Manuel’s Arm-y

MAE162E Project Report

Group 1

Shane Monney

Phillip Ng

Alexander Pak

Lucas Ton

Matthew Quinn

Jason Kong

Jonathan Kuriakose

June 16th 2023

Abstract

Manuel is a manually controlled robot that was designed to perform gestures with its arm and spell in ASL with its hand. The arm was designed to mimic the full 7 degrees of freedom that a human arm has. The hand and fingers were designed to perform finger spelling, applying only enough degrees of freedom to do so. These objectives were to be completed within 5 months with a $600 budget plus part donations from the RoMeLa lab. Unfortunately, these ambitious goals were not fully met. Manuel managed to perform exactly one gesture with its arm and hand: flipping someone off.

Table of Contents

List of Figures

1 Project Scope

1.1 Problem Statement

American Sign Language is used by the deaf in order to communicate without using sound. Unlike other languages, sign language is an important tool for accessibility for the deaf. During public announcements, there is often an ASL translator that translates messages in real time. However, this is only done at the highest levels, where human resources are plentiful. However, this is a luxury for lower levels of communication. Our project idea aims to bridge that gap, by making an affordable robot that can communicate in ASL.

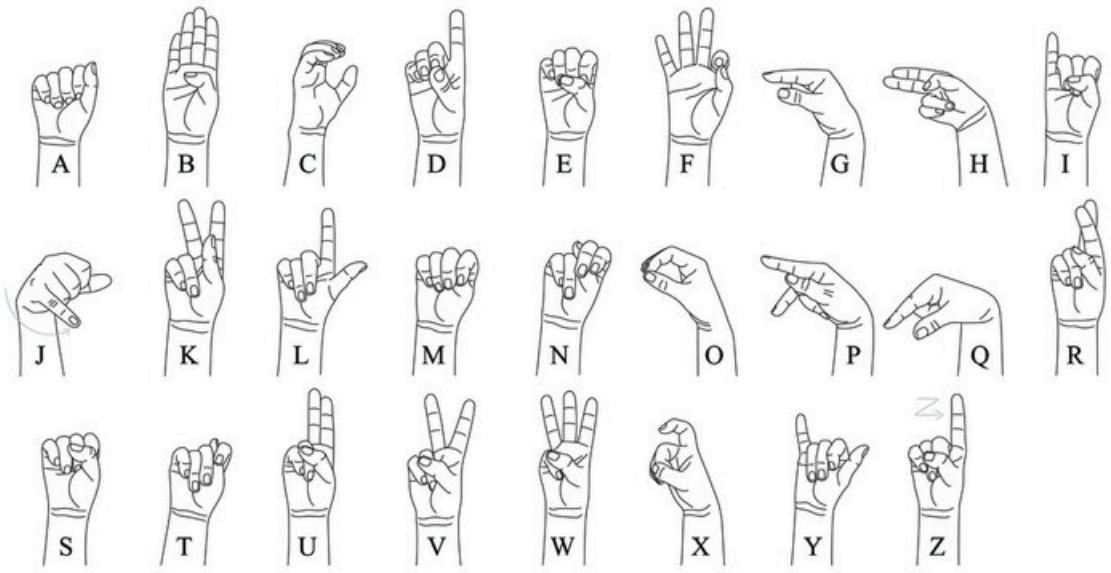


Figure 1: ASL Spelling

1.2 Client Identification and Recognition of Need

An ASL robot would be used for communication. Our robot must look human-like in order for communication to be effective, since ASL was constructed with the human body in mind. Our main clients would be government and public services, who would need to make announcements to the general public. This product would need to be mass produced, hence our material choices must keep abundance and low cost in mind, while being easy to manufacture using current mass manufacturing processes.

In order to make our robot more efficient, we must minimize the number of DOFs we need to actuate in order to use as few motors as possible. This will keep our costs down, since the motors are likely to be one of the biggest costs. The fingers DOF is of particular interest since the hand has a small design space to work with. Eliminating as many DOFs here is beneficial since each DOF for one finger means 5 motors for the hand. The following is the DOF required for each finger:

1. Thumb - Two degrees of freedom (Left/Right - In/Out)
2. Index - Two degrees of freedom (Up/down - Left/Right)
3. Middle, Ring, Pinky - One degree of freedom (Up/down)

The DOFs were chosen after considering the various finger movements required to get ASL spelling to the extent feasible for our timeline. These movements can be seen in Figure 1. Some gestures may be difficult to achieve using these DOFs, however we have decided to compromise on these gestures in order to get something close to it.

When it comes to motor actuation in the hands, we will not require strong motors since we are not designing it to hold anything, all we require it to do is to move the fingers in a certain direction. The fingers are light; according to Solidworks, the index finger subassembly weighs a mere 10.2g. Moreover, friction is greatly reduced due to the addition of dowel pins between each finger joint, decreasing the required force needed to actuate each finger. Hence, we decided to choose SG90 Positional Servo Motors and FS90R Continuous Servo Motors for the finger actuation in the hand. They are common and cheap, and output a stall torque of 1.4 kg-cm, which is likely enough to actuate the fingers. These motors are often found in amateur Arduino Kits, which many of us already own. Moreover, the servos are lightweight (9g each) and work well with available microcontrollers.They are compact (22.5 x 12 x 30mm) and are thus ideal for placement in the hand or forearm, where space is limited.

For the arms, while 6 DOF is all that is needed to be able to move the hand to any position we want, we concluded that 7 DOF is still needed in order to get redundant configurations for the same position. This is important because our human arm works in a similar fashion and without it, our human arm would look far less human-like, making it difficult for it to do ASL.

For the arm actuation, it will be more important for us to choose stronger motors since it will need to move the weight of the entire arm, including the motors used for the fingers. We plan on using Dynamixel motors, however, due to the varying torque requirements for each part of the arm, we have chosen different models for each actuation. Further discussion on this design can be found in section 4.

For material choices, we have decided to use PLA for the hand parts, while using aluminum rods as dowel pins to join them together. This allows us to rapidly prototype designs for the hand in a cheap and time efficient manner. For the arm, we will be using aluminum rods due to their strength and availability.

1.3 Project Goals and Objectives

Our main goals for this project began as follows:

- Manufacture and assemble a hand and arm that can mimic human arm movements.
- Actuate the hand and arm so that the system can be controlled using microcontrollers.
- Code our hands to do ASL spelling and play rock paper scissors.

Our stretch goals were as follows:

- Implement Battery Operation (as compared to drawing power from the wall)
- Code Manuel to express complex sentences.
- Manufacture a neck to express human emotions on our humanoid robot.

We accomplished the following:

- Manufactured and assembled a hand and arm that can mimic human arm movements and close its fingers
- Actuate the hand and arm so that the system can be controlled using microcontrollers
- Coded our hand and arm to individually close each finger, give onlookers the middle finger, a closed fist, and bring the arm up to present the middle finger

2 Project Planning and Task Definition

2.1 Task Identification

When deciding on our approach to our project, we initially came up with a few main categories of work that needed to be done; mechanical, electrical, and software. We discussed our individual strengths and weaknesses in regards to these areas in order to more effectively delegate tasks. Keeping this in mind, we made two subteams during winter quarter, hands and arms, with a reasonable balance of skills between them. The hands subteam was charged with designing the hands of the robot, while the arms subteam was responsible for designing the arms. Both teams worked together to determine final design choices like motor positions and specs.

The assignment of these subteams was not permanent, since the needs of each subsystem will change as this project goes on. However, the following was the tentative placement of our labor to each sub-team:

1. Hand Subteam:
    a. Shane Monney: CAD, Analysis of finger DOF and finger structure design
    b. Matthew Quinn: CAD, String and elastic routing
    c. Jason Kong: CAD for Motor Placement in hand
    d. Phillip Ng: CAD for string routing and motor placement in forearm
    e. Jonathan Kuriakose: Documentation and Lateral Finger Motion CAD
2. Arm Subteam:
    a. Lucas Ton: Arm CAD, Integration of hand and arm subsystems and Kinematics Calculations
    b. Alexander Pak: Motors research and Torque Calculations

During spring quarter, when we began manufacturing and testing our designs, we had the same main categories of work; however we needed to create more subteams as the project progressed. We maintained the hand subteam, focused on working out any problems with the hand design that arose during prototyping, as well as the arm subteam, which focused on manufacturing the arm and coding it. We also added a head subteam focused on designing a head and coding an LED screen to give the robot expressions, and a body subteam to create the body. A coding subteam was also created to focus on creating the code required for the hand and arm actuation, and a power subteam to determine how the robot would even acquire the energy needed to do anything.

1. Hand Subteam:
    a. Shane Monney: further development of CAD, prototyping and necessary design changes
    b. Philip Ng: Prototyping and design changes to integrate spring steel & fishing line
    c. Jason Kong: Prototyping and optimizing CAD design
    d. Lucas Ton: Bevel gear integration for direct non-string actuators on the fingers: thumb yaw and pitch, index yaw
2. Arm Subteam:

   a. Lucas Ton: Designing, Manufacturing, Assembly of the Arm
   b. Jonathan Kuriakose: Dynamixel Troubleshooting and Testing
   c. Matthew Quinn: Designed shoulder mount for stepper motor to attach to the body
3. Head Subteam:
   a. Shane Monney: designed the head structure
   b. Jason Kong: Wiring of the RGB display and developed code for certain faces
   c. Philip Ng: Wiring of RGB display and developed code for certain faces
4. Body Subteam:
   a. Matthew Quinn: Designed body options, retrofitted stool to meet body needs
5. Coding Subteam:
   a. Jonathan Kuriakose: Developed preliminary code for use and testing. Code troubleshooting.
   b. Lucas Ton: Kinematic Calculations of the Arm
6. Power Subteam:
   a. Alexander Pak: Electronics, motor integration, and wiring
   b. Jonathan Kuriakose: Wire soldering, power supply and wiring for hand actuation.

2.2 Timeline

Winter quarter started with lengthy brainstorming sessions to decide on our final project idea. We started with a wide variety of ideas, but during week three we narrowed our ideas down to a BB8 droid aimed at being a guide for those with disabilities, a guitar playing robot, a bob ross robot, and an emote bot. During week four we further developed our bob ross, emote, and guitar playing robots to prepare for our meeting with Dr. Hong to pick our final idea. After our meeting with Dr. Hong we decided to pursue the emote bot idea for our final project, and we also decided to focus on making it able to fingerspell (spelling in sign language). In week five we developed our expected timeline for the project, performing the necessary background research on similar examples of our robot that currently exist, and further finalized our idea for our mid-quarter presentation. Our progress in week six was slowed due to midterms starting for our group members, so that week we focused on creating our initial hand design and splitting up the work in that subsystem. Week seven was also a slower week during which we made edits to our timeline and determined how to meet the unique requirements for a fingerspelling and rock-paper-scissors hand. During week eight we began creating CAD models of our hand and arm as well as determined the types of motors we might need for each part of our robot. In week nine we created our final presentation and 3D printed some prototypes of our hand (a print in place model and a linkage model) to test our plans for actuation. This stage helped us finalize our motor placement and hand design. After our presentation in week ten we met with the TAs where they gave us feedback on our project thus far and suggested some improvements. Finals week was spent working on the initial version of this report.

This quarter (spring) we focused on prototyping, coding, and manufacturing. By week 2 we had nearly completed the first prototype CAD for the hand, printed our first finger prototypes, and began testing power sources, motors, and shields. The arm CAD also made a great deal of progress. In week 3 we began testing our hand with springsteel, and created a finger design with built-in slots for the springsteel and fishing line. The palm was also modified to fit necessary servos. Testing with the motors, shields, and power sources continued, and the arm CAD was further developed to house the servos required for actuation of the fingers. A preliminary body CAD was also produced this week. During week 4 we continued testing new iterations of fingers with the springsteel and fishing line, and added routing for the fishing line in the palm. The body CAD underwent further development, and electronics testing continued. Coding began in week six, in addition to arm assembly. At this point, due to budget constraints, it was decided to modify a stool to use as the body instead of purchasing 8020 and other aluminum. The final hand prints were completed week seven, and they made use of additional printed components to screw into the fingers and hold the springsteel instead of the slot design being used the previous weeks. MATLAB code was also prepared to test the Dynamixel motors for the arms, and the arm construction was mostly completed. Progress on modifying the stool continued to be made, the focus this week was on creating brackets to reinforce the stool. In week eight code was created to control the arm as “gestures,” essentially taking specific position values for each motor in the desired arm positions. Some work on creating servo horns for the hand’s continuous servos was also done. In week nine the arm code was further developed, as was the servo horn design. During week ten we began assembling all of the motors and other electronics and performing any final edits and tests necessary.

3 Literature Review

Our research on humanoid hands led us to discovering 3 different projects that have helped us draw inspiration in our current design of our hands.

3.1 Prosthetic Arms

One of the first avenues for hand design would be prosthetics, since prosthetics aim to achieve similar movements to the actual human hand. The main difference between prosthetic arms and our hands is the actuating force needed and the DOF of the fingers. Firstly, prosthetics' primary objective is to be able to hold items. Hence, their motors need to be stronger in order for them to grip onto objects. Secondly, prosthetic arms are less concerned with mimicking human hand motion, which means they may not be able to do all the movements required for our humanoid hand.

Regardless, prosthetics have some key features that we have drawn inspiration from. Firstly, most prosthetics arms are string actuated in order for the motors to be placed in a convenient location. Secondly, the location of these motors need to also be decided in order for us to make our design as effective as possible. Some models placed their motors in the palm of the hand, while others placed them in the forearms. Placing them in the palm puts the motors closer to the fingers, which could lead to a cleaner design with less string routing. However, placing them in the forearm would give more freedom in the design of the hand since the palm has a very limited design space for us to add motors. The forearms have a greater capacity to hold the motors.

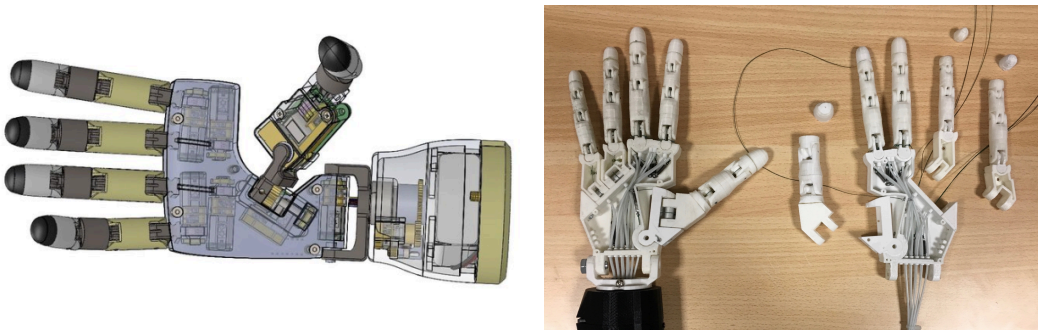


Figure 2: (left) Actuating motors in palm; (right) Actuating motors routed to forearm

3.2 Integrated Linkage-driven Dexterous Anthropomorphic (ILDA) robotic hand

The ILDA design uses a complex linkage system with three actuation inputs per finger to achieve complex finger control. It would easily achieve all the finger motion we require for our hand. However the complex nature of its linkage mechanism would make it difficult to manufacture and be prone to error due to the accuracy required for the mechanism to function effectively.

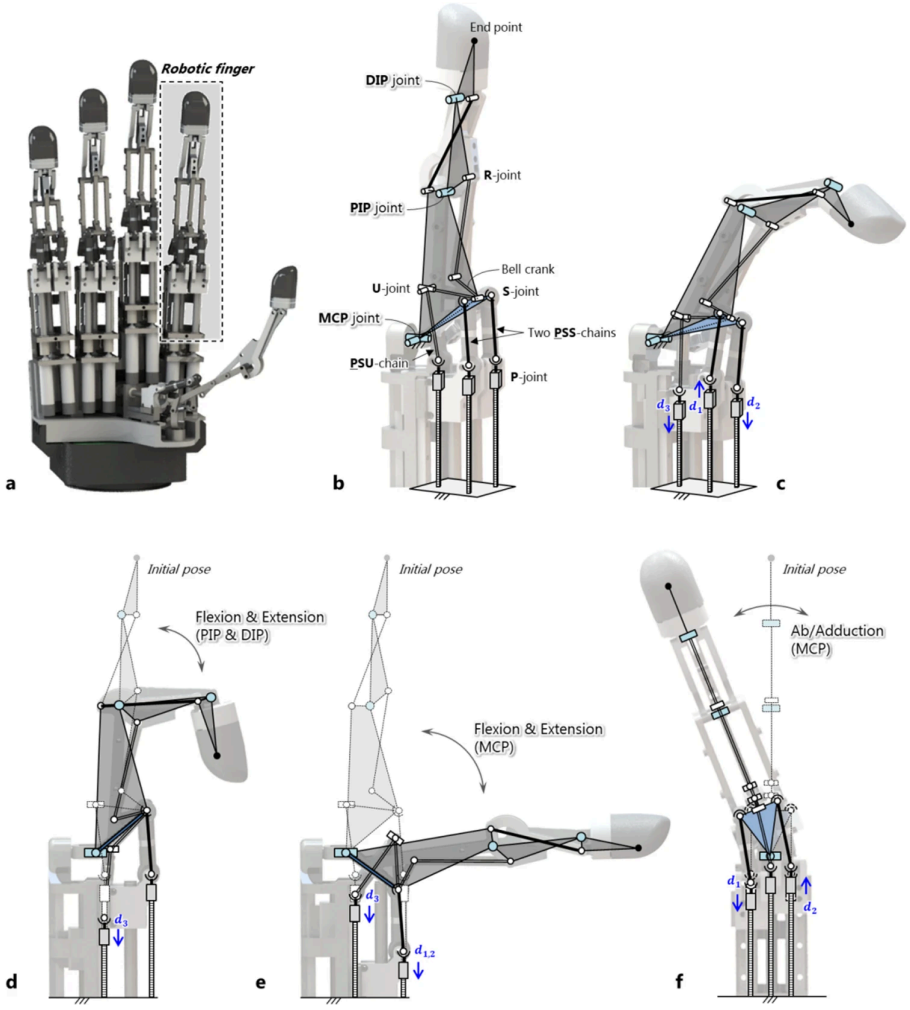


Figure 3: ILDA Hand Mechanism

3.3 Project ASLAN

The ASLAN project has a very similar aim to what our group is trying to do, which makes this design of particular interest to us. It has a very similar design to many prosthetic arms, however it uses up to 4 strings per finger in order to achieve the desired motion. Apart from some articles talking about the project, very little technical information seems to be available about this system, leaving us with very little to work with here.

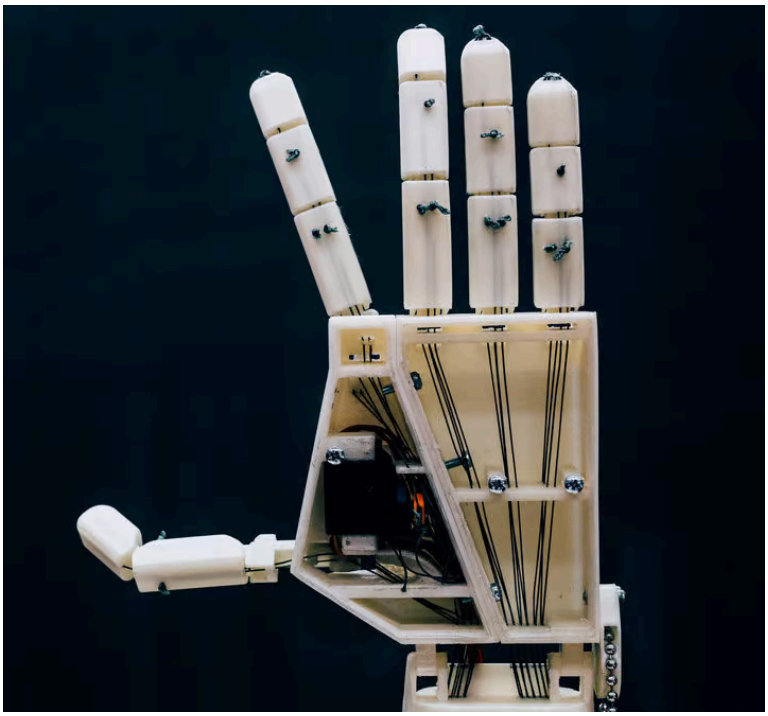


Figure 4: ASLAN Hand Project

4 Preliminary Design

4.1 Concept Generation and Evaluation

Our design process for this project was mainly focused on different types of hands and actuation systems for them. Our first hand model was created from individual links for each segment of every finger which would be connected using pins to allow rotation at each joint. The initial model for this lacked the ability to move the index and thumb laterally, but it was intended to simply test if such a design could work and we were pleased with the results.

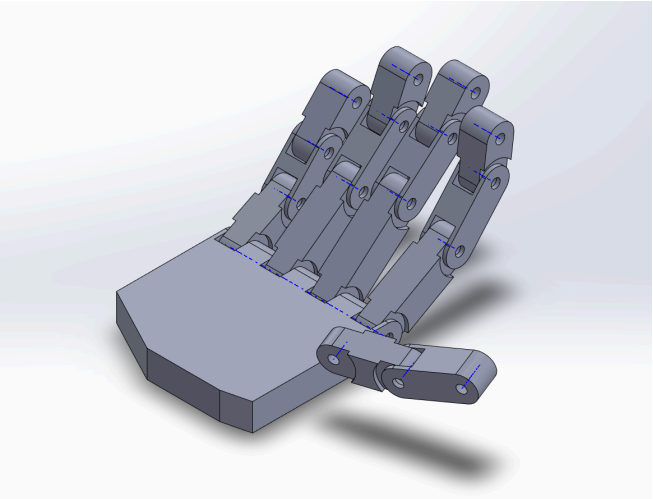


Figure 5: Initial Linkage Hand Design

Another design we considered was a model that could be 3D printed in place, meaning the entire thing could be made in one print without the need for any assembly. The design for this lacked the thumb and index finger because we were unsure how to accomplish our desired movements from those fingers with a print in place design. The allowed movements with the below design worked very well, and if the lateral movements were unnecessary we likely would have chosen to go with this design.

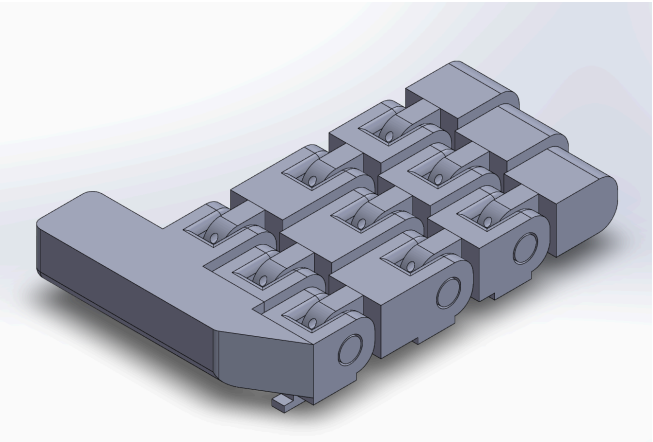


Figure 6: Initial Print in Place Hand Design

For each hand we had lengthy discussions and performed various tests to determine the best plan for how to actuate the fingers. We planned from the start to use some type of string to pull the fingers, but we debated between pulling them up or pulling them down (i.e. the rest state for the hand either being the fingers fully extended or a closed fist). For a resting state of extended fingers we decided we would need some sort of material at the backs of the fingers that would be compliant enough to allow the fingers to bend into a fist but strong enough to return them to their initial extended position when the motors were no longer pulling the strings. For this design we planned to put the motors on the front of the forearm (in the same plane as the palm) and attach the string to the tips of each finger. If we used the closed fist resting state we would use gravity to return to the resting state after pulling the fingers up to form signs, and the motors would be on the back of the forearm (in the same plane as the back of the hand). The figure below shows a rough sketch of the motor placement for the extended fingers resting state, as well as a plan to route the strings and move the index finger laterally.

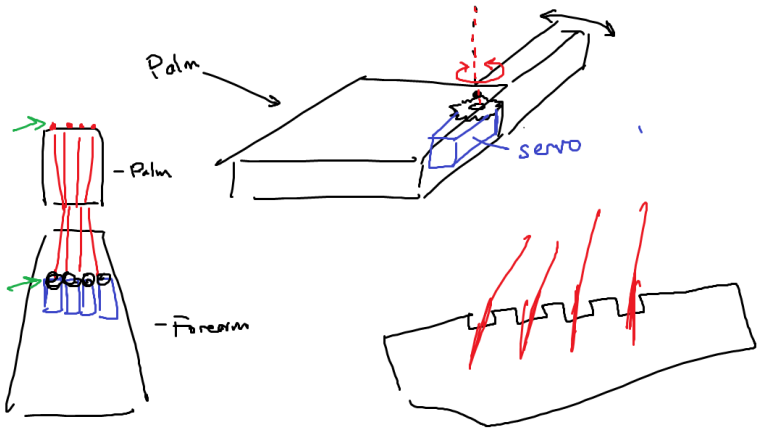


Figure 7: Rough Sketches of Hand Actuation Ideas

After considerable deliberation, we decided to pursue the linkage hand design and the extended finger resting state. We chose the linkage hand design mainly because we determined it was not possible to achieve the desired lateral movement of the index finger or the movements of the thumb if we continued with the print in place design. The TAs were also in favor of the linkage design so we could gain more practical experience. We chose the extended finger resting state plan for actuation because the closed fist design would limit us to only being able to sign and play rock-paper-scissors correctly if the hand was held straight up (palm facing out from the body), which we deemed quite limiting and undesirable.

In order to achieve our lateral motion in the index finger, we extended the finger partially into the palm and added an extra joint for the rotation. The mechanism for this is sketched above. For the thumb, we removed a portion of the palm so the thumb's rotation could be moved far enough that the end of the thumb could reach the opposite end of the palm after the full rotation. This is necessary to sign the letter m. We also removed one of the joints in the thumb and replaced the other with a joint that would allow lateral movement, which is also necessary to sign letters like m and e.

Each finger will have one servo to bend the finger down towards the wrist, and the index finger and thumb will have one extra servo each for their lateral movements, as shown in the figures below. The servo controlling the lateral movement of the thumb is located at the base of the palm and connected to the base of the first knuckle via a spur gear. The motion of the end knuckle of the thumb is controlled by a servo mounted on the back of the thumb itself, specifically the base knuckle. The shaft of this servo is connected to the revolute joint of the respective thumb knuckle with spur gears as well. The lateral motion of the index finger is driven by a servo mounted on the back of the hand connected to the base knuckle. The three motors for the thumb base, thumb end-knuckle, and index finger lateral motion are mounted very close to the moving piece to allow for a simple gear drive linkage. The bending motion of the index finger and three outer fingers is controlled using line attached to proximal servos.

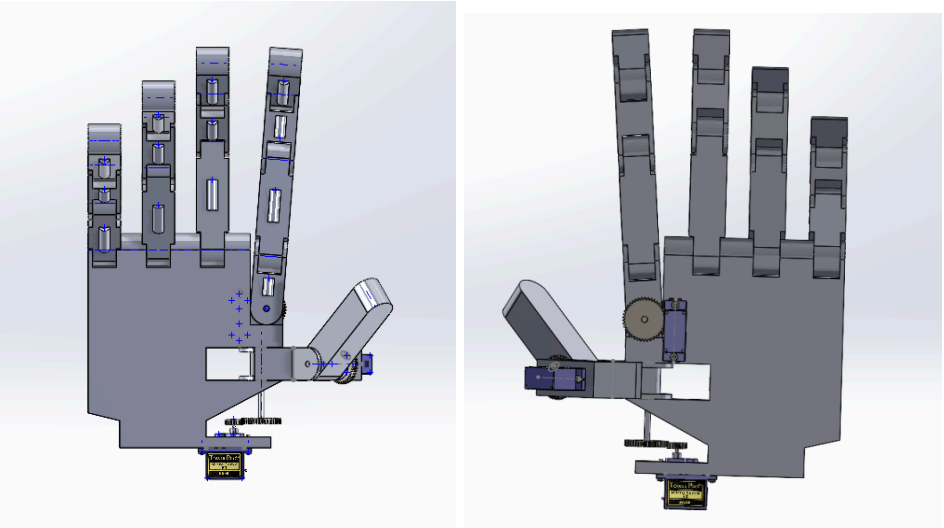


Figure 8: Front and Back Views of Hand with Mounted Servos

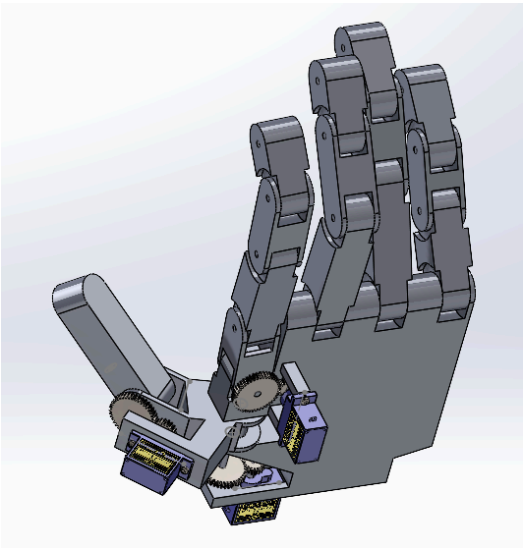


Figure 9: Hand Servos with Gearing

We decided to use fishing line for our string, and for the normal bending of the index, middle, ring, and pinkie fingers we plan to simply tie it from the tip of the finger down to the servo. We considered running the line inside the fingers themselves but we had worries about the stress and friction this would put on the joints since the force would be pulling the fingers into themselves more than bending them down, and we also ran into issues with meeting our desired range of motion. Our hand also has no need to grab or hold anything, so there is no need for the palm to be clear of obstructions (like the fishing line crossing it).

Moving further down into the overall design of the forearm and the upper arm, our goal is to keep the design as lightweight yet strong as possible. Only a small portion of the forearm needs to house motors, so the remaining space is dedicated to optimizing strength and weight.

In regards to the arm design, as mentioned before we adapted a model with 7 degrees of freedom, with 3 DOF in the shoulder, 1 DOF in the elbow, and 3 DOF in the “wrist”. Although only 6 DOF is required to fully actuate a robotic manipulator to any orientation desired, the extra DOF allows for the arm to access the same end effector orientation with multiple arm configurations. In addition, we want to model our arm as close to the human arm which is modeled with 7 (sometimes 8) DOF.

4.2 Preliminary Analysis

In choosing the motors for our design, the primary factor was the holding torque required to maintain the arm midair. To account for the worst case scenario, we modeled the arm fully outstretched (orthogonal to the direction of gravity), maximizing torque. We calculated the torque required at each joint in the arm with the arms modeled as completely filled cylinders of PLA at the size of an average human arm. A 3D printed link of PLA will not produce the same weight because the hollowed pockets created by the prints’ infill will reduce the weight by nearly 80%.
About the shoulder, the maximum torque required to maintain the weight of the total arm structure (the links and electronics) estimates to 7.48 Nm, with only 1.37 Nm coming from the electronics alone. By reducing the weight of the links by half to produce less than 4 Nm of total torque, we can actuate the entire arm with a stepper motor rated for 4.7 Nm of holding torque.

About the elbow, the maximum torque required to maintain the weight of the remaining arm structure (forearm and hand) is estimated to be 2.56 Nm, with only 0.52 Nm coming from the electronics alone. By reducing the weight of the links in half to produce less than 1 Nm of total torque, we can actuate the entire arm with a dynamixel servo motor rated for 1.5 Nm of holding torque.

The decision to use sg90 servo motors to actuate the fingers of the hand was not made through analysis but rather through testing. We had motors of similar strength available to us, which we confirmed to work with the current prototypes we had produced. The sg90s are small, inexpensive, easy to code, and get the job done.

Our current Kinematics model demonstrated in figures 11 and 12 assumes that the shoulder joint has all of the frame origins intersecting. The same goes for the wrist joint. Table 1 lists the DH parameters, and following it is the forward kinematics matrix.

Table 1: Initial DH Parameter

Joint	Link Twist	Link Length	Link Offset	Link Angle
1	-π/2	0	0	θ_1
2	-π/2	0	0	$\theta_2 + \pi/2$
3	π/2	0	0	$\theta_3 + \pi/2$
4	-π/2	a_3	0	$\theta_4 - \pi/2$
5	-π/2	0	d_5	θ_5
6	π/2	0	0	$\theta_6 + \pi/2$
7	-π/2	0	0	θ_7

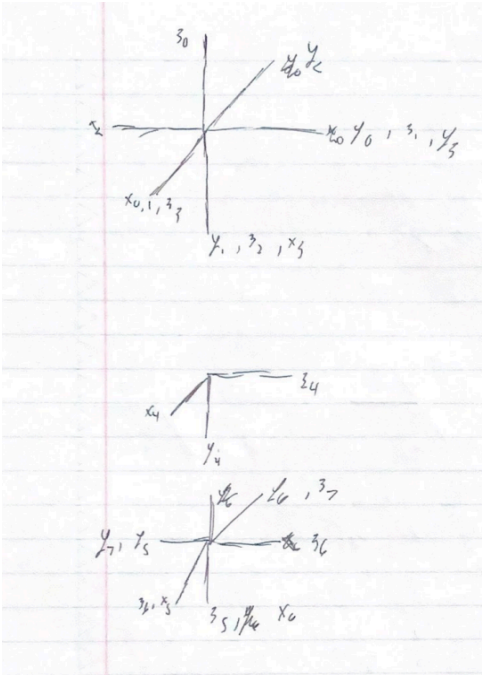


Figure 10: Initial Arm Kinematics

Currently the bicep arm length (a_3) is about 308 mm and the forearm length (d_5) is about 223 mm. The lengths will change as the arm design continues to be optimized, but these current values keep the housing numbers of the forearm frame and the bicep frame simple.

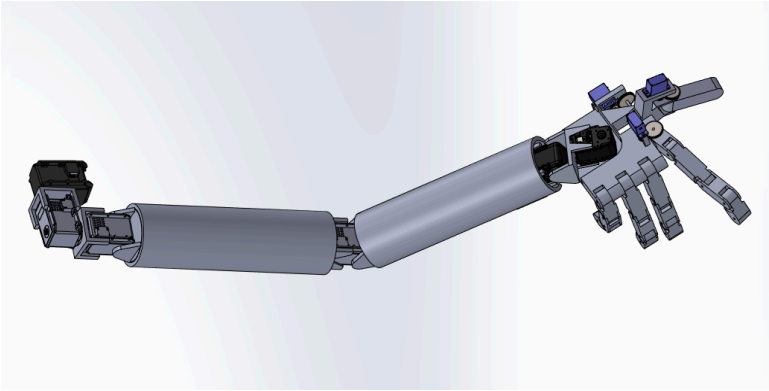


Figure 11: Integrated Hand and Arm CAD

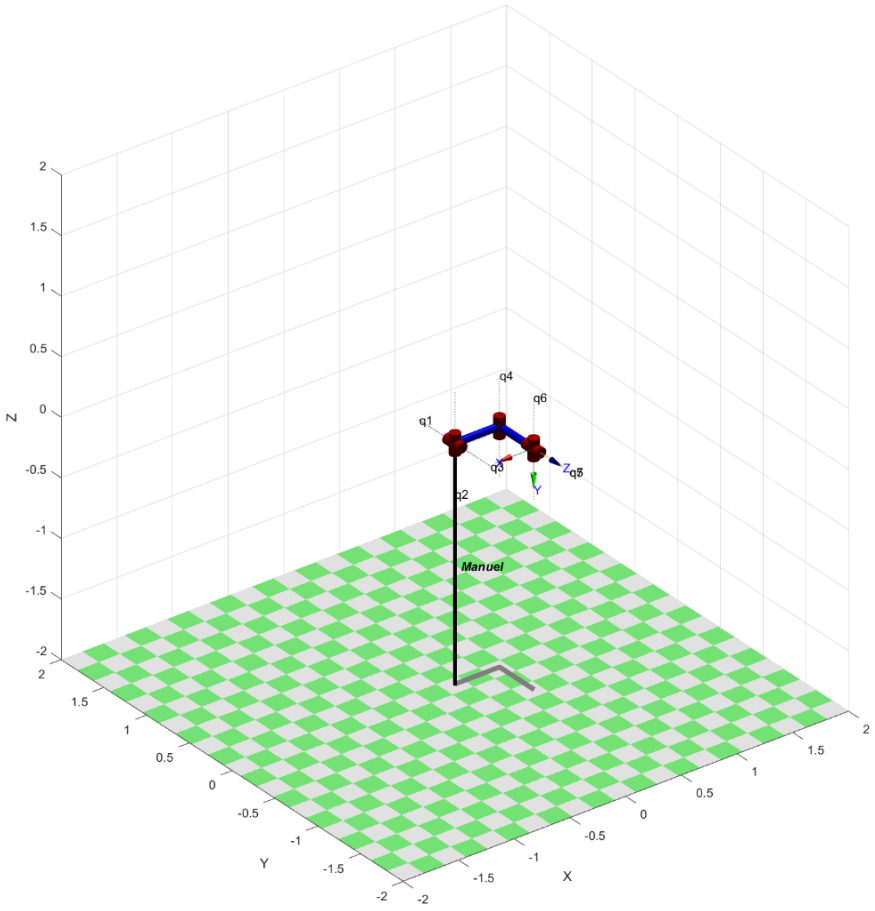


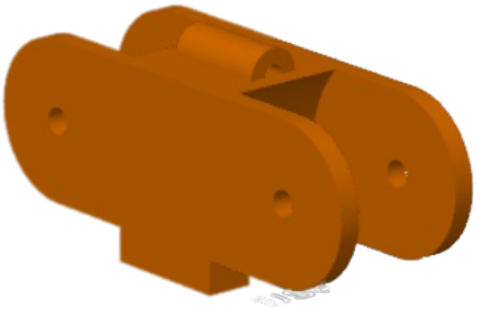
Figure 12: Preliminary MATLAB model of the Arm

The wrist joint joint and possibly the motor joint have been slightly adjusted since the kinematics were developed, so parameters will need to be added to to represent the change. Inverse kinematics have yet to be developed in the case this situation has occurred.

5 Detailed System Design

5.1 Hand Subsystem

In order to maintain an upright resting state for the fingers, we decided to go with springsteel slotted into the back of each finger. Initially we modified the finger prints themselves to add these



slots, but the print quality suffered due to the location of the slots and the print orientation

Figure 13: Initial Slot and Fishing Line Channel Design for a Knuckle

required because of them. We also added channels for the fishing line into the front of each knuckle print, which was essentially a half cylinder added onto each piece. This also gave us trouble due to the print quality, and also simply because it was hard to remove excess print material from inside the small channel. After a few printing attempts and small redesigns we switched to a snap-on slot idea. This idea relied on ‘snap-on’ 3D printed pieces that would allow the springsteel to slide in the back, but also keep the string around the middle of the finger in the

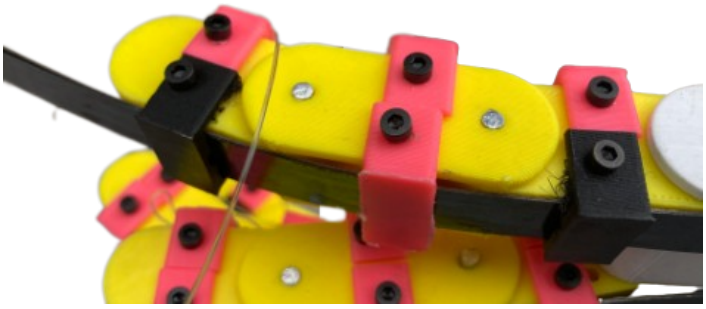


Figure 14: ‘Snap-on’ Closeup View

front. These ended up working extremely well, and the only change we made to them was to screw them into the fingers instead of simply snapping them on. This was required due to deformation of the snap-ons and inconsistent print sizes.

Each finger uses one continuous servo to bend the finger down towards the wrist, and the index finger has one extra, positional servo for its lateral movements. The thumb has two positional

servos, one controlling its movement across the palm, and the other controlling its lateral movement with respect to the link. The servo controlling the movement of the thumb over the palm is located within the base of the palm and connected to the base of the first knuckle via a bevel gearing system. A gear is attached to the end of a pin crossing through the base knuckle, and this meshes with a gear attached to a pin attached to the motor in the base of the hand. The motion of the end knuckle of the thumb is controlled by a servo mounted in the thumb itself, specifically the base knuckle. The shaft of this servo is connected to a pin crossing the end knuckle by more bevel gears.

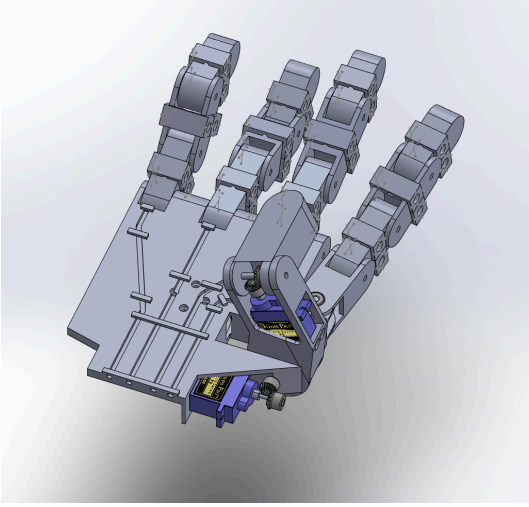


Figure 15: Motor and Gearing System for Thumb Movements

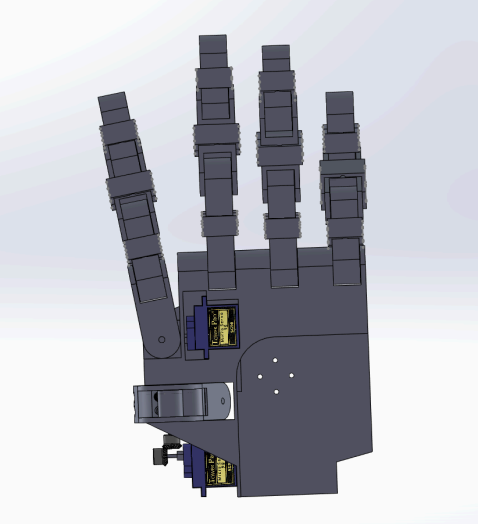


Figure 16: Index Finger Motor

The lateral motion of the index finger is driven by a servo mounted in the back of the hand and connected through the wall of the palm to the index finger. A bevel gear is attached to the servo and meshes with another gear on a pin within the base knuckle of the index finger.

We continued our use of fishing line for our string, but we decided to run the line through channels in the front of each finger instead of having it connect straight to the wrist like we had planned at the end of Winter quarter. We decided to do this simply for aesthetic purposes, but we did experience increased friction as anticipated. Due to this increased friction, our index finger is usually unable to bend down almost at all.

In terms of the hand, if we were to continue this project, in future iterations we might reconsider running the fishing line inside of the fingers. This could allow us to actually bend the index finger, and would decrease the load on each continuous servo. We might also try to use something besides springsteel to maintain our upright resting position. Another group used an interesting elastic string for a similar purpose, so we could experiment with that, or even simply rubber bands. If we had more of a budget we could also aim to control both the opening and closing of the hand, and in that case we would just use fishing line on both sides but would need twice as many continuous motors. There would undoubtedly be further challenges with each of these ideas.

5.2 Arm Subsystem

Moving further down into the overall design of the forearm and the upper arm, our goal is to keep the design as lightweight yet strong as possible. Only a small portion of the forearm needs to house motors, so the remaining space is dedicated to optimizing strength and weight.

At the shoulder, we have a true 3 DOF spherical joint that allows for rotation about the x, y, and z axes (roll, pitch, yaw). We accomplish this by mounting the 3 motors of the shoulders so that their axes of rotation all intersect at the same point. This spherical joint also makes our kinematic calculations much easier for when we decide to move the arm around in space. The elbow joint only has 1 DOF so we can place a motor at the connecting point between the upper arm and forearm. And finally, for the 3 DOF wrist we have another spherical joint. However, we positioned the motor controlling the hand/forearm's roll (rotation about its long axis pointed towards the hand) in the forearm closer to the elbow joint for two reasons. By doing so, we save space for the wrist and palm areas where our finger actuation mechanisms will be placed as well as shift the weight of the motor closer to the shoulder to minimize the torque required for the entire arm to actuate.

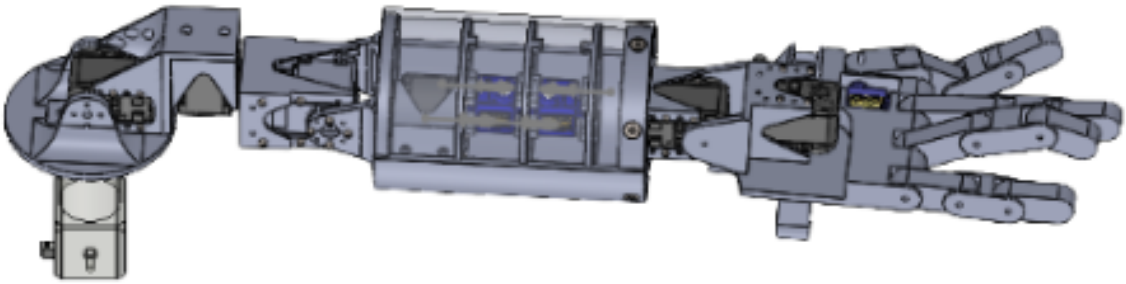


Figure 17: Integrated Hand and Arm CAD

Since the preliminary design, there has been changes to the DH parameters. These changes were made to reflect a small change in motor selection. Originally, Dynamixel MX64T motors were used for the shoulder yaw and roll in addition to elbow. However, we were on short supply of said motors, and had to switch to Dynamixel AX12A motors instead for the shoulder actuators. As a result, the roll and yaw motor positions were switched to ease up arm motion. However, as we found out with later testing, the shoulder yaw actuator was not strong enough to perform the entire range of motion we had intended it to.

DH parameters listed in table 2 sets the resting position, or zero position, of the arm so that the arm hangs directly downward with its palm facing the body. From here joint limits listed in table 3 were arranged so that it would mimic a human's arm's joint limits along side meeting inherent joint limits of the actuators and arm design.

Table 2: Final DH Parameters

Joint	Link Twist	Link Length	Link Offset	Link Angle
1(Shoulder Pitch)	$\pi/2$	0	0	$\theta_1 - \pi/2$
2(Shoulder Yaw)	$-\pi/2$	0	0	$\theta_2 - \pi/2$
3(Shoulder Roll)	$-\pi/2$	0	0.142 m	$\theta_3 + \pi/2$
4(Elbow)	$\pi/2$	0	0	θ_4
5(Wrist Roll)	$-\pi/2$	0	0.218 m	$\theta_5 - \pi/2$
6(Wrist Pitch)	$\pi/2$	0	0.007 m	$\theta_6 + \pi/2$
7(Wrist Yaw)	$-\pi/2$	0.065 m	0	θ_7

Table 3: Joint Limits

Joint	**Minimum Joint Limit**	**Maximum Joint Limit**
1(Shoulder Pitch)	$-\pi/2$	π
2(Shoulder Yaw)	π	$\pi/2$
3(Shoulder Roll)	$-3\pi/4$	$\pi/2$
4(Elbow)	0	$4\pi/9$
5(Wrist Roll)	$-\pi/2$	$\pi/2$
6(Wrist Pitch)	$-\pi/4$	$\pi/2$
7(Wrist Yaw)	$-\pi/4$	$\pi/4$

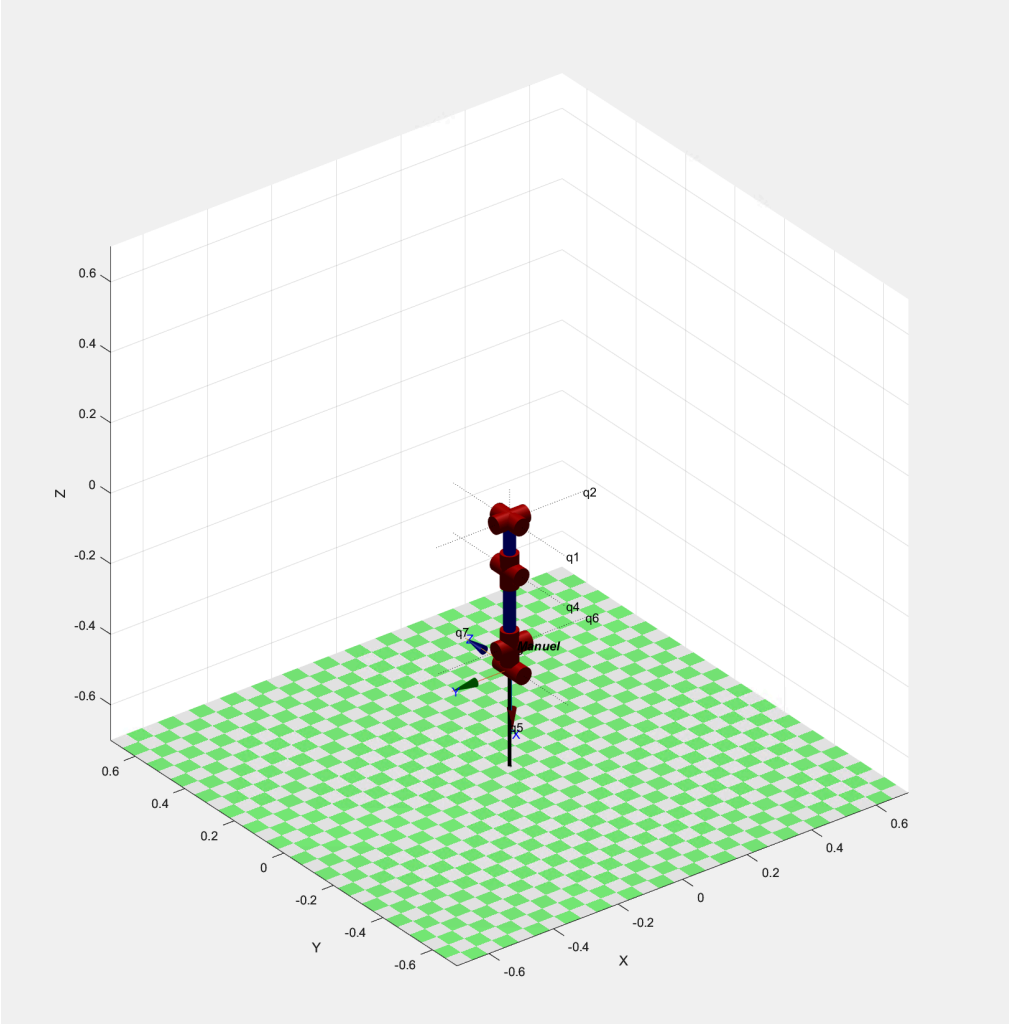


Figure 18: Updated MATLAB model of the Arm at Zero Positions

5.3 Head Subsystem

For the design of the head, we printed a box to velcro on the RGB LED matrix board. The dimension of the box is 6 x 6 x 6 inches and has a hole that is the same width of the LED board but the height is about 2.5 inches. The LED has a matrix of 32 x 32. With super glue, we then put together the non velcro side of the patch to the head and did the same with the LED board.

The setup for the RGB Display is relatively complex. The Arduino Mega microcontroller connects to the LED screen via jumper wires connected to a ribbon wire. The LED is connected to an Adafruit matrix shield via jumper wires connected to the shield's pins. The power source of the 5V 2A power source stayed constant for the LED matrix.

The setup was intended to be simple, but because of the lack of RAM from the Arduino Uno, we needed to switch to the use of an Arduino Mega. Unfortunately, the Adafruit matrix shield used was only compatible with microcontrollers with the same form factor as the Arduino Uno.

Because the Arduino Mega has a different form factor, the shield could no longer fit directly over it, so jumper wires were required to connect it with the shield.

The head is programmed to make simple facial movements and expressions, such as blinking, winking, looking in different directions, and even an angry face. This addition provides a humanoid aspect to the robot that makes it more appealing to users. The code for programming the face will be discussed in the “Code” section of the report.

5.4 CAD Models

At the conclusion of the project, we have a full arm, hand, body, and head modeled in our CAD. Included in the CAD assembly is the repurposed wooden stool that serves as the body of Manuel.

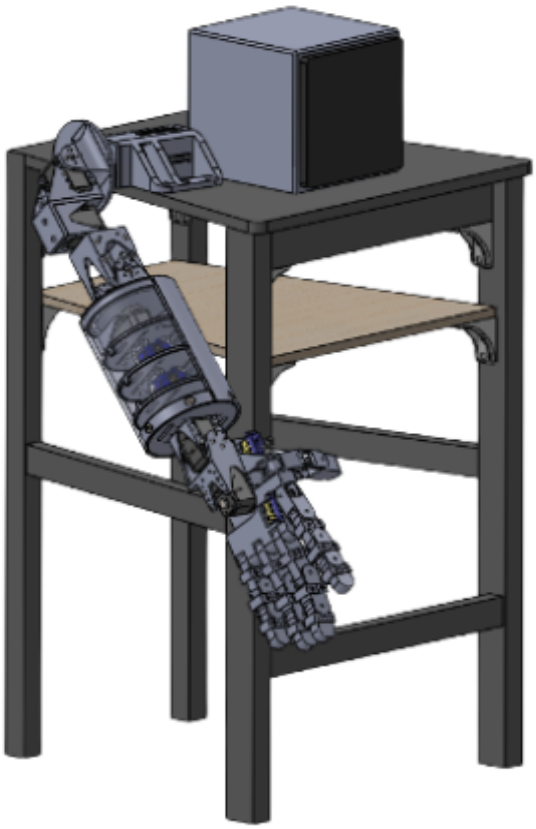


Figure 19: Full body CAD assembly

The motors driving the system are all located throughout the arm and hand. The remaining electronics, including Arduinos, controllers, and batteries, are all located on the body of Manuel on the platform just underneath the top level where the head is placed. Connecting all of the electronics are wires that will feed from the required motors down the arm and into the shoulder,

where the wires will then lead to the required electronics. This will prevent any unnecessary weight from being added to the arm, decreasing the torque placed on the motors.

5.5 Electronics Systems

The electronics systems of Manuel can also be divided into the 3 main subsystems of the hand, arm, and head. Every electronics system is powered by a respective power source at a different voltage and current based on the needs of its components. Each power source is plugged into a power strip that then feeds into a wall outlet to power the entirety of the robot.

For the hand, the mini servos all receive three wires: power, ground, and signal. Both the power and ground wires consist of jumper cables that have been extended through splicing and soldering, each delivering current from a 5V/10A power source on the body that delivers at least 750mA of current to each servo. The signal jumper cables are also extended through soldering to reach all the way to the body where they converge at an Ardunio Uno.

A similar process used for the hand servos is used for the Dynamixel servo motors and stepper motors located in the arm. The dynamixels take a specific set of wires to connect it to a Dynamixel shield for another Arduino Uno, which allows the Ardunio to interface with and control the motors. This Arduino shield supplies the power to all of the Dynamixel motors with a 12V/5A power source and also provides the actuation signals all in one bundle of cables. Dynamixel motors can operate by daisy chaining to each other, meaning that only one cable needs to be connected to the Ardunio shield at the base of the chassis. Because of this system, only a handful of Dynamixel cables had to be extended -- namely the wire that reaches from the shoulder joint all the way to the body of Manuel.

Finally, the head subsystem's electronics also require an Arduno microcontroller to control the system; however, it uses an Arduno Mega in combination with an Adafruit RGB Matrix Shield to control the output of the LED matrix of the face. The shield takes an additional 5V/2A power source to provide enough power to the LEDs, which is also plugged into the same power strip as the power sources for the other two subsystems. Unfortunately, the Arduino Mega is not completely compatible with the 32x32 LED Matrix used in the system, so the connection is bypassed by manually wiring all of the header pins from the shield to its corresponding pin on the Arduno Mega. The electronics for the head is wrapped up with a USB cable that feeds additional power to the LEDs from a peripheral computer that will also provide the code for the matrix as well.

5.6 Code

Manuel was coded mainly using the Arduino IDE, with the exception of the arm trajectories being coded in MATLAB, after deriving the values from MATLAB, it was copied into the Arduino IDE using an array. The plan was for us to use the array to set the position of the arm. However, due to problems discussed later, this was not done.

The code for Manuel can be divided into 5 categories, and the pseudocode for each will be given here. Each pseudocode's goal is to move the associated part of the arm to the correct value.

1. Servos:

Input: Position for each DOF

// gesture

for each continuous servo

turn in one direction

wait for time period based on position input

stop moving

for each positional servo

move to desired position input

// return to default

for each continuous servo

turn in opposite direction

wait for time period based on position input

stop moving

for each positional servo

move to default position

With the positional servos, the code is pretty straightforward since you can directly input the goal position. However, with the continuous servo, it was trickier to implement since we do not have direct control of the position. Hence, time was used as the variable to control the motor. This was an imperfect solution due to the speed of the motor being easily affected by the resisting torque. This made it hard to predict exactly how long we needed to run the servo to reach a desired position. This also meant that everytime we did a "gesture", we would need to return to the default position so that we have a proper reference at the start of every gesture on where the fingers are.

In the end, it was decided that implementing this code to do specific gestures on its own was too unreliable, and hence a program to manually control the fingers using the serial monitor was created for demonstration purposes.

2. Stepper Motor:

for each step of the MATLAB trajectory value

take difference of value at i and i-1

determine number of steps needed using stepper constants

move stepper by determined steps

Moving the stepper by a defined number of steps is a standard function that was derived from a tutorial website listed in the Reference Section. It is basically a for loop that turns the stp signal on for a millisecond to move the stepper by one step. The direction of the stepper is determined using the dir pin.

However, due to not being able to use the array values with the Dynamixels (explained below) we implemented a program which could manually move the stepper by a predetermined amount using the serial monitor. This was done for demonstration purposes and testing.

3. Dynamixel:

for each step of the MATLAB trajectory value

set goal position of each dynamixel to associated value in MATLAB array

wait for motors to reach goal position

Theoretically, this code was the simplest since the Dynamixels could accept direct position values unlike the stepper and servos. However, we ran into major issues when implementing this code. The code for the Dynamixels was written with the DynamixelShield library, whose documentation is in the Reference Section. Since the code is using a for loop to enter values into the dynamixel code, the array values are treated as dynamic values at run time. Based on extensive testing of code, it was determined that the function to set the goal position of the dynamixel did not accept dynamic values. Hence, we were unable to implement the for loop as stated in the pseudocode above. In order to get the code to work as intended, we would need to write the same set of code for all 50 array positions, which would take up too much space on the arduino, hence it was then decided to use only 5 values. However, after testing that, it was found that the movement of the arm was too jittery and concerns of the structural integrity led us to decide only using the start and end values of the array to avoid the constant stop/start movement of the arm. Unfortunately, this still causes the arm to arrive at the final position with jittering, an issue that we were warned of at the beginning of the project.

4. Face Code

The code for the face does not require significant pseudocode since the implementation of this code is strictly defined using the RGB Matrix Panel Library. The documentation for this library is in the Reference Section. The code is simply manipulating values of a 32x32 matrix that represents each LED on the RGB Matrix.

In order to draw a shape on the LED screen, the functions drawLine, drawRectangle, or drawCircle are used with parameters to define start and end position, radius (if applicable), and color. These functions were used to create all the facial expressions mentioned in Section 5.3. To update the face, the LED screen was cleared and redrawn, since there is no function in the library for deleting specific elements from the screen.

5. Integration

The integration of codes proved to be a challenge due to the hurdles encountered in the subsystems of code. The initial plan was to use 2 Arduino Unos to control Manuel, since the RGB matrix required its own shield to work. The arm motors would be controlled by one arduino, allowing us to move all motors in the order we want it to move in using code. However, we encountered 3 issues:

- The RGB Matrix Panel would not work with an Arduino Uno due to space constraints. Hence, we switched the face microcontroller to an Arduino Mega.
- Due to issues with the Dynamixel code, we needed to manually control the stepper and the servos with the serial monitor for demonstration purposes. The problem is that the dynamixel shield prevents us from using the serial monitor due to the shield using the IO ports that are typically used to interact with the serial monitor. Hence, we decided to use 2 Arduino Unos to control the arm motors. One for the Dynamixels, the other for the stepper and servos. We can pre-program the dynamixels, while manually controlling the other arduino in order to prevent any clashes between the two codes.
- The dynamixel shield appeared to be bleeding dynamixel signals into the regular Arduino pins. This was determined due to the fact that the stepper would make unexpected noises while we ran dynamixel code on the arduino. Hence, we needed to split the microcontroller for the dynamixel control. This was already done due to the issue mentioned in point 2.

6 Bill of Materials

6.1 Purchases Made Through the Mech&AE Department

Quantity	**Description**	**Total Price**
2	STEPPER DRIVER 0.75A 30V LOAD	$ 43.53
2	Multipurpose 6061 Aluminum (1/8" Diameter)	$ 16.97
1	Wear-Resistant 1095 Spring Steel (0.0040" Thick, 1/2" Wide, 25 Feet Long)	$ 49.90
2	FEETECH FS90R (5 Pack) - 360 Degree Continuous Rotation Servo 9g for Robotic Helicopter Airplane Boat	$ 50.34
1	607 Adafruit Addressable Lighting - 1024 (32 x 32) LED Matrix Red, Green, Blue (RGB) 128.00mm L x 128.00mm	$ 40.05
1	DFR0379 20W ADJUSTABLE DC-DC BUCK CONVERTER	$ 10.10
1	Dynamixel Shield	$ 37.79
1	Fasteners	$ 149.84
1	42STH38-1684B - 1.8 Degree - 1.68A Stepper - 100:1 Gearbox	$ 61.83
1	Robot Cable-X3P 180mm (Convertible) (10pcs)	$ 19.00
1	Robot Cable-X4P 180mm (Convertible) (10pcs)	$ 20.20
1	Adafruit RGB Matrix Shield for Arduino	$ 12.00
1	5V 2A (2000mA) switching power supply - UL Listed	$ 16.00
1	Female DC Power adapter - 2.1mm jack to screw terminal block	$ 4.00
1	Robot Cable-3P 200mm 10pcs	$ 17.20
	Total Price	**$ 548.75**

6.2 Items Borrowed from RoMeLa Lab

Quantity	**Description**	**Total Price**
6	Dynamixel AX-12A	$ 299.40
1	Dynamixel MX-64R	$ 369.90
	Total Price	**$ 669.30**

6.3 Personal Items Used in Project

Quantity	**Description**	**Total Price**
1	Fishing Line	$ 14.00
1	Gorilla Super Glue Gel XL, 25 Gram, Clear, (Pack of 2)	$ 19.67
2	Arduino Uno	$ 20.00
1	Arduino Mega 2560	$ 48.20
1	BTF-LIGHTING AC100-240V to DC5V10A Max50W	$ 26.27
1	ALITOVE DC 12V 5A Power Supply Adapter Converter Transformer AC 100-240V Input	$ 13.13
1	Henxlco AC 100-240V to DC 24V 2A 48W Power Supply Adapter Converter Transformers with 5.5 x 2.5mm Plug for LED Strip Flexible Lights, CCTV Security Camera System(1Pack)	$ 8.75
	Total Price	**$ 150.02**

Total Cost of Project: **$ 1368.07**

7 Summary and Discussion

Unfortunately, Manuel barely met the minimum expectations we had set out for him. Originally, Manuel set out with the intention to be able to perform simple sign language statements along with smooth arm gestures. As the project progressed, the difficulty of each objective became much more apparent.

The fingers were produced to mimic the size of an actual human hand. However, with constant redesign to promote smoother movement and incorporate the dowel pins, the size of the fingers was increased, resulting in relatively heavy fingers. As a result, rubber bands were insufficient to tension the fingers, and spring steel was used instead. To incorporate spring steel and string to pull the finger, we printed slots on each finger, but this resulted in poor prints. Thus separate "screw-on" pieces were printed to hold the spring steel. After incorporating the continuous servos and connecting the string with the palm, it became apparent that the servos were too weak to fully contract the fingers. Moreover, it was difficult to find a "zero" position for the fingers and control how much the fingers contracted, since continuous servos are speed-controlled and not position-controlled. In the future, stronger servos (>5 kg) and encoders to store the zero-position, allowing for more controlled movement.

The faults with the arm lie in the selection of motors for both the finger and joint actuation. Due to budget limitations, we had elected to use motors provided by the RoMeLa. However, despite the quality of motors RoMeLa had provided, torque provided by the AX-12A motors did not meet our expectations for the shoulder actuation. Fortunately, the stepper motor used for the shoulder pitch allowed us to move the arm to a position where it can visibility perform the middle finger gesture.

The structural design of the arm created a heavy forearm and weakly supported upper arm section. The weak upper arm is a result of minimizing weight and minimizing how much 3D printing needed to happen before assembly. While it did not fail during testing, the brackets did yield during the arm movements. In addition, the jitter that resulted from motion that wasn't smooth exaggerated the yielding demonstrated in the video.

The heavy forearm is a result of storing the finger actuators in the arm. Besides the weight of the forearm, the string used to actuate the fingers limited the range of motion on the wrist. Essentially, the wrist yaw and pitch locked a the zero positions, removing them for the range of motion from the arm.

The code for operating Manuel is not complex on paper. However, due to physical constraints and library incompatibilities, coding Manuel to operate on its own was not feasible. Hence, we reverted to a manual control approach for demonstration and testing. With stronger motors and a more reliable finger pulling mechanism, coding Manuel to operate automatically should be a feasible goal.

While Manuel did not get to do sign language or gesture as intended, it did manage to accomplish the intended gesture that Manuel was promoted with. Potential exists to create a robotic arm that can gesture and perform sign language, but with the time and resources provided, our overall goal was too ambitious.

References

"Compact Bionic Hand." *Mahdi Designs*. https://mdesigns.space/3d-printing-cad-files/p/compact-bionic-hand. 24 March 2023.

Kim, U., Jung, D., Jeong, H. et al. Integrated linkage-driven dexterous anthropomorphic robotic hand. *Nat Commun* 12, 7177 (2021). https://doi.org/10.1038/s41467-021-27261-0

"Hubs x Aslan Project - Sign Language Humanoid Robot." YouTube, uploaded by Hubs, 15 Sep 2017, https://www.youtube.com/watch?v=S1eljmSxGRA

"Easy Driver Hook-up Guide." Sparkfun, https://learn.sparkfun.com/tutorials/easy-driver-hook-up-guide

"Dynamixel Shield E-Manuel" Robotis, https://emanual.robotis.com/docs/en/parts/interface/dynamixel_shield/

"RBG Matrix Panel Github" Github, https://github.com/adafruit/RGB-matrix-Panel