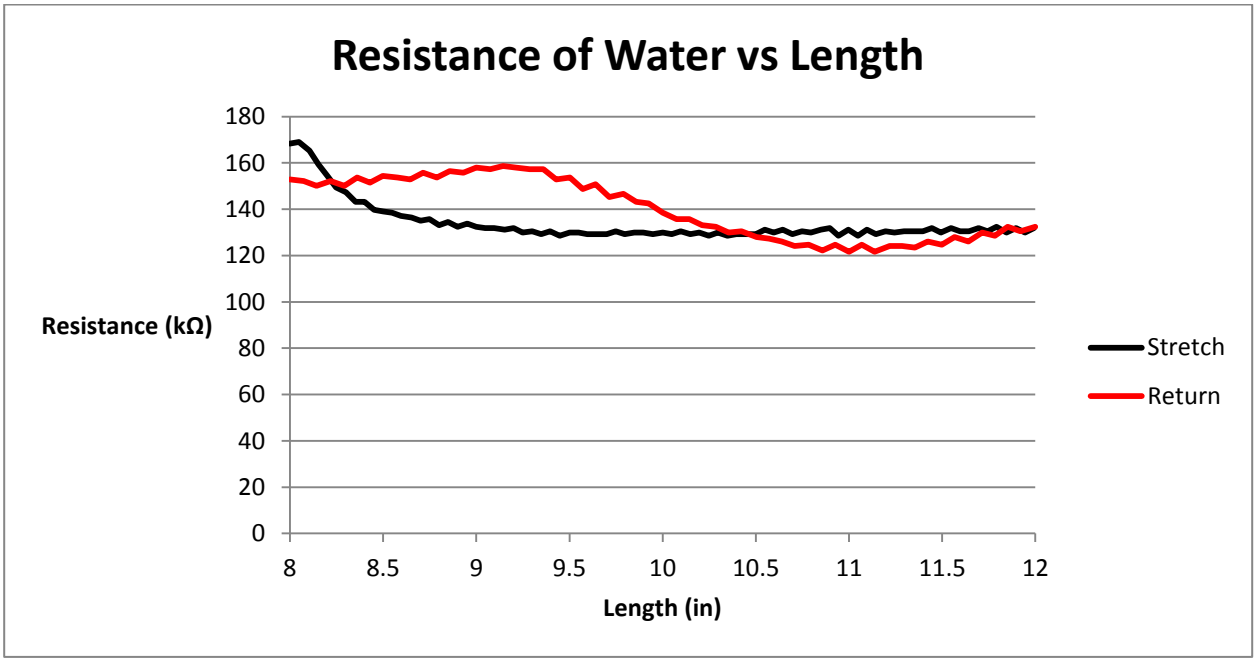

Figure 33: hysteresis of water resistance in the muscle. The black line is while extending, the red line is while contracting

As seen in Figure 33, the water has some hysteresis. This will make it difficult to determine where the muscle actually is based on the resistance of the water. This may be caused by the radial expansion of the tubing that occurs while the tubing is also expanding longitudinally.

5.10.4.3 Settling time

To test the settling time of the water resistance, an 8 in muscle was stretched from 12 in to 8 in, and then the resistance was measured over time. The result is show in Figure 34 below.

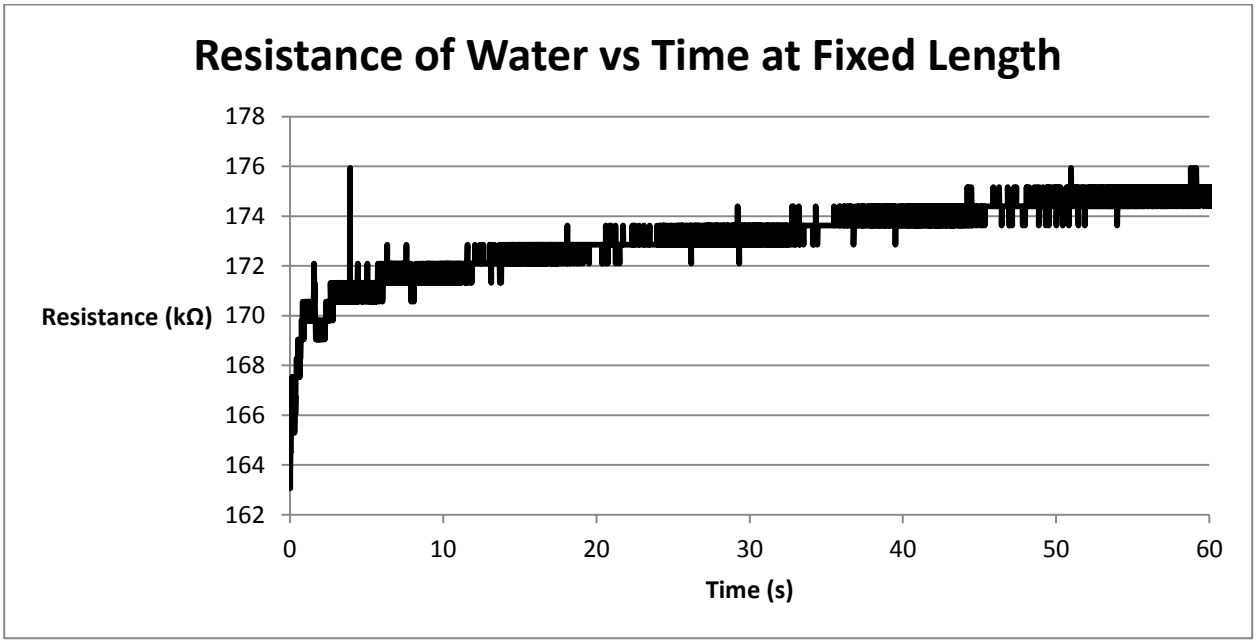

Figure 34: resistance of water over time at constant muscle length

The readings are a bit noisy, but that could easily be solved by filtering. The settling time of the water resistance was not too bad. It got within 5% of the final resistance in about half a second. This settling time may present problems when trying to be very precise, or trying to move very quickly.

5.10.5 Encoder Testing

To determine if the encoder is accurate enough for our needs, tests were run on it to test for hysteresis, repeatability, and settling time.

5.10.5.1 Repeatability

The number of ticks received from a 7.5 in long muscle was measured while at rest. The muscle was then stretched to 10.5 in, and then brought back to a resting position. The number of encoder ticks was recorded, and then the muscle was stretched again. This was repeated 10 times. The numbers of ticks is shown in Table 4 below.

trial	encoder ticks
resting	0
1	0
2	0
3	1
4	0
5	0
6	0
7	0
8	0
9	0
10	0

Table 4: Encoder Ticks in 7.5 in muscle

The encoder is very repeatable, and the error of 1 tick that occurred in trial 3 is still accurate enough to satisfy Requirement 2.

5.10.5.2 Hysteresis

To test the encoder for hysteresis, a 7.5 in long muscle was stretched from resting to 10.5 in, and then brought back to its resting length. The muscle was stretched at a rate of 2 in per second, taking a total time of 3 seconds to stretch the muscle and then bring it back. A graph of the ticks vs length is shown in Figure 35 below.

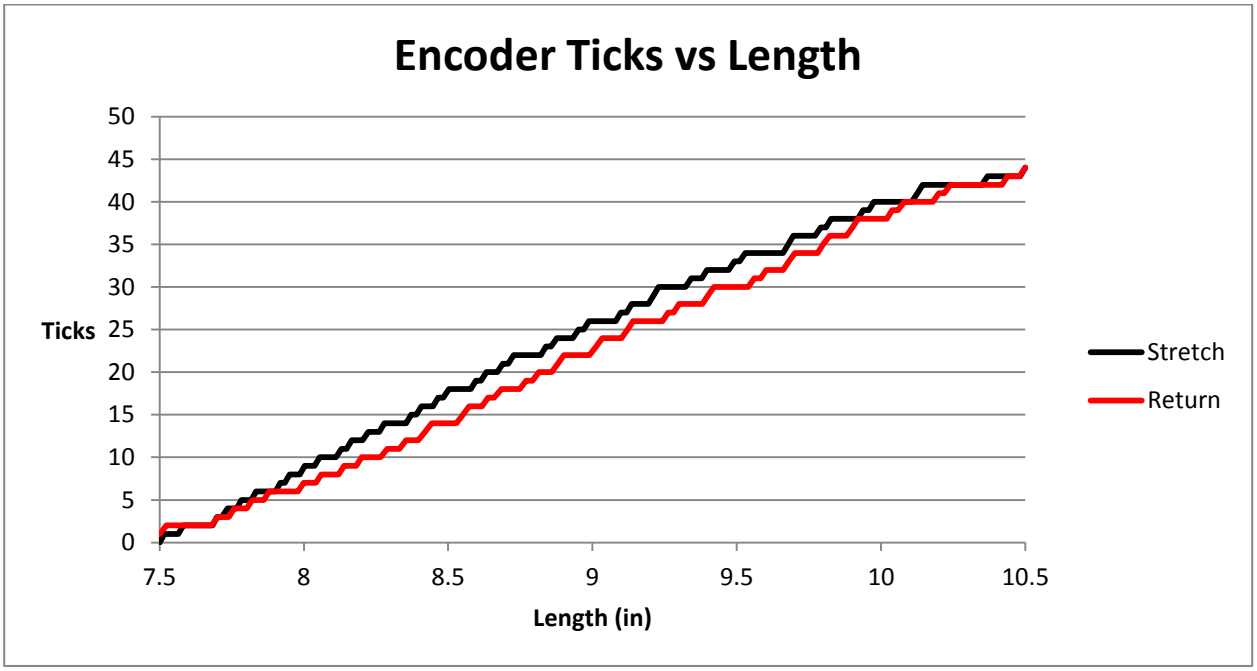

Figure 35: hysteresis of encoder. The black line is while extending, the red line is while contracting

As seen in Figure 35, the encoder has very little hysteresis. This makes it very easy to use the encoder to determine the actual length of the muscle. In theory, an encoder should have no hysteresis at all. The small hysteresis observed may be due to small errors while testing, or due to hysteresis in the return spring inside the retractable reel.

5.10.5.3 Settling time

To test the settling time of the encoder, a 7.5 in muscle was stretched from 10.5 in to 7.5 in, and then the encoder ticks were measured over time.

The encoder has practically no settling time. The encoder reached the final value within 0.1 seconds, well within Requirement 4. This makes it good for our application, as it will immediately update to reflect the correct length of the muscle, without adding extra delays.

5.10.6 Teleoperation testing

To test our teleoperation, two of the arm braces were used. One brace was the master, and the other was the slave. The master brace was moved up and down several times, and the positions of both the master and slave were recorded. Figure 36 shows the positions of both the master and slave.

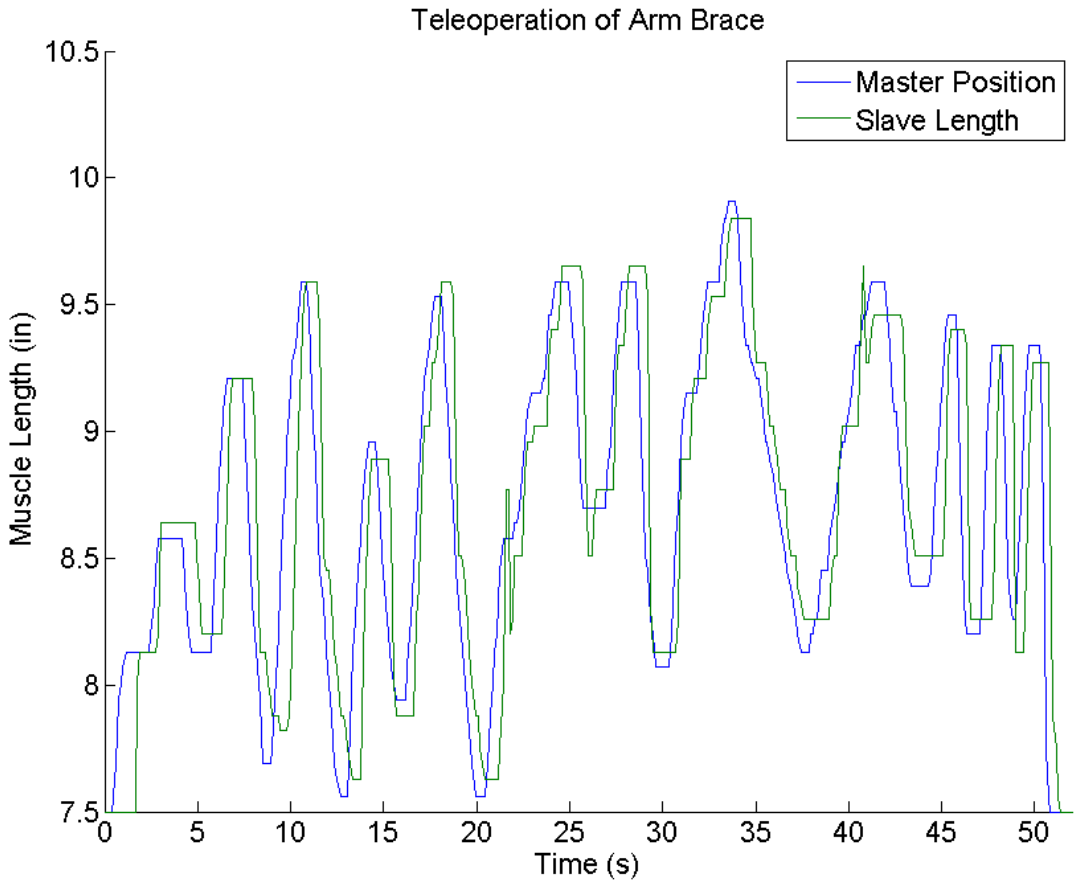

Figure 36: lengths of master and slave muscles. The master is in blue and the slave is in green.

The slave brace closely follows the master. The slave often reaches the master position within half a second. Figure 37 shows the tolerances applied to the master position (0.15 inch tolerance, and 1 second tolerance).

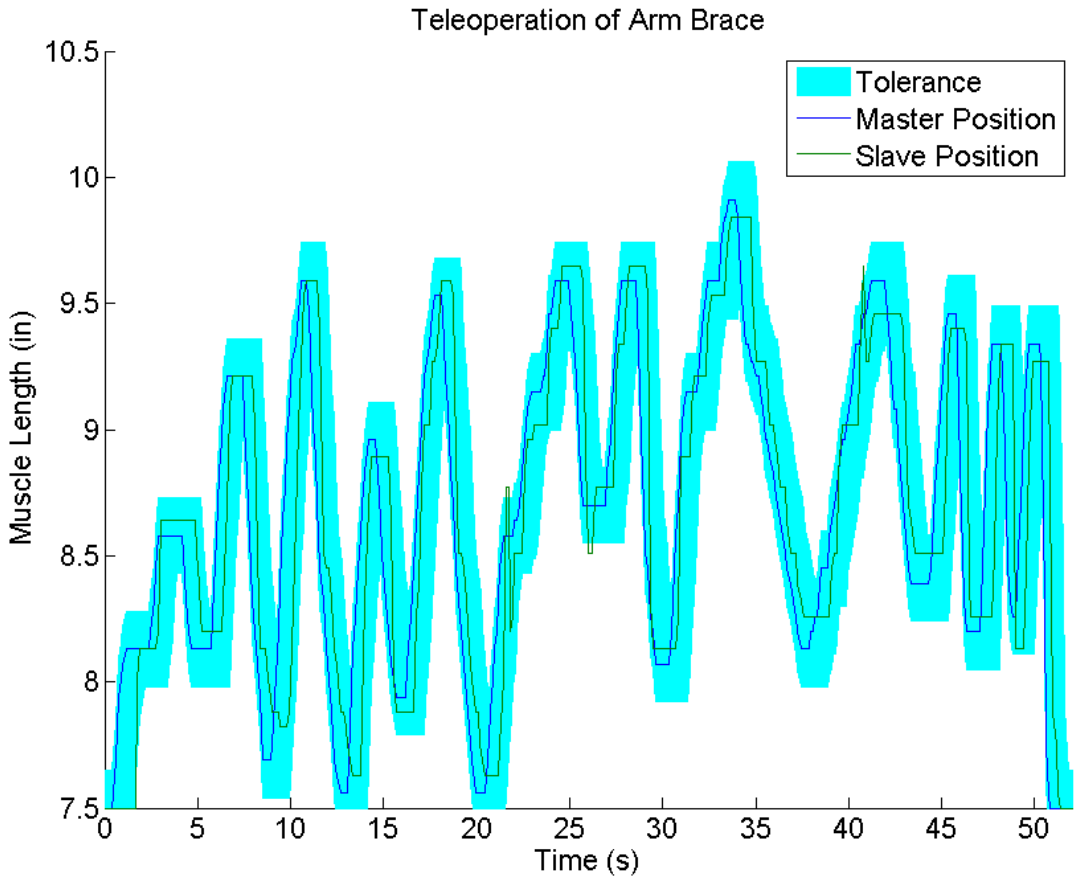

Figure 37: lengths of master and slave muscles. The master is in blue and the slave is in green. The tolerance is shown in cyan.

The system stayed within the tolerance for 98.6% of the test. The slave brace went slightly out of the allowed tolerances at around 1.5 seconds, 21 seconds, 26 seconds, and at 40 seconds.

5.10.7 Path Following

This test was run on the arm brace. The system was told to go to the (length (in), time (s)) points (8, 0), (8.5, 3), (9, 5), (8, 7), (10, 9), (8, 11), (8.5, 13), (8.5, 16). The length of the muscle was recorded during this time duration, and is shown in Figure 38 below.

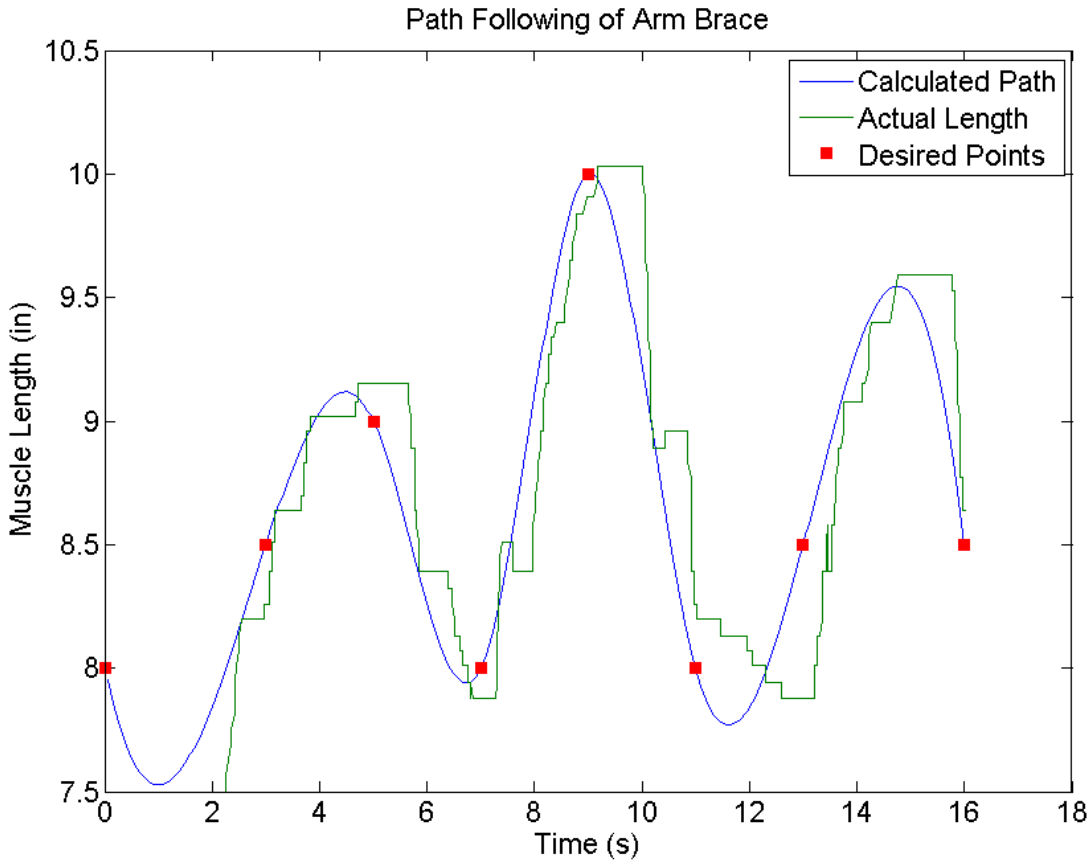

Figure 38: length of muscle over time. Goal points shown in red, calculated smooth path shown in blue. Actual path shown in green.

The system was able to generate a smooth path for the muscle to follow. The muscle closely followed the planed path. The muscle started at 7.5 in, but the path started at 8 in. Besides that, the muscle was able to get within 0.15 inches and half a second of each goal point, while following the smooth path. The tolerances are shown in Figure 39 below.

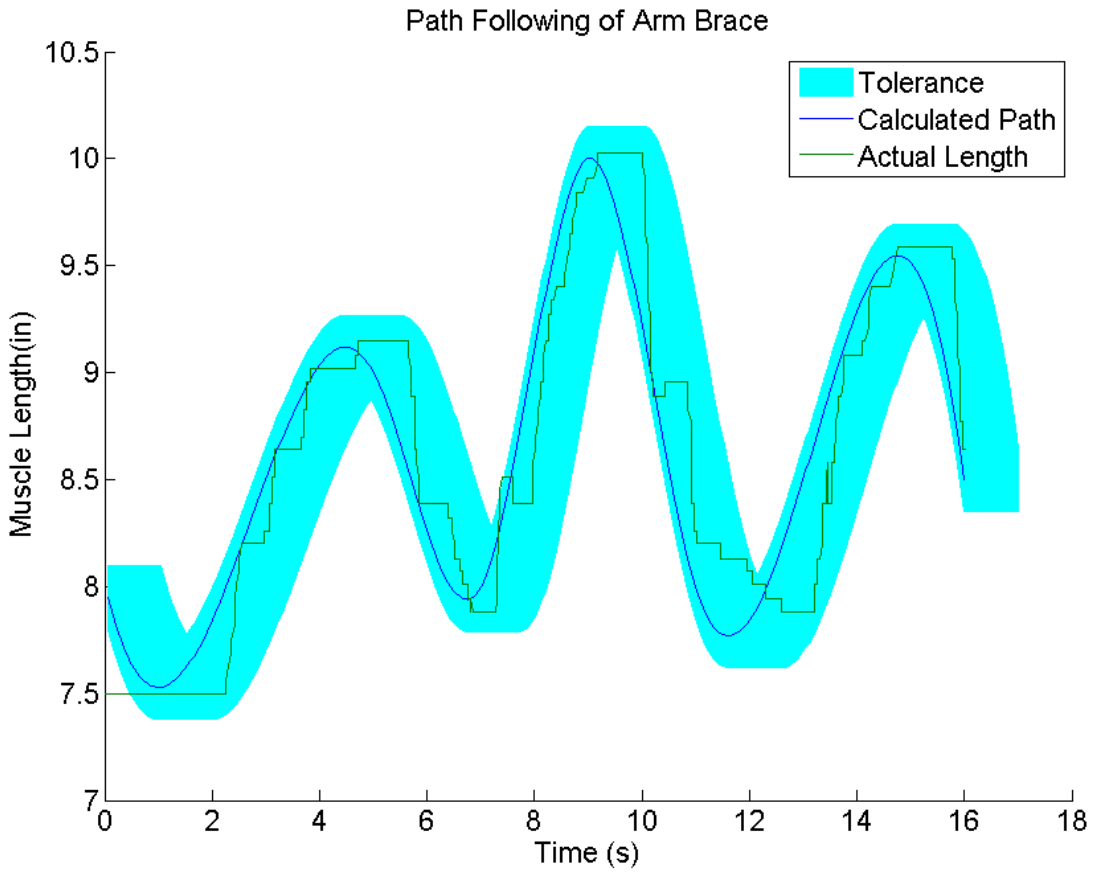

Figure 39: length of muscle over time. Calculated smooth path shown in blue, path expanded by tolerance shown in cyan. Actual path shown in green.

The tolerances for our system are 0.15 inches, and 1 second. That is, the system should reach within 0.15 inches of the set point within 1 second. Figure 39 shows, in cyan, the calculated path expanded by 0.15 in in the positive and negative y directions, and expanded by 1 s in the positive x direction. The muscle slightly left the tolerance at around 8 seconds, but aside from that, the system was able to stay within tolerance of the calculated path, satisfying Requirement 2 and Requirement 4.

5.11 Summary

Different sensor options were explored and tested, and the best ones were chosen for the muscles. For length sensing, the KY-040 encoder was chosen. For pressure sensing, the TBPDPNS100PGUCV pressure sensor was chosen. For the control system, the Arduino platform was chosen, and then a control program was written for the Arduino. Testing showed that the muscles can be accurately controlled using this setup. The system was able to control the muscles during both teleoperation and while following a planned path.

6 Conclusion

6.1 Introduction

This section will discuss how the results compared to the original requirements of the project. Overall, the project was successful in completing the goals that were set. Then, the section discusses possible future work related to this project.

6.2 Summary

The original requirements can be found in Table 1. Table 5 shows how this project met each requirement. Seven of the requirements were completely met. Requirement 2 was relaxed part way through due to available sensors, and the relaxed requirement (0.15 in) was met. Requirement 7 was not met.

Requirement 1:	Our flow control valve was successful. It is able to fully open or close within half a second, which meets this requirement.
Requirement 2:	Our length sensor meets this requirement; it has a resolution of 0.0633 inches which is within the 1/8 inch requirement. However, the control system was chosen to have an accuracy of 0.15, to allow for quicker movements and less jitter. This did not satisfy the original requirement, but was still acceptable for this application.
Requirement 3:	Due to the pressure sensor breaking, a force sensor was used during the force control tests. This sensor was able to determine the force to within 1 N accurately, meeting the requirement.
Requirement 4:	The results in Figure 37 and Figure 39 show that the system reaches the set point within 1 second, satisfying the requirement.
Requirement 5:	The system is able to teleoperate the muscles, as shown in Figure 36. The controlled muscle is usually about half a second behind, satisfying the requirement.
Requirement 6:	We were able to successfully demonstrate the control system by creating two arm braces with muscles on them, and using one brace to teleoperate the other. This met the requirement.
Requirement 7:	The current cost of $32.89 is higher than the $10.00 requirement. However, a muscle without sensors costs only $5.57, which is below the $10.00 requirement.
Requirement 8:	The muscles were able to survive 4 hours of constant running at the Cambridge Science Festival, with children occasionally grabbing the muscles as they were expanding and contracting, which proves the muscles are durable. The muscles also survived project presentation day, which was another 4 hours of testing and demonstrations, which shows that the requirement is satisfied.
Requirement 9:	The muscles are modular. The sensors can be easily separated from the muscle, and one part can be replaced while using the others. This meets the requirement.

Table 5: Meeting of requirements

6.3 Future Work

One possible future project would be to use EMG readings to control the goal of the muscles. Instead of teleoperation or having a recording, the system would determine where the user was trying to move their arm, and use that as the target position.

Another possible idea for future work is to look into different length sensors. Our group used an encoder and a retractable reel to measure the length. There are several possible stretch sensors besides the one we tried, and this field seems to have lots of development right now, so in the future there may be better sensors available for cheaper. Also, EGaIn sensors would probably be worth looking into.

Instead of a rigid brace, one could try using a soft brace, or just strapping the muscles directly to the users arm without a separate structure.

Another idea that we did not have time to pursue is to design a flow control valve that can control the flow to multiple muscles at once, instead of needing a separate valve for each muscle.

6.4 Conclusion

Hydro-muscles are soft actuators that, with the addition of two sensors, are capable of precise measurements of their current length and pressure. The muscles and sensors are constructed from readily-available, low-cost materials, and their modular design allows for them to be easily swapped in and out of systems. In addition to the muscles, a hydraulic system capable of providing controllable flow rates to a muscle was developed. The system allowed for quick (within 1 second) and accurate control of the muscles to a precision of 0.15 inches. To demonstrate this, hydro-muscles were used to actuate an arm-brace about a double jointed pivot with both precision and speed. Since hydro-muscles are light weight and flexible, they can be used in a wide range of robotic applications, and can often be directly attached to the components that they are actuating.

References

"AnalogRead." *Arduino*. N.p., n.d. Web. 14 Apr. 2015. <http://arduino.cc/en/Reference/analogRead>.

"Basic Board Mount Pressure Sensors" *Honeywell International Inc*. Feb. 2014. Web. 19 Nov. 2014. <http://media.digikey.com/pdf/Data%20Sheets/Honeywell%20Sensing%20&%20Control%20PDFs/TBP_Series_DS~.pdf>

"A Basic Guide: How Feedback Encoder Devices Operate." *Micromo*. N.d., Web. 21 Apr. 2015. <http://www.micromo.com/technical-library/encoder-tutorials/basic-guide-how-feedback-encoder-devices-operate>

"Biomimetics" *European Space Agency* April. 2015 <http://www.esa.int/gsp/ACT/bio/index.html>

Co, Thomas B., "Ziegler-Nichols Method." *Michigan Technological University*. 13 Feb. 2004, Web. 19 Apr. 2015. < http://www.chem.mtu.edu/~tbco/cm416/zn.html>.

Cooper, Douglas J., "Using Signal Filters In Our PID Loop." *Controlguru*. 2006, Web. 19 Apr. 2015. < http://www.controlguru.com/wp/p82.html>.

Corso, N. D., Effraimidis, D., Jennings, B. C., & McCarthy, G. D. (2014). Hydro Artificial Muscle Exo-Musculature (Undergraduate Major Qualifying Project No. E-project-050114-143641). Retrieved from Worcester Polytechnic Institute Electronic Projects Collection: http://www.wpi.edu/Pubs/E-project/Available/E-project-050114-143641/

"Dap 09114 Xhose 50-Feet Incredible Expanding Hose" Amazon.com, Inc. 2015. Web. 4 Apr. 2015 http://www.amazon.com/09114-Xhose-50-Feet-Incredible-Expanding/dp/B00AOVH60M/ref=sr_1_1?s=hi&ie=UTF8&qid=1429990054&sr=1-1

G. K. Klute, J. M. Czerniecki, and B. Hannaford, "McKibben artificial muscles: Pneumatic actuators with biomechanical intelligence," in Proc. IEEE/ASME Int. Conf. Adv. Intell. Mechatron., 1999 <http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=803170>

"How to Measure Pressure With A Pressure Sensor" *National Instruments*. 15 Nov. 2012. Web. 20 Nov. 2014 <http://www.ni.com/white-paper/3639/en/>

Nichols, N. B., Ziegler, J. G., "Optimum Settings or Automatic Controllers." Nov. 1942, Web. 21 Apr. 2015. < http://chem.engr.utc.edu/Student-files/x2008-Fa/435-Blue/1942-paper.pdf>

"Pneumatic Artificial Muscles." *Soft Robotics Toolkit*. N.p., n.d., Web. 15 Apr. 2015. <http://softroboticstoolkit.com/book/pneumatic-artificial-muscles>.

"XHose" XHose Extendable Hose, 2015, <www.xhose.com>

Zhong, Jinghua, "PID Controller Tuning: A Short Tutorial." *K.N.Toosi University of Technology*. Spring 2006, Web. 19 Apr. 2015. < http://saba.kntu.ac.ir/eecd/pcl/download/PIDtutorial.pdf>.

Appendix A: Code

This is the main program for controlling muscle length or force on the Arduino. There are two switches connected to the Arduino, one controls length vs force, and the other controls record vs playback. When the Arduino is controlling length, the system will use the length from the encoder. When controlling force, the system will calculate the force based on the length and pressure readings.

In recording mode, the system will send the current length or force over the serial port, so that it can be recorded by the computer, or used as the set point by a second Arduino for teleoperation. In playback mode, the system will listen for set points on the serial point, and use PID control to move the system to that set point.

```
#include <Servo.h>
#include <Encoder.h>

// PID gains
#define P 200
#define I 0
#define D 120

// resting length of muscle
#define LENGTH 7.5

// length muscle moved per encoder tick (in inches)
#define ENC_LEN_PER_TICK (0.0633)
#define ENC_MAX_TICKS 48

#define LOOKUP_TABLE_SIZE 5

// what pins components are connected to
int slaveInValvePin = 9;
int slaveOutValvePin = 8;
int slaveServoPin = 10;
int slavePressurePin = A0;
int lenForceSelectPin = 4; // TODO
int recordPlaybackSelectPin = 5;

int encoder1Pin = 2;
int encoder2Pin = 3;
Encoder myEnc(encoder1Pin, encoder2Pin);

Servo slaveFlowControlServo;
Servo masterFlowControlServo;

float goal = LENGTH;

float forceTableInput[LOOKUP_TABLE_SIZE] = {0, 500, 1000, 1500, 2000};
float forceTableForce[LOOKUP_TABLE_SIZE] = {0, 5, 10, 15, 20};

boolean recording = false;
boolean lengthControl = true;

// setup inputs and outputs
```

```
void setup() {
  pinMode(slaveInValvePin, OUTPUT);
  pinMode(slaveOutValvePin, OUTPUT);
  pinMode(lenForceSelectPin, INPUT);
  pinMode(recordPlaybackSelectPin, INPUT);
  pinMode(slavePressurePin, INPUT);
  slaveFlowControlServo.attach(slaveServoPin);
  Serial.begin(9600);

  // check what mode we are in
  recording = digitalRead(recordPlaybackSelectPin);
  lengthControl = digitalRead(lenForceSelectPin);
}

// do PID forever
void loop() {
  slavePIDControl();
}

void slavePIDControl() {
  // attempt to move to goal location using PID
  float pTerm = 0;
  static float iTerm = 0;
  float dTerm = 0;

  static float prevDiff = 0;

  static boolean prettyClose = false;

  static float prevSetPoint = 0;
  static boolean aboveSetPoint = true;
  static float scaleFactor = 1.0;

  static int slavePressure = 0;

  // read length from encoder
  int ticks = myEnc.read();
  // if we lost ticks, reset to a reasonable value
  if (ticks < 0) {
    myEnc.write(0);
    ticks = 0;
  } else if (ticks > ENC_MAX_TICKS) {
    myEnc.write(ENC_MAX_TICKS);
    ticks = ENC_MAX_TICKS;
  }
  float length = ticks * ENC_LEN_PER_TICK + LENGTH;

  // use value from serial for goal in playback mode
  if (!recording) {
    if (Serial.available() > 1) {
      goal = Serial.parseFloat();
      // read the \n that comes after the float
      int n = Serial.read();
    }
  } else {
    goal = length; // don't do anything
```

```
    }

    // calculate the error
    float diff = goal - length;

    // filter the pressure, because the readings are a bit noisy
    slavePressure *= 0.9;
    slavePressure += analogRead(slavePressurePin) * 0.1;

    // for playback, always send our current length and pressure
    // for recording, send lengths every 50 ms
    if (!recording) {
      Serial.print(length);
      Serial.print(",");
      Serial.println(adcCountToPressure(slavePressure));
    } else {
      static long lastTime = 0;
      long currentTime = millis();
      if (currentTime - lastTime > 50) {
        Serial.println(length);
        lastTime = currentTime;
      }
    }

    // if we're pretty close to the setpoint, don't jitter back and forth
    if (abs(diff) < 0.15) {
      prettyClose = true;
      setFlowRate(0,             slaveInValvePin,             slaveOutValvePin,
slaveFlowControlServo);
      return;
    } else if (prettyClose && abs(diff) < 0.15) { // if we were close but have
strayed a bit
      setFlowRate(0,             slaveInValvePin,             slaveOutValvePin,
slaveFlowControlServo);
      return;
    } else {
      prettyClose = false;
    }

    pTerm = diff;
    iTerm += diff;
    dTerm = diff - prevDiff;

    // calculate PID output
    int rate = P * pTerm + I * iTerm + D * dTerm;

    // decrease movements if we are oscillating
    if (goal == prevSetPoint) {
      if (diff > 0 && !aboveSetPoint) {
        // we were below and now went above
        aboveSetPoint = true;
        scaleFactor /= 1.5;
      } else if (diff < 0 && aboveSetPoint) {
        // we were above but now are below
        aboveSetPoint = false;
        scaleFactor /= 1.5;
```

```
      }
    } else {
      // setpoint changed
      prevSetPoint = goal;
      aboveSetPoint = diff > 0;
      scaleFactor = 1.0;
    }

    // send flow rate to the flow control valve
    setFlowRate(rate   *   scaleFactor,   slaveInValvePin,   slaveOutValvePin,
slaveFlowControlServo);

    prevDiff = diff;
  }

  // opens both valves
  void muscleOpen(int inPin, int outPin) {
    digitalWrite(inPin, true);
    digitalWrite(outPin, true);
  }

  // closes both valves
  void muscleClose(int inPin, int outPin) {
    digitalWrite(inPin, false);
    digitalWrite(outPin, false);
  }

  // expands the muscle
  void muscleExpand(int inPin, int outPin) {
    digitalWrite(inPin, true);
    digitalWrite(outPin, false);
  }

  // contracts both valves
  void muscleContract(int inPin, int outPin) {
    digitalWrite(inPin, false);
    digitalWrite(outPin, true);
  }

  // sets the flow control valve
  void setFlowRate(int rate, int inPin, int outPin, Servo servo) {
    if (rate > 0) {
      muscleExpand(inPin, outPin);
    } else if (rate < 0) {
      muscleContract(inPin, outPin);
    } else {
      muscleClose(inPin, outPin);
    }

    // when contracting, go more slowly, because the muscle naturally wants to
contract faster
    if (rate < 0) {
      rate = -rate / 4;
    }
    setServo(abs(rate), servo);
  }
```

```
// opens the valve the appropriate amound
void setServo(int amount, Servo servo) {
  amount = min(amount, 100);
  int value = map(amount, 0, 100, 40, 100);
  servo.write(value);
}

// converts ADC counts to pressure in PSI
float adcCountToPressure(int adcCount) {
  float pressureVoltage = adcCount * 5.0 / 1023.0;
  float pressurePressure = pressureVoltage * 23;
  return pressurePressure;
}

// uses a lookup table to calculate force based on length and pressure
float lengthPressureToForce(float length, float pressure){
  float value = length * pressure;
  // use lookup table
  for (int i = 0; i < LOOKUP_TABLE_SIZE; i++) {
    if (value == forceTableInput[i]) {
      return forceTableForce[i];
    } else if (value < forceTableInput[i]) {
      // interpolate
      return (forceTableForce[i-1] * (forceTableInput[i] - value) +
              forceTableForce[i] * (value - forceTableInput[i-1])) /
              (forceTableInput[i] - forceTableInput[i-1]);
    }
  }
}
```

This function runs on a computer connected to the Arduino records the length or force of a muscle. The recording is saved to a file on the computer, and can be played back later.

```
function [ ] = record( serialPort, filename )
%record starts recording lines from the serial port
%   records lines and timestamps into csv format

    %set default filename if it wasn't set
    if nargin < 2
        filename = strcat('recording from ', datestr(now,'mm-dd-yyyy HH-MM-
SS FFF'), '.csv');
    end

    %close all open ports
    if ~isempty(instrfind)
        fclose(instrfind);
    end

    %open serial port
    port = serial(serialPort);
    fopen(port);
    %open output file
    f = fopen(filename, 'w');
```

```
    %start timing
    tic

    %read from the serial port, and save the data
    while 1
        line = fgetl(port);

        disp([line(1:end-1) ',' num2str(toc)]);

        %write data
        fwrite(f, num2str(toc));
        %write time
        fwrite(f, [',' num2str(line) char(10)]);
    end
end
```

This function takes in desired lengths or forces, and timestamps. It creates a smooth curve that passes through all the desired points. This smooth curve can then be sent to the Arduino to have it follow the calculated path.

```
function [ ] = manualInput( lengths, times, filename )
%creates a 'recording' from the input lenghts and times
%   creates a smooth path to reach all those lengths at the given times

    %set default filename if it wasn't set
    if nargin < 3
        filename = strcat('recording from ', datestr(now,'mm-dd-yyyy HH-MM-
SS FFF'), '.csv');
    end

    %time step in seconds
    timeStep = 0.05;

    %open output file
    f = fopen(filename, 'w');

    s = spline(times, lengths);

    %write data to file
    for i = 0:timeStep:max(times)
        %write time
        fwrite(f, num2str(i));
        %write data
        fwrite(f, [',' num2str(ppval(s, i)), char(10)]);
    end

    %close the file
    fclose(f);
end
```

This function runs on the computer and is used to read a recorded file, and send the set points over the serial port to the Arduino.

```
function [ ] = playback( serialPort, filename, filename2 )
%playback playsback a previously recorded file
%   plays back lines at the same times they were received

    %close all open ports
    if ~isempty(instrfind)
        fclose(instrfind);
    end

    %set default filename if it wasn't set
    if nargin < 3
        filename2 = strcat('recording from ', datestr(now,'mm-dd-yyyy HH-MM-
SS FFF'), '.csv');
    end

    %open serial port
    port = serial(serialPort);
    fopen(port);
    %open previously recorded file
    f = fopen(filename, 'r');
    %open file to save this data
    f2 = fopen(filename2, 'w');
    %skip first line
    fgetl(f);

    goal = 0;

    tic;
    waiting = false;

    try
        %read stuff
        while 1
            %read a line if we need another
            if ~waiting
                line = fgetl(f);
            end

            %record the serial communication from the arduino
            if port.BytesAvailable > 0
                fwrite(f2, strrep(fread(port, port.BytesAvailable), char(13),
[',' num2str(toc) char(13)]));
            end

            splitLine = strsplit(line, ',');

            %wait until it is time to send this goal
            %pause(str2double(splitLine(1)) - toc);
            if str2double(splitLine{1}) > toc
                waiting = true;
                continue;
            end
            waiting = false;

            %no need to resend the same goal
            if strcmp(goal,splitLine{2})
```

```
                continue;
            end

            goal = splitLine{2};

            fprintf(port, splitLine{2});
        end
    catch E
        disp(E);
    end

    % graph how well the system followed this recording
    figure;
    data = csvread(filename, 1, 0);
    data2 = csvread(filename2, 1, 0);
    plot(data(1:end,1),data(1:end,2),data2(1:end,3),data2(1:end,1));
    %axis([7 12 7.5 10.5]);
    fclose(instrfind);
end
```

This function runs on the computer and reads the length and pressure from two Arduinos during teleoperation. It saves the data to a file, and then calls the armSim function to draw each arm.

```
function [ ] = drawArm( slavePort, masterPort )

    %close all open ports
    if ~isempty(instrfind)
        fclose(instrfind);
    end

    %open serial ports
    port1 = serial(slavePort);
    port2 = serial(masterPort);
    fopen(port1);
    fopen(port2);
    %skip first line
    fgetl(port1);
    fgetl(port2);

    prevLen1 = 0;
    prevPres1 = 0;

    prevLen2 = 0; % master doesn't have pressure

    fig1 = figure;
    fig2 = figure;

    filename = strcat('recording from ', datestr(now,'mm-dd-yyyy HH-MM-SS
FFF'), 'teleop', '.csv');
    f = fopen(filename, 'w');

    tic;

    %read stuff
    while 1
```

```
        %read a line or two
        fgetl(port1);
        fgetl(port2);
        line1 = fgetl(port1);
        line2 = fgetl(port2);

        %parse the line
        splitLine1 = strsplit(line1, ',');
        splitLine2 = strsplit(line2, ',');
        curLen1 = str2double(splitLine1(1));
        curPres1 = str2double(splitLine1(2));
        curLen2 = str2double(splitLine2(1));

        %save the data to a file
        fwrite(f, strcat(splitLine1{1},',',splitLine2{1},',',num2str(toc)));
        fwrite(f, char(13));

        %if anything has changed, update the figure
        if prevLen1 ~= curLen1 || prevPres1 ~= curPres1
            figure(fig1);
            armSim(curLen1, curPres1);
            prevLen1 = curLen1;
            prevPres1 = curPres1;
            drawnow();
        end

        %if anything has changed, update the figure
        if prevLen2 ~= curLen2
            figure(fig2);
            armSim(curLen2);
            prevLen2 = curLen2;
            drawnow();
        end

        %clear buffer
        if port1.BytesAvailable > 0
            fread(port1, port1.BytesAvailable);
        end
        if port2.BytesAvailable > 0
            fread(port2, port2.BytesAvailable);
        end
    end
end
```

This function takes the length and pressure of a muscle, and calculates the position of the arm brace that the muscle is attached to. It then draws the arm brace.

```
function [] = armSim( length, pressure )

    if nargin < 2
        pressure = 20;
    end

    %   various constants
    topLinkLen = 9;   % the length of the top part of the arm brace
```

```
    botLinkLen = 11;  % the length of the bottom part of the arm brace

    topAttachLen = 5; % how far from the elbow the muscle is attached to the
                      % top part

    botAttachLen = 5.5; % how far from the elbow the mucles is attached to
the
                      % bottom part

    topAngle = 2 * pi / 3; % the angle the top part is held constant at

    %   calculate the angle the bottom part of the brace is at, using law of
    %   cosines
    botAngle = topAngle - acos((length^2 - topAttachLen^2 -
botAttachLen^2) ...
                               / (-2 * topAttachLen * botAttachLen));

    %   arm (centered at origin)
    armX(1,1) = topLinkLen * cos(topAngle);
    armX(1,2) = 0;
    armX(1,3) = botLinkLen * cos(botAngle);
    armY(1,1) = topLinkLen * sin(topAngle);
    armY(1,2) = 0;
    armY(1,3) = botLinkLen * sin(botAngle);

    %   muscle
    muscleX(1,1) = topAttachLen * cos(topAngle);
    muscleX(1,2) = botAttachLen * cos(botAngle);
    muscleY(1,1) = topAttachLen * sin(topAngle);
    muscleY(1,2) = botAttachLen * sin(botAngle);

    %   plot the position of the arm and muscle
    plot(armX,armY,'k','LineWidth',4);
    hold on;
    widthFactor = 10; % width depends on pressure
    plot(muscleX,muscleY,'g','LineWidth',pressure / widthFactor + 2);
    hold off;

    %   label the graph
    title('Hydro Muscle Arm Brace');
    xlabel('X Position');
    ylabel('Y Position');
    grid on;
    plotSize = max(topLinkLen, botLinkLen) + 1;
    axis([-plotSize,plotSize,-plotSize,plotSize]);
    text(-5, -1, ['Length = ' num2str(length) ' in'], 'FontSize', 15);
    if nargin >= 2
        text(-5, -3, ['Pressure = ' num2str(pressure) ' psi'], 'FontSize',
15);
    end
 end
```

Appendix B: Materials

Item	Cost	Seller
Super Soft Latex Rubber Tubing 3/8" ID, ½"OD, 1/16" Wall, Semi-Clear Amber	$1.46/ft	http://www.mcmaster.com/#5234k51/=wtqhy2
Nylon Tubular Webbing	$1.12/ft	https://www.xhose.com/
Conductive Rubber Cord Stretch Sensor	$9.95/m	https://www.adafruit.com/products/519
KY-040 Rotary Encoder Module	$2.98	http://www.amazon.com/KingSo-KY-040-Encoder-Development-Arduino/dp/B00OQ0EHJK
NBPDANN150PAUNV Pressure Sensor	$20.81	http://www.mouser.com/ProductDetail/Honeywell/NBPDANN150PAUNV/?qs=gKd32bbgGM82ymAJlOg8pg%3D%3D
Hinged Elbow Brace Support Guard	$29.99	http://www.ebay.com/itm/Flexibrace-Hinged-Elbow-Brace-Support-Guard-/191203374515?pt=LH_DefaultDomain_0&var=&hash=item2c849bf5b3
2W-025-08 12V DC ¼" Solenoid Valve	$20.89	http://www.amazon.com/2W-025-08-Position-Water-Electric-Solenoid/dp/B00843V9N2
6380490 PVC Ball Valve	$4.31	http://www.ebay.com/itm/THE-SPECIALTY-MFG-CO-6380490-PVC-Ball-Valve-Inline-FNPT-x-MNPT-1-4-In-/221602099777?pt=LH_DefaultDomain_0&hash=item339883b241
Arduino Uno	$21.62	http://store.arduino.cc/product/A000066
S3305 Standard HT Servo	$33.98	http://www.amazon.com/Futaba-S3305-Standard-Servo-Gears/dp/B0015H4C24
Nylon Push-on Hose Fitting	$5.62/10	http://www.mcmaster.com/#5228k24/=wtr1mp
Worm-Drive Hose Clamp with Zinc Plated Steel Screw	$5.78/10	http://www.mcmaster.com/#5388k14/=wtr2ms
80/20	$0.53/in	http://download.8020.net/8020_Price_List_2011.pdf
Mid Size Mirror Finish Light Duty Metal Reel	$3.39	http://www.retractablereels.com/light-duty-reels/size-mirror-finish-light-duty-metal-reel-p-695
Medium-Pressure Brass Threaded Pipe Fitting	$1.38	http://www.mcmaster.com/#50785k162/=wtrboq

5.5 yards PARACHUTE RIPSTOP NYLON ORANGE MATERIAL FABRIC RO92	$0.42/ft^2	http://www.ebay.com/itm/5-5-yards-PARACHUTE-RIPSTOP-NYLON-ORANGE-MATERIAL-FABRIC-RO92-/380109713072

Table 6: Materials used to construct hydro-muscle