RoboXO

Tic Tac Toe Robot

MAE C263A - COURSE PROJECT

Fall 2021

TEAM 15

Rubin Jacob

Maharshi Pathak

Yash Trivedi

Shubham Wani

Raj Intwala

I. Introduction

Have you ever been in a situation where you had a deep desire to play Tic Tac Toe, but no one to play the game with? Our team understands how devastating it is to be in such a situation. Therefore, we came together to eliminate this problem by creating a robotic system that can play Tic Tac Toe with you when you are alone. The system is in the form of an open loop robot arm mechanism that plays on a Tic Tac Toe board attached to the frame of the robot. To play the game, the user will indicate the spot where they would like to place their game piece. Then the robot will pick up the user's game piece (either X or O) and place it in that spot. Then the robot will select it's game piece and place it. This cycle continues until either the robot or the player wins or when the game board is full.

The robot arm uses a combination of prismatic and revolute joints to navigate the end effector along the plane of the game board and move it to pick up and place game pieces. An electromagnet is attached to the end effector, which is used to pick-up and place the desired piece. A majority of the pieces were 3D printed from the makerspace and the remaining components were purchased. The servos provided were used to control the revolute and prismatic joints and they were programmed through MATLAB and RoboPlus Manager. All in all, though we faced many challenges, we were able to successfully create a robot arm that is capable of playing Tic Tac Toe with a user. This report details the various aspects of the design, analysis, and testing.

II. Design

A. Design Choices

Our robot arm is depicted in Figure 1 below. All of the components were modeled and assembled in Solidworks; the full assembly is also shown in Figure 1.

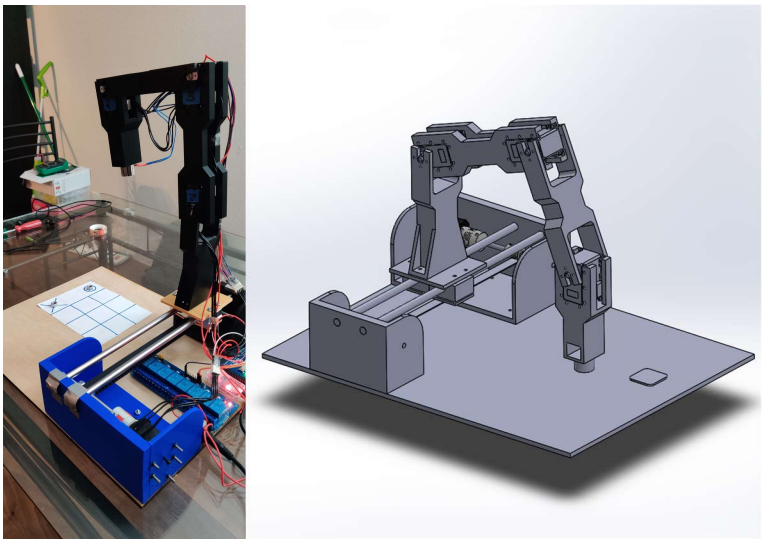

Figure 1: Fully assembled robot arm with game board (left). Solidworks assembly of robot arm (right).

The first joint in the design is a prismatic joint actuated with a belt drive that allows horizontal movement relative to the game board. Two linear rails sit between two 3D printed bases and linear bearings move across the rails. The top face of the linear bearings attaches to a platform that is coupled to the ground link, as depicted in Figure 2. A belt clamp is used to couple a timing belt to the bottom face of the linear bearings which allows them to move along with the belt. The timing belt is connected to shafts on each base. On one base, a servo, mounted to it, is used to drive the rotation of the belt using a timing pulley and motor shaft coupler. On the other base, an idler pulley allows rotation of the belt along that shaft.

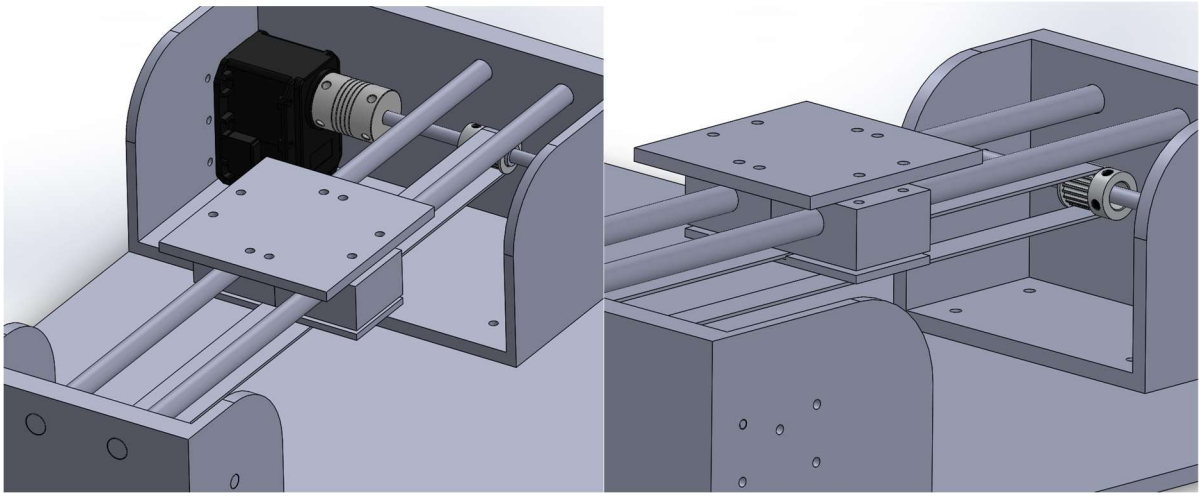

Figure 2: Belt drive system with platform that connects to linear bearings.

The next link is mounted to the prismatic platform with nuts and bolts and the other end connects to the rotating horn on the second servo with screws. The next link attaches to the second and third servo base on each end using nuts and bolts, and it was split into two parts for mass reduction and more efficient 3D printing. The following link attaches to the rotating horn of the third and fourth servo. The final link attaches to the base of the fourth servo on one end and the other end attaches to the electromagnet with a single screw. Each link that attaches to a servo horn was given 53 mm of clearance to allow the joint to rotate 264 degrees. All of these are depicted below in Figure 3.

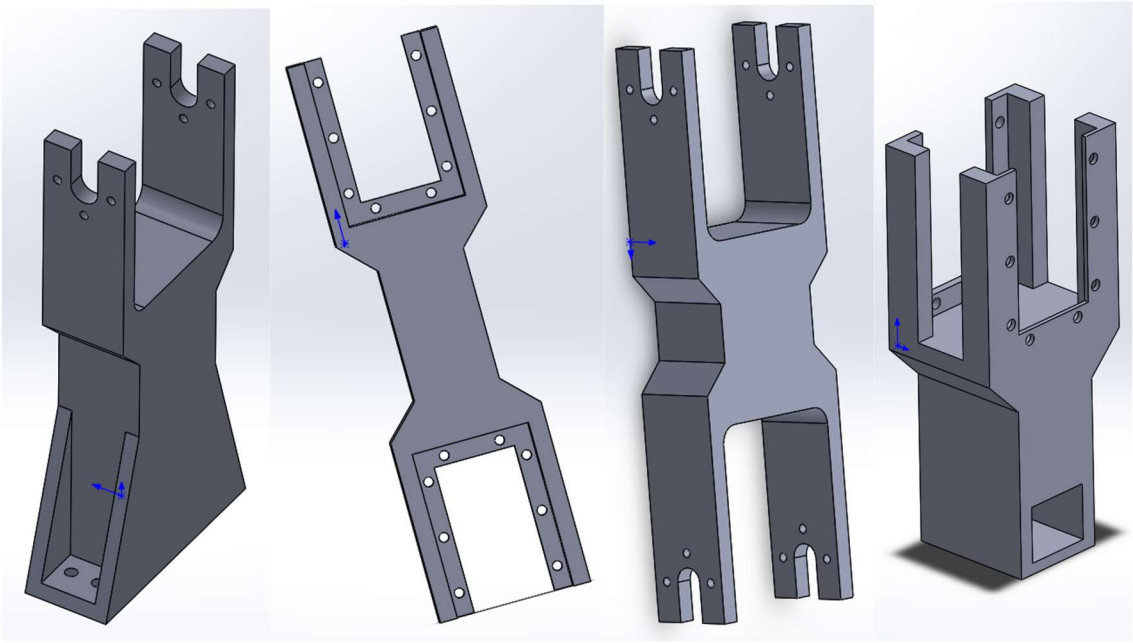

Figure 3: Link 1, 2, 3, and 4 in order from left to right. Note that only one of the two identical components for link 2 is shown.

The prismatic joint at the base was implemented so that the horizontal motion achieved could be decoupled from movement in the perpendicular plane achieved by the revolute joints. This simplified the calculations and programming of the arm. A belt drive was chosen for the prismatic joint because it is one of the most simple, easy-to-implement, and cost-effective methods to achieve linear translation with a motor. Lastly, an electromagnet was chosen to pick and place the metal pieces because this method allows for better repeatability and eliminates the risk of component failure.

B. Workspace Analysis

When determining the workspace of the manipulator, the workspace of the revolute joints was determined first. Due to the design choice mentioned earlier, the angle of each joint could range from +134 degrees to -134 degrees. Starting from the last joint and working our way back, the reachable workspace of the revolute joints is shown in the first part of Figure 4. Due to self-collision, the end effector cannot reach a circular section in the middle of the workspace, as indicated. Since the first joint is prismatic, the 3D workspace is achieved by extruding the revolute joint workspace along the direction of the linear rails until the reachable positions of the belt drive are reached. The front view of the 3D workspace is shown in the second part of Figure 4. Due to the lack of 360 degree joints and self-collision, there is no dexterous workspace.

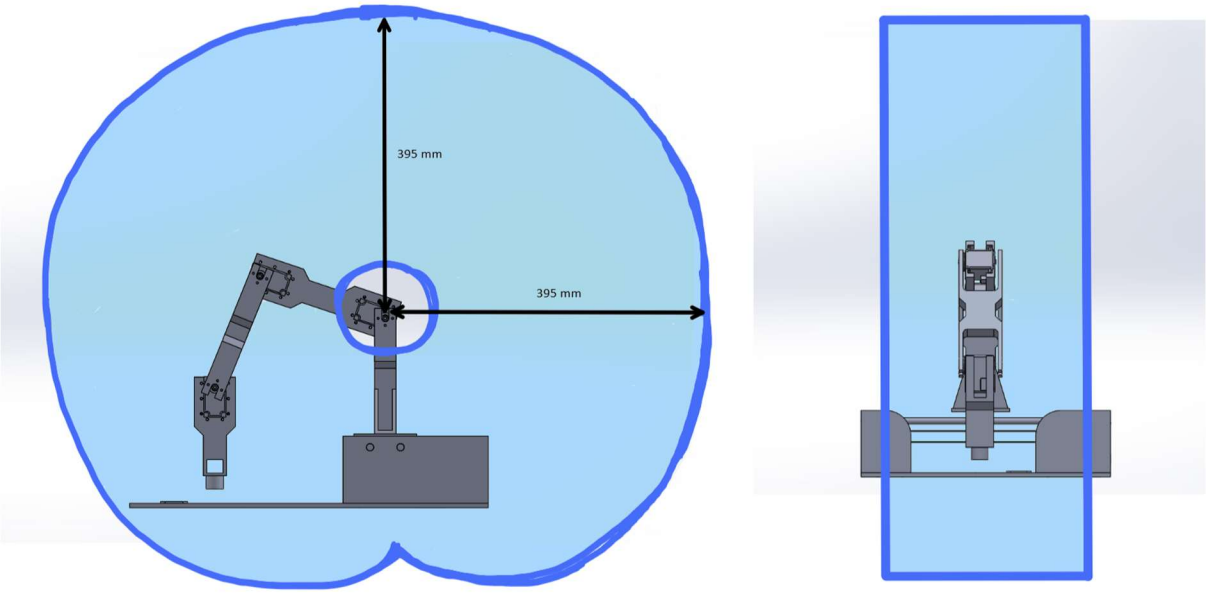

Figure 4: Side profile of the 3D workspace achieved by the revolute joints (left). Front profile of the 3D workspace achieved by the prismatic joint (right).

C. Static Force Analysis

The holes (attached to servos) at one end of each link were considered as fixed supports while a moment of 3000 Nmm (well above Stall torque of the servos) was applied to the holes at the other end. The factor of safety was also calculated to be over 2 in each case. It is evident that the parts are structurally well suited for our application.

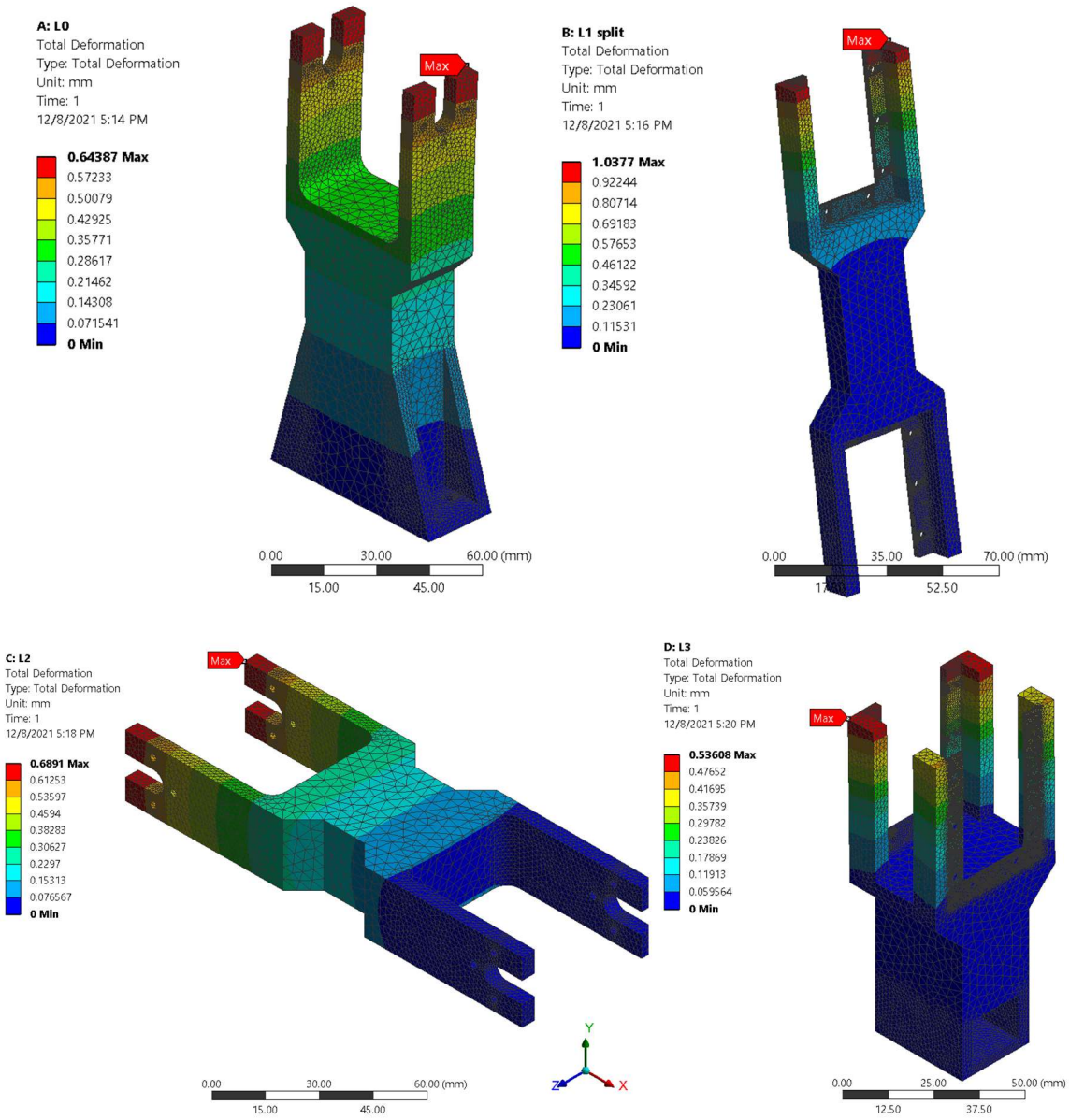

Figure 5: Static Structural Analysis (Total Deformation) on Ansys Workbench for each link - L0, L1, L2, L3

D. Design Highlights and Difficulties

1. Wooden base was laser cut at makerspace
2. 3D printed components from makerspace - blue base (to mount servo #1 and timing pulley connected via a rod), black base (to mount idler pulley connected via a rod), robotic arm links - L0, L1, L2, L3, platform (to mount link L0), bush (to rigidly attach the timing pulley

to the rod connected to servo #1), belt clamp (to attach belt to the bottom of the linear bearings)

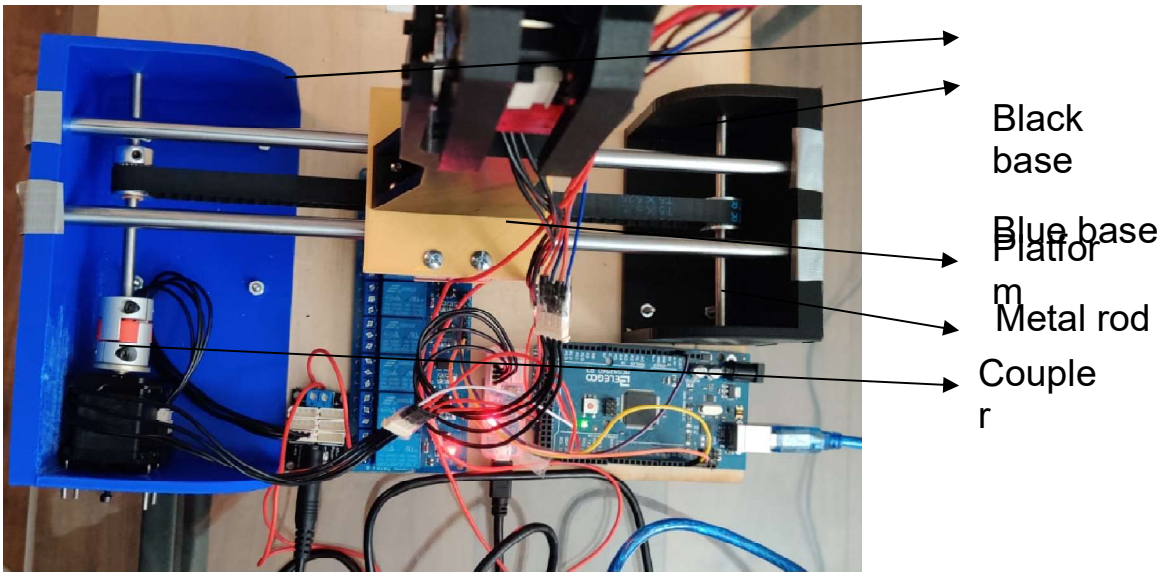

Figure 6: Some components labelled

3. *Components ordered* - electromagnet, relay, T5-500 timing belt, timing pulley, idler pulley, flexible shaft coupler, linear rail, metal rods

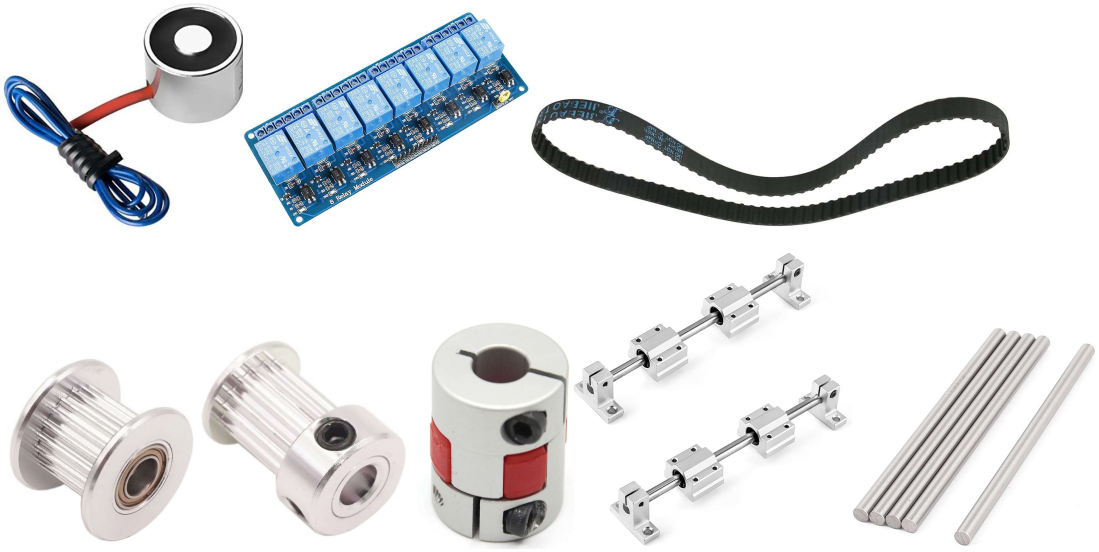

Figure 7: Components ordered

4. Clearances for holes = 0.4mm. Example - for an M3 bolt, the hole size was designed as 3.4mm in the CAD model for the 3D printed part.
5. *Difficulties:* Most of the difficulties revolved around the prismatic joint, which included the following components - the belt, timing pulley, idler pulley, pulley rods, linear bearing and aluminium shafts. The belt needed to be properly tensioned to prevent the belt teeth slipping over the pulley teeth when driven by the servo. This required us to make the following modification. The black base (on which the idler pulley is mounted) was attached to the wooden board via nuts and bolts through holes. It was difficult to attach the belt as it was too tight. So we converted the holes on the wooden base into slots. Then, we attached the belt and pulled the black base away to appropriately tension it, and subsequently tightened the bolts. This ensured that there was no slipping.

III. Forward and Inverse Kinematics

A. Forward Kinematics

RoboXO has a PRRR type of mechanism. The first joint, which is prismatic, helps the robot cover distance in the YZ plane - the plane of the board. Given a goal $[P_x\ P_y\ P_z]$, the robot first moves along the Z axis via the prismatic joint to reach P_z. Once it reaches the desired location along the Z axis, it only needs to move in the XY plane to lift/place the piece. This is achieved through a series of RRR pairs. The frame assignments for RoboXO are shown in Fig 8.

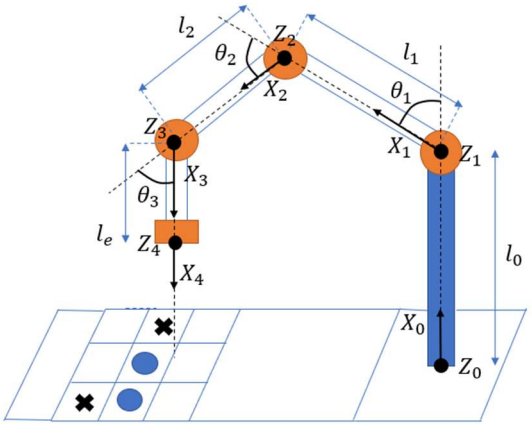

Figure 8: Frame assignments for RRR mechanism

The manipulator configuration has been intentionally chosen in such a way as it decouples the P and RRR portions of the mechanism and allows us to treat them separately. This makes the kinematics much more simple to solve. We do not need to explicitly involve the prismatic joint in the kinematics. The *DH* parameters of the mechanism (RRR) are as given in the table below:

Table 1: DH Parameters for RRR mechanism

$i-1$	i	α_{i-1}	a_{i-1}	d_i	θ_i
0	1	0	l_0	0	θ_1
1	2	0	l_1	0	θ_2
2	3	0	l_2	0	θ_3
3	4	0	l_e	0	0

The transformation matrix linking the base frame {*0*} to goal frame {*4*} is:

$$^0T_4 = {}^0T_1 \ {}^1T_2 \ {}^2T_3 \ {}^3T_4 =$$

$$\begin{bmatrix} C_{123} & -S_{123} & 0 & l_0 + l_1C_1 + l_2C_{12} + l_eC_{123} \\ S_{123} & C_{123} & 0 & l_1S_1 + l_2S_{12} + l_eS_{123} \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

B. Inverse Kinematics

To further simplify the inverse kinematics, the constant link of length l_0 (all DH parameters for the first link are constant) is not explicitly required to be considered while solving for the angles, i.e. inverse kinematics is calculated about (X_1, Z_1) origin. We can later incorporate l_0 in the goal position coordinates of the robot.

$$^1T_4 = {}^1T_2 \ {}^2T_3 \ {}^3T_4 =$$

$$\begin{bmatrix} C_{123} & -S_{123} & 0 & l_1C_1 + l_2C_{12} + l_eC_{123} \\ S_{123} & C_{123} & 0 & l_1S_1 + l_2S_{12} + l_eS_{123} \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Thus,

$$p_x = l_1C_1 + l_2C_{12} + l_eC_{123}$$
$$p_y = l_1S_1 + l_2S_{12} + l_eS_{123}$$
$$p_z = 0$$
$$\varphi = \theta_1 + \theta_2 + \theta_3$$

where φ is the final orientation of the end effector in the X-Y plane. For our application, $p_x,\ p_y, \varphi$ is known. Since φ is known, S_{123} and C_{123} are constants.

Let $p_x' = p_x - l_eC_{123}, p_y' = p_y - l_eS_{123}$,

$$p_x' - l_1C_1 = l_2C_{12} \quad \ldots(1)$$
$$p_y' - l_1S_1 = l_2S_{12} \quad \ldots(2)$$

Squaring and adding the above equations,

$$2l_1p_x'C_1 + 2l_1p_y'S_1 = (p_x')^2 + (p_y')^2 + l_1^2 - l_2^2 \Rightarrow PC_1 + QS_1 = R$$

$$\theta_1 = atan2(\pm\sqrt{P^2 + Q^2 - R^2}, R) + atan2(Q, P)$$

To calculate θ_2,substitute θ_1 in (1) and (2),

$$\theta_2 = atan2(\frac{p_y' - l_1S_1}{l_2}, \frac{p_x' - l_1C_1}{l_2}) - \theta_1$$

Thus,

$$\theta_3 = \varphi - \theta_1 - \theta_2$$

IV. Code

As Linus Torvaldus, the creator of Linux said, “Talk is Cheap, Show me the code.”

a) Selection

The programming is done in MATLAB, by Mathworks Inc, due to its efficient matrix manipulations and built-in hardware support packages for Arduino and Dynamixel.

b) **Animation**:

The IK program was tested in MATLAB. A simple straight line was achieved. The IK program was called at multiple points on the straight line and solved at every step. The plots were created at every step and compiled into an animation.

c) **Task Programming**:

The Dynamixel motors in Daisy Chain mode were connected via a USB through U2D2 using the RS-485 protocol, which needed to be initialized with the MATLAB SDK provided by ROBOTIS. During initialisation, we established the serial communication at a baud rate of 57600 and opened the port. We enabled the torque of the motor by writing to the RAM area of the control table using the write1ByteTxRx function. We similarly reduced the speed of the motors by using the write4ByteTxRxfunction, which was also in the RAM area. Note that the parameters in the RAM area of the control table need to be initialized after every reset, but those in the EEPROM do not need to be set again. The electromagnet, to pick up the pieces, was operated via a relay, which was controlled via an Arduino Digital pin.

As we were controlling a physical structure, we parameterized all the variables (e.g. link lengths, offsets, board size) for easy manipulation in case of design changes. The home position is achieved after every movement to ensure that the arm does not collide with other X/O pieces and avoid configurations which will cause structural damage. A while loop is used to constantly switch between player 1 and 2. The game starts with the input from the player 1 (X token by default), which tells us the block number we need to place the X/O token. The robot calculates the position and the motor angles i.e. inverse kinematics for the position where it needs to pick up the token based on the number of tokens and the token height. The X token pickup location is block 10 and the O token pickup location is block 12. The {x,y,z} are calculated based on the offset and the block number, with the block size defined in the program.

We get two solutions for the IK{Θ2,Θ3,Θ4} based on the elbow up/ down configuration. The optimal solution is chosen based on the least motor movement. The prismatic joint is defined by the Θ1, which is calculated based on axle radius (r) and the commanded z coordinate using formula Θ1=z/r . The motion is executed using the write4ByteTxRx function by supplying the angle value mapped between 0 and 4095 typecasted to the uint32 data format. Sufficient pause commands are used to ensure the motion is achieved before the next command execution. The electromagnet is turned on by writing a digital HIGH to the D13 GPIO, which triggers the relay circuit. This ensures that the token piece is attached to the arm. The goHome() function is used to

command the arm to go to its home position. The new goal location is based on the user's input location, where the token is to be dropped. The same procedure is followed to drop the token by writing digital LOW to the D13 GPIO, which causes the token to detach from the electromagnet. The goHome() function is commanded again and the Player 2 user input is awaited. This program runs in a loop until the game is over, at which point, we may reset the game.

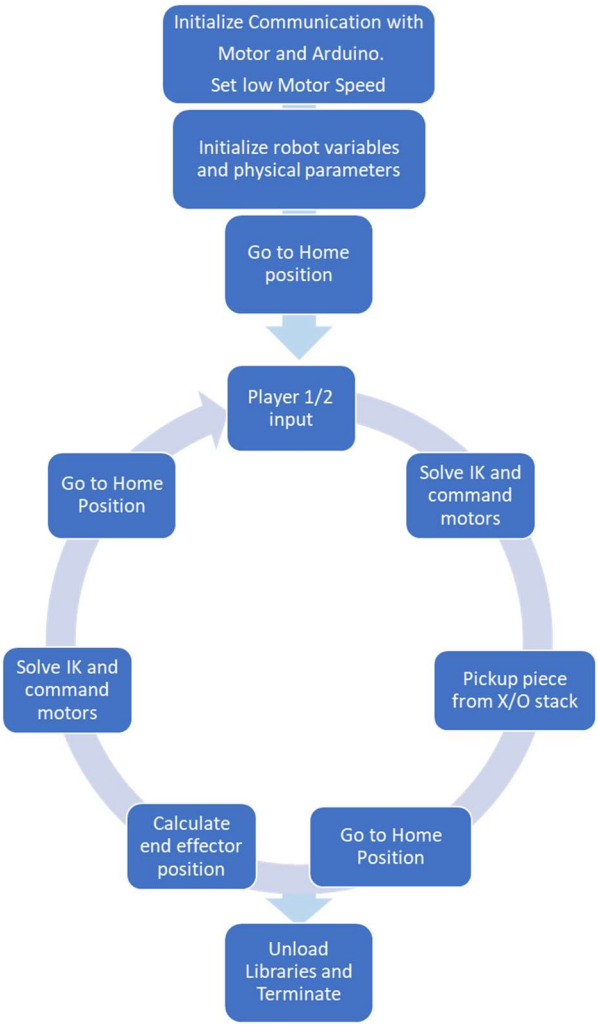

Figure 9: Process flow, which was implemented in code

V. Conclusion

In the end, we were able to create a robot manipulator that can successfully play tic tac toe with a user by traversing a game board using prismatic and revolute joints. The end effector was successfully able to reach individual pieces on the game board and get close enough to pick them up and place them.

Areas for improvement

As we look towards the future of our system, we envision a system with integrated computer vision to make it even more user friendly. The user just has to place his/her piece on the board, and using a camera, the robot could identify the played move and decide what to play next without any external input from the player. Additionally, we would like to make the overall game more aesthetically pleasing. This can be done by changing the link design to hide and protect wires, developing an enclosure for the belt drive and electronics, printing our own game board with slots for pieces, and creating game piece enclosures with the X or O symbol for the nuts we are currently using. We are hopeful for the future of RoboXO and are confident in its future development.

VI. References

1. https://www.youtube.com/watch?v=w5Oftm3l2uQ
2. https://www.youtube.com/watch?v=Mux63vZbmj4
3. https://www.youtube.com/watch?v=q_gAewi_dyY
4. https://www.youtube.com/watch?v=RcnVYqhGYiQ
5. https://www.youtube.com/watch?v=8NQ1h0gGgX8
6. https://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.65.5698&rep=rep1&type=pdf
7. https://www.youtube.com/watch?v=NL4DKNoJlgk&ab_channel=ServoCity