Final Capstone Report

Robotic Telepresence

University of Washington, Department of Mechanical Engineering

June 8, 2017

From left to right:

Jess Carr
Ashley Combs
Nathan Isaman
Eze Klarnet
Bryan Beard

Table of Contents

I. Executive Summary

Many professions and industries pose great risk to their skilled work force. Human safety can be increased by automating these dangerous professions, but many jobs are too delicate or difficult to fully automate. This project sought to develop a middle ground with a human-controlled telepresent robotic arm.

The applications of a human-controlled robot are vast. Moderating the risk of bodily harm must be balanced by the necessity of human skill and instinct in dangerous fields such as bomb defusal, hazmat operations, and military applications. The introduction of robotic telepresence would guarantee no risk to the human in these applications, while their unique and crucial skills could still be directly applied. Professionals in industries with high levels of skill or technical expertise, such as surgeons and machinists, would have the opportunity to apply their talent remotely without the cost or risk of relocation.

The team focused on developing a proof of concept due to the diversity in applications of this device. The overarching goal was to produce a system that would take positional input from a 3D camera, interpret and normalize the data, and recreate the motion via a robotic arm. MATLAB and Simulink were the programs used to communicate between the camera and servos. Multiple avenues of data interpretation were investigated, as many methods for translating the data currently exist or are in development. A straightforward method of translating input joint positions to output servo angles was utilized in the final product due to time and funding constraints.

Testing required examination of both accuracy and repeatability. Two tests were performed to obtain quantifiable data. First, the robotic arm was outfitted with a marker and a paper bullseye target was placed in front of it. The human, using motions the robot could achieve with its more limited degrees of freedom, attempted to touch the target with the marker. This test demonstrated the repeatability of the robot's motion with results in a simple pass/fail format. The goal was to have 100% of the attempts hit within the 22cm target. The second test performed was to determine the accuracy of the prototype. Photos were taken of the human controller alongside the robot and the angles of their respective limbs were compared. The human and robot were deliberately moved in the plane perpendicular to the camera to minimize out of plane distortions of the angle being measured. These results were compared to the respective angles as read by the camera to determine where the largest discrepancy occurred. The goal was to have an average absolute error of less than ten degrees for each joint.

The robot slightly underperformed in both accuracy and repeatability tests. The repeatability test had a sample size of seven attempts and would require additional attempts to determine a statistically significant success rate. Out of these, four successfully landed within the target resulting in 57% repeatability. Improved future testing methods and a larger sample size may reveal better repeatability than these results indicate.

The accuracy test results varied between joints and between points in the system. Over the entirety of the system, that is, between the human input and the robot output, the best performing joint was the elbow with an average absolute error of 2.63 degrees. The wrist was within the desired margin at an error of 6.48 degrees. Both shoulder joints underperformed with errors of 15.75 and 24.68 degrees. It is probable that this large error is due to the joints' intended functionality. The arm as represented by its manufacturer is to be mounted horizontally. For the purpose of mimicking human motion, this project mounted it vertically. The introduction of a counterweight assisted the joints' rotation, however more robust servos may be necessary to reduce the remaining error.

Data was also recorded for the interim steps in the system. The largest error occurred in the wrist joint at 12.25 degrees between the human input and the camera interpretation. This is likely due to the small distance between the wrist and palm from which angle is determined. The smallest error in this step is in the first shoulder joint at 1.67 degrees. The error between the camera and the servo positions more closely resembles that of the overall system with the notable exception of the wrist joint. This reinforces the theory that stronger servos would reduce the overall error, since the system is correctly interpreting the user input.

As this project was intended to provide proof of concept, there are many future developments that may be pursued. An essential aspect of better mimicking human movement is the inclusion of more degrees of freedom. Rotator cuff, forearm rotation, a more functional wrist joint, and lateral finger motion will be critical to many applications and should be pursued in future iterations. Additionally, the inclusion of haptic feedback and a virtual-reality interface for the human user would greatly reduce the amount of training and practice required for effective use. Also, it may be beneficial to allocate additional time to the development of data interpretation and trajectory calculation methods that were not feasible with this project's time constraints. Methods like inverse kinematics may result in higher accuracy for end-effector position and orientation. Finally, the issue of real-time performance must be addressed. The current necessity of extrinsically defined functions in Simulink excludes the implementation of a microcontroller and has likely greatly increased the computation time of the model. Should these functions be successfully bypassed, real-time movements may be possible as the program would no longer require the use of the MATLAB Interpreter.

The development of a proof of concept prototype was largely successful. The camera took in joint positions from the human input, the program interpreted those positions and then translated them to joint angles, and the servos attempted to achieve those angles. There are many opportunities for further development and improvement, as there should be from any proof of concept. Once the current mechanical and software limitations are reduced, additions and innovations specific to the intended application may be developed. There are many future possibilities for this technology, and this project constitutes a definitive initial step.

II. Project Goals

The main goal of this capstone was to create a telepresent robot that could be used in a variety of applications. The focus was on a proof-of-concept design, which could be built upon for a variety of end uses. This project focused on integrating multiple pieces of hardware and software into a continuous system using human-in-the-loop control. MATLAB and Simulink code was utilized for sensing, data filtering, and servo control. The device was designed to successfully meet testing parameters, including the device's capacity to hit target locations and replicate large-scale human movement within ten degrees of accuracy. Another main concern was that the device could execute these tasks repeatedly without losing accuracy or consistency.

To quantify robot performance, the arm was designed to hit a series of parameters. The quantified goals are as follows:

1. Device consistently hit points within a 20 cm target.
2. Device replicates human body angles within 10 degrees of accuracy.
3. Hand is capable of holding a load that approaches the max recommended load for the arm.
4. Hand can grip up to 50 grams.

Final robot performance was measured against these goals.

III. Background

a. Previous Developments

The XBOX One Kinect is a 3D camera developed by Microsoft. It uses a combination of infrared and visual spectrum sensors to detect movement, and it is able to export color and depth digitized visual information to a variety of hardware and software packages. One of these software packages is in MATLAB. The MATLAB Kinect Toolbox and Image Acquisition Toolbox can be used to collect Kinect information, track a human body, and obtain joint information from the body.

When it comes to motion detection and telepresent robotics, there are a few recent developments that span across multiple disciplines, which demonstrates the diverse applications of this technology. At NASA JPL, engineers have created a three-fingered robot that is controlled using an Xbox Kinect as the input sensor and an Oculus Rift for a first person point of view. The intent of the robot was to assist astronauts in their daily tasks around the space station.[15]

Microsoft had a development group known as "Hand Pose" that advanced the algorithm for the Xbox Kinect to better recognize hand positions and movements. The original software recognizes only the palm and tips of the middle finger and thumb. This group developed software capable of determining the location of every joint within the hand. Although their

development did not continue into a robotic application, it opened many doors for the ability to have a more lifelike robotic hand.[10]

Another example of past work is another University of Washington student capstone design, with applications in medicinal fields. In this project, a Kinect was used to watch how a person could move their joints in order to diagnose any problems with tendons, muscles, or bones within the hand. In contrast to JPL's accomplishments, motion detection data was examined via statistical analysis rather than outputted to a robot.[22]

b. Additional Applications

There are numerous applications for telepresent robotics, ranging from hazardous situations to delicate surgical work. It is an extremely versatile technology that can improve human safety and efficiency by expanding the reach of specialized skills. The primary application of a telepresent robot is to remove a human from a dangerous environment while retaining use of their skills or special training. Bomb disposal is one example of a field where this technology will be extremely important. Police or military bomb squads face great personal risk in their profession. A mobile, telepresent robot outfitted with necessary equipment or attachments would allow a skilled technician to diffuse a bomb remotely with no threat of harm or loss of life.

Another profession that could benefit from the use of telepresent robots is the hazardous materials industry. The precautionary measures that are already taken, including hazmat suits and material containment, are never completely failsafe. A human worker could be completely removed from harm's way and can still accomplish the task at hand through the use of a telepresent robot.

Similarly, loss of life in dangerous regions could be significantly reduced through military applications of this technology. Many robots are currently used by the United States military, including exploratory and weaponized machines. Robots that can interact with its environment in a more hands-on manner has recently been put to use.[5] These robots can perform tasks that require the delicacy of human control thanks to the introduction of telepresence. They can execute the job of a scout or perform risk-assessment tasks without risking a soldier or civilian. They can react like a human in situations where even the most thought out programming cannot outweigh human instinct because they are actively controlled by a trained professional.

Industrial applications are also widespread. Serious injuries can occur in factories, particularly to skilled machinists working with complex tools.[21] Machining is not easily automated.[12] which therefore necessitates human presence during the operation. A telepresent robot with specific kinematic capabilities would be uniquely useful in this industry, allowing a person to apply their skills with no bodily risk.

The medical field is another industry that could benefit from the use of telepresent robotics. Top surgeons could utilize their skills to improve the lives of patients all over the world,

without relocating either party. A summary of the applications that our team conceived for this project are listed below in Table 1.

Table 1: Project Applications

Application	Option 1	Option 2	Option 3	Option 4
Military	Bomb Defusing	Telepresence in danger situations (i.e. hazmat)	Training	Maintenance
Academic	At home learning	In classroom learning (part of curriculum)	ASL learner	
Medical	Robotic Surgery	Surgical Training		
Toy	Tickle Me Elmo			
Manufacturing	Improved method of "teaching" robotic arms	Telepresence in dangerous operations (i.e. hazmat)	Maintenance (lower downtime due to safety precautions)	

c. Risk and Liability Issues

There are a number of liability issues associated with robotic telepresence. One example is the possibility of an individual or group maliciously hacking into surgical robots for the purpose of harming the patient. This is a potential risk with any computerized system despite the ever increasing levels of cyber security. Signal lag is another issue with interconnected systems, whether they be wireless or connected via the internet. Data cannot be transferred instantaneously and the magnitude of the signal lag will increase as the user and the robotic system are moved further apart. Connection stability is also a risk with this system since lag spikes could have a tremendous impact on the performance of the system, especially with delicate maneuvers. In a time sensitive situation, such as defusing a bomb, every second counts and if the robot is a couple seconds behind the user, it could lead to mission failure. Lag during a surgery could cause a surgeon to accidentally cut past the target point causing undue harm to the patient. Another stand-out risk with robots of this nature is the potential for general failures, including mechanical, electrical, and communication. Depending on the situation, there may not be a human around to intervene and the failure can cause greater problems. If the robot was in a bomb defusal situation and it accidentally cuts the wrong wires because a motor skipped while moving in for the cut, it could mean death for those still in the blast radius; medical applications would have similar issues with unintended movement.

d. Ethical Issues

There are numerous codes and standards that could come into play with the use of this device; they vary based on the application that robot would be operated within. For complete list of codes and standards, see Appendix A. Below is a brief overview of relevant codes and standards based on the application.

Medical: ISO 13402 covers metals used in surgical applications so all exposed metal on the robot would have to conform to this standard. AAMI TIR.38, AAMI TIR.57, and AAMI TIR.65 deal with medical device safety, security, and sustainability, specifically as it relates to device lifecycle and material usage. The cleaning and treatment of medical devices is discussed in ANSI 11135 and ANSI 11737, and it is essential that the device is designed to be cleaned up to medical code.

Bomb Defusal: When working with mines, the United Nations has established the International Mine Action Standards, which involves the research, development, construction, and use of demining equipment. Bomb diffusion could be done by either military or private groups, and the military has an independent system of codes and standards that would apply.[1]

Hazmat: UL 674 and UL 698 cover the use of electrical motors and generators and industrial control in division 1 hazardous locations. UL 877 regards circuit breaks, and UL 894 has to do with switches in hazardous locations. Additionally, various components regarding the use of industrial robots and robotic systems are found in ANSI R 15.01-1, 15.02, 15.05-1, 15.05-2, 15.05-3, and 15.06.

e. Impact on Society

Robotic telepresence will enable human exploration in regions that were previously too remote for immediate assistance. For example, a Russian surgeon famously had to remove his own appendix while on an Antarctic expedition as he was the only one there capable of performing the surgery.[13] A surgical robot being operated by a surgeon remotely could improve the health and safety of explorers found in similar need of medical intervention. Future space explorers could also benefit from this technology as there is no guarantee that crewmembers or colonists will have the required medical or technical skills in some emergency situations. It is also far more acceptable to have a teleoperated robot perform dangerous tasks in place of a human. This is true for bomb-defusing robots, manufacturing robots (high voltage areas), and robots operating in spaces where humans would be susceptible to harmful chemical or radiation exposure. Telepresent robotics will ultimately will make life on Earth and, eventually, space exploration safer for humanity.

f. Impact on the Environment

Whether this robot would be produced in small batches or mass produced, the materials and efforts required for its construction will have an impact on the environment. The important thing is that engineers minimize the overall negative side of that impact as best as possible. There are no components to the prototype system that have a notable negative impact, but future versions will require some sort of energy source. Common batteries for high power robotics are lithium-ion polymer batteries (LiPo) of various cell sizes. Lithium battery disposal and specifically lithium extraction have a negative impact on the environment.[6] The disposal aspect is not a primary concern as there are various implemented methods for safely disposing of spent or faulty batteries.

One misconstrued environmental concern is the use of leaded solder in the electronics. According to an article written by the Environmental Protection Agency, leaded solder and lead-free have similar effects on the environment, depending on the other metals within them. With respect to the user, the temperature used to melt the solder is not hot enough to aerosolize the lead, meaning the user sees no effects unless they ingest the lead.[8]

g. Cost and Engineering Economics

Many robotic applications are viewed negatively by the general public. This negativity stems from a variety of fears ranging from robots replacing all human labor to the dystopian future seen in various Terminator-esque movies. Some of this anxiety is reasonable as some estimates say up to 47% of total US employment is at risk to be lost due to automation.[9] This project's application, however, is free from either concern as this robotic system relies on human user input to function; there is no artificial intelligence or automation involved. The intent of this robotic system is to allow the skills of a trained person to be used anywhere from a separate location. This will expand the effective range of many niche skills, especially seen in the medical and other technical fields, by cutting out travel time and other relocation concerns.

In regards to the financial burden of the system itself, it has the potential to reduce time and monetary costs for whomever may utilize it. The direct application, such as medical or bomb disposal, will dictate the required materials and accuracy of the machine. The price will increase for higher precision or special material requirements.

IV. Rationale for Design Choices

The complexity of the project requirements as well as the time constraint imposed upon the team required the team to prioritize. The team decided that it could save a lot of time and work if it leveraged the existing commercial robotics market and purchased the hand, arm, and various electrical components for the system. Fabricating the system from scratch would not

have added anything worthwhile to the project outcome and would have taken valuable time away from working on the real challenge: system integration and programming.

For the motion detection aspect of the system, there were two main choices brainstormed by the group: the Xbox Kinect and a wearable device containing sensors to go over the hand and arm. The Kinect is already capable of gathering skeletal data using an infrared sensor to detect the location and movement of the person in front of its sensor. It is purchasable with no additional assembly required for the unit to work. On the other hand, the wearable would have to be taught the "home" location of the arm and be able to determine relative positions based on the movement picked up by the sensors. It would have required sensors to be purchased and a prototype to be made. This prototype would have taken a considerable amount of effort to design in order to move with the user while still having the sensors correctly pick up motion data. In the end, the group decided on focusing on using the Kinect as the method of detection. The biggest selling point was that the Kinect can be used for any user, no matter their shape or size. With the wearable, there was the issue of what hand size to design to, or how long the arm should be. It was not an adaptable enough design element for the wide range of potential users.

To decide which arm and hand combination to purchase, we constructed decision tables. These tables are shown below in Tables 2 and 3.

Table 2: Arm Decision Table

Name	Cost($)	Lifting Capability (hand is 380g)	Arm Length (ft) human is around 2ft	DOF	Notes	Source
RobotShop M100RAK V2	650	500g	2	4	• No electronics • Metal mount on wrist (no EOAT to take off) • All metal gears (durable) • No software • No up/down wrist movement	http://www.robotshop.com/en/robotshop-m100rak-v2-modular-robotic-arm-kit-no-electronics.html?gclid=CJPg78aA0tECFYhufgod7sYKog#productsReviewBoxTitle
Lynxmotion AL5D 4DOF Robotic Arm SSC-32U	350	370g (+ gripper) (seen varying numbers from 300-370g)	1.5	5	• Includes software and servo controller • Includes power supply • Includes EOAT • Inverse kinematics	http://www.robotshop.com/en/lynxmotion-al5df-kt-robotic-arm-flowbotics-studio.html
U Arm metal	350	500g	1	4	• Bluetooth • Arduino, python, mouse control • Currently developing new arm with great precision (ships in April)	http://ufactory.cc/#!/en/about/contact
Global Specialties R680 Banshi Robotic Arm w/ Power Supply	200	60g	.85	6	• Includes processor/software/power supply • Includes keyboard to control • Plastic arm	https://www.instrumentation2000.com/global-specialties-r680.html?fee=2&fep=3482&gclid=CL35-YOgz9ECFYlffgodGq8HZw
Arduino Braccio Robotic Arm	230	150g	1.5	6	• Sample code given • Includes power supply/ no controller • 30ms delay	http://www.robotshop.com/en/arduino-braccio-robotic-arm.html?gclid=CP-BkKOiz9ECFUiDfgodBFYGXw#Specifications

Table 3: Hand Decision Table

Name	Cost($)	Weight (g)	Hand length (ft)	Notes	Source
5DOF Robot Five Fingers Metal Mechanical Paw Left and Right Hand	127 ea	380g	.66	• Individual 5 finger control by servos • 1 month window for shipping	http://alexnld.com/product/diy-5dof-robot-five-fingers-metal-mechanical-paw-left-and-right-hand/?gclid=CN_k09Koz9ECFRB4fgodcgIIXg
DIY 5DOF Robot Five Fingers Metal Manipulator Arm Left and Right Hand QDS-1601	97 ea	170g	.6	• Individual 5 finger control • Long wrist mechanism for the servos (included in length) • 1 month shipping	http://www.bonanza.com/listings/DIY-5DOF-Robot-Five-Fingers-Metal-Manipulator-Arm-Left-and-Right-Hand-QDS-1601/365340000?goog_pla=1&variation_id=130675543&gpid=18283950120&keyword=&goog_pla=1&pos=1o6&ad_type=pla&gclid=CMOJxuy58tECFUNafgod2EkLxw
Bionic Robotic Hand Kit	90ea	Not given but light	"human size"	• Can only control fingers simultaneously • Very cheap looking	https://www.scientificsonline.com/product/bionic-robotic-hand-kit?gclid=CPWc9s3J8tECFZCcfgodzjECLg

Our first decision was to choose a hand that would function as close to a human hand as possible. Many of the robotic hands that we found were expensive and there was only a few choices that could fit in our budget. From the three options in the Hand Decision Table, we chose the "5DOF Robot Five Fingers Metal Mechanical Paw" which can be seen in Figure 1.

Figure 1. Chosen Robotic Hand (Mechanical Paw)[7]

This was chosen because it appeared much sturdier than the other choices as the DIY hand was operated based on pulley systems, which we felt would be a significant point of failure and can be seen in Figure 2.

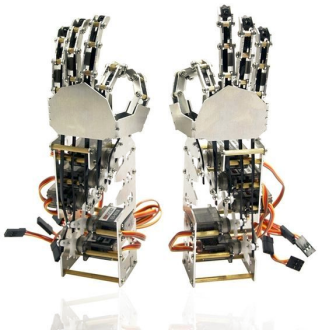


Figure 2. DIY Hand[3]

The Bionic hand was made almost entirely out of plastic and thus deemed unacceptable and is shown in Figure 3.

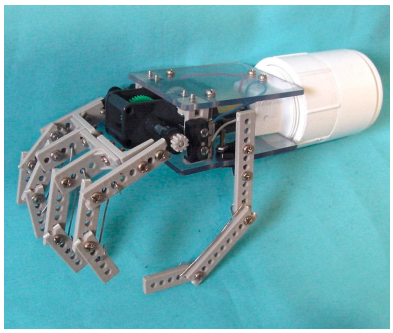


Figure 3. Bionic Hand[18]

After our hand decision, we needed to choose an arm that lifted at least 380g due to the weight of the hand. From our Arm Decision Table, we only had two options that could lift this weight. Both of them would be able to to lift the hand in addition to 120g. From here, the choice for the arm was based on the length. The Robotshop option was most similar to a human arm and was therefore selected as the final choice. The final arm choice is shown in Figure 4.

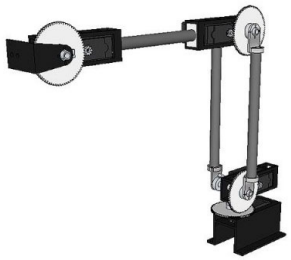


Figure 4. RobotShop 4 DOF Robot Arm (chosen arm)[17]

V. Modeling and Analysis of System

There was not a large amount of mechanical design to be done since the emphasis was placed on software development. The only mechanical component remaining was the final mount for the system since the arm and hand were purchased. The only requirements for the mount were that it could support the weight of the arm, hand, and a reasonable load (i.e. a pen or dry-erase marker) while remaining stationary.

The design process began with analyzing the requirements, specifically the parameters associated with the arm and hand. The full length of the arm and hand, when attached to each other is 33.5 inches, with a total mass of 2.3 kg, and with the maximum load being 130 grams. When fully extended, this creates a moment of 0.9 N-m, which would need to be supported by the base. Calculations for which can be seen in Appendix B. In order to optimize the amount of material used, the support needed to be a combination of expanding the tipping point further from the source of the moment and increasing the counterbalance weight to directly counteract the moment. From the weight of the mount, a counter-moment of 0.786 Nm was created. To deal with the rest of the moment arm, the mount was placed on the plywood sheet with two and a half feet of wood in front of the tipping point to keep the base stable. The other point of failure could present itself in the form of bolt shearing. The weight of the arm, hand, and load are attached to the mount by six bolts. These bolts shearing would cause the entire mount to fail, so some machine design analysis techniques were implemented in determining the factor of safety on the bolts based on the given load. In all, the shear stress on each bolt is 17.21 psi, which in comparison to the shear strength of the bolt, reveals a factor of safety of 1872. These calculations can be seen in Appendix B. As evidenced, bolt shearing is not an expected mode of failure.

Previous design considerations allowed for the triangular portion of the mount to be tiltable in order to place more of the downward, mass force into the frame instead of straight through the bolts holding the arm to the mount. This idea was decided against as it would add complexity to the design without a large benefit, as the max allowable angle the arm would be set at would be too small to affect the distribution of forces. It would also allow for more methods of failure as the number of moving parts increased.

Figure 5 below is a render of the final CAD model. It is a basic structure designed to manage the loads of the arm without tipping or warping. The two extruded T-slot pieces are fixed to the frame, but due to their geometry, allow the arm bracket to be repositioned anywhere along them, giving flexibility to the distance from the table to the arm's base. The final design of the mount will weigh twenty pounds between the metal frame and the wooden base. It occupies a three foot by three foot square, due to the wooden base, and reaches up 28 inches in height. The mount was welded to ensure structural stability between the butt joints of the metal segments.

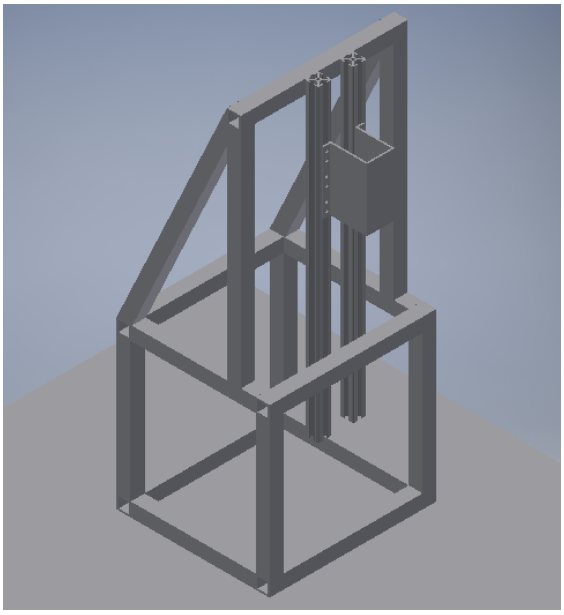


Figure 5: Final Arm Mount Render

There was little concern as to the structural stability of the purchased components as they were already parts of kits that were being commercially sold. All of the servos were rated for specific loads, and the system as a whole had a load rating so as long as these were not exceeded, nothing should need replacing. A backup plan was in place in the case that something broke, but no major modifications were needed for any of these parts.

It became clear after initial arm testing that the shoulder servos were struggling to support the full weight of the arm. This difficulty largely stems from the vertical orientation of the arm, instead of its intended horizontal configuration. A counterweight was designed to decrease the required torque load from the motors and was attached to the shoulder as shown in Figure 6.

Figure 6. Model of arm with counterweight (left most component)

The counterweight was machined out of aluminum and weighed 465 grams. This allowed for an additional 2.5 Nm of torque supplied by the counterweight when the arm was fully extended. The counterweight was attached using a threaded rod so that its position could be adjusted as needed using nuts on either side. A smaller counterweight was also manufactured to be used if a smaller moment arm was required. For counterweight torque calculations, see Appendix B.

VI. Controller Design, Analysis and Implementation

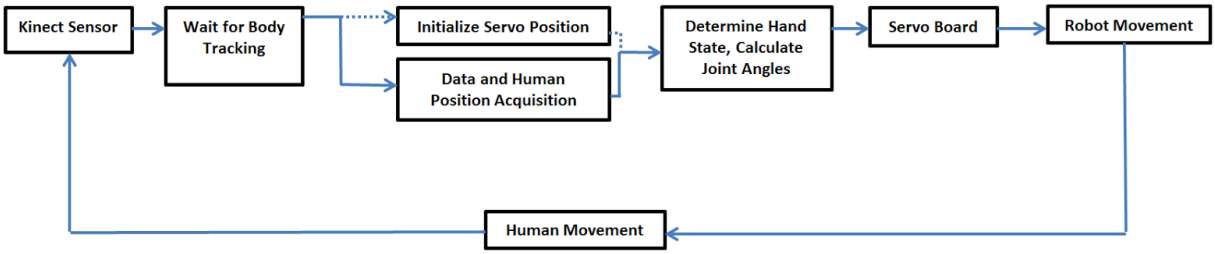


Figure 7: Control System Block Diagram

The controller was designed using a human-in-the-loop feedback paradigm, as shown in Figure 7 above. As the user's arm shifts, the robotic arm moves to match their position. A Kinect camera was used to detect human movement and Simulink was used to collected information from the kinect, calculate device position, and output position commands to the servo controller board. The controller board sends voltage to the servos and the servos respond. The bulk of the data processing, filtering, and calculation were performed in Simulink using MATLAB code function blocks.

Effective control was achieved primarily through software and human in the loop. Instead of implementing hardware filters, MATLAB code function blocks were used to achieve the same result. After the Kinect Sensor block outputted data in Simulink, the next function waited until a body had been detected by the Kinect to proceed. The general process for the control loop and software implementation can be found in the description below.

1. Initialize servo connection
    a. Send arm to initial, hardcoded position.
2. Check if a body is being detected by the Kinect. If it is, then:
    a. Collect a snapshot of joint locations in XYZ coordinates from Kinect function block.
    b. Calculate arm angles (both shoulder, elbow, and wrist) from XYZ joint positions using vector algebra. Also, check if the hand is in an open or closed state.
    c. Check with previous position. If the current angles detected were more than a 0.5 degree different than the previous position:
        i. Transform arm angles into servo position, and get hardcoded hand servo positions based on state.
    d. Send positions to servos board, servo board outputs voltage to servos.
3. Arm and hand response occurs.
4. Human views the robot's response and corrects their own position as necessary.
5. Kinect re-collects again, and the process loops starting from step two.

Simulink's Kinect Toolbox provides function blocks that enable the collection of standard Kinect data, such as an operator's joint positions in cartesian coordinates. The final Simulink block diagram used for this project is shown in Figure 8 below.

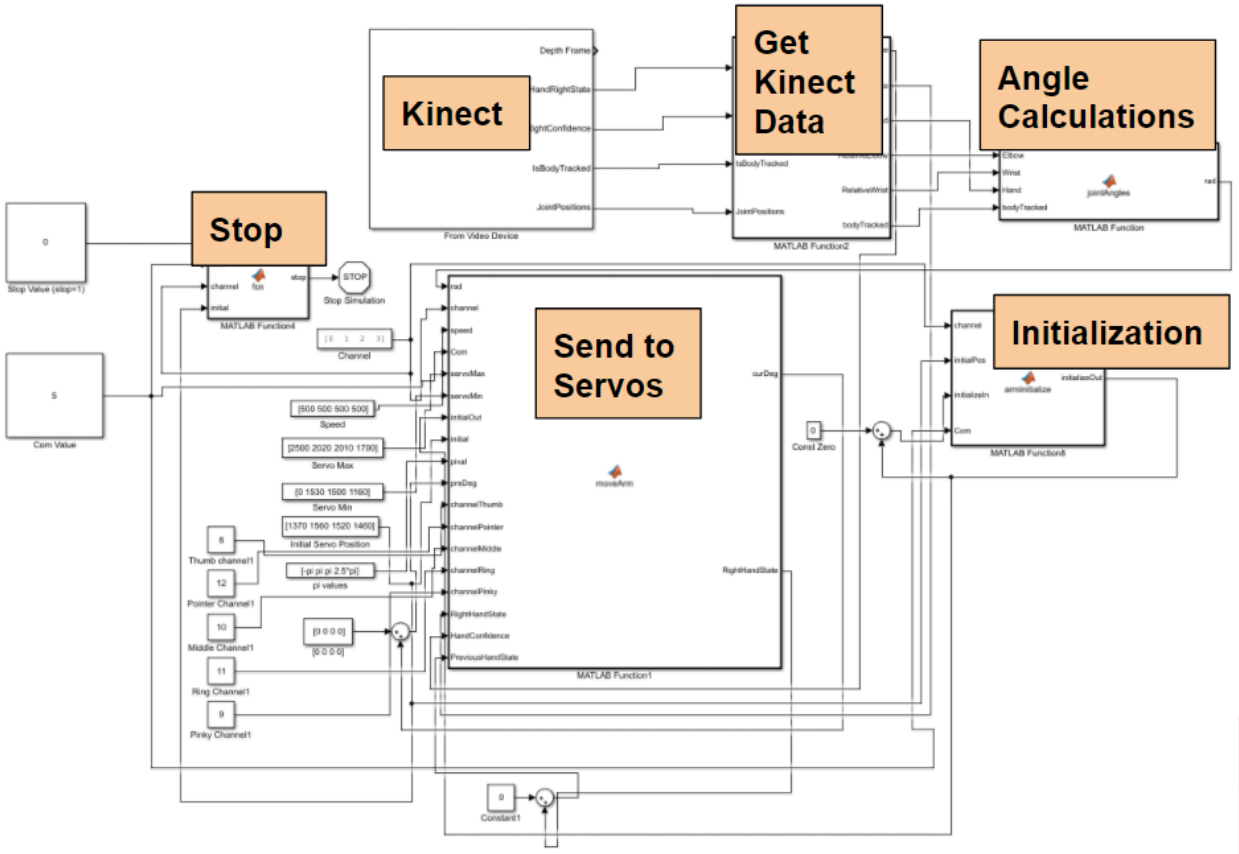


Figure 8: Simulink Diagram of System

VII. Design and Testing of Prototype

The plastic and metal surfaces of the hand needed to be covered to recreate the grip and compressibility of human skin in order to make the hand as lifelike as possible. A series of tests were conducted to determine the best combination of materials to achieve an appropriate amount of compliance and grip. To have quantifiable results, the testing procedure was organized in the following way: A wooden block was fixed with two eye-hooks to allow for a cheesecloth bag to be hung from it. The hand servos, excluding the thumb due to hand geometry, were brought slightly above the minimum PWM signal that causes the hand to shut. The timing for the PWM signal for each to remain closed is in Table 4.

Table 4: PWM Pulse Length for Closed Fingers [μs]

Index	Middle	Ring	Pinky
1440	1220	1295	1110

The materials tested during the above trial were affixed to the fingertips with rubber bands outside of the gripping area so the rubber bands would not interfere with the testing. The block was then placed within the hand and small weights were slowly added into the bag until the block slipped under the grip of the fingers. The weight of the block and bag was recorded at this point.

Table 5: Material Testing Results

		Grip								
		Yoga Mat	**Polyfil**	**Quilt Padding**	**Drawer Liner**	**Medical Tape**	**Gardening Gloves**	**Vinyl Gloves**	**Rubber Bands**	**Hot Glue**
	Nothing	298 g	-	x		167 g	x			254 g
Filling	**Yoga Mat**	299.5 g	x	x	462.5 g	x	x			300 g
	Polyfil	x	x	x	x	x	x	128 g	x	105.5 g
	Quilt Padding		x	179 g		x	x	231 g		178 g
	Drawer Liner		x	x	360.5 g	x	x			216 g
	Medical Tape	x	x	x	x	168.5 g	x	x	x	x
	Gardening Gloves	-	-	-	-	-	-	-	-	-
	Vinyl Gloves	-	-	-	-	-	-	-	-	-
	Rubber Bands	324.5 g	x			x	x		285.5 g	326.5 g
	Hot Glue	-	-	-	-	-	-	-	-	-

Table 5 shows the experimental results. The upper row show the materials used as the grip while the leftmost column represents the materials used as filling. Grip corresponds to one

layer of the material being used while filling is two layers; for blocks labeled as one material being both grip and filling, that means three layers were used. The dashes seen within the table represent the tests that were not conducted due to initial analysis of usefulness. For example, hot glue, once cooled, is a rigid material. It would not do well as a filling, which is meant to compress, so no tests were conducted with hot glue as a filling. The x marks represent tests being excluded after a material was used in at least one test and proved to be ineffective. An example of this is the self-stick medical tape. It seemed a reasonable option for filling due to its soft surface. In its first test, however, it was noticed that the tape compressed when in contact with the block, but never rebounded to its original shape, losing its compressive quality for further uses. The gardening gloves were removed from testing without actual test data because they constricted the movement of the hand and prematurely caused servos to overheat while installed.

Not all data was collected for both the sake of time as well as the fact that there was a clear trend towards one combination. The results of this testing concluded that the drawer liner and yoga mat materials were the best for use as the grip and filling materials, respectively. THe team decided the results were conclusive enough to continue on so these materials were then secured to the hand to be used in the accuracy tests described below.

The team devised a series of tests to inspect different functional outcomes and expectations within the system in order to test the interaction between the mechanical and software components of the prototype. Initial testing involved a qualitative perspective of basic movement to see if the arm would react in the same manner as the user was prompting. These tests included basic arm panning in three-dimensional space as well as opening and closing of the hand. The human and robot arm positions were collected in a number photographs and angles to assess the success of this behavior. The results of the photograph analysis were used to compare arm angles between the human and the robot. The hand state was also noted, and the number of times the hand was in the correct state (either open or closed) was recorded. The degree difference between the robot and the human was compared in both absolute differences and also percent error. These data can be in seen in Section IX.

Further testing included analyzing the accuracy and repeatability of the device. A marker was placed in the robot's hand, and a bullseye target was placed on the table below the arm. The human attempted to get to hit the center of the target, and the average distance from the target was measured and recorded.

Additional tests for accuracy were devised but not completed due to time restraints. However, the tests would be a great next step in the development of this prototype. The tests consist of drawing a variety of shapes and lines, both on the user end and with the robot with the goal of increasing shape difficulty to vary the number of motors required for the movement, as well as the timing between motors when operating simultaneously. The first, simple test is a straight line along an axis of the arm. From there, squares, triangles, and circles in any orientation can be tested, with the circle being the most difficult more than one motor is constantly running at the same time. To quantify the results, compare the drawing done by the

user and the drawing done by the robot. Maintaining orientation, as the user didn't create a perfect shape either, compare the variance between line shape, location, and size. For consistency, line up the starting points of the drawings and note the variance from that point.

VIII. Microcontroller Programming and Interfacing

A microcontroller was initially going to be used to run the code generated from compiling the Simulink model file. It was, however, concluded that a microcontroller was not necessary as the system response speed was slow enough that the additional speed from a microcontroller would not actually benefit the system. The Kinect and servo controller board were instead connected to an individual laptop, from which the Simulink program was run. Simulink was used to integrate the Kinect sensor data, servo initialization, filtering, calculations, and output to servo board. The programming in Simulink was largely achieved through a series of MATLAB function blocks. These blocks performed tasks like signal filtering, calculating joint angles from cartesian joint coordinates, calculating servo position from angles, and printing to the servo board via a serial connection. Arm movement was entirely based off of cartesian coordinates for arm joint locations which were then converted into joint angles; hand position was determined by discrete states detected by the Kinect.

Programming was broken into three subsections: Kinect input, path determination, and servo output. Kinect information was obtained using a Kinect function block found in the Kinect Toolbox for Simulink. The Kinect block was set to trigger a series of image captures from the camera, but only save the second-to-last frame. This frame was analyzed using the provided function getData(), which provided a structure of matrixes that saved cartesian coordinates for points throughout the body. The function also returned information about hand state (open, closed, lasso, or undetected). The desired joints positions of the shoulder, elbow, wrist, and hand palm, along with hand state were extracted from this structure.

Inverse kinematics were initially pursued for path determination, but was abandoned partway through the project due to function support difficulties. See Appendix C for a complete discussion of the inverse kinematics researched and developed for this project. The team transitioned to a "direct joint angle" method for determining and commanding the robotic arm positions. The program sets the shoulder to the origin [0,0,0], and calculates the elbow and wrist cartesian coordinates relative to the shoulder origin. Vector math was used to determine the appropriate angle for each servo using normalized arm vectors; these geometric calculations are shown in Figures 9 and 10.

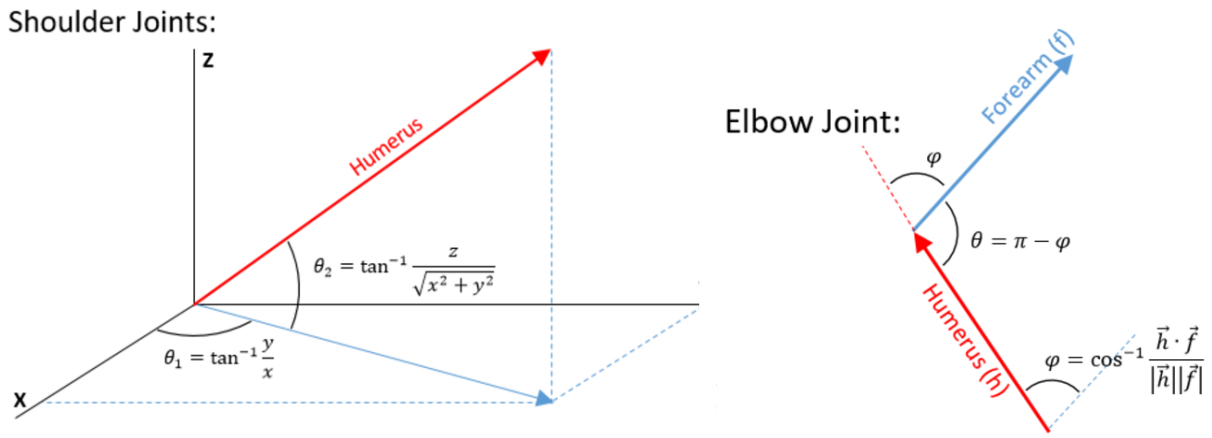


Figure 9. Vector math for shoulder and elbow angle.

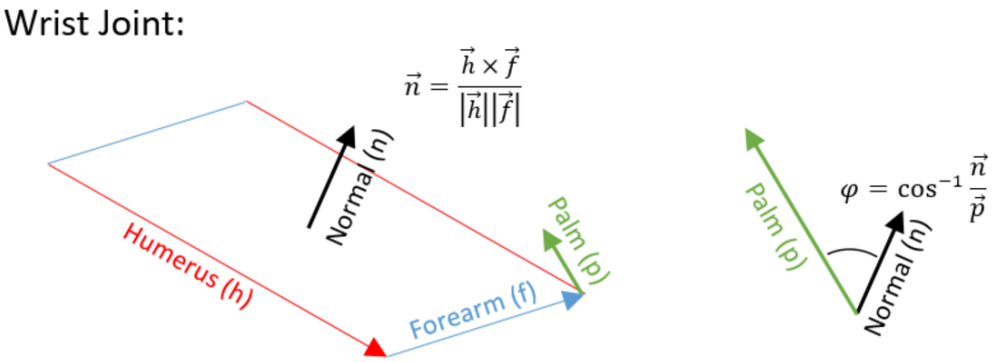


Figure 10. Vector math for hand rotation angle.

The angles were transformed into absolute servo positions after the necessary joint angles were calculated. The servos took in absolute position as numerical values (usually ranging 650-1250), which were linearly correlated with an angle (in degrees), using Equation (1).

$$Position_{current} = Position_{initial} + \pi * \theta \qquad (1)$$

The initial servo positions were set and hard coded based on individual servo center position and orientation for each motor. The servo position, target channel, and servo speed were sent to the servo board in a string format, and the servos responded. An example of servo command string is shown in Figure 11, where a position of 1600 is sent to channel 0 at a speed of 500.

$$'\#0P1600S500'$$

Figure 11. Example command string to servo controller.

Hard coded limits were also implemented to ensure that the robot did damage itself or the base. The limits were based on the geometry of the device and base, as well as the orientation of the servo wiring. The team was especially concerned that the arm could rotate around the first shoulder joint and subsequently disconnect or damage the arm's wiring.

IX. Results

Multiple tests were carried out to measure the device's ability to accurately and consistently replicate human movement. A series of photos were collected with the human and robot in the same positions for the first test. The human and robotic arm angles in the photos were calculated and the results were compared with the Kinect's measurements and calculations to determine the differences between the three steps. An example of this calculation is shown in Figure 12.

Angles (degrees):

Kinect	111
Human	108.43
Robot	98.23
Difference	10.2

Figure 12. Collected image and resulting angles.

See Appendix D for all analyzed images. This analysis was performed for a series of images, and the averages of the results are summarized in Table 6 below.

Table 6. Average Absolute and Percent difference between angles for Human, Kinect, and Robot.

Average Values	**Shoulder 1**		**Shoulder 2**		**Elbow**		**Wrist**	
Human to Kinect	1.7°	2.1%	7.8°	53.9%	4.6°	12.0%	12.3°	69.5%
Kinect to Robot	17.4°	20.8%	16.9°	75.7%	1.9°	3.0%	12.7°	69.9%
Robot to Human	15.8°	19.1%	24.7°	170.3%	2.6°	10.9%	6.5°	24.8%

Repeatability of the device's performance was also measured. A human user guided the robot to hit the target seven times and the distance from the mark to the bullseye was calculated. The results are shown in Figure 13 and summarized in Table 7.

Figure 13. Results of repeatability test.

Table 7. Summary of repeatability test results.

Sample Size	**Percent within target**	**Average distance to bullseye (cm)**	**Max distance (cm)**	**Min Distance (cm)**
7	57%	9.3	14.5	3.0

The amount of weight the device could successfully hold, without either dropping the object or the servos failing, was also an important consideration. The device was tested with the hand holding an increasing number of marbles in a mesh bag, and was judged on a how long it was able maintain its grip before the bag fell. The results are shown in Figure 14, where a time of -1 indicates that the bag never fell.

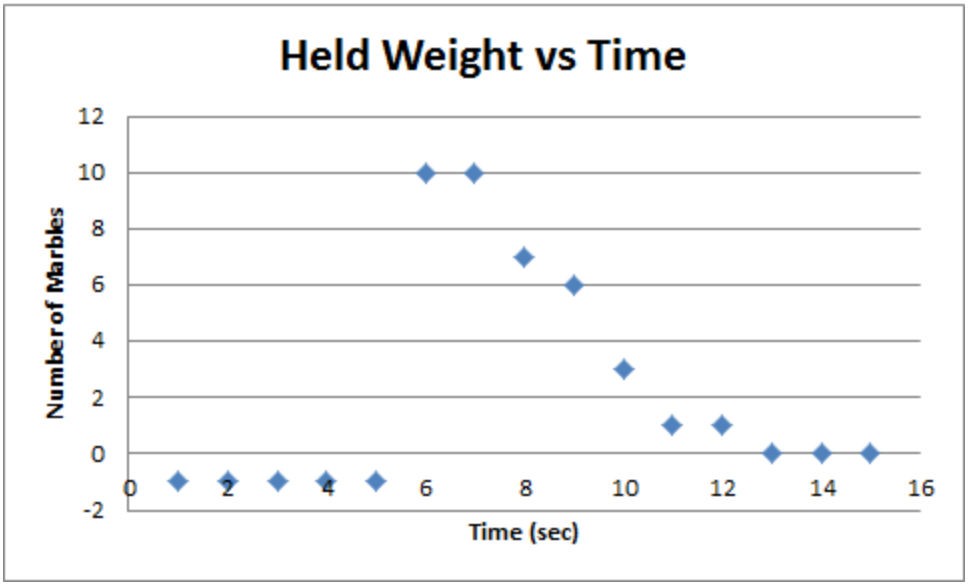


Figure 14. Marbles held by device hand vs time

Overall, the testing results were very informative regarding the functionality and accuracy of the device. Most joint movements tested were within twenty degrees of the human movement, which demonstrated generally good performance of the device. Different joints performed at different levels of accuracy as expected. The elbow was the most successful joint, which makes sense because the mathematical equations used to derive elbow angle relied on the simplest vector math and the joint positions used to create the needed vectors were the most reliable. The wrist was the hardest joint to obtain a consistently accurate angle for. This was largely due to the small vector created between the palm and the wrist in addition to the occasional corruption of palm or wrist data by finger position. Accurate wrist angles were even harder to obtain when the user closed their hand as the hand and wrist positions become even harder to distinguish.

The largest impact on our results was the degrees of freedom at the shoulder and in the forearm. Humans have three degrees of freedom at the shoulder, which also allow for bending and rotation at the elbow and wrist. Since our device has only two degrees of freedom at the shoulder and no rotator cuff, this had a significant impact on the robot's ability to mimic human movement exactly as most human arm movements involve rotator cuffs. The team had originally designed a series of tests involving drawing with the robot. These tests became impractical given the number of degrees of freedom that were available; it was extremely challenging for the robot to even maintain contact between the pen and paper, let alone actually produce a continuous drawing. The new series of tests allowed for more accurate performance measurements of the overall arm as well as each joint individually.

X. Budget

Our initial estimated budget was sectioned into subsystems including budgeting for the arm system, hand system, kinect system, miscellaneous materials, and the virtual reality (VR) system. Because we estimated that the VR system would cost around $880 and would be hard to implement in our time schedule, we decided that the VR system would best kept as a stretch goal. The team estimated in our College of Engineering grant application that we would need about $1000, in addition to the class's $200 budget, so that we would have a final budget of $1200 to complete this project. Of this $1200 budget, we attributed $740 for the arm system, $200 for the hand system, $155 for the Kinect system, and $50 for miscellaneous materials. This initial budget estimated that, of the $1200 budget, we would spend $1145 with $55 left for unexpected cost.

Our actual budget for the project was similar to our initial budget. A major difference was we did not need to buy most of the items we initially budgeted for in the miscellaneous materials subsystem because we did not need any solder or heat shrink. Also, we were provided electrical wires in the mechatronics lab. In addition, all of the materials we used for hand grip testing were scavenged by team members from items we already owned. As a result, we saved $50 that was budgeted for miscellaneous materials and another $42.86 on anticipating a higher cost for items we did buy. Another major difference between the actual and initial budget was we forgot to budget for the mounting system for the robot arm and hand. Luckily, the amount we initially budgeted for in the miscellaneous materials subsystem ($50) was similar to the amount we needed for the mounting subsystem ($67.92). All of this considered, our actual budget still fell below our initial budget of $1200 and we spent a total of $1129.96. The actual vs initial budget is shown in Table 8. The initial budget can be found in the "Anticipated Cost" column where items with a nonzero value were in our initial budget. The actual budget is shown in the "Actual cost" column. The values in the "Difference" column are green if the item was less than the initial budget while it is red if it was either more than the initial budget or not accounted for in the initial budget.

Table 8: Initial and Actual Project Budget

	Component	Anticipated Cost	Actual Cost	Quantity	Difference	
Arm	RobotShop M100RAK V2 Arm	$650.00	$649.99	1	$0.01	
	Lynxmotion SSC-32U Servo Controller	$45.00	$44.95	2	$0.10	
Hand	DIY 5DOF Robot Five-Finger Mechanical Paw	$100.00	$91.79	2	$16.42	
Camera	Kinect Camera	$110.00	$99.08	1	$10.92	
	Kinect Adaptor	$45.00	$39.49	1	$5.51	
Miscellaneous	Palm/Fingertip Materials	$20	$0	n/a	$20	
	Solder	$10	$0	n/a	$10	**Allowed Budget**
	Wires	$10	$0	n/a	$10	$1200
	Heat Shrink	$10	$0	n/a	$10	**Total Expected Cost**
Arm Mount	Arm Mount Material	$0.00	$42.15	n/a	($42.15)	$1145
	Servo Motor Extension Cables	$0.00	$8.59	3	($25.77)	**Total Expenditure (Excluding Labor)**
Labor	Estimated Wages	$30,608	$0	n/a	$30,608	**$1129.96**

The total theoretical cost of the project is an important consideration. Estimated labor costs can be developed using the average salary of a control engineer in Seattle, which is approximately $84,883 a year or around $40.81 an hour.[11] We can estimate that 5 of us spent an average of 15 hours a week for approximately 10 weeks on the project if we consider the time spent from winter quarter as well. The labor cost of our engineering team is around $30,608 based on this estimate. This figure includes paying ourselves for welding and machining, but does not include the cost to use a machine shop if we did not have free access to one with sufficient tools. Wages would be the largest expense, excluding potential workspace rental fees, and would increase the final cost of our project to around $31,738.

XI. Project Schedule

At the beginning of this project, the distribution of tasks fit into two groups: the hand team and the arm team. Eze, Nathan, and Jess made up the hand team while Ashley and Bryan made up the arm team. This was quickly changed because we discovered that the basic Kinect SDK and Simulink Toolbox functions would only provide coordinate data for the end of thumb, end of middle finger, and the middle of the hand. The team's previous attempt to obtain higher precision SDKs from Microsoft's Hand Pose research group were denied. We thought it would be very hard to do any programming in the hand to individually control each finger as a result. Instead, we used the hand state data returned by the Kinect, which only returns if the hand was open or closed in the sampled image, to open and close the hand accordingly. This got rid of the need to have an entire team focused on the hand. As such, we thought it made more sense to split into groups based on subject matter for each subsystem (i.e. programming, fabrication,etc.).

Through the development of the project, we divided into groups for input/output, data to servo position, and fabrication. Eze and Bryan headed the input and output group, focusing on Kinect data acquisition and sending the correct signals to the servos to match the input. Jess and Nathan focused on the functions in between the input and output. Originally, this involved very involved inverse kinematics. The method of inverse kinematics was implemented successfully in MATLAB, but could not be implemented in Simulink due to support issues. This required the group to work on vector math to follow human movement. Ashley was then left in charge of all required fabrication because of her extensive machining experience. This meant machining and designing the arm and hand mounts as well as doing materials testing for the hand. Although we were split into these groups, most everyone contributed to each part of the final product. Specifically, everyone had to work in the machine shop at one point for the project.

We were able to mostly follow the initial Gantt chart for this quarter. There were, however, some differences between our planned schedule and our actual schedule. In particular, the programming part of the project was very extensive. This was, in large part, because of the difficulty in implementing code in Simulink that we previously got to work in MATLAB (as mentioned above). This happened both in our code to send signals to the servo controller and in our code for the inverse kinematics. A significant amount of time was spent on trying to find workarounds for the various roadblocks we encountered in implementing our code in Simulink. The servo controller requires the input to be an ASCII string, which we found out is not very compatible with standard Simulink serial output methods. We had to use extrinsic functions "sprintf" and "fprintf" in Simulink to fix this issue. This created a new problem, however, because these extrinsic functions severely slow down the execution of the code and resulted in about a 5-10 second delay for human to robot movement. Use of extrinsic functions in the Simulink file stops the program from generating executable code; instead, the model is compiled and run through the MATLAB interpreter where required. This process appeared to take a significant amount of the computational time for our given system. We tried many different

methods to replace the extrinsic functions so that Simulink would properly compile and generate the necessary code for our system, but were ultimately not able to. The programming part of this project took approximately two to three weeks longer than originally anticipated for the aforementioned reasons.

Another major difference between the initial and actual Gantt charts is the time spent on the hand and arm mounts. This ended up being because of a switch in our priorities during the project. We were able to quickly create preliminary arm and hand mounts that were functional enough to use for program writing. Once these were completed, we prioritized working on the programming and finished the final arm and hand mounts later. This also gave us more time to focus on designing good mounts. This resulted in our final mounts being completed around six weeks later than expected but this did not affect our project.

The last major difference between the initial and actual Gantt charts is the time for the finger and palm material testing. This took around four and a half weeks longer than anticipated because the small servos would overheat so that we could not carry out tests as quickly as anticipated. Also, the material testing was ultimately not as important for the project outcome so we focused more of our energy on the programming aspects. The initial and actual Gantt charts can be found in Appendix E for comparison. The only difference is the actual Gantt chart has long red lines for the tasks that took longer than expected with an x on the completion date. For the arm and hand mounts, there are 2 x's indicating when we finished both the preliminary and final arm and hand mounts.

We would not change how the work was designated for future projects, but we would change some things on our Gantt chart. Particularly, we would plan for working on the programming almost the entire quarter because this is where a majority of the work was spent.

XII. List of Parts

This part list only includes the essential items to build our project and not any items we ordered for redundancy. The prices also reflect the source cost, not what we actually paid from the budget. These are summarized in Table 9.

Table 9: Project Part List

Item	Quantity	Cost ($)	Vendor Name	Item Source
Arm System				
RobotShop M100RAK V2 Arm	1	649.99	RobotShop	http://www.robotshop.com/en/robotshop-m100rak-v2-modular-robotic-arm-kit-no-electronics.html
Lynxmotion SSC-32U USB Servo Controller	1	44.95	RobotShop	http://www.robotshop.com/en/lynxmotion-ssc-32u-usb-servo-controller.html
Hand System				
DIY 5DOF Robot Five Fingers Metal Mechanical Paw	1	89.99	Banggood	https://www.banggood.com/DIY-5DOF-Robot-Five-Fingers-Metal-Mechanical-Paw-Left-and-Right-Hand-p-1099018.html
Kinect System				
Kinect V2	1	83.09	Amazon	https://www.amazon.com/Xbox-One-Kinect-Sensor/dp/B00INAX3Q2/ref=sr_1_1?ie=UTF8&qid=1495007483&sr=8-1&keywords=kinect
Kinect Windows Adaptor	1	39.99	Amazon	https://www.amazon.com/Xbox-Kinect-Adapter-One-Windows-10/dp/B01GVE4YB4/ref=pd_sim_63_1?_encoding=UTF8&pd_rd_i=B01GVE4YB4&pd_rd_r=WNZYPW96K7PTZ4DFEW4T&pd_rd_w=EIe6N&pd_rd_wg=NiNnm&psc=1&refRID=WNZYPW96K7PTZ4DFEW4T
Mounting System				
Servo Extension Cable	3	25.77	Amazon	https://www.amazon.com/gp/product/B00MIQRN2C/ref=oh_aui_detailpage_o01_s00?ie=UTF8&psc=1
Metal for Arm Mount (length=18 ft)	n/a	42.15	OnlineMetals	http://www.onlinemetals.com/merchant.cfm?pid=18014&step=4&showunits=inches&id=1270&top_cat=60

XIII. References

1. About IMAS. (n.d.). Retrieved June 08, 2017, from https://www.mineactionstandards.org/about/about-imas/
2. AISI 1018 Steel, cold drawn. (n.d.). Retrieved June 01, 2017, from http://www.matweb.com/search/datasheet_print.aspx?matguid=3a9cc570fbb24d119f08db22a53e2421
3. Banggood. (n.d.). DIY 5DOF Robot Five Fingers Metal Manipulator Arm Left and Right Hand QDS-1601. Retrieved June 08, 2017, from https://www.banggood.com/DIY-5DOF-Robot-Five-Fingers-Metal-Manipulator-Arm-Left-and-Right-Hand-QDS-1601-p-1063112.html
4. Buss, Samuel. (2000) *Introduction to Inverse Kinematics*. La Jolla, CA: University of California, San Diego.
5. Current Use Of Military Robots Briefly Explained And Discussed. (n.d.). Retrieved May 18, 2017, from http://www.armyofrobots.com/current-use-military.html
6. Dijk, M., Wells, P., & Kemp, R. (2016). Will the momentum of the electric car last? Testing an hypothesis on disruptive innovation. Technological Forecasting and Social Change, 105, 77-88. doi:10.1016/j.techfore.2016.01.013
7. DIY 5DOF Robot Five Fingers Metal Mechanical Paw Left and Right Hand. (n.d.). Retrieved June 08, 2017, from http://alexnld.com/product/diy-5dof-robot-five-fingers-metal-mechanical-paw-left-and-right-hand/?gclid=CN_k09Koz9ECFRB4fgodcgIIXg
8. Environmental Protection Agency . (2013). Solders in Electronics: A Life-Cycle Assessment Summary. Retrieved June 8, 2017, from https://www.epa.gov/sites/production/files/2013-12/documents/lead_free_solder_lca_summary.pdf
9. Frey, C. B., & Osborne, M. A. (2013, September 17). THE FUTURE OF EMPLOYMENT: HOW SUSCEPTIBLE ARE JOBS TO COMPUTERISATION? . Retrieved June 8, 2017, from http://www.oxfordmartin.ox.ac.uk/downloads/academic/The_Future_of_Employment.pdf
10. Fully Articulated Hand Tracking. (n.d.). Retrieved June 08, 2017, from https://www.microsoft.com/en-us/research/project/fully-articulated-hand-tracking/
11. GlassDoor.com. (n.d.). Salary: Controls Engineer in Seattle, WA. Retrieved June 06, 2017, from https://www.glassdoor.com/Salaries/seattle-controls-engineer-salary-SRCH_IL.0,7_IM781_KO8,25.htm
12. Harrison, C. (2014). A review of automation in manufacturing illustrated by a case study on mixed-mode hot forging. EDP Sciences, 1(15). doi:10.1051/mfreview/201401

13. Lentati, S. (2015, May 05). The man who cut out his own appendix. Retrieved June 08, 2017, from http://www.bbc.com/news/magazine-32481442
14. McCarthy, Michael. (2000) *Geometric Design of Linkage*. Irvine, CA: University of California, Irvine.
15. OPSLab. (2013, December 23). NASA Robot Arm Control with Kinect. Retrieved June 08, 2017, from https://www.youtube.com/watch?v=pqNC72fgetc&feature=youtu.be
16. Rigid Body Tree Robot Model. (n.d.). Retrieved May 18, 2017, from https://www.mathworks.com/help/robotics/ug/rigid-body-tree-robot-model.html
17. Robotshop. (n.d.). Retrieved June 8, 2017, from http://www.robotshop.com/eu/en/robotshop-m100rak-v2-modular-robotic-arm-kit-no-electronics.html
18. Scientificsonline. (n.d.). Bionic Robotic Hand Kit. Retrieved June 08, 2017, from https://www.scientificsonline.com/product/bionic-robotic-hand-kit?gclid=CPWc9s3J8tECFZCcfgodzjECLg
19. Shigley, Joseph. (1981) *Theory of Machines and Mechanisms*. Madison, Wisconsin: McGraw-Hill Book Company.
20. Siciliano, Bruno. (1999) *The Tricet Robot: Inverse Kinematics, manipulability, and closed-loop kinematic algorithm*. Napoli, Italy: PRISMA Lab.
21. UNITED STATES DEPARTMENT OF LABOR. (n.d.). Retrieved May 18, 2017, from https://www.osha.gov/pls/imis/AccidentSearch.search?acc_keyword=%22Machinist%22&keyword_list=on
22. UW Capstone Discussion [Personal interview]. (2017, March 23).

Appendix A: Relevant Codes and Standards

Relevant codes and standards researched for the project are summarized below:

- ISO 13402
  - metals used in surgical applications
- AAMI TIR.38, AAMI TIR.57, and AAMI TIR.65
  - medical device safety, security, and sustainability, specifically as is relates to device lifecycle and material usage.
- ANSI 11135 and ANSI 11737
  - cleaning and treatment of medical devices
- UL 674 and UL 698
  - use of electrical motors and generators and industrial control in division 1 hazardous locations.
- UL 877
  - Circuit breakers in hazardous locations
- UL 894
  - Switches in hazardous locations
- ANSI R 15.01-1, 15.02, 15.05-1, 15.05-2, 15.05-3, 15.06.
  - Components and use of industrial robots and robots systems

Appendix B: Arm Diagrams and Machine Design Calculations

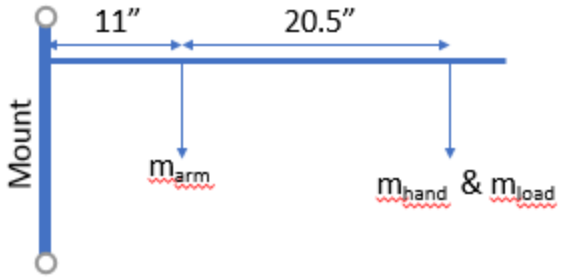


Figure B.1 Free body diagram of arm on mount

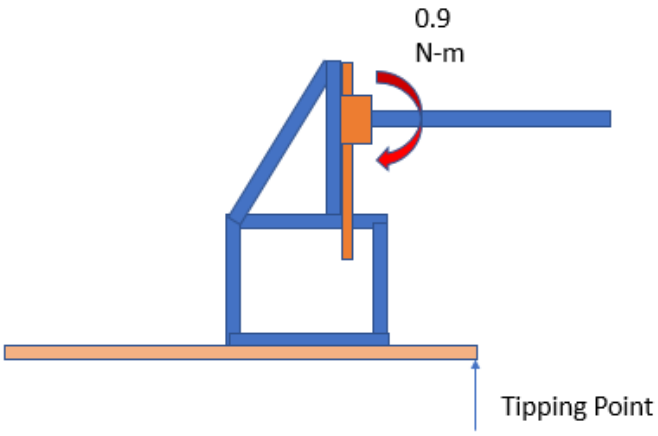


Figure B.2 Free body diagram of final mount

$\Sigma M_{arm} = m_{arm} * x_1 + m_{hand/load} * x_2 = 0.279m(1.792kg) + 0.8m(0.505kg) = 0.904Nm$

$\Sigma M_{mount} = m_{mount} * y = 0.102m(7.71kg) = 0.786Nm$

Where x’s and y’s are distances from center of mass to interface point. Center of masses estimated.

Bolt Shearing Calculations

The following calculations are analyzing the shear stress on the bolts created by the arm, hand and load.

$$\tau = \frac{F}{A} = \frac{m*g/_{\#bolts}}{\pi*r^2} \quad \text{(Eq. B.1)}$$

$$n = \frac{Allowable\ Stress}{Actual\ Stress} = 1872 \quad \text{(Eq. B.2)}$$

Where mg = 5.07 lbs, #bolts = 6, r = 0.125in (system parameters). For the bolt material, the minimum tensile strength is 53700 psi.[2] Shear strength is generally 60% of tensile strength so the allowable stress in the factor of safety calculation is 32220 psi.

Therefore, $\tau = 17.21\ psi$ and $n = 1872$

Counterweight torque calculation:

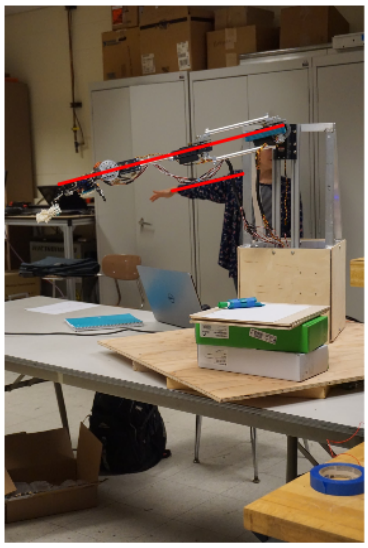
$$Max\ Torque = Fxr = (0.458\ kg)\ (9.314\ m/s^s)\ x\ (0.6\ m) = 2.5\ Nm$$

Appendix C: Inverse Kinematics of Project

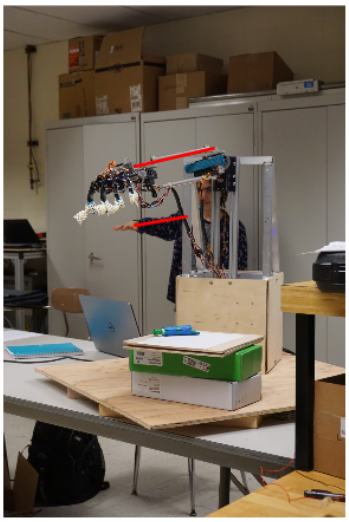
The initial goal was to determine an appropriate path for the arm to move along based on inverse kinematics. This method uses the inverse Jacobian matrix of the system to interpolate a smooth path for the robot's end-effector to reach the indicated target.[4,14,19,20] The indicated target, in this case, is the position of the human's palm in three-space. The robot was constructed in MATLAB as a rigid body tree using Robotics System Toolbox functions.[16] This method defines the robotic arm as a series of joints and rigid bodies, with defined geometry and geometric constraints as well as parent-child relationships. The target position was defined as an additional rigid body extending from the base of the structure. The Robotic System Toolbox function for "Generalized Inverse Kinematics" was utilized to minimize the path between the end effector (the hand of the robot) and the target position (the hand of the person). The output from this function was a matrix of joint angles at a specified number of waypoints for each servo that created a smooth transition from the current position to the target position.

This method worked perfectly in MATLAB simulation, but difficulty arose when trying to port this over to a Simulink function block. Simulink was determined to be the best option for both retrieving Kinect input and outputting joint positions to the servos. We attempted to feed the current position of the servos and the target position of the hand to a MATLAB function block containing the inverse kinematics trajectory calculation. The Robotics System Toolbox, however, only has code-generation supported functions for the Robot Operating System (ROS). This meant that all of the rigid body tree definitions and inverse kinematic functions used in the team's MATLAB implementation had to be defined as "extrinsic" for Simulink to recognize them. When functions are defined as extrinsic in Simulink, the data results are contained in an "mxArray". This type of array cannot be accessed by calling specific rows and columns and we therefore could not access the trajectory data we needed. We attempted to bypass this problem in several ways. The first was to define the rigid bodies and joints as global variables, but the data type was unsupported by Simulink. Then, we attempted to define them in the MATLAB workspace by separating the robot definition from the inverse kinematics, running the robot definition, and accessing the MATLAB workspace from Simulink, but again the data type was unsupported. We tried to rewrite the object definition for joints and rigid bodies so that they appeared as structures, which Simulink could process, but the inverse kinematics functions could not operate with structure input. Finally, we attempted the utilize the "Interpreted MATLAB Function" block in Simulink. Unlike the standard MATLAB Function block, this block does not generate code and relies entirely on MATLAB processing. Because our system does not require C code to function, this was a viable option. The limiting factor, however, was the run-time. This single block ran successfully and produced viable results, but it took over 30 minutes to do so. The inverse kinematics approach was subsequently abandoned due to project time limitations.

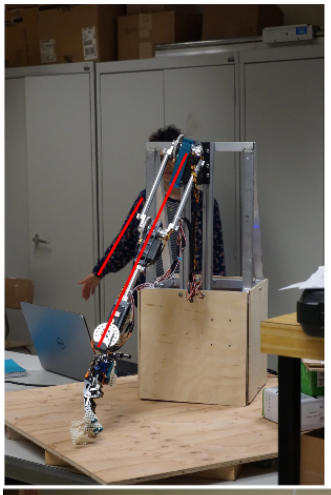
Appendix D: Analyzed Images



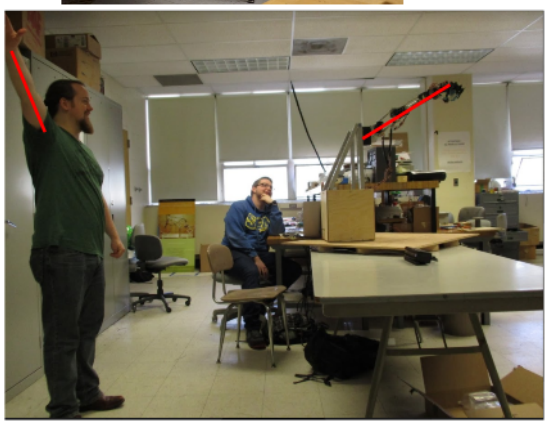
Angles:
- Human arm: 75.2 deg
- Robot arm: 76.6 deg



Angles:
- Human arm: 77.0 deg
- Robot arm: 79.8 deg



Angles:
- Human arm: 29.8 deg
- Robot arm: 24.5 deg



Angles:
- Human arm: 67.2 deg
- Robot arm: 63.6 deg

Angles:
- Human arm: 41.8 deg
- Robot arm: 65.2 deg

Angles:
- Human arm: 32.2 deg
- Robot arm: 51.1 deg

Angles:
- Human arm: 81.7 deg
- Robot arm: 85.6 deg

Angles:
- Human arm: 90 deg
- Robot arm: 90 deg

Angles:
- Human arm: 10.5 deg
- Robot arm: 5.3 deg

Appendix E: Scheduling of Tasks

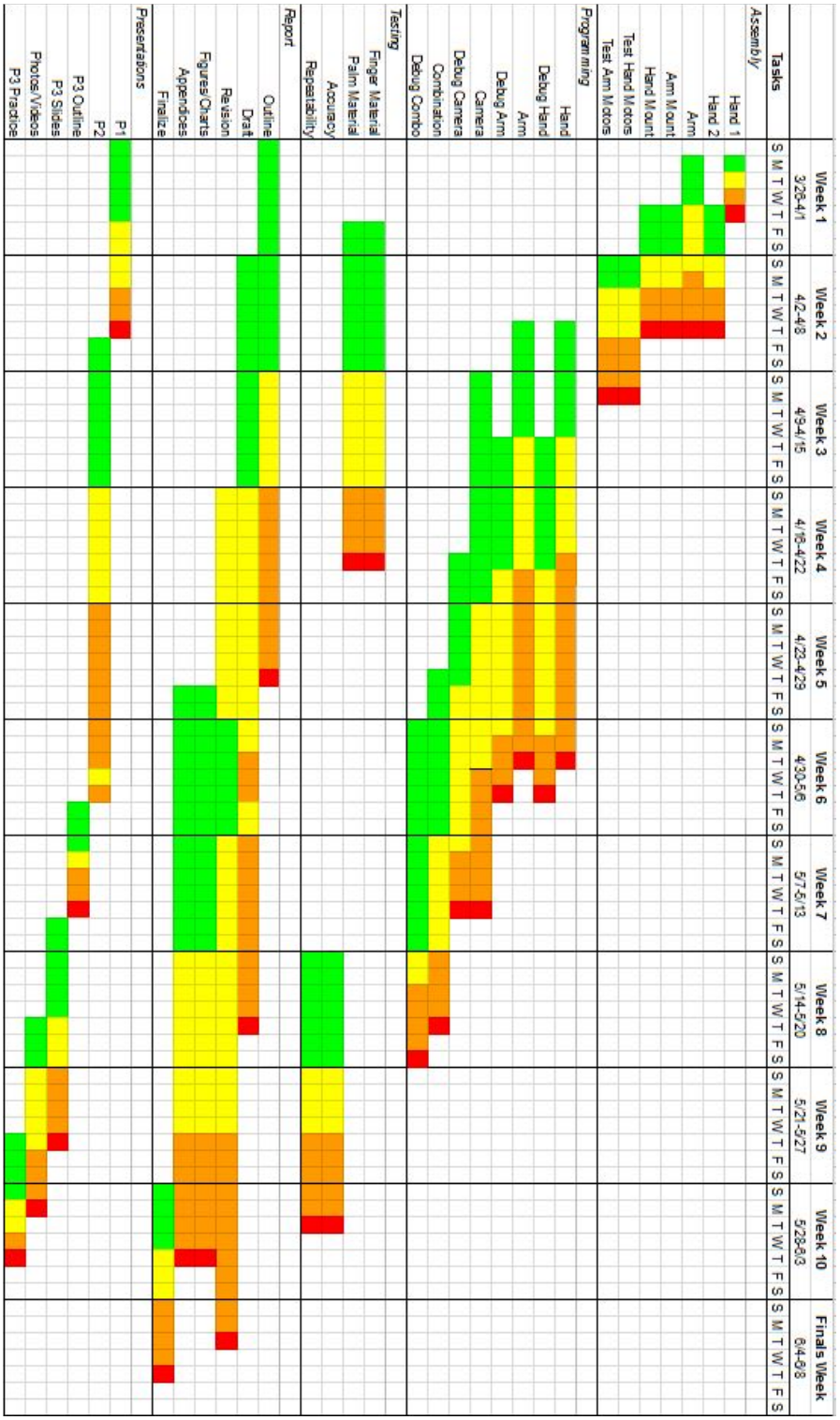


Figure E.1 Initial Gantt chart of project

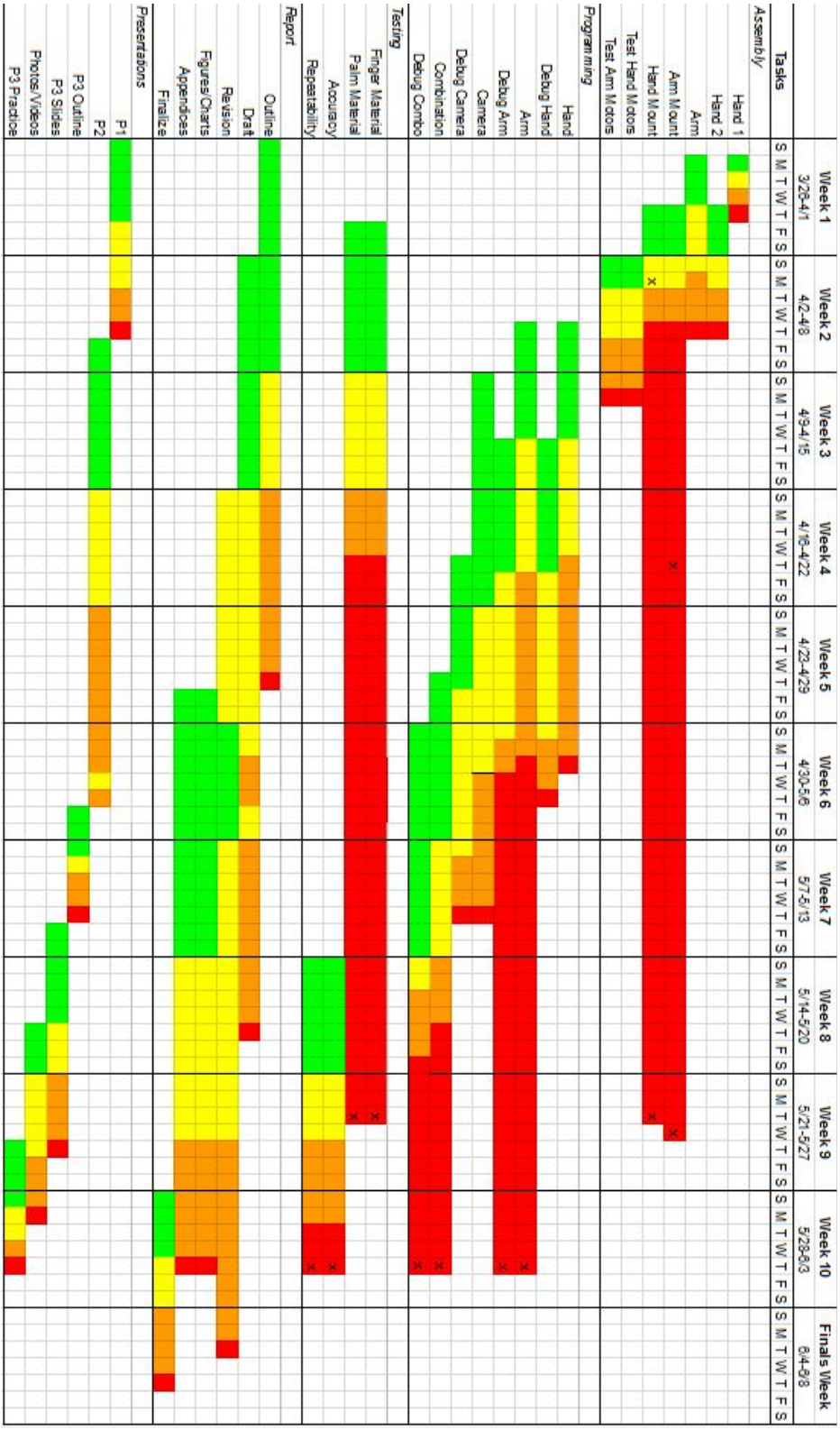


Figure E.2 Final Gantt chart of project

Appendix F: Operation Guide

Part 1: File Setup

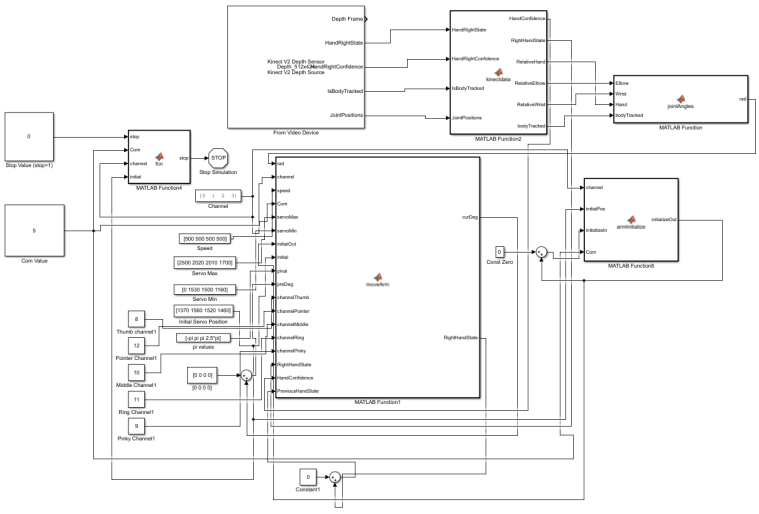
The file that operates the robot is a Simulink file. When this project was done, MATLAB 2017A and Simulink version 8.9 were used. In addition, a few add on toolboxes were required. The Image Acquisition Toolbox is required to use the Kinect. Also, the Image Acquisition Toolbox Support Package for Kinect for windows sensor is required. In order to install these, open MATLAB. Click on the "Home" tab on top, and click on the "Add-Ons" drop down box. The Image Acquisition Toolbox can be downloaded by clicking "Get Add-Ons" and searching for the toolbox. Similarly, the Kinect hardware package can be found by clicking the "Get Hardware Support Packages" and searching for it.

Part 2: Setting up the Hardware (Requires laptop with 3.0 USB and a regular USB/ Power supply that can output 6V DC with greater than 3A max current).

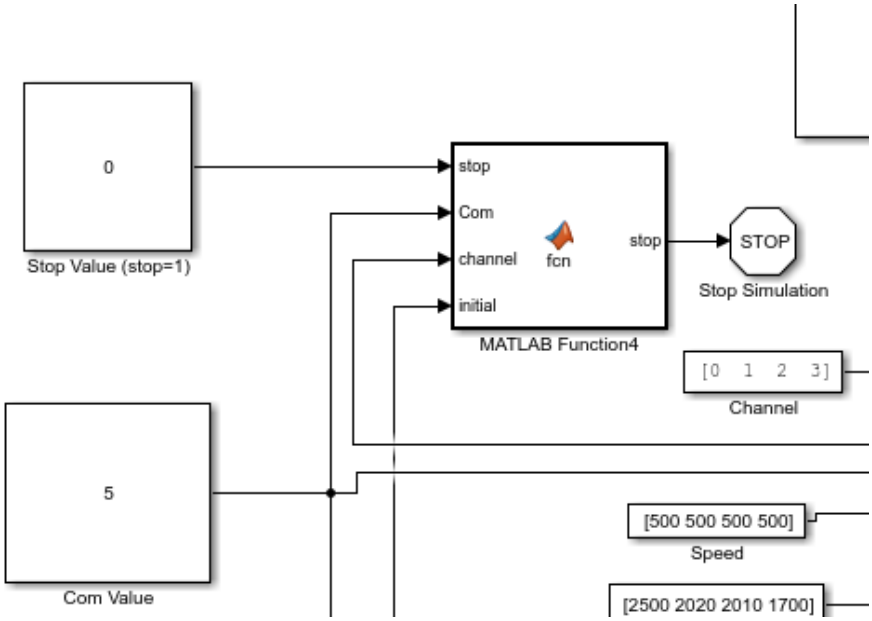
1. The Red lead coming out the back of the Robot Arm panel is the positive lead. This lead must be hooked up to 6V DC power supply for the robot to function. The black lead is the negative lead. Turn the power supply on and hook up the leads. *NOTE: The arm can draw in excess of 3.5A and when a 3A power supply is used, it will not operate correctly.*
2. The kinect needs to be plugged into a wall outlet and the USB for the kinect requires the use of a USB 3.0 Port for it to work. Plug in the Kinect.
3. The servo controller wire is the USB cord coming out the back of the robot arm. Plug the servo controller into your laptop. This can be a regular USB.

Part 3:Running the code

1. Open the "ControlArm" Simulink file. It should look like the following:



2. Notice on the top left a few boxes that look like the following:



3. The Com Value is currently 5. This value will need to be changed depending on what the computer assigns the com value to be for the servo controller. The Stop Value is also currently 0. If you want to stop the execution, change this value to 1 and allow the file to end before removing power. Make sure this value is 0 before the file is run.

4. Press the run button at the top of simulink: 
5. The file may give a Com error and tell you which Com to use. If it does, change the Com Value shown above to the appropriate Com that is available and rerun the file.
6. The file may also give a Kinect (unable to detect input video) error. If this happens, see the troubleshooting section of this appendix.
7. When the file runs, the arm will move to its upright neutral position until it receives input from the kinect camera.
8. Stand in front (facing) the Kinect and move your right arm. The robot will follow with about a 7 second delay. You can also change your hand to open, close or lasso. Lasso is when your pointer and pinky finger are out only- “rock on” hand gesture. *NOTE:It is suggested you operate the arm at max 10 mins at a time. Also, it is suggested you allow the motors to rest 3X longer than operation time (10mins operation = 30 mins rest).*
9. To stop the file, set the Stop Value to 1 and allow the robot to get to its neutral position.
10. When cutting the power, hold the arm so it doesn't smash into the mount.

Troubleshooting

Problem: Cannot detect video input from the Kinect

Solution: Try connecting and unconnecting the Kinect from the Microsoft Kinect Studio software (available at link) and rerunning the simulink file. If this does not work, try closing and opening Simulink or restarting your computer. Make sure the Kinect is plugged into power and it is plugged into a USB 3.0. This error seems somewhat random.

Problem: No COM/Ports are Available

Solution: Close MATLAB/Simulink and reopen. This error occurs when the file stops execution without setting the stop value to 1.

Problem: The arm goes to the initial position and the file stops execution.

Solution: The Stop Value is probably 1. Change it back to 0 and rerun.

Improvements

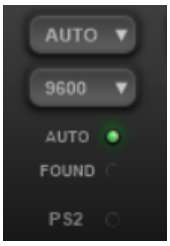
The major improvement to be made is to eliminate the time delay. The main thing causing the delay is the extrinsic functions needed in the simulink code. You can view the code by double clicking on the function blocks. Currently, the extrinsic functions are used to create a string output, which is required for the servo sequencer. If this can be done another way or with a simulink block, it will dramatically improve how quick the robot responds.

Jogging the Arm without Kinect Input

To jog the motors individually without kinect input, use the free servo sequencer software (available at link).

1. After downloading and opening the software, it will appear like:

2. Change the Baud Rate in the bottom right to 9600 and the Com to Auto as shown:



3. The servo values range from 500-2500 and correspond to set servo positions. Turn on the appropriate servos by clicking the check marks. The setup for the robot arm is:

Motor	Servo #
Shoulder 1	0
Shoulder 2	1
Elbow	2
Wrist	3
Thumb	8
Pinky Finger	9
Middle Finger	10
Ring Finger	11
Pointer Finger	12

4. Jog each servo by clicking the up or down buttons on the servo sequencer **NOTE if the servos are sent around 600 or around 2400, they will spin continuously and will damage the robot.*