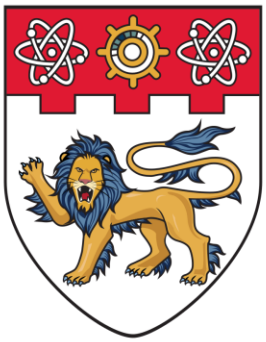


NANYANG TECHNOLOGICAL UNIVERSITY
SINGAPORE

MA4825 Robotics

Group Project Report

Date: 17-11-2023

Tic-Tac-Toe Playing Robot

AY2023/24 Semester 1

Team members:

1. Goh Wei Liang Terence U2021720C
2. Loh Xin Zhi U2021438D
3. Loong Chor Teng U2022241H
4. Loke Kum Yew U2021683L
5. Thio Zheng Yang U2021597J
6. Raymondal Sankalpa U2023694F

Contents

List of figures

List of tables

1. Introduction

1.1. Background and Motivation

Robotic arms are used in the industry because they can work 24 hours in a day, minimizes usage of floor and can do dangerous work without putting any risk on humans [1]. With the inclusion of computer vision, robotic arms can do additional tasks like picking and placing objects as shown in [2]. These systems can be applied to sort for products according do different features, for products with defects or other such applications. There is an opportunity of making difficult tasks more convenient by making robotic arms more intelligent.

This project aims to demonstrate a system capable of making complex decisions by making a robot arm that can play the game “tic-tac-toe” or “noughts and crosses” with a user. The robot arm is able to pick an object, move within a 3-dimensional coordinate system and is able to perform trajectory planning in conjunction with kinematics analysis.

The robot arm is actuated using Dynamixel motors and utilised the provided Dynamiel frames along with 3D printed parts to minimise weight while maintaining strength. The claw of the robot was also 3D printed.

An ESP32 camera sensor provides vision to the system, allowing detection of the current stage of the game. The robot can then be directed to place a block depicting a nought or a cross at the best location to win the game.

Tic-tac-toe is a two-player game of placing noughts or crosses in a three-by-three grid each turn. The player to first place three marks in a straight line, either diagonally, horizontally, or vertically wins the game.

1.2. Objectives and Scope

The objective of this project is to create 5 degree of freedom (DOF) robotic arm that is capable of playing tic-tac-toe with a human player. To achieve this, both the physical configuration of the robot arm, as well as the detection and control algorithm must be designed with this in mind.

On the physical design of the robot arm, the team decided to narrow down the specification to a gripper end effector to pick and place tic-tac-toe pieces. The range of motion required of the robot arm end effector was also a square playing field of roughly 20cm by 20cm. Lastly, it was decided that the end effector should be positioned vertically during pick or place operations to ensure the highest chance of success in accurately picking or placing our tic-tac-toe pieces, shown in Figure 1.

The noughts and crosses of the game are represented by blue and red cylinders.

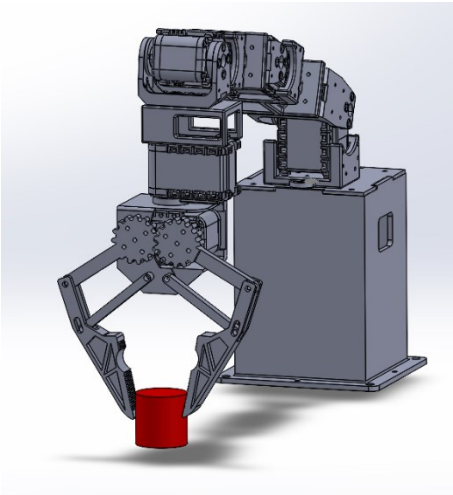


Figure 1: Figure of robot in motion of picking play piece

2. Robot Description

Figure 2 shows the setup of the entire system. The global origin is located at the base of the robot, where the Z axis coincide with joint 1 axis of rotation.

The process of making this robot split into 2 segments: hardware design and robot control.

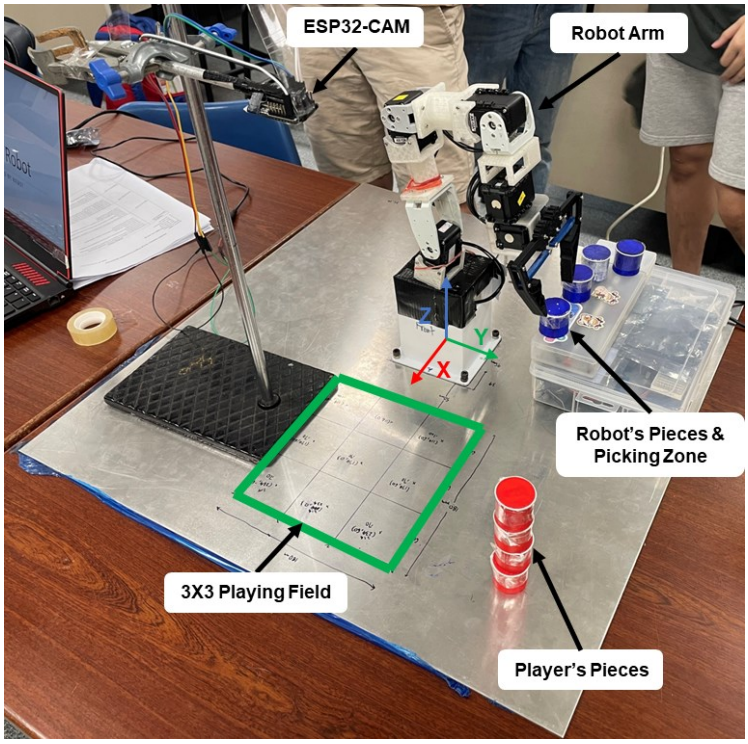


Figure 2: System Setup

2.1 Overall Flow

Step 1: Initialization

First, we initialise the entire system which includes setting up the necessary connections and configurations of the robotic arm and the game logic. At the same time, the Tic Tac Toe game board and the initial state of the Tic Tac Toe game is set up.

Step 2: Visual Processing and Game State Detection

Using the ESP32 CAM and OpenCV, the image of the gameboard is captured and processed to detect the current state of the game (the positions of blue and red pieces on the board). This visual information is translated into positional data understandable by the robotic system

Step 3: TTT Robot Decision

The Minimax Algorithm is used to decide the next best move for the robot. This algorithm considers the current state of the board and computes the optimal move to either win the game or block the opponent.

Step 4: Calculating Robotic Arm Movements

Inverse Kinematics calculates the required joint angles for the robotic arm to reach the specific position on the game board while motion planner plans the movement trajectory, ensuring the smooth motion of the arm from its current position to the target position.

Step 5: Actuating Robotic Arm

The calculated joint angles are converted into data and used to control the motors of the robotic arm. The planned movement from motion planner is executed and command the robotic arm, placing the red pieces on the game board.

Step 6: Updating the Game State

After the move is made, the game state is updated to reflect the new position placed by the robotic arm

Step 7: Repeat or End game

The system checks if the game has been won, lost, or drawn. If the game is still ongoing, it returns to Step 2 for the next move. If the game is concluded, the system either resets for a new game or shuts down.

2.1 Hardware design

In addition to the components from the original set of provided to the teams, we also used a camera, and several 3D printed parts to enhance the set up.

Table 1 List of items for hardware

No.	Description of Item	Quantity	Remarks
1	Robotis Dynamixel Robot Servo, AX-12A	4	
2	Robotis Dynamixel Robot Servo, AX-18A	2	
3	Robotis U2D2 converter (Including USB cable and 3Pin-Dynamixel cable)	1	
4	Robotis SMPS 12V/5A Power Supply and Power cord	1	
5	Robotis SMPS2Dynamixel w/ DC input and 3P/4P connector	1	
6	Dynamixel cables	8	
7	Frame (Bioloid FP04-F1)	1	
8	Frame (Bioloid FP04-F2)	6	
9	Frame (Bioloid FP04-F3)	6	
10	Frame (Bioloid FP04-F4)	2	
11	Frame (Bioloid FP04-F5)	1	
12	Screwdriver	2	
13	M2 and M3 screws and nuts	1 lot	
14	Bioloid BPF-WA/BU Washer/Bushing and Screws	6 set	
15	Cable ties	1 lot	
16	Toolbox	1	

17	ESP32-CAM	1	
18	Clamp	1	To position camera
19	3D printed linkages and fittings	As required	
20	3D printed Tic-tac-toe play pieces	8	4 red, 4 blue
21	Laptop	1	

The team planned to use all 6 of the Dynamixel motors in the construction of the arm, with 5 motors to engage in motion, and 1 motor for the claw end effector, to maximise the degrees of freedom on the arm.

Figure 3 depicts the full robot arm assembly, with critical dimensions labelled. The overall dimensions of the robot arm were 135mm by 582mm by 95mm when the arm is fully vertical with an open gripper.

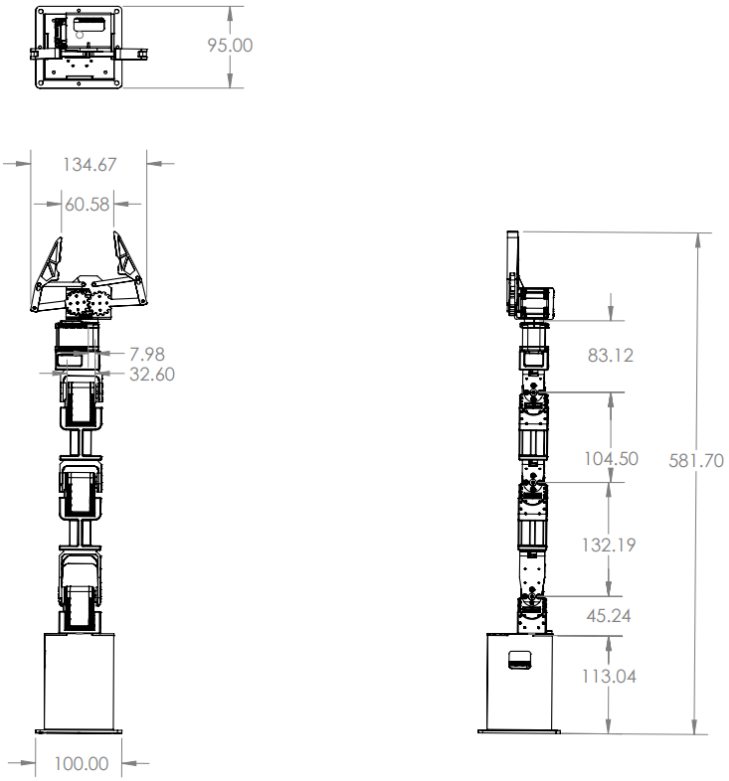


Figure 3: Assembly drawing of the robot arm with critical dimensions

2.1.1. Linkages and Joint Configuration

The original configuration for the arm was decided to consist of 6 linkages (Links 0 to 5), and 5 joints (Joints 1-5). Figure 4 depicts the layout of the linkages. The configuration gave the robot arm 5 degrees of freedom, and the 3-elbow joint configuration allowed us to ensure that the end effector is vertical during pick and place operations. The linkages utilised a combination of the standard Dynamixel brackets provided, and 3D printed parts to interface the brackets with the Dynamixel motors.

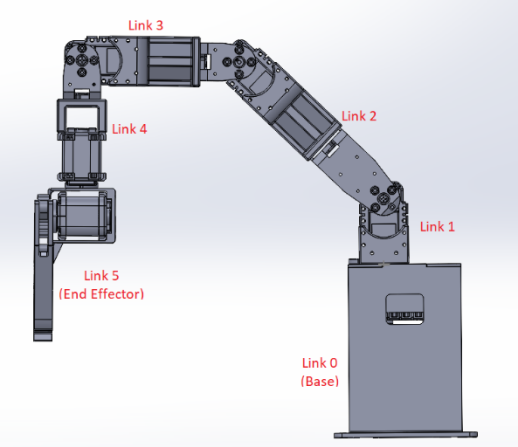


Figure 4: Robot arm with the various links labelled

Link 0 (base_mount), show in Figure 5, is the base link, which is a 3D-printed part of PLA material with 60 percent infill. This base linkage was designed with sufficient height so that the end effector of the robot arm may have sufficient reach to achieve the pick and place tasks. This 3D-printed part is mounted onto an aluminium sheet via screws to ensure that it is firmly secured to the playfield and has a fixed origin relative to the tic-tac-toe playfield. As this is a relatively large part to 3D-print, the part was broken down into two pieces that may be printed separately. This helped to avoid printing with supports, which would lengthen the printing time significantly.

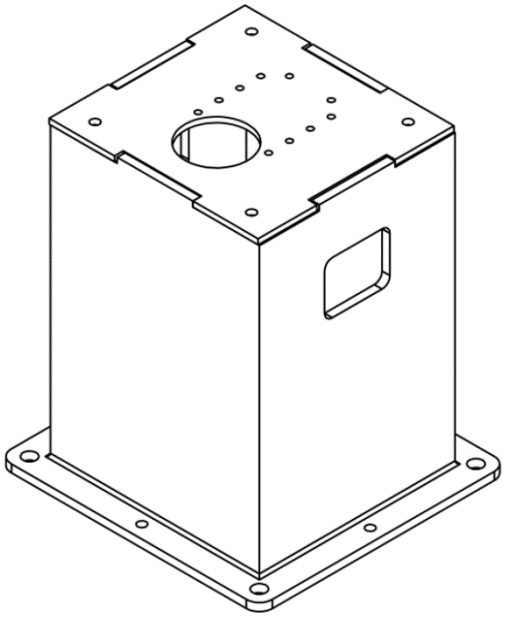


Figure 5: Drawing of the ‘base_mount’ part

Link 1 (linkToBase) consists of a 3D-printed part as seen in Figure 6, which allows one Dynamixel motor to be directly mounted onto the flange of another Dynamixel motor. This was printed with PLA of 60 percent infill to ensure the strength of the part. Fillets were also applied to the edges to reduce stress concentrations. A loop was also added to attach rubber bands that would stretch to the next link. These rubber bands may be added to offset the torque due to the weight of the arm, hence reducing the overall load on the motor in joint 2.

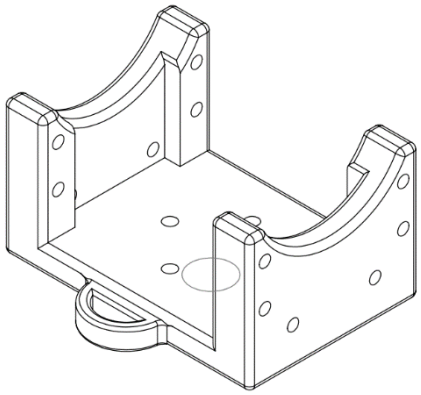


Figure 6: Drawing of the 'linkToBase' part

Links 2 and 3 (link_adapter1) utilised the same 3D-printed part, as seen in Figures 7 and 8, to interface the motors with the brackets provided. The top section of the link_adapter1 part allows a Dynamixel motor to slide into place and be bolted down, while the bottom flat portion allows it to interface with the brackets. The large chamfer of the centre stem helps to ensure structural strength of the part against the bending moment it will experience during the operation of the robot arm. The filleted edges also serve to reduce stress concentrations that may occur, and the part was printed with PLA material of 60 percent infill to further ensure the strength of the part.

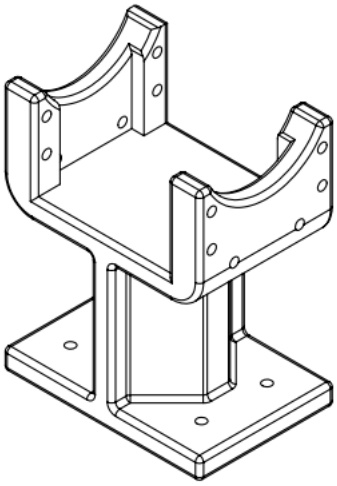


Figure 7: Drawing of the 'link_adapter1' part

Figure 8: Photograph of the actual 'link_adapter1' part

Link 4 (link_adpater2) also consists of a 3D-printed part that interfaces with a bracket. The 3D-printed part, as seen in Figure 9, allows a motor to be mounted to the bracket such that the axis of rotation is perpendicular to the previous motor's axis of rotation. Once again, this was printed with PLA material of 60 percent infill, and fillets were applied to ensure the strength of the part and to reduce stress concentrations.

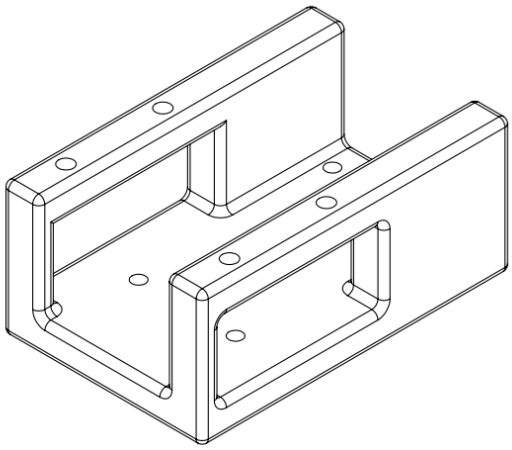


Figure 9: Drawing of the 'link_adapter2' part

The robot was designed such that the joints will lead the end effector to face vertically downwards during pick and place operations. Figure 10 highlights the respective joint motion at each joint.

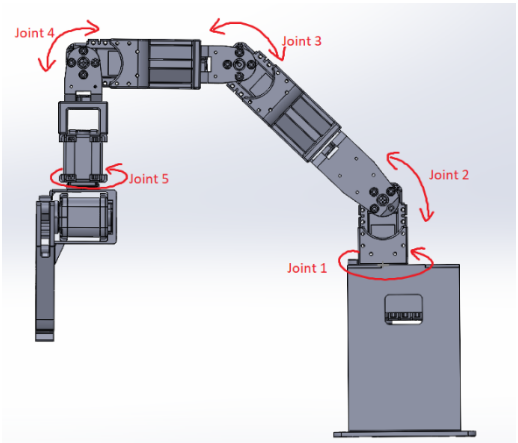


Figure 10: Robot arm with the various joints and their range of motion labelled

Joints 0 and 1 were employed with the AX-18A motor, due to its higher stall torque compared to the AX-12A motors (Appendix A and B). Due to their proximity to the base link, these joints, particularly Joint 1, will undergo the highest torque during the robot arm operation, and hence would benefit from using the motor with higher stall torque.

2.1.2. Claw End Effector

For the Claw end effector, shown in Figure 11, the AX-12 Dynamixel motor is used to control the rotation of the 3D printed spur gear linkage. The design of the end effector utilises the concept of 4-bar linkage to convert rotation motion to a linear motion where the gripper.

The end effector consisted of 7 3D printed pieces; 4 parallel links with 2 having a spur gear attached at the end, 2 pieces for the claws, 1 piece as housing for the motor

Optimisation of the 3D printed parts to ensure light weight and strength was done. Lastly, the parts were printed with ABS material with 15 percent infill.

The claw end effector can grip up till 6cm diameter object, and for our project, we decided to dimension the Tic-tac-toe play piece with a diameter of 3cm.

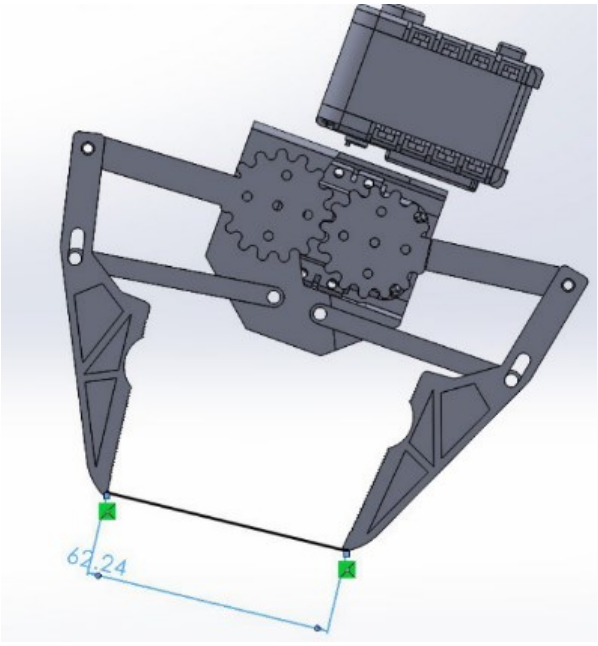


Figure 11: Drawing of the Claw end effector with maximum span measurement

2.1.3. Design Calculations

To understand the feasibility of our robot arm configuration, we conducted design calculations based on the planned configuration of our robot arm. With a design such as ours, the weight of the robot arm itself is a concern. Hence, we conducted calculation on the expected torque on joint 2 due to the weight of the arm, as it was the joint that would experience the highest torque. This was then compared against the stall torque of the AX-18A.

The set-up was designed to consider the worst-case scenario, as shown in Figure 12 when the robot arm is fully extended along the Y-axis.

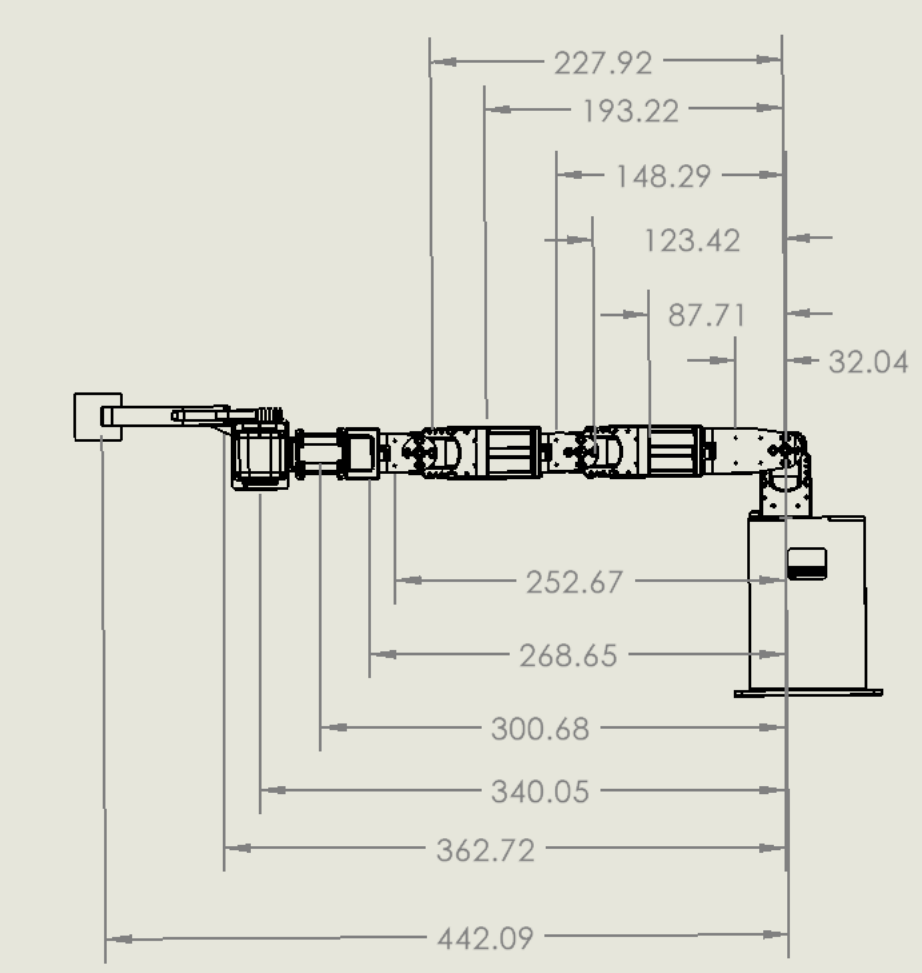


Figure 12: Layout of the robot arm under worst-case scenario

Based on the material properties of the printed parts, and datasheets of the Dynamixel components, the following masses were obtained.

Mass of link_adapter1 = 32g

Mass of link_adapter2 = 14g

Mass of FP04-F2 Frame = 30g

Mass of FP04-F4 Frame = 40g

Mass of AX-18A motor = 55.9g

Mass of AX-12A motor = 54.6g

Mass of Claw Mechanism = 43.4g

Mass of Tic-Tac-Toe pieces = 7.9g

Hence the calculation on maximum torque due to the weight of the arm is as follows.

$$Torque\ on\ Joint\ 2\ from\ weight\ of\ arm$$

$$= Bending\ Moment_{F4\ Frame} + Bending\ Moment_{linkAdapter1_1}$$

$$+ Bending\ Moment_{AX12A_{Joint3}} + Bending\ Moment_{F2\ Frame_1}$$

$$+ Bending\ Moment_{linkAdapter1_2} + Bending\ Moment_{AX12A_{Joint4}}$$

$$+ Bending\ Moment_{F2\ Frame_2} + Bending\ Moment_{linkAdapter2}$$

$$+ Bending\ Moment_{AX12A_{Joint5}} + Bending\ Moment_{AX12A_{Claw}}$$

$$+ Bending\ Moment_{Claw\ Mechanism} 9.81(0.03204 * 0.04 + 0.08771$$

$$* 0.032 + 0.12342 * 0.0546 + 0.14829 * 0.030 + 0.19322 * 0.032$$

$$+ 0.22792 * 0.0546 + 0.25267 * 0.030 + 0.26865 * 0.014 + 0.30068$$

$$* 0.0546 + 0.34005 * 0.0546 + 0.36272$$

$$* 0.0434)$$

$$= 0.941469Nm$$

With the stall torque of the AX-18A being 1.5Nm, this means that the motor at joint 2 will be able to handle the weight of the arm even at it maximum extended position. We can then proceed to calculate the maximum advisable payload that our gripper should pick up, based on the AX-18A stall torque.

$$Max\ Torque\ on\ Joint\ 2$$

$$= Torque\ from\ from\ weight\ of\ arm + Torque\ from\ payload$$

$$= 1.5Nm$$

$$Torque\ from\ payload$$

$$= Max\ Torque\ on\ Joint\ 2 - Torque\ from\ from\ weight\ of\ arm$$

$$= 1.5Nm - 0.941469Nm = 0.558531Nm = m_{payload}gx$$

$$= m_{payload}(9.81 * 0.44209)$$

$$m_{payload} = \frac{0.558531}{9.81*0.44209} = 0.128785kg = 128.785g$$

With our planned tic-tac-toe pieces, in Figure 13, being 7.9g, much lighter than the maximum payload of 128.8g, the robot arm should be able to pick up the pieces without issues.

Figure 13: Tic-tac-toe play piece (Diameter: 3cm, Height: 3cm)

2.1.4. Electronics Components and Connections

A laptop running the Ubuntu 22.04.3 LTS operating system is used as the master controller of the system. An ESP-32 CAM is used as the primary sensor input, which will host the frame data on a web server through Wi-Fi. The robot will power by a 12V DC supply and be connected to the laptop via USB serial communication.

2.2 Robot Control Flow

Robot Operating System 2 (ROS2) is used as the main communication architecture. Figure 14 shows the overall program pipeline, taking in camera input and driving the motors to control the robot.

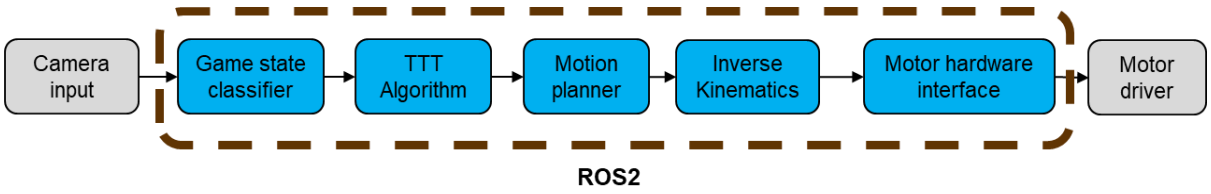


Figure 14: Program pipeline

ROS2 was utilised for its following features and benefits:

a) Modularity (nodes and topics) – easy to test and debug each node program individually without affecting other nodes

b) Multiple language support (Python and C++) – provide access to multiple libraries and example codes of written in each language

c) Open-sourced – full documentation is available online, with access to existing ROS2 packages like Moveit2 and Gazebo Simulation

2.2.1 Camera input

To capture playing field scene, an ESP32-CAM is used as the primary input sensor of the system. The camera hosts the live captured frame on a private server and a ROS2 node is written to request the frame and published to a topic for further processing.

2.2.2 Game State Classifier

From the image data, OpenCV is used to process the image. Hough Circle Transform is used to detect the game pieces as circles from the image, which returns their coordinates. By thresholding the HVS values of the circle's center pixel, the colour of the piece is determined.

Ternary classification is done on each box in the 3 by 3 grid, where it is either empty, contains a red piece (player) or blue piece (robot). Therefore, the current game state can be represented as a 3x3 matrix with integer value 0 which represents empty box, 1 for player's piece and 2 for robot's piece. The process is shown in Figure 15, from the Camera to Game state.

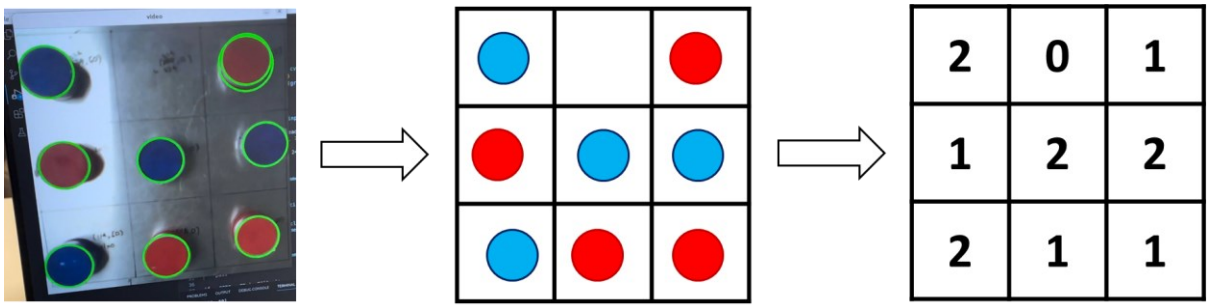


Figure 15: From Camera to Game State

2.2.3 TTT Algorithm

The main concept behind the TTT algorithm is the Minimax algorithm. Minimax is a type of adversarial search algorithm for generating and solving game trees. It is mainly used to solve zero-sum games where one side's gain equals the opponent's loss. As such, adding all the gains and subtracting all the losses ends up in zero. The Tic-Tac-Toe's game tree, as depicted in Figure 16, is recursively generated by exploring all the next possible move for each board state.

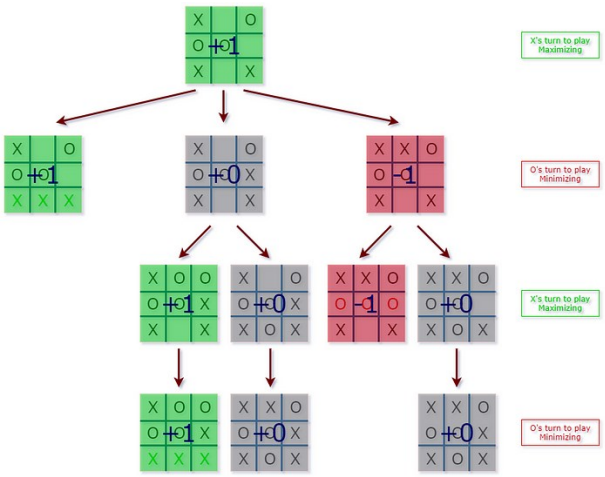


Figure 16: Tic-Tac-Toe game tree

Upon reaching a terminal state, a score can be assigned to each move, 1 for winning, -1 for losing and 0 for draw. The player will then be assigned as the maximizer while the opponent

will be the minimizer. Depending on whose turn it is, the move prioritised will be picked based on score, with 1 for optimal for the maximizer and -1 for the minimizer. As the algorithm runs through all possible moves, it adds its return value to a list of scores and undos said move. Only after considering all possible scenarios, do the Minimax algorithm return the maximum or minimum of the scores list, depending on whose turn it is.

In this project, the minimax algorithm is used to determine the robot's next best move. It will output robot's decision represented as the indexes of the matrix grid. As each matrix position is mapped to a (x, y, z) coordinate as reference to the physical coordinate (Fig. 17), the indexes can be translated and published as the target pose for the gripper to place the game piece. This grid in Fig. 17 is a part of the intended workspace.

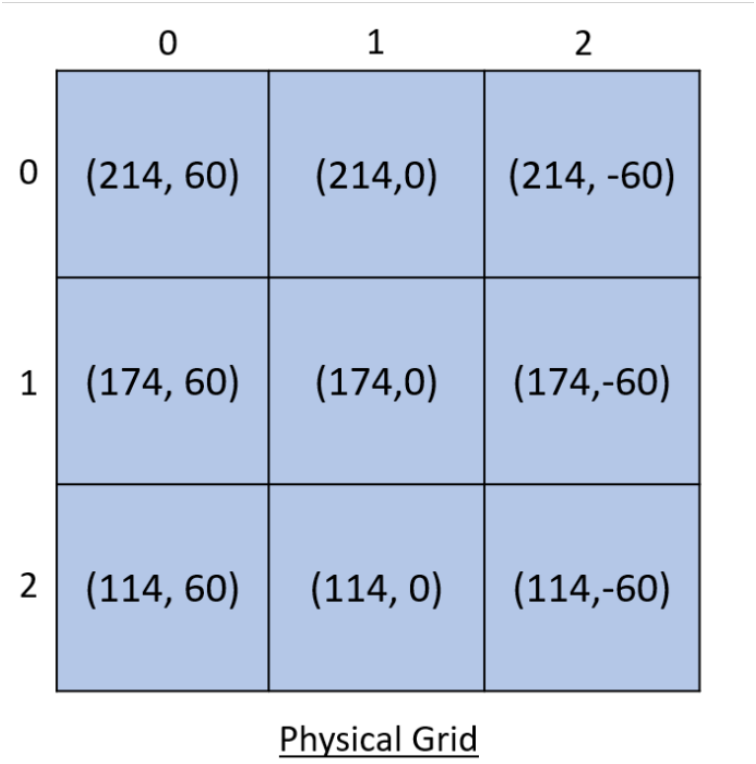


Figure 17: Mapping of matrix position to physical coordinate

2.2.4 Motion Planner

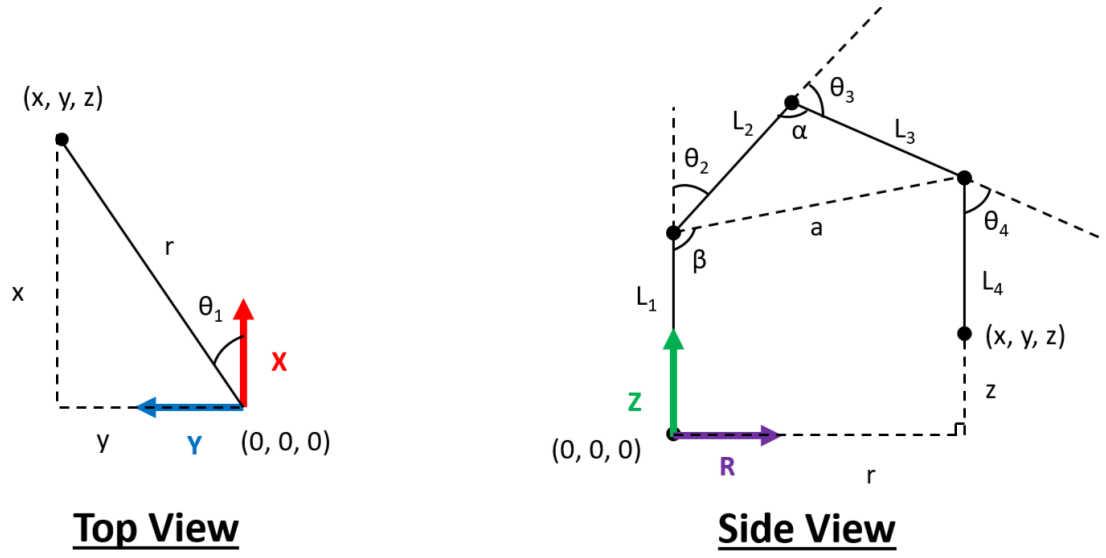
After receiving the initial and target pose, the Motion Planner module will plan the path of the end effector as a series of waypoints. This is achieved through the following steps, assuming initial pose = (x_i, y_i, z_i) and target pose = (x_t, y_t, z_t) and home pose = (x_h, y_h, z_h) :

1) Go to $(x_i, y_i, z_i + 10)$, gripper opened
2) Go to (x_i, y_i, z_i), gripper opened
3) Go to (x_i, y_i, z_i), gripper closed
4) Go to $(x_i, y_i, z_i + 10)$, gripper closed
5) Go to $(x_t, y_t, z_t + 10)$, gripper closed
6) Go to (x_t, y_t, z_t), gripper closed
7) Go to (x_t, y_t, z_t), gripper opened
8) Go to (x_h, y_h, z_h), gripper opened

The node will publish the cartesian coordinates in each step and wait until the end effector reaches that point before publishing the next step.

2.2.5 Inverse Kinematics

With the cartesian coordinates from motion planner in the task space, the Inverse Kinematics module converts this pose in the task space to joint angle space as the angular position of the 4 revolute torque joints (motors) as $(\theta_1, \theta_2, \theta_3, \theta_4)$. θ_5 will be fixed at 0° (refer to Chapter 3.3) We use the trigonometry method to calculate the joint angles, illustrated in Figure 17.



$$r = \sqrt{x^2 + y^2}$$

$$a = \sqrt{r^2 + (L_4 + z - L_1)^2}$$

$$\beta = \cos^{-1}(r/a) + 90 \quad \text{for} \quad \beta > 90\ (L_1 < L_4 + z)$$

or

$$\beta = \sin^{-1}(r/a) \quad \text{for} \quad \beta \leq 90 (L_1 \geq L_4 + z)$$

$$\alpha = 180° - \theta_3$$

Figure 18: Inverse Kinematics Calculation with Trigonometry

With the requirement that Link 4, which is connected to the gripper, be vertically down, we have the constraint:

$$\theta_2 + \theta_3 + \theta_4 = 180°$$

This gives us the following simultaneous equations for the joint angles:

$$\theta_1 = \tan^{-1}\left(\frac{y}{x}\right)$$

$$\theta_2 = 180° - \beta - \sin^{-1}\left(\frac{L_3 \sin\alpha}{a}\right)$$

$$\theta_3 = \cos^{-1}\left(\frac{a^2 - {L_2}^2 - {L_3}^2}{2L_2L_3}\right)$$

$$\theta_4 = 180° - \theta_3 - \theta_2$$

2.2.6 Motor Hardware Interface and Motor Driver

The Dynamixel motors provides a software development kit (DynamixelSDK), which include a position control function, to control the motor by writing integer values to the various registers through serial communication. Each motor is assigned a unique ID, and data can be written to the position control register by referencing its address. After receiving the joint angles from Inverse Kinematic module and adjusting for offsets and scaling, the integer data values is written to relevant registers for all motors. The motors then travel to the target position with a preset velocity limit.

3. Robot Performance

The robot was able to analyse the playing field and execute its decisions of picking and placing its piece to the position decided by the algorithm. After the player makes a move, the vision system of the robot sees the game state and then calculates where to place the robot's piece. Then, the robot picks up its piece and places it on the correct place. We observe that generally, the games played against the robot always end up in a draw, or even a win if the human player makes a mistake. The designed system has various benefits but are also met with some limitations.

3.1 Benefits

1. The system exhibits feature of a combination logic, where the output of the algorithm is only dependent on the current state of the playing field. It does not hold memory of the previous state or moves like a sequential logic. This allows the game to be reset at any time and the player can arrange the board to test out the best move by the algorithm at any time.
2. The design of the robot allows for modification to the playing field, as long as the camera is facing normal to the surface of the playing field. This is due to joint 5 of the robot arm which can rotate accordingly about the y-axis.

3.2 Limitations

1. During the operation and control of the robot arm, it was discovered that the mounting of the brackets around the width of the motor effectively reduced the range of motion to approximately between -90 degrees and 90 degrees. Although the robot is still able perform pick-and-place tasks with its current range of motions, it limits the available workspace due to the joint constraints.

2. For certain motion paths, the gripper may collide with the existing pieces on the playing field. This is due to the lack of collision detection within the motion planner module. The motion planner is only designed for simple way-points generation.
3. The combined weight of the motors and 3D printed parts caused a bending moment in the 3D printed base connected to the first joint. This caused imbalance due to the elastic deformation of the 3D printed base as the rigidity of the material was insufficient in countering the bending moment. Rubber bands had to be added as a counter force to increase stability of the overall structure.

3.3 Other Issues Encountered

1. When implementing the kinematics of the robot for the Tic-Tac-Toe game, it was deduced that the motion of joint 5, which controls the roll orientation of the gripper, is redundant. The robot arm was still able to grip the pieces due to the game pieces' cylindrical shape. Therefore, to simplify the kinematics calculations, joint 5 was fixed at 0°, effectively reducing the robot DOF from 5 to 4.
2. The circle detection algorithm had some difficulties in detecting the cylindrical game pieces due to poor lighting conditions. White wire strips were added encircling around the top of the cylinders to increase the contrast between the cylinder and the background, therefore increasing detection rate (refer figure 13).

4. Conclusion and Future Works

The robot arm designed in this project can fulfil its intended task, displaying elements of computer system, intelligence, motion planning and kinematic analysis. It is able to proficiently pick and place objects in a 3-dimensional coordinate system and does it with the appropriate kinematic analysis. In addition, the system successfully implements a vision component as its primary sensor for perception of the environment.

4.1 Improvements

A better hardware design of the brackets around the motor flange could be developed to increase the range of motion for each of the joints. A bracket that mounts only to the flange side of the motor would ensure a much larger range of motion of the joint, without risk of the link impacting the motors or impinging the wires.

Instead of creating the motion planner and inverse kinematics modules from scratch, we can utilise existing ROS2 packages like Moveit2, which tools for motion planning, collision checking and inverse kinematics. This will allow for more complex control, collision avoidance and high DOF robot integration. We can also utilise Gazebo Simulation to simulate and visualise the motion before implementing the physical system.

4.2 Future Works

A similar system can be used in other applications like automation of manufacturing processes, packaging or other pick and place applications, with the integration of computer vision and intelligence for smarter systems.

5. References

[1] H. A. Almurib, H. F. Al-Qrimli, and N. Kumar, “A review of application industrial robotic design,” in *2011 Ninth International Conference on ICT and Knowledge Engineering*, IEEE, 2012, pp. 105–112. Accessed: Nov. 16, 2023. [Online]. Available: https://ieeexplore.ieee.org/abstract/document/6152387/

[2] M. Intisar, M. M. Khan, M. R. Islam, and M. Masud, “Computer Vision Based Robotic Arm Controlled Using Interactive GUI.,” *Intelligent Automation & Soft Computing*, vol. 27, no. 2, 2021, Accessed: Nov. 16, 2023. [Online]. Available: https://cdn.techscience.cn/ueditor/files/TSP_IASC-27-2/TSP_IASC_15482/TSP_IASC_15482.pdf

6. Appendix

Appendix A: Datasheet Specifications for AX-18A Dynamixel Servo Motors

1. Specifications

Item	Specifications
Baud Rate	7,843 [bps] ~ 1 [Mbps]
Weight	AX-18F (54.5 [g]), AX-18A (55.9 [g])
Dimensions (W x H x D)	32 X 50 X 40 [mm] 1.26 X 1.97 X 1.57 [inch]
Resolution	0.29 [°]
Motor	Coreless
Gear Ratio	254 : 1
Stall Torque	1.8 [N.m] (at 12 [V], 2.2 [A])
No Load Speed	97 [rev/min] (at 12 [V])
Running Degree	0 ~ 300 [°] Endless Turn
Operating Temperature	-5 ~ +70 [°C]
Input Voltage	9.0 ~ 12.0 [V] (**Recommended : 11.1 [V]**)
Command Signal	Digital Packet
Physical Connection	TTL Level Multi Drop Bus Half Duplex Asynchronous Serial Communication (8bit, 1stop, No Parity)
ID	254 ID (0~253)
Feedback	Position, Temperature, Load, Input Voltage, etc
Gear Material	Engineering Plastic(1, 2, 3), Precious Metal(4)
Case Material	Engineering Plastic(Front, Middle, Back)

NOTE : Stall torque is the maximum instantaneous and static torque. Stable motions are possible with robots designed for loads with 1/5 or less of the stall torque.

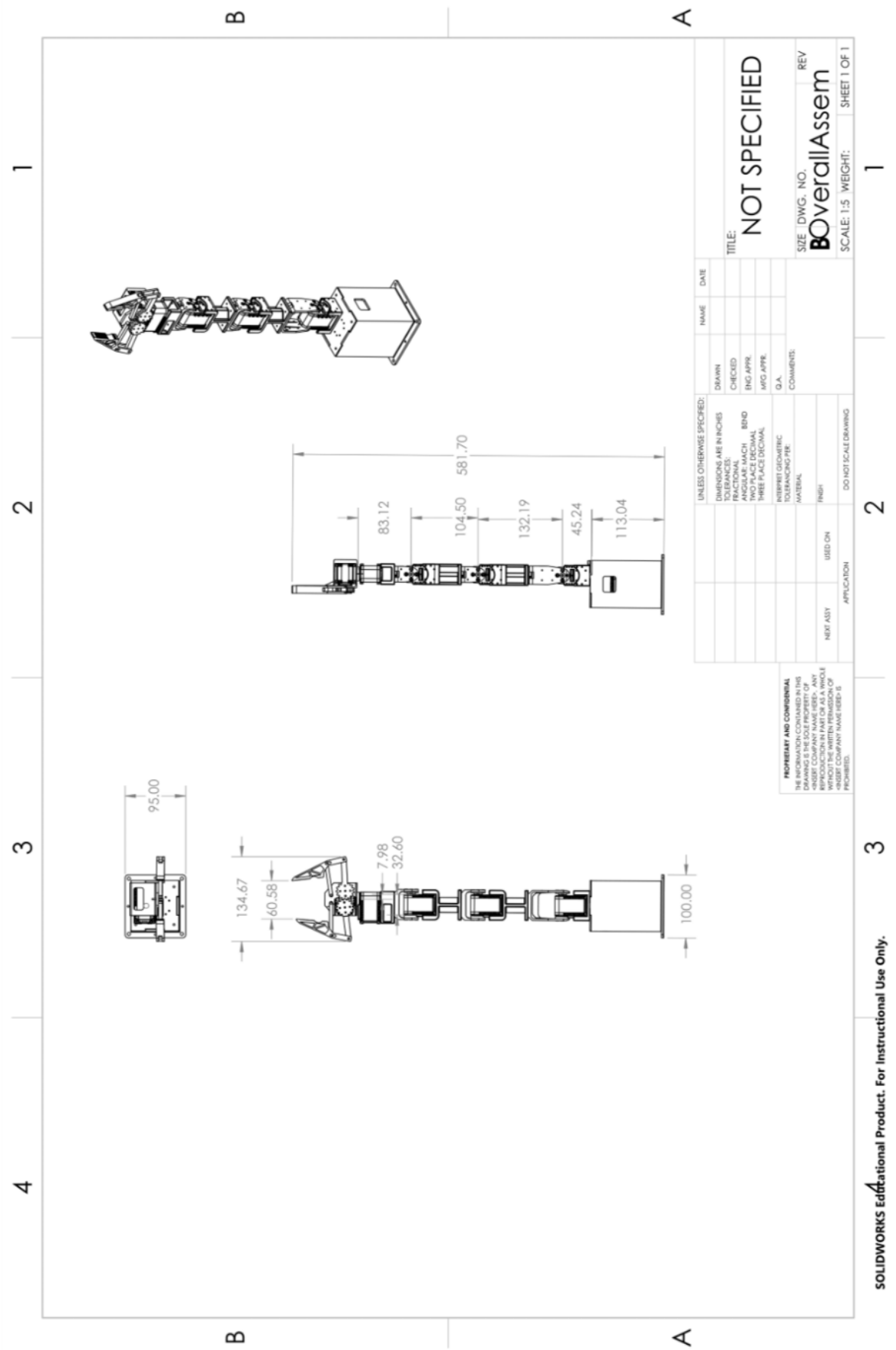
Appendix B: Datasheet Specifications for AX-12A Dynamixel Servo Motors

1. Specifications

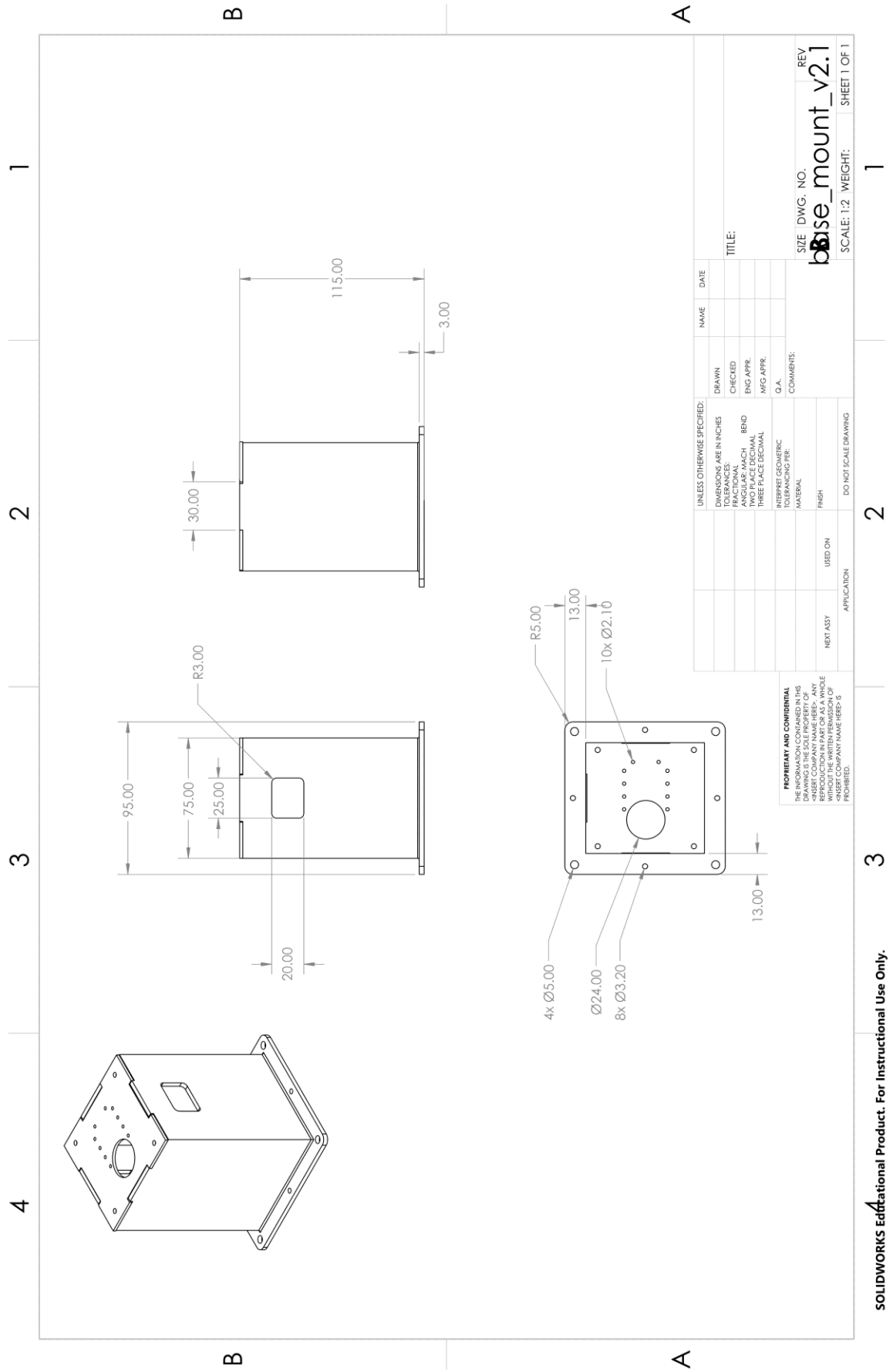
Item	Specifications
Baud Rate	7,843 [bps] ~ 1 [Mbps]
Weight	AX-12 (53.5 [g]), AX-12+ (53.5 [g]), AX-12A (54.6 [g])
Dimensions (W x H x D)	32 X 50 X 40 [mm] 1.26 X 1.97 X 1.57 [inch]
Resolution	0.29 [°]
Running Degree	0 ~ 300 [°] Endless Turn
Motor	Cored
Gear Ratio	254 : 1
Stall Torque	1.5 [N.m] (at 12 [V], 1.5 [A])
No Load Speed	59 [rev/min] (at 12V)
Operating Temperature	-5 ~ +70 [°C]
Input Voltage	9.0 ~ 12.0 [V] (**Recommended : 11.1V**)
Command Signal	Digital Packet
Physical Connection	TTL Level Multi Drop Bus Half Duplex Asynchronous Serial Communication (8bit, 1stop, No Parity)
ID	254 ID (0~253)
Feedback	Position, Temperature, Load, Input Voltage, etc
Gear Material	Engineering Plastic(Full)
Case Material	Engineering Plastic(Front, Middle, Back)

NOTE : Stall torque is the maximum instantaneous and static torque. Stable motions are possible with robots designed for loads with 1/5 or less of the stall torque.

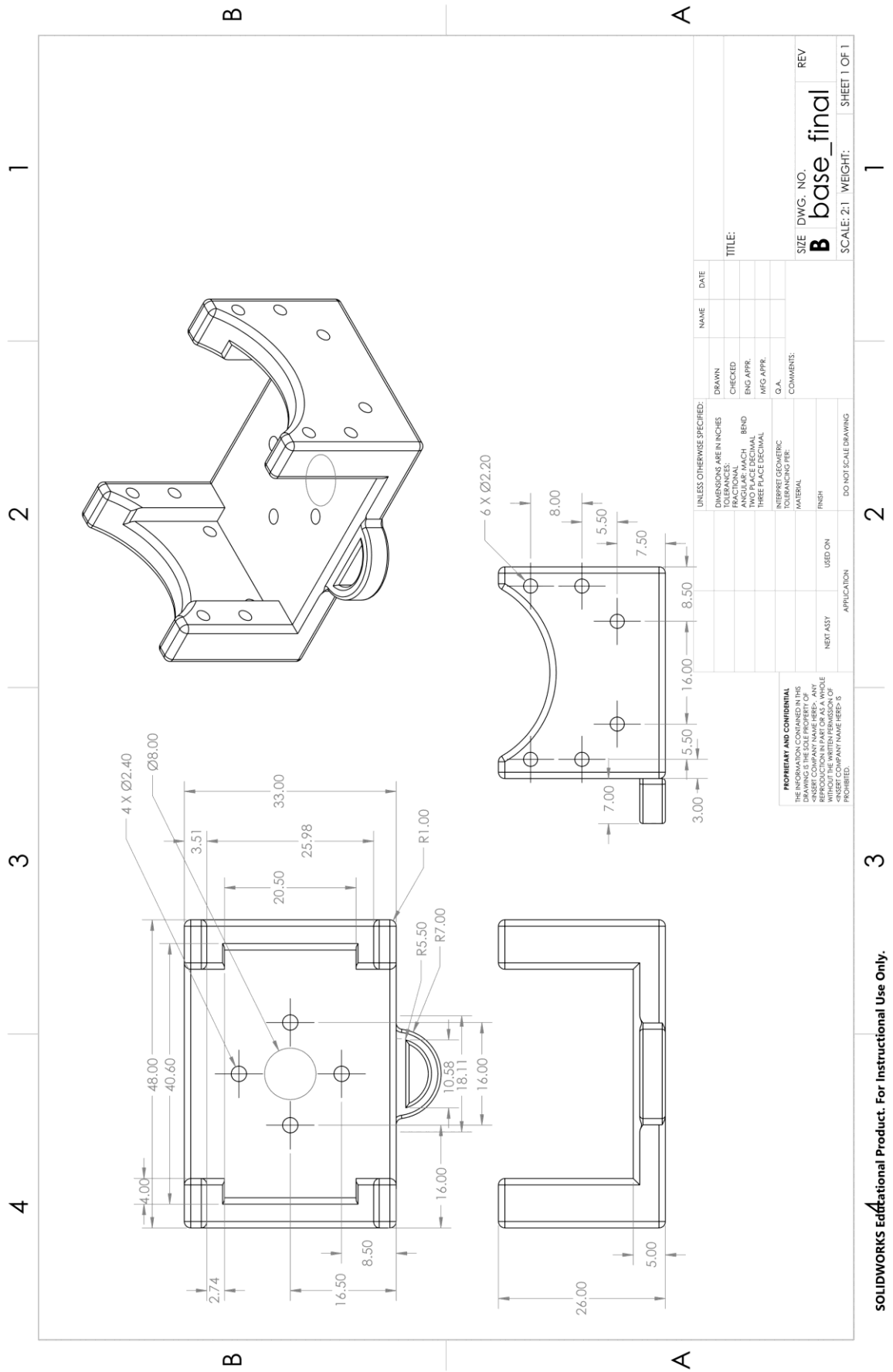
Appendix C: Overall Assembly Drawing



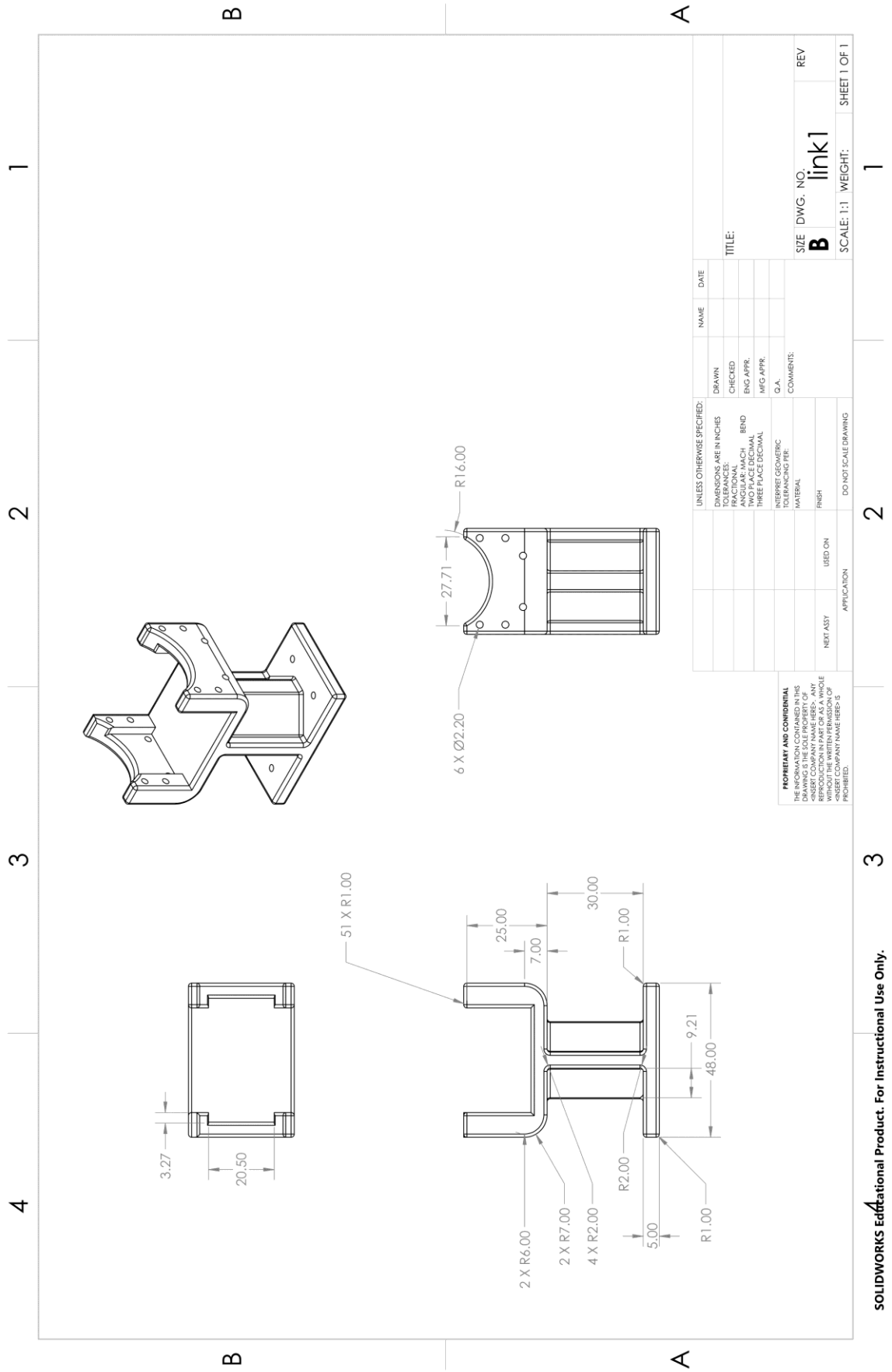
Appendix D: Part Drawing (base_mount)



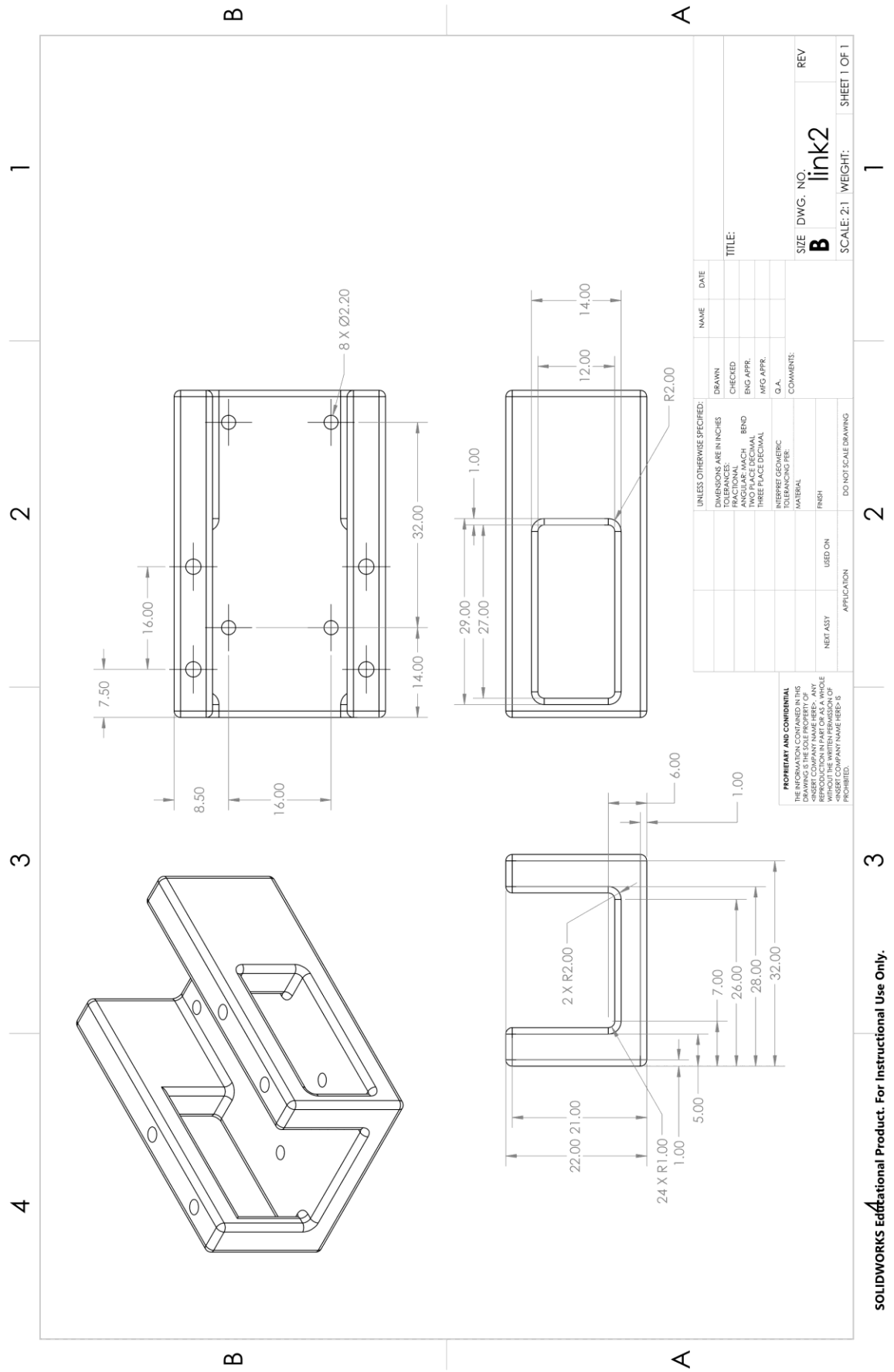
Appendix E: Part Drawing (linkToBase)



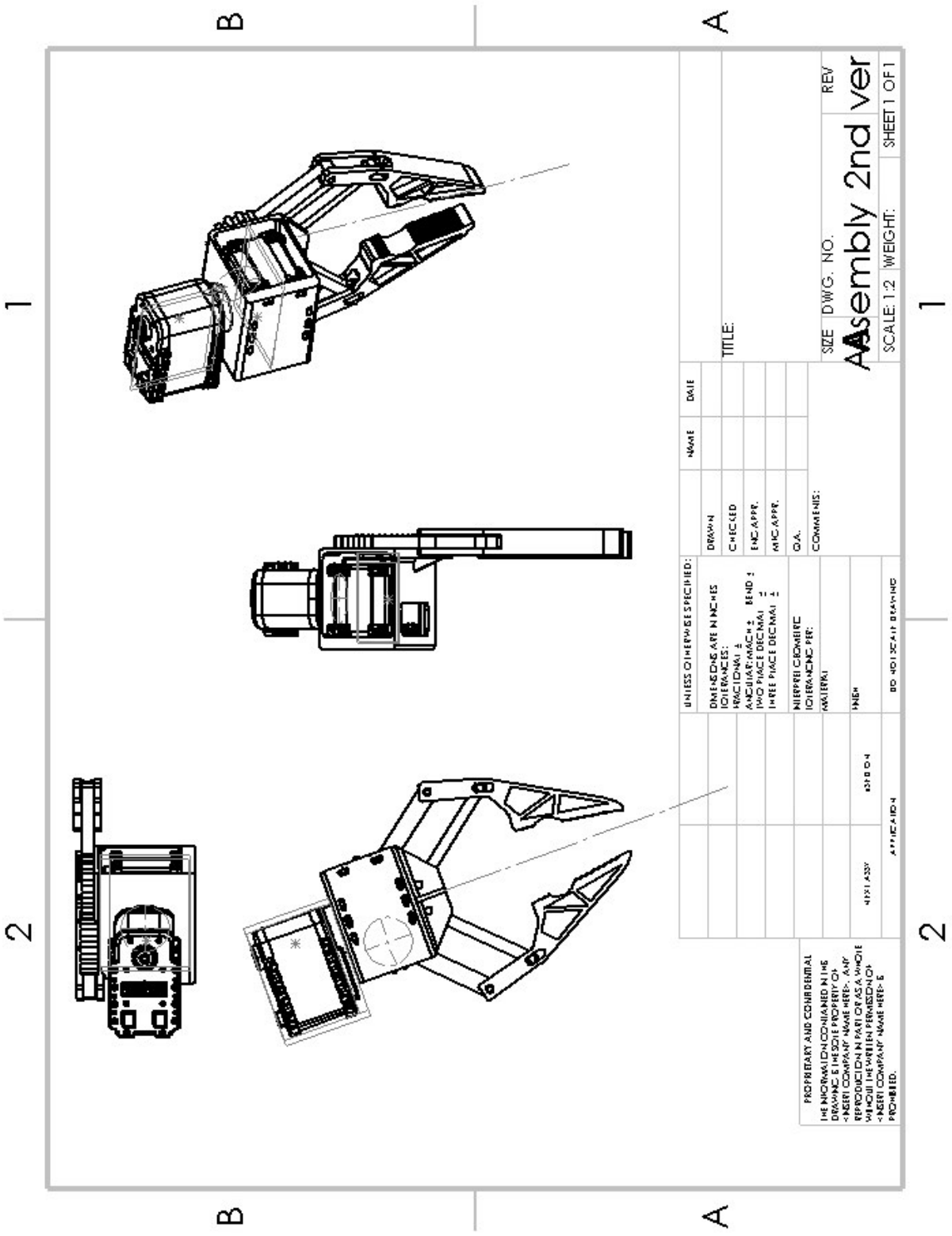
Appendix F: Part Drawing (link_adapter1)



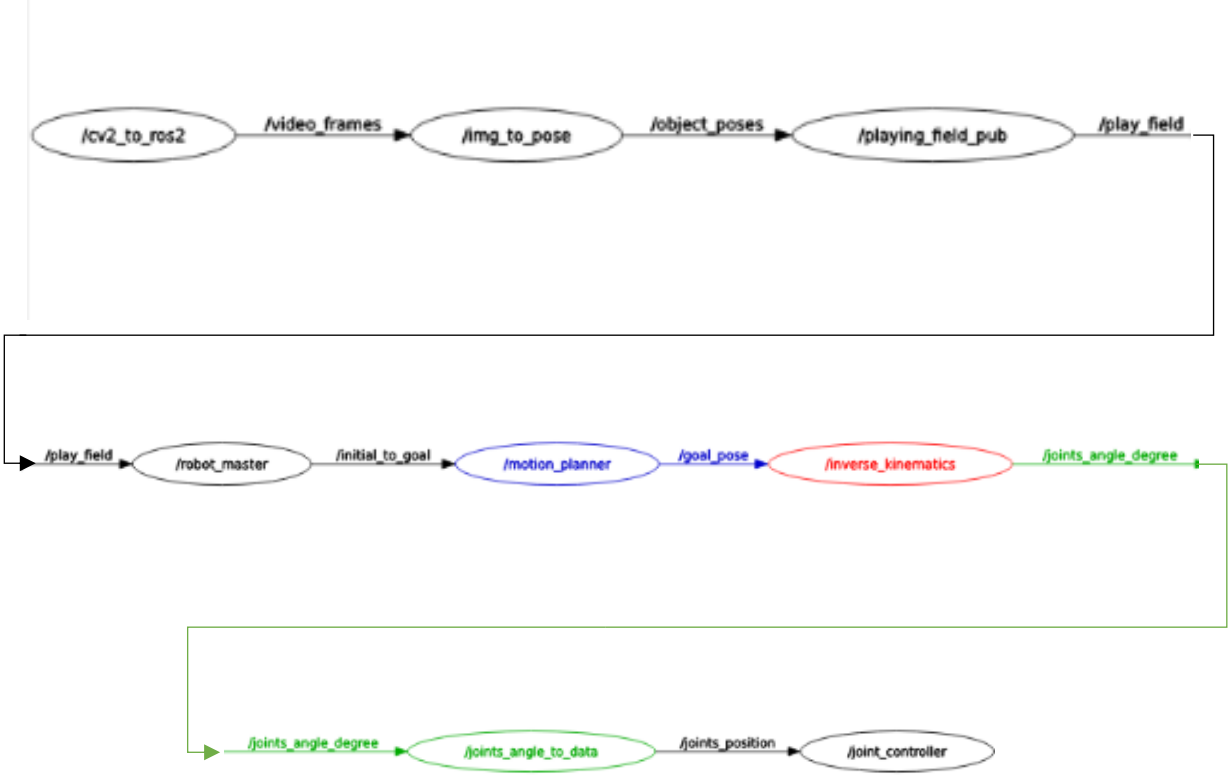
Appendix G: Part Drawing (link_adapter2)



Appendix H: End Effector Gripper Drawing



Appendix I: ROS2 Programme Details



The graph above shows the ROS2 nodes and topics that are active when the program is running. Each node is design to do a specific task so that it can is modularise for easy debugging and modification of program. Refer to the code README.md file for more details:

Nodes Description:

Nodes	Subscription topic	Publish topic	Description
/cv2_to_ros2	-	/video_frame	Request frame from ESP32-CAM webserver and publish image data to topic.
/img_to_pose	/video_frame	/object_poses	Read the image data and determine the game piece positions and color as circles. The pose is published to topic.

/playing_field_pub	/object_poses	/play_field	Determine the current game state (e.g. which piece occupy which spot) and publish the state to the topic.
/robot_master	/play_field	/initial_to_goal	Based on current state of the playing field, generate the next best move for the robot (TTT algorithm). Each box in the 3x3 grid is mapped to a (x,y,z) coordinates. Publish the (x,y,z) coordinates to the topic.
/motion_planner	/initial_to_goal	/goal_pose	From the initial and goal pose, generate the waypoints for the path of the end effector. Publish each waypoint in sequence to the topic iedddait for 3 seconds.
/inverse_kinematics	/goal_pose	/joint_angle_degree	Convert the pose from task space (x,y,z) to joint space (θ_1 ,θ_2 ,θ_3, θ_4, $\theta_{gripper}$) and publish to topic.
/joints_angle_to_data	/joint_angle_degree	/joints_position	Convert the joint angles in degree to integer values to be written to the motor register for position control. The data is published to topic
/joint_controller	/joints_position	-	The hardware interface to read and write data to the registers in the Dynamixel smart actuators. Takes in the joints position data and write to the relevant registers for position control