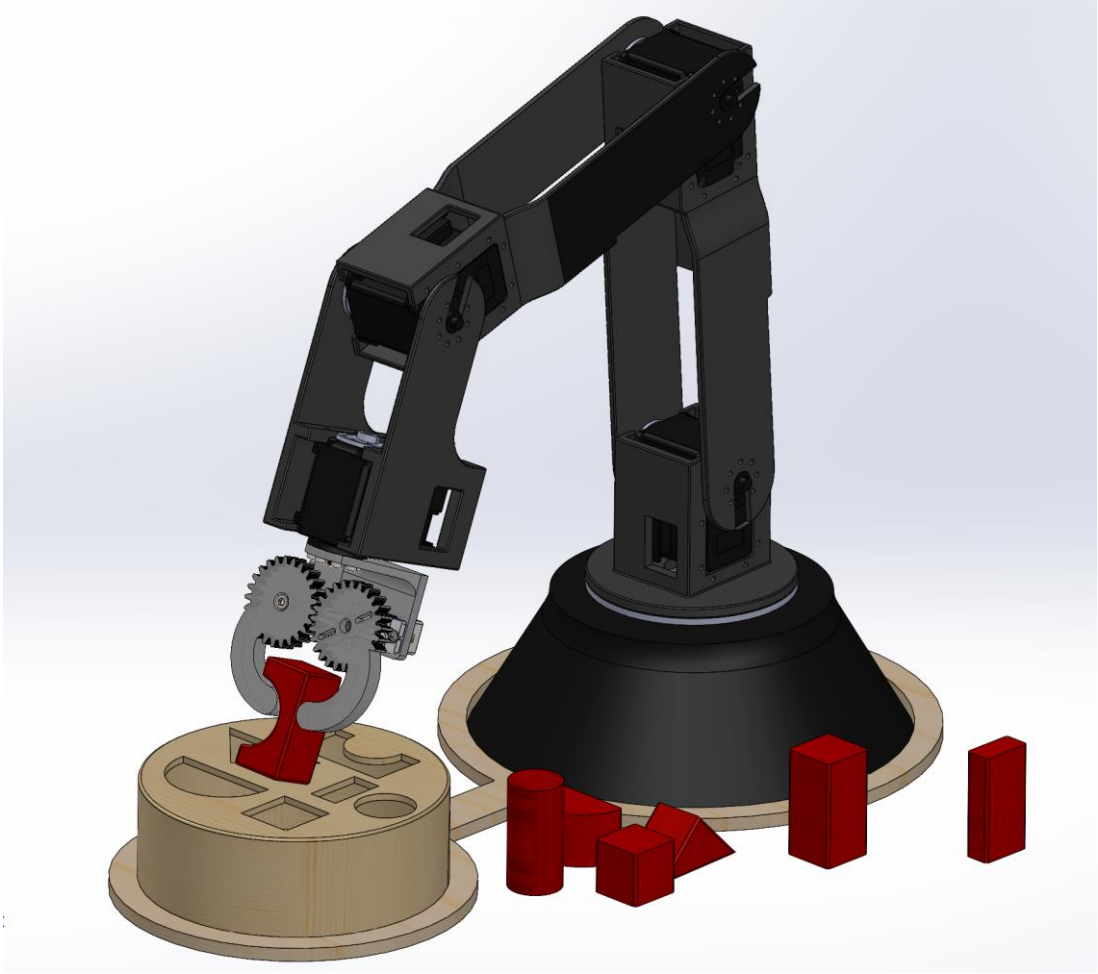
It Goes in the S.Q.U.A.R.E Hole!



MAE 263A

Divik Bhargava
Federico Parres
Julia Yuan
Jane Liu
David Bughman
Atharva Shetye

Intro

The purpose of this project is to test the students' understanding and ability to plan, design, and assemble a functioning robotic manipulator utilizing the methods of determining the forward and inverse kinematics of a robotic manipulator. In addition to the skills learned in class, the students must also demonstrate their ability to create manufacturable and assembleable designs that can withstand the normal operations of a robotic arm. To demonstrate the understanding and skills required, this team chose to design a robotic manipulator whose purpose was to recreate the internet viral video "It Goes in the Square Hole!", in which a robotic pick-and-place arm would grab blocks of various shapes and place them not only into their designated repository but also ensure that they also erroneously fit into the square hole. A 5 DoF robotic arm, named **S**uperior, **Q**uick **U**tility **A**rm **R**eplicating **E**ntertainment (SQUARE) , was designed, programmed, and manufactured to achieve this goal and did so successfully with minor alterations to the design.

Design

The arm consists of five degrees of freedom controlled by five revolute joints, shown in Figure 1. Numbered starting from the base, the joints are referred to as Joints 1-5.

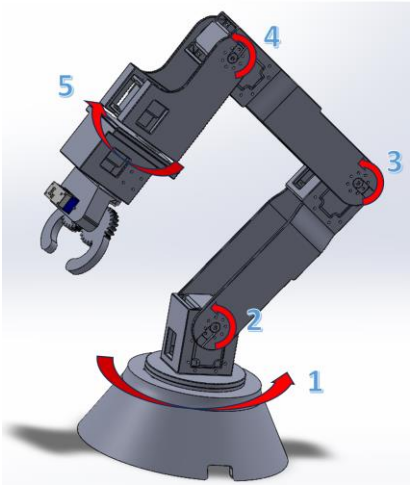


Figure 1

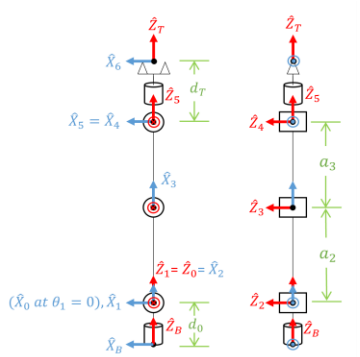
The three degrees of freedom provided by Joints 1-3 ensured the placeability of the end effector in 3D space. The last two degrees of freedom, Joints 4 and 5, were encapsulated in the wrist in order to provide tip-and-tilt capability at the wrist. These two additional degrees of freedom were added to simplify the pick and place operation. Joint 4 allows us to restrict the gripper in a vertical configuration and Joint 5 provides axial rotation to turn the shape to align to the hole.

During design of parts, care was taken to reduce the number of offsets between links. This was done by integrating the motors into the center of the structure, and aligning the axes of each motor when possible. In the end, only four nonzero link lengths and offsets were necessary to describe the arm.

Kinematics

Forward Kinematics

To derive the forward kinematics, coordinate frames were assigned and Denavit-Hartenberg (DH) parameters determined to determine the frame transformations between joints. To account for the offset between the robot mounting surface, as well as the distance between the last joint and the gripping location of the end effector, two additional frames not associated with joint parameters were added. The end result of the forward kinematics calculation is shown below.



$$ {}^{B}_{0}T = \begin{matrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & d_0 \\ 0 & 0 & 0 & 1 \end{matrix} \quad {}^{1}_{2}T = \begin{matrix} c_2 & -s_2 & 0 & 0 \\ 0 & 0 & -1 & 0 \\ s_2 & c_2 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{matrix} \quad {}^{3}_{4}T = \begin{matrix} c_3 & -s_3 & 0 & a_2 \\ s_3 & c_3 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{matrix} \quad {}^{5}_{T}T = \begin{matrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & d_T \\ 0 & 0 & 0 & 1 \end{matrix} $$

$$ {}^{0}_{1}T = \begin{matrix} c_1 & -s_1 & 0 & 0 \\ s_1 & c_1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{matrix} \quad {}^{2}_{3}T = \begin{matrix} c_3 & -s_3 & 0 & a_2 \\ s_3 & c_3 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{matrix} \quad {}^{4}_{5}T = \begin{matrix} c_5 & -s_5 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ -s_5 & c_5 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{matrix} $$

$$ {}^{0}_{T}T = \begin{matrix} c_1c_{234}c_5 - s_1s_5 & -c_1c_{234}s_5 - s_1c_5 & -c_1s_{234} & c_1(a_3c_{23} + a_2c_2 - d_Ts_{234}) \\ s_1c_{234}c_5 - c_1s_5 & -s_1c_{234}s_5 + c_1c_5 & -s_1s_{234} & s_1(a_3c_{23} + a_2c_2 - d_Ts_{234}) \\ s_{234}c_5 & -s_{234}c_5 & c_{234} & d_0 + a_3s_{23} + a_2s_2 + d_Ts_{234} \\ 0 & 0 & 0 & 1 \end{matrix} $$

Figure 2.

The link lengths were chosen to be d_0= 10.2 cm, a_2= 14.5 cm, a_3= 14.5 cm, d_T= 14.8 cm. The workspace of the arm not accounting for collisions is a sphere centered around the second joint. In 3D the dexterous workspace is only along the centerline of Joint 1.

Inverse Kinematics

To determine the joint positions to reach a desired end effector position, the following equality must be solved.

$$Goal = \begin{matrix} r_{11} & r_{12} & r_{13} & x \\ r_{21} & r_{22} & r_{23} & y \\ r_{31} & r_{32} & r_{33} & z \\ 0 & 0 & 0 & 1 \end{matrix} = {}^{0}_{T}T = \begin{matrix} c_1c_{234}c_5 - s_1s_5 & -c_1c_{234}s_5 - s_1c_5 & -c_1s_{234} & c_1(a_3c_{23} + a_2c_2 - d_Ts_{234}) \\ s_1c_{234}c_5 - c_1s_5 & -s_1c_{234}s_5 + c_1c_5 & -s_1s_{234} & s_1(a_3c_{23} + a_2c_2 - d_Ts_{234}) \\ s_{234}c_5 & -s_{234}c_5 & c_{234} & d_0 + a_3s_{23} + a_2s_2 + d_Ts_{234} \\ 0 & 0 & 0 & 1 \end{matrix}$$

This is done by following the process explained in the flowchart shown in Figure 3. For details of the inverse kinematics derivation, see the handwork in the attached zip file.

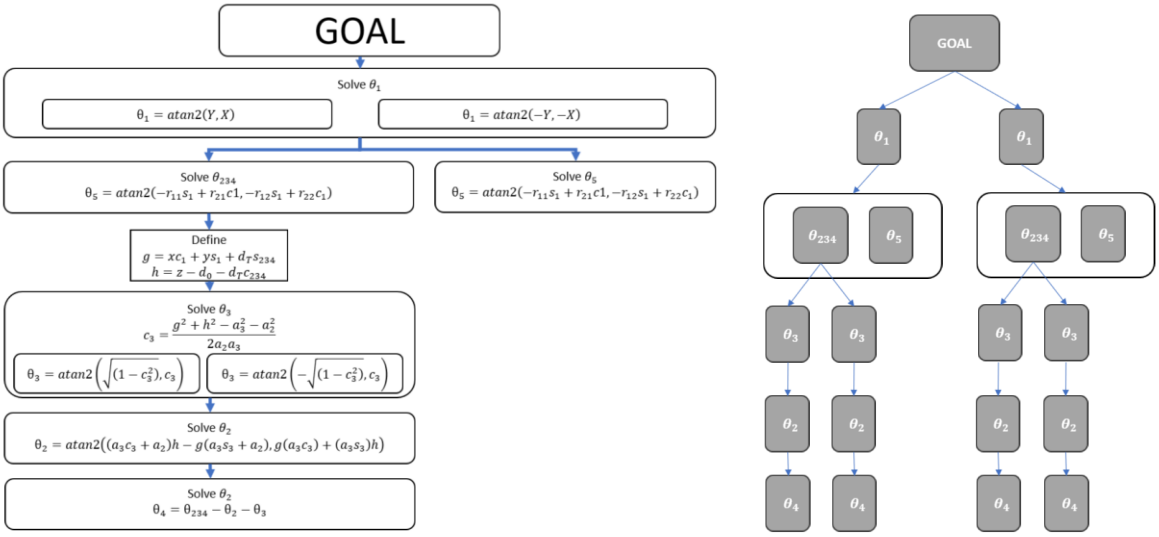


Figure 3

As can be seen from the flowchart, there are two deviation points during the inverse kinematics that results in four solutions for most goal positions in the workspace. One of the divergences occurs because the “elbow”, the location of Joint 3, is associated with an “elbow up” and an “elbow down” configuration. The other occurs because Joint 1 can turn 180° and the arm can reach behind itself to achieve the same tool position. These four solutions are shown in the diagram in Figure 3. The four solutions reduce to two solutions when the elbow is fully extended, removing the “elbow up” and “elbow down” configurations. In addition, if the tool tip is directly above the base such that the joint axes of Joint 1 and Joint 5 are aligned, there are infinite solutions due to there being infinite acceptable combinations of Joint 1 and 5.

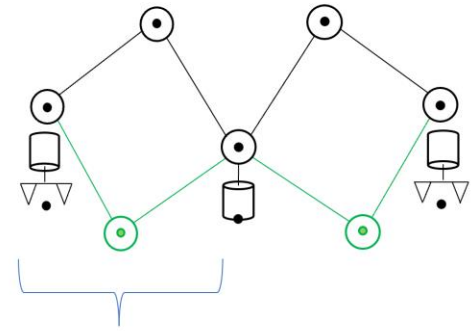


Figure 4

Solution Selection

For most parts in the workspace, there are four solutions possible. There are two configurations possible for θ_1, which determines if the arm reaches forward or backwards to the goal position. For each of the two θ_1 values, there are two possible elbow configurations. To choose one solution, one of the θ_1 values was eliminated. Since there's no benefit to one or the other solution because they're 180° apart, the first θ_1 value was chosen. We are then left with two solutions, one with elbow up and the other with elbow down. We chose to bias the elbow up position to allow the gripper to be pointed straight down, and to also avoid any potential interference with the elbow hitting the floor.

There are two possible solutions when the goal position is at the edge of the workspace. In this case, there will be two possible θ_1 values. Since there's no benefit to either one, we choose to pick the first θ_1 solution. When the goal position is on top of the base, there are infinite solutions due to the free rotation of joint 1 and the wrist joint. In this case, we lock θ_1 to 0°, which gives us one solution.

Computer Aided Design

In this robotic manipulator arm, importance was given on the moments generated due the different weight of the motors and the arm components. Hence, a conservative factor of safety of 3 was chosen. This ensures that any kind of weight changes over the links are accounted for. The design then follows to make the assembly easier. Specifically, motor housing was integrated with the links using press fit to make the assembly easier and secure the motor within the link. Having decided the assembly, preference was then given to the overall range or workspace of the robotic arm. The links are designed in such a way that the whole robot can rotate 360 degrees around its base and the subsequent joints can make about 270 degree rotation. This increases the overall range, which enables the existence of a dextrous workspace as previously defined. Wall thickness of the components was decided based on the printing capabilities of the 3D printer and material. Throughout the design, it is fixed at 2.5 mm. The overall design of the links is a simple C cross section making them more resistant towards bending. The design has been inspected for stress points and interference. Accordingly, changes were made to add filets wherever necessary or provide smooth natural curves to make the design more robust.

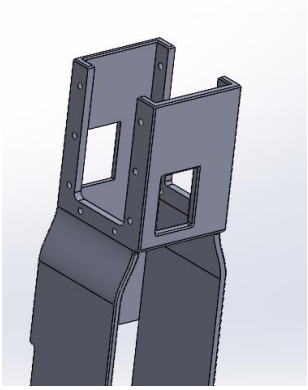


Figure 5

The gripper is a simple two claw mechanism, consisting of three 3D printed parts, the two claws and the mount. The claws are geared together, with one driven by a SG90 servo. The mount is designed so that the center of the gripping fingers is aligned with the rotational axis of Joint 5.

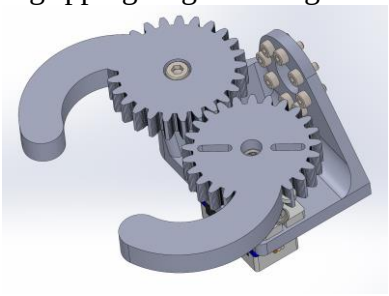


Figure 6

Although simplistic, designing the props to be accurate to those from the video was important in order to fit the theme of the project. In addition to being screen accurate, they must also be compatible with 3D printing manufacturing. Each shape was made to scale based on the cube, which was a predetermined 20 mm across. This ensured that every shape would be able to fit through the square hole, so long as it was oriented properly. In the video, the shapes have bevels on every edge, however for some shapes this was excluded to ensure improved bed adhesion to the 3D Printer.

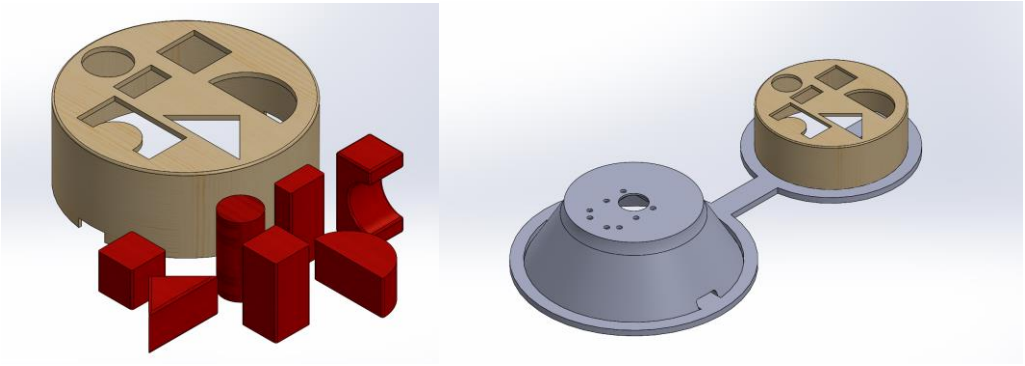


Figure 7

The base of the robotic arm, as well as the alignment jig were modeled in such a way to ensure ease of programming for the inverse kinematics. The alignment jig placed the shape

repository a known 175mm away from the center of the robotic arm, and each hole had a 2mm gap to allow for inaccuracy in the robotic arm's motion. The base of the robot arm was modeled to be a large cone for ease of printing, as well as increasing the footprint of the surprisingly heavy robotic arm to reduce tilting. Connecting the base and the alignment jig is a clamping ring which matches the angle of the cone, with enough offset to ensure a safe grip when clamped between the table. The base as well as the shape repository included notches which fit snugly into the alignment jig to ensure that they remain at a fixed angle and distance to each other, thus allowing any troubleshooting to be done lie solely with the inverse kinematics and code for the robotic motion.

Programming

For scripts, functions, and details of code, see the provided zip file.

Implementation of Kinematics

The forward and inverse kinematics are implemented in several MATLAB scripts and functions. First, a script that takes the DH parameters in the form of a DH table calculates all matrix multiplications in a symbolic form. It saves each transformation matrix into a cell array as a function handle that may be evaluated by feeding in the joint angles, as well as a vector of the DH constants. All following scripts are able to load in the saved transformation matrices and DH parameters.

The forward kinematics function simply takes joint positions as input and feeds it to the necessary transformation matrix as previously calculated. For visualization purposes, a script that plots a line representation of the arm given a set of joint angles, and a script that creates a video of the arm moving between given sets of joint angles were written. Images of the outputs of these scripts are shown in Figure 8.

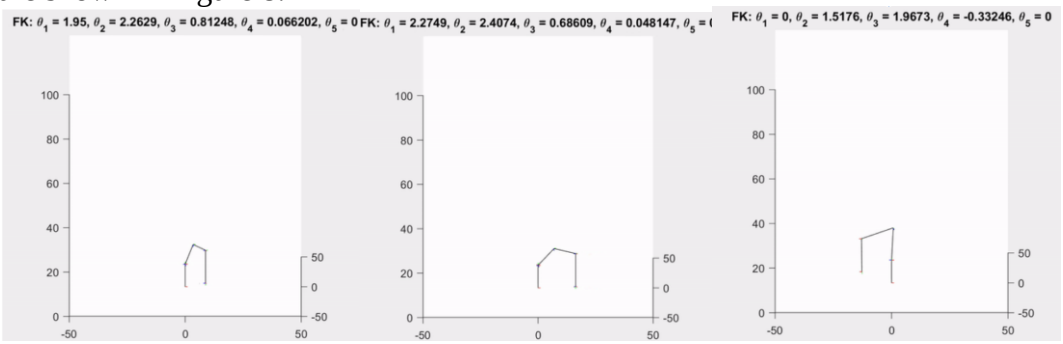
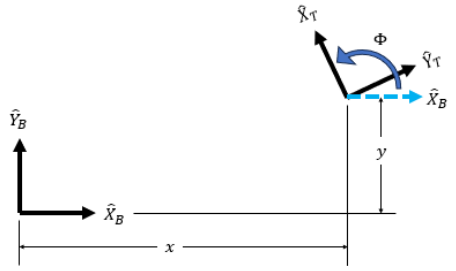


Figure 8

The inverse kinematics consists of two nested functions. The first implements the calculation process described in the Inverse Kinematics section. It calculates all four solutions– with duplicate solutions in the case of two solutions–and performs no checks. In the case of infinite solutions, Joint 1 is prescribed at 0 and two solutions are returned. The first function is nested into a script that selects a single solution to be used. Solution selection is implemented by choosing a solution out of the available selection. First, the evaluation of Joint 1 is restricted to one consistent atan2 function. This ensures the arm does not need to flip over itself to reach any new position. The script also checks if any of the solutions causes the robot to clip the ground. That is checked

by evaluating the vertical component of each joint. Lastly, if both remaining solutions are considered valid, the one associated with "elbow up" behavior is chosen.

For generation of goal frames to be fed to the inverse kinematics is done by specifying each goal position as an 4-entry vector [x y z ϕ], where ϕ is the angle between the x-axis of the base and the gripper x-axis. Since the gripper is restricted to the vertical orientation, these 4 degrees of freedom are sufficient to generate the tool frame using the matrix, shown below.



$$\text{Tool Frame} = \begin{bmatrix} c\phi & s\phi & 0 & x \\ s\phi & -c\phi & 0 & y \\ 0 & 0 & 1 & z \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Figure 9

Arm Motion and Motor Implementation

S.Q.U.A.R.E makes use of 5 Dynamixel MX-28AR motors for its main appendage and a single arduino hobby servo motor for its gripper. One advantage of the MX-28AR's is their ability to be daisy-chained together, which simplifies the wiring of the system significantly. The single chain of these motors is fed into a 6 port MX power board- supplying 15V from a wall outlet- which is also connected to a U2D2 usb communication converter. This is then connected to a computer and allows each Dynamixel motor to be controlled via MATLAB. The servo controlling the gripper is on a separate system as it is wired independently to an Arduino Mega 2560. The Arduino supplies power to the servo and a controller pin, and the whole board is also connected to the computer which utilizes the MATLAB-Arduino interface to control the servo's position.

MATLAB was used to control the U2D2- and consequently Dynamixel motor chain- as well as the Arduino Mega- which in turn controlled the gripper servo. The main MATLAB script begins by loading necessary Dynamixel libraries and connecting to the COM port used by the U2D2. The desired pick-up, drop off, and intermediate positions are passed to an inverse kinematics solver which returns the appropriate joint angle of each motor. The program then enables the torque of each dynamixel so their positions can be updated before moving S.Q.U.A.R.E into a predetermined home position. This makes use of a separate MATLAB script which adjusts the speed of each motor and sequentially updates their position in an adjustable order. The main program then moves S.Q.U.A.R.E. into the pickup position (as determined from inverse kinematics) before querying the user for a specific block and its corresponding hole. Regardless of the answer, a function is called to actuate the gripper, the arm is moved to the intermediate and drop of position before releasing the block into the square hole. On its way back to the pick up location, S.Q.U.A.R.E. returns to the intermediate position once more. This cycle repeats for a specified number of blocks, or until the user enters "End" when prompted. After this, the program asks the user to confirm the arm is in a safe position before pressing enter to deactivate the motors and disconnect from the U2D2.

Manufacturing

The components for the robotic arm were manufactured on two 3D printers, the first of which was a heavily modified CR-10S, and the second being a Prusa MK3s. The Prusa printer was primarily used to rapidly print a last minute redesign of the claw out of standard PLA, as the initial design was found to be too large and effected the dexterity of the arm. The rest of the arm was printed on the CR-10S, utilizing carbon-fiber laced PLA, the improved the rigidity and reduced brittleness of the material allowing for a more lightweight design, with thinner walls and minimal reinforcement. However, this improved material also brought difficulties to 3D printing that were not expected, such as poor layer adhesion, inconsistent extrusion, and significant stringing. The prints were done at a standard 0.2mm layer height, at a relatively high temperature of 225C and increased extrusion by 15% to counteract the negative aspects of the carbon fiber PLA.

3D models of the robotic arm were not found to be particularly friendly to additive manufacturing, as the design focused on ease of assembly and reduced usage of material for the final structure. However, this philosophy resulted in significant usage of support material when printing the design, specifically within the motor casings, which were troublesome to remove. Despite the use of significant support material, the motor casings were within acceptable tolerances when completed and still allowed for the motors to be press fit safely into their final location.

The most difficult design element to print were those of the links themselves, their thin walls made for delicate printing which required the printer to be slowed to ensure that the M2 screw holes on the their sides were printed correctly. In addition to their thin walled design, due to a small lip located on the bottom of the link, required them to be printed entirely on support material. This negatively affected the finish as well as the accuracy of the final product, however the end result was acceptable enough to be utilized for final assembly.

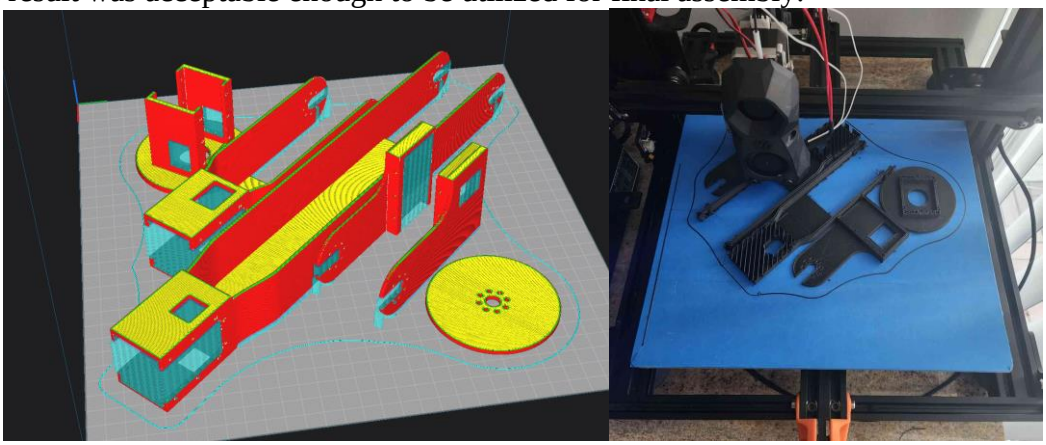


Figure 10

Results

The main obstacles of the project were programming issues and troubleshooting using the Dynamixel MX-28AR's. When initially attempting to control the Dynamixel motors individually, the computer being used was unable to connect to the chip in the U2D2. After attempting to resolve the issue, a separate computer was used and immediately able to connect to the motors using the Dynamixel Wizard software. The motors were originally intended to be controlled using the Arduino Mega and a MAX485 serial communication converter, but further connectivity issues and increased wiring complexity made this impossible. Consequently, the system returned to using the

U2D2 to be controlled by a MATLAB script to simplify the design. This too proved more difficult than anticipated, as the computer used to initially connect to the Dynamixels was unable to transmit information through MATLAB. Consequently, a third computer was used which was able to both communicate with and transmit information from the program. One of the final programming issues faced was a directory issue, in which a function called in the main script of the program was unable to be accessed, likely due to a conflicting function name along the file path. This was eventually resolved by changing the function to an entire separate main script which was then called in the original main function after updating necessary parameters.

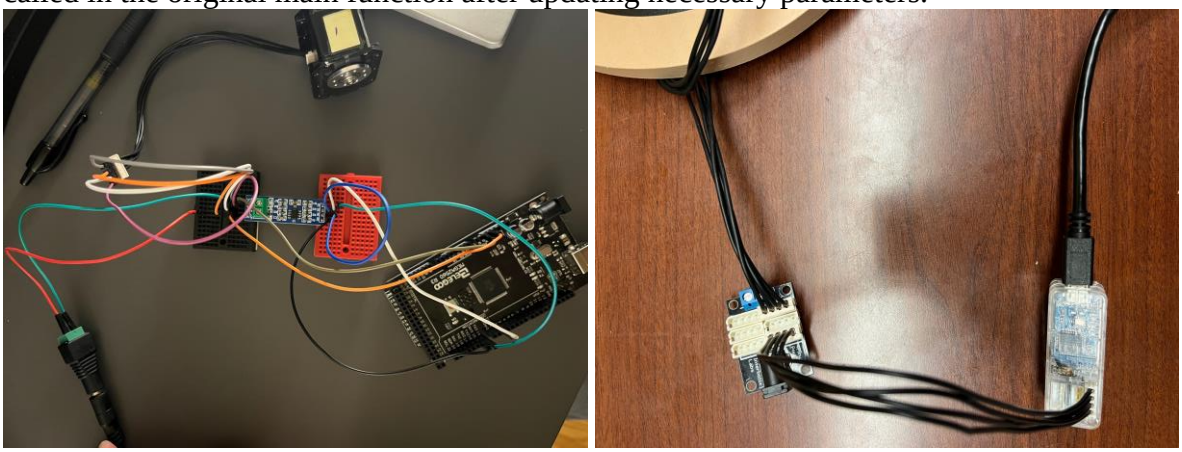


Figure 11

The final assembly was completed successfully and able to run the demo. The final assembly is shown in Figure 12.

Figure 12

The reliability of the shape dropoff was less than ideal due to some deformation in the structure of the arm, but mainly due to the shape pickup resulting in a slightly misaligned arm. When the gripping phase completed properly, the arm was able to consistently drop the shape into or very near the square hole goal.