Project Final Submission evaluation criteria

Course Code:	MCT344	Course Name:	Industrial Robotics	Project Final Submission	.....	Date:	....
Student Name:	Omar Mahraus Elhawary			Student ID:	19P1060		

	A (89-100)				B (76-88)				C (67-75)				D (60-66)				F (0-59)			
	100	96	92	89	88	84	80	76	75	72	69	67	66	64	62	60	59	40	20	0
Report (50%)	• Clear and relevant research methodology with complete implementation • Excellent analysis of results and complete relevant conclusions		✓		• Clear and relevant research methodology missing few components • Good analysis of results missing some minor conclusions				• Clear research methodology missing several components some structure. • Normal analysis of results missing some basic conclusions				• Inappropriate research methodology Structure not clear. • Incomplete analysis or results with some conclusions				• Lack of clear research methodology • Missing proper analysis or results and no conclusions at all.			
Simulation Quality (50%)	• A perfectly running simulation with no errors, • Express a deep understanding of each step to prepare the simulation, • Applying the learned knowledge to prepare a perfectly running simulation with no errors.		✓		• A good running simulation, • Express a very good understanding of each step to prepare the simulation, • Applying most of the learned knowledge to prepare a good running simulation.				• A moderate ability of preparing the simulation, • Express a good understanding of the steps to prepare the simulation, • Applying most of the learned knowledge, correctly, in the prepared solution.				• A fair ability of preparing the simulation, • Express a fair understanding of the steps to prepare the simulation, • Applying some of the learned knowledge, correctly, in the proposed solution.				• An inability of preparing the simulation, • Failed to define the correct parameters of preparing the simulation, • Failed to correctly apply the learned knowledge for proposing a valid simulation.			

1st marker Total	28	1st marker Signature		ASU Agreed Mark	28
2nd Marker Total	28	2nd marker Signature		UEL Agreed Mark	28

...ts and Feedback:

.....

.....

.....

.....

.....

.....

.....

.....

UEL Grading System	Agreed Mark	ASU Grading Scale	
% equivalent at UEL	Range	% at ASU	Grade
95% and higher		97% and higher	A+
82% to less than 95%	✓	93% to less than 97%	A
70% to less than 82%		89% to less than 93%	A-
66% to less than 70%		84% to less than 89%	B+
63% to less than 66%		80% to less than 84%	B
60% to less than 63%		76% to less than 80%	B-
56% to less than 60%		73% to less than 76%	C+
53% to less than 56%		70% to less than 73%	C
50% to less than 53%		67% to less than 70%	C-
45% to less than 50%		64% to less than 67%	D+
40% to less than 45%		60% to less than 64%	D
Less than 40%		Less than 60%	F

MCT344- Industrial Robotics

TEAM 15

Supervised by:

Dr. Shady Ahmed Maged

Dr Diaa Emad

Eng. Mina Yousry

Submitted by:

Kirolos Thabet Fouad	19P6754
Omar Mahrous Eltoutongy	19P1060

Table of Contents

Table of Figures:

Introduction:

In this documentation for this project milestone1, we will outline the steps required to program a robot to move according to a set of joint angles and publish its end effector position. This project milestone focuses on using ROS (Robot Operating System) to control a robotic arm and calculate the end effector position based on its joint angles.

To achieve this goal, we will first need to choose a set of four reasonable joint angles and move the robot using a ROS node that publishes on joint command topics. Then, we will calculate the DH (Denavit-Hartenberg) parameters and use them to determine the forward kinematics matrix to the base of the end effector. With this information, we can calculate the end effector position and publish it to a ROS topic in the form of a std_msgs/Float32MultiArray.

Throughout this documentation, we will go through the steps required to complete this project milestone in detail, including the necessary background information and code examples. We assume that you have a basic understanding of ROS and programming concepts such as matrices and arrays. By following the steps in this documentation, you will gain a deeper understanding of how to control a robotic arm using ROS and calculate its end effector position based on its joint angles.

Screenshots of code:

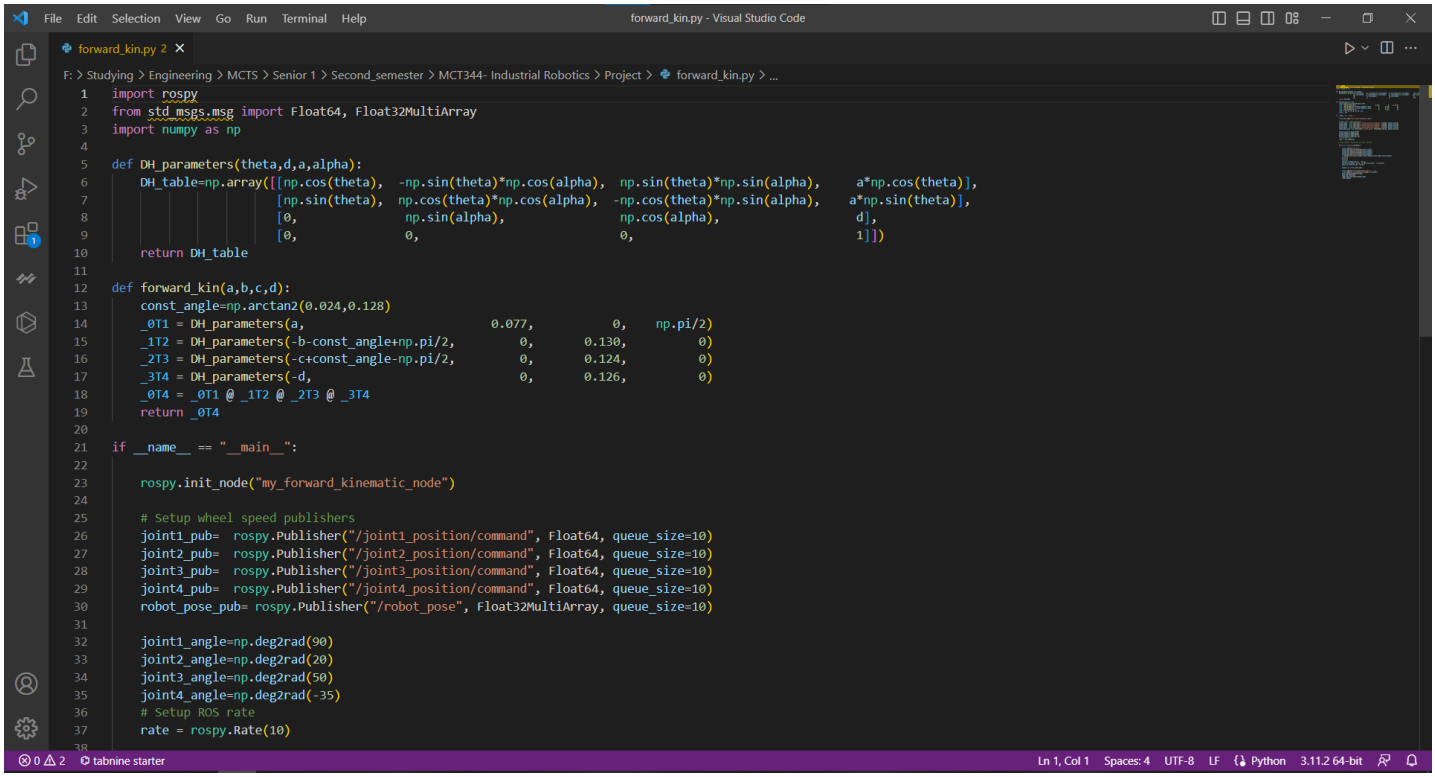


Figure 1: Code part1

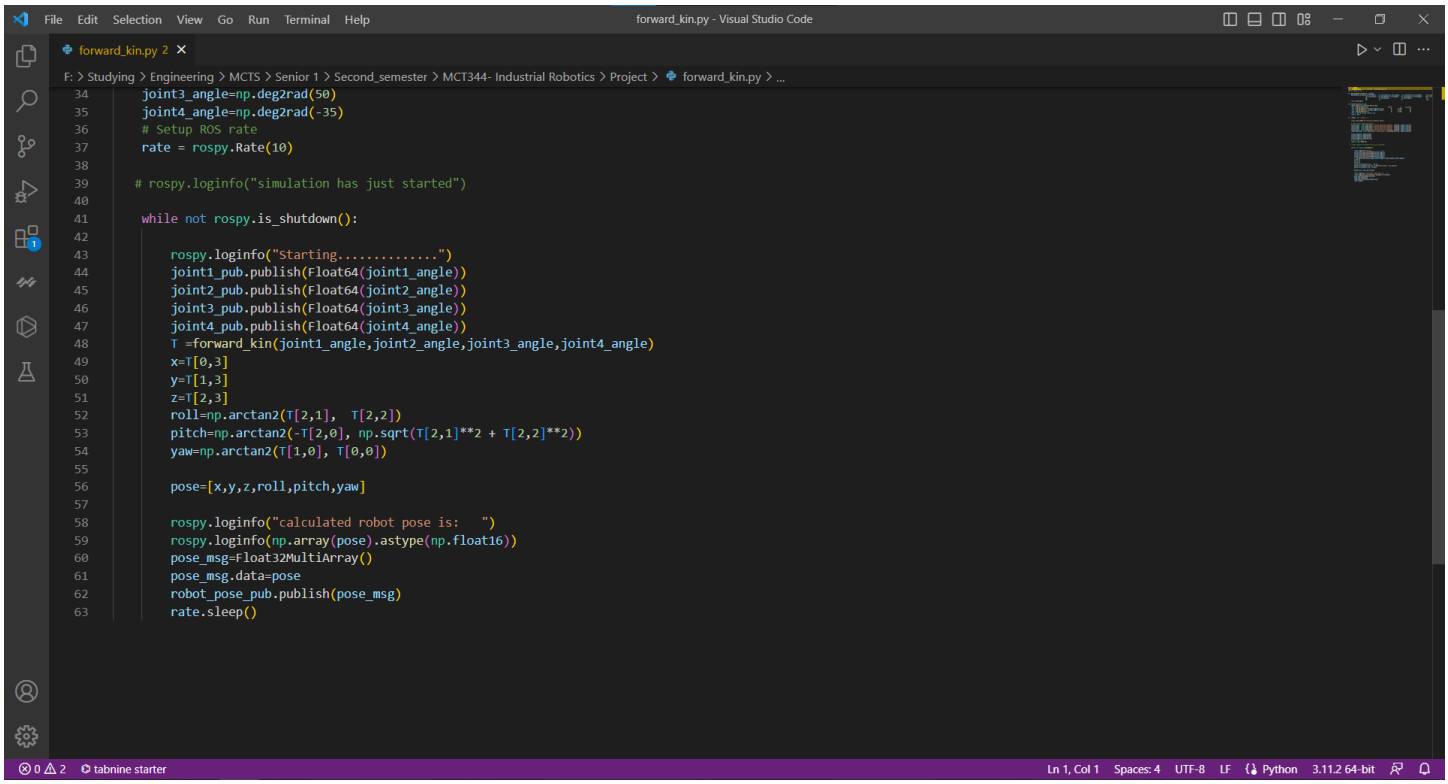


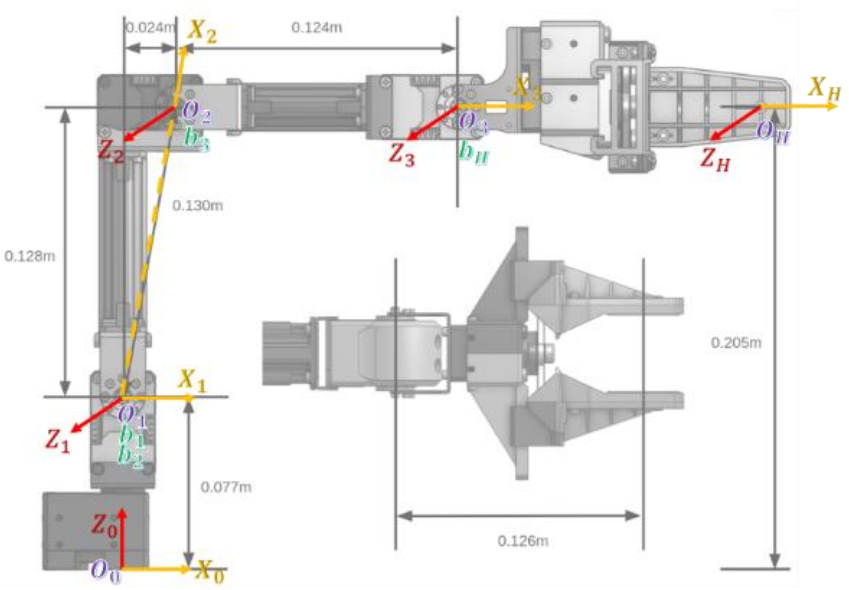
Figure 2: code Part 2

This Python script is designed to control a robotic arm's forward kinematics and publish the robot's position to ROS topics. The DH_parameters() function calculates the DH transformation matrix based on the input values of theta, d, a, and alpha. The forward_kin() function calculates the transformation matrix between the robot's base frame and end-effector frame based on the DH parameters and the joint angles of the four robot joints.

The script initializes ROS nodes and publishers for joint angle and robot pose data. The joint angles are initialized and published to ROS topics. The forward kinematics function is called to calculate the robot's position and orientation. The calculated position and orientation are published to a ROS topic as a Float32MultiArray message. The script runs continuously at a rate of 10 Hz using the rospy.Rate() function.

DH parameters:

Joint	θ	d	a	α
1	θ_1	0.077	0	90
2	$-\theta_2 + 90 - c$	0	0.130	0
3	$-\theta_3 - 90 + c$	0	0.124	0
4	$-\theta_4$	0	0.126	0



$$C = \tan^{-1}\left(\frac{0.024}{0.128}\right) \approx 11$$

Figure 3: DH Parameters tables

DH parameters explaining:

This is the link to the video explaining the DH parameters in depth. Please download it to listen to the explanation.

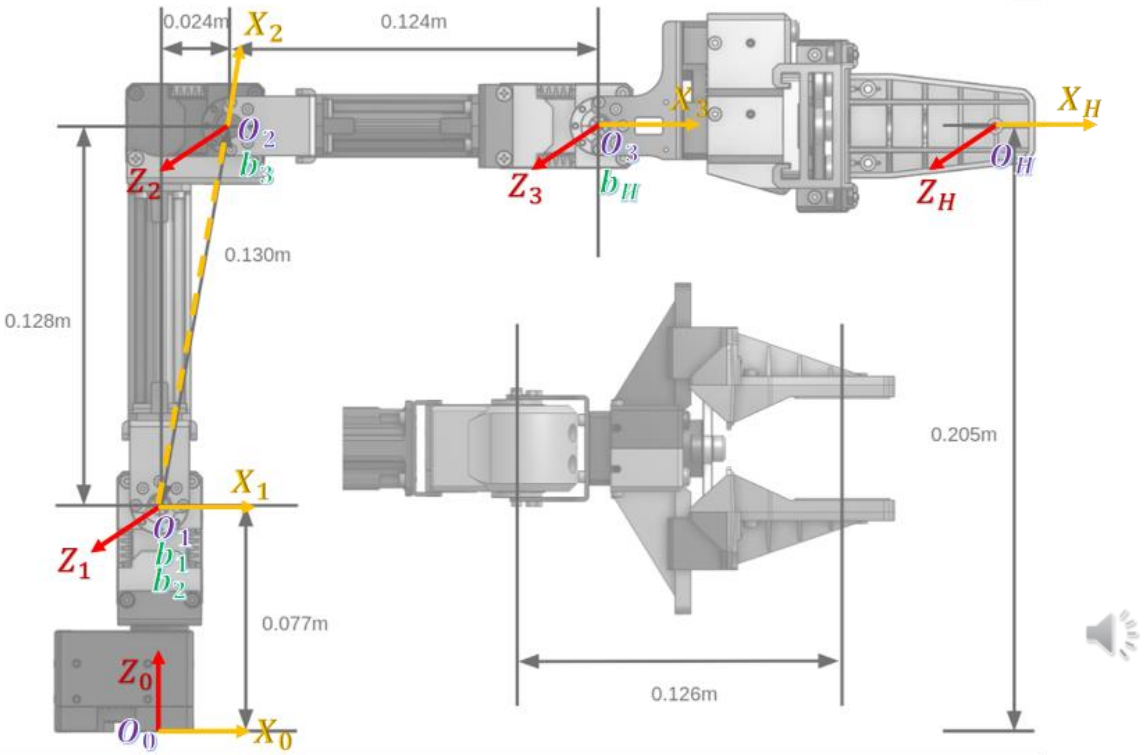


Figure 4: Robot with axis

Screenshot for the robot:

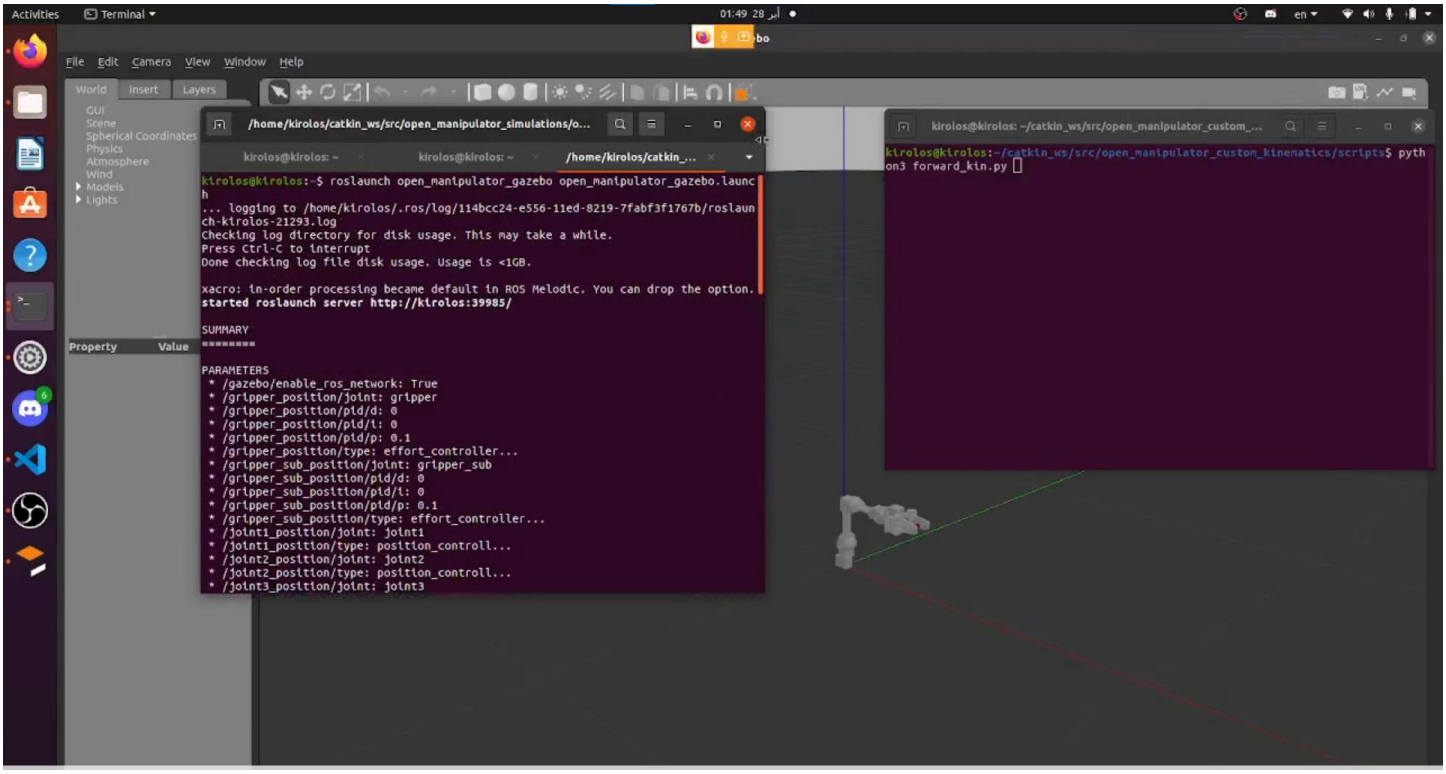


Figure 5: the Initial Position of the robot

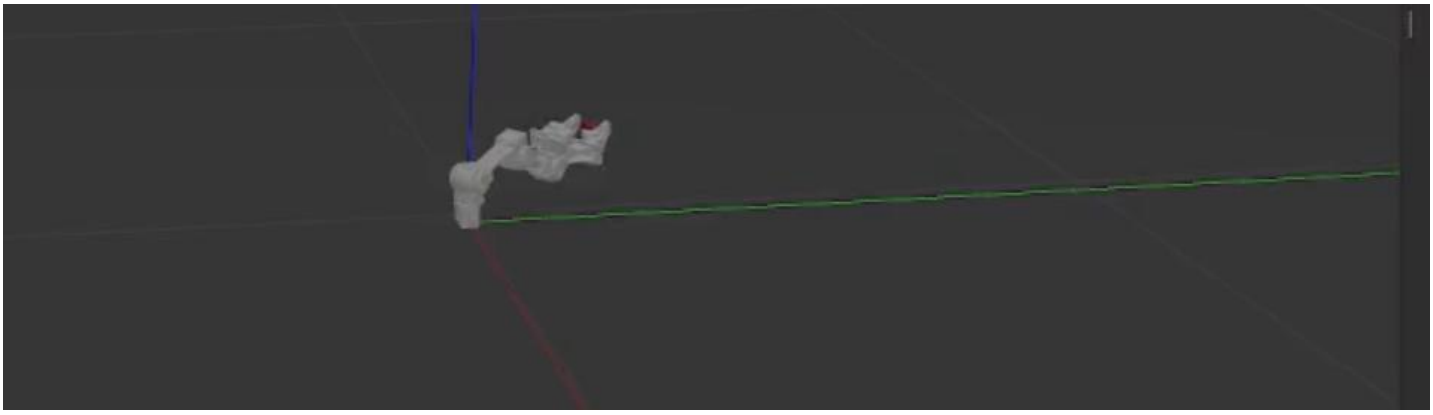


Figure 6: moved position of the robot.

Screenshots for the outputs:

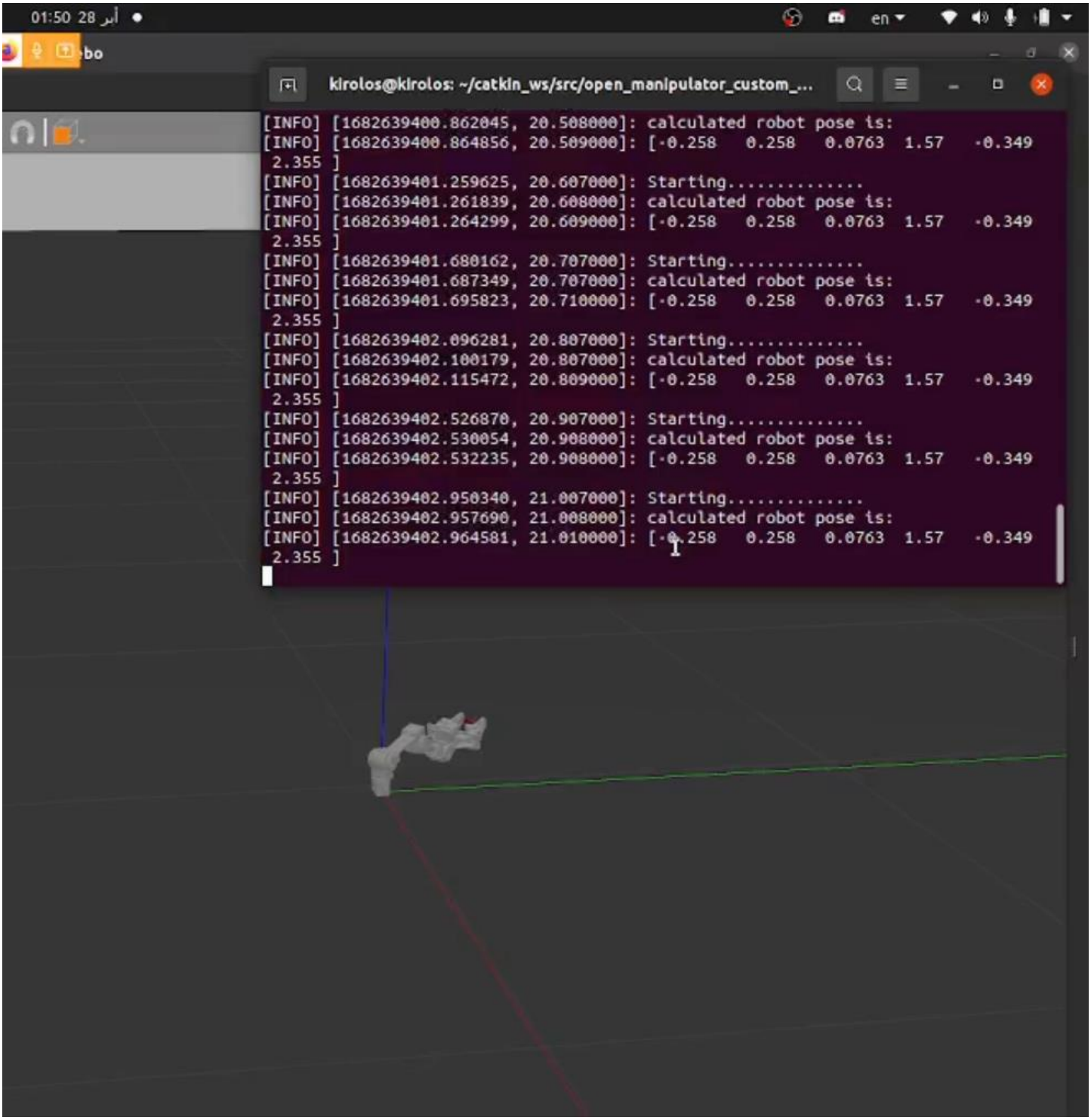


Figure 7: outputs with terminal

Screenshot for the published topic:

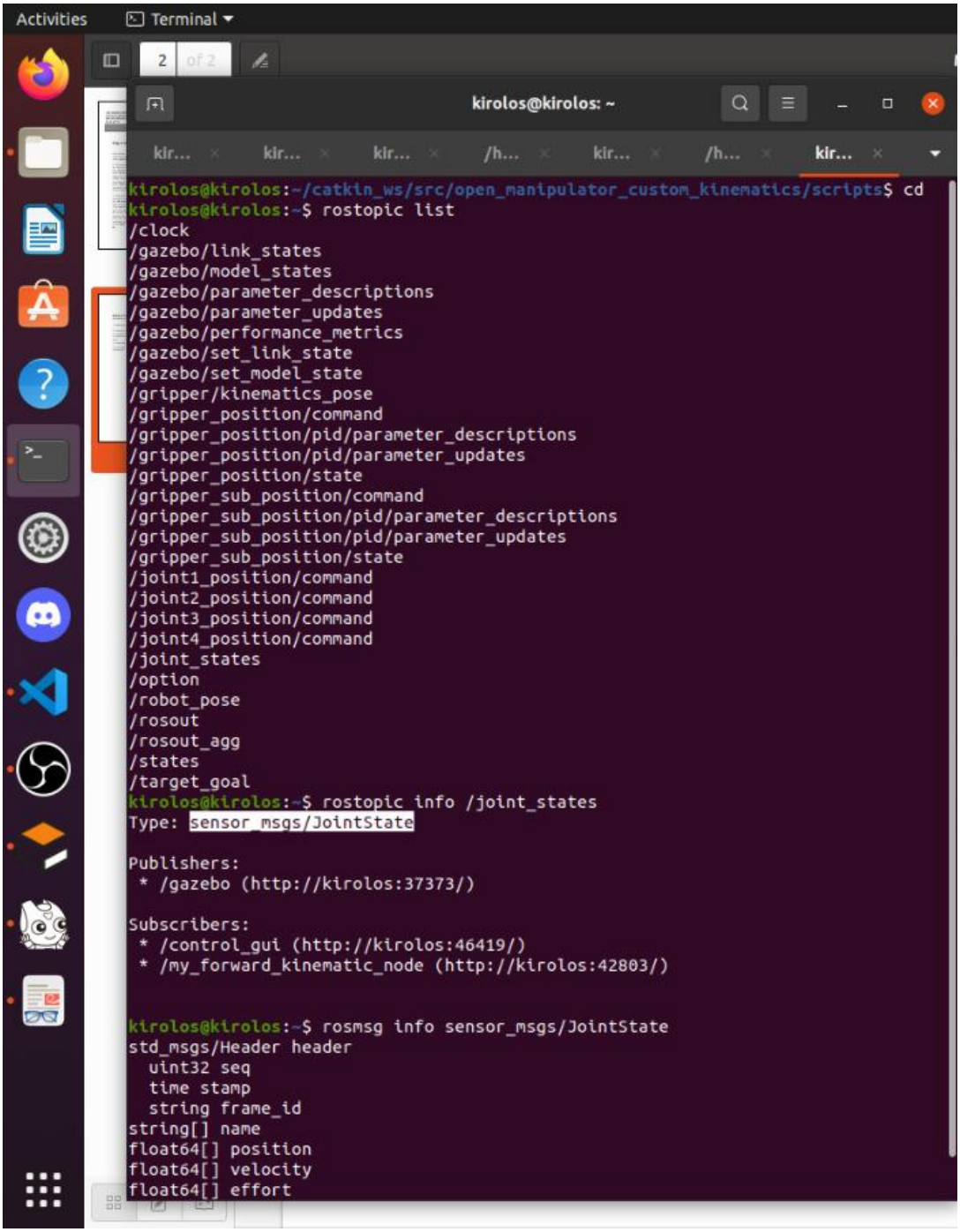


Figure 8: Topic lists

Video link:

In this link, you can watch our video.

And in this link the drive folder containing all data.

References:

OpenMANIPULATOR-X (robotis.com)

MCT344- Industrial Robotics

TEAM 15

Supervised by:

Dr. Shady Ahmed Maged

Dr Diaa Emad

Eng. Mina Yousry

Submitted by:

Kirolos Thabet Fouad	19P6754
Omar Mahrous Eltoutongy	19P1060

Table of Contents

Table of Figures:

Introduction:

Project Milestone 2 is a robotics project where each team is required to submit a working package named **open_manipulator_custom_kinematics**. The package consists of two nodes, **fkine_node** and **ikine_node**, that perform forward kinematics and inverse kinematics calculations, respectively, for a robot. The project involves using DH parameters and geometric methods to calculate joint angles and transform matrices and publishing them on specific topics in Gazebo. The robot can be controlled using a GUI controller to verify the accuracy of the calculations.

Screenshots of code:

1. Position node:

   This node is responsible for holding the target goal values and publish it to a topic named target_goal which the ikine node subscribes to it.

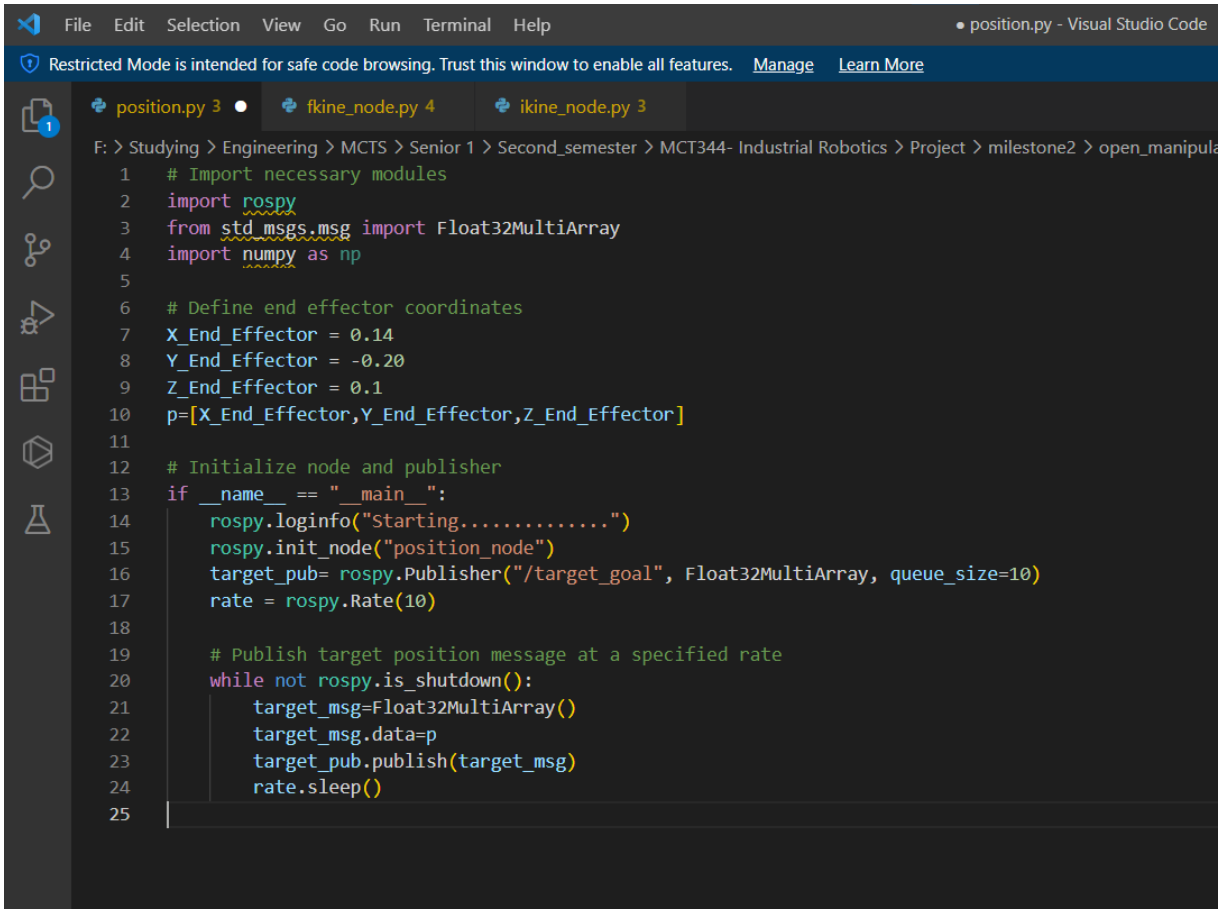


Figure 1: position node

2. fkine node:

this node read the angle values from the gazebo topic Jointstates and do the regular algorithm to calculate the position of end effector in Euler form.

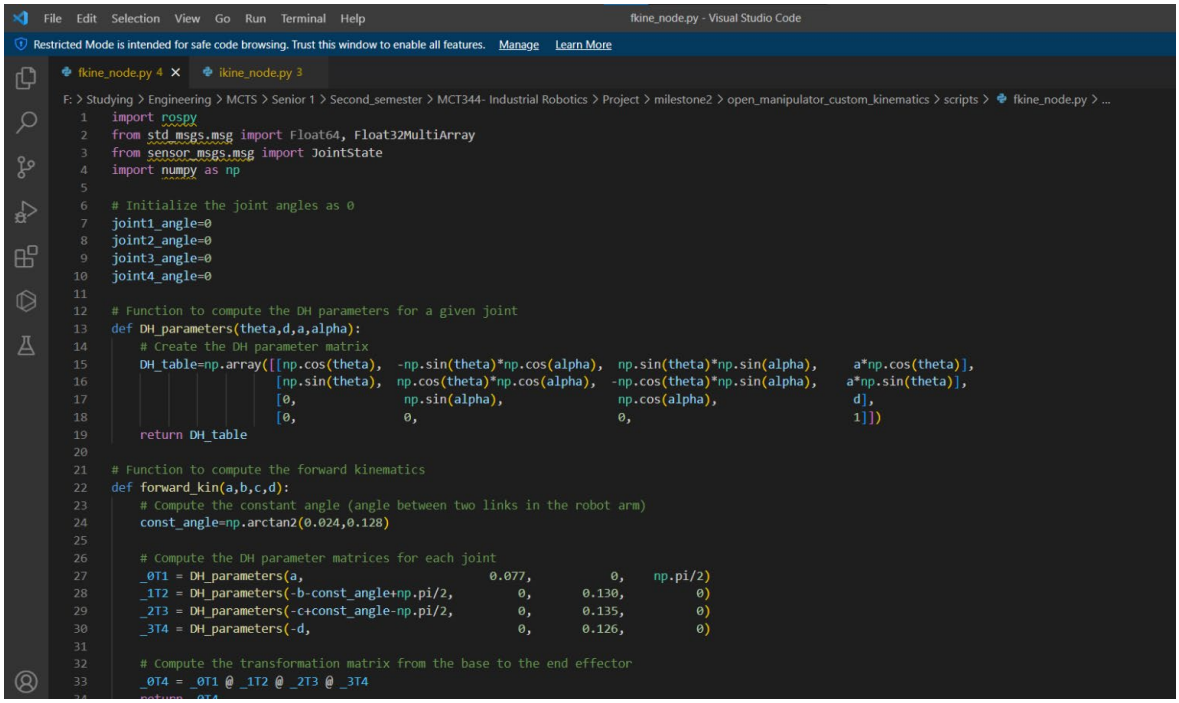


Figure 3: fkine node part1

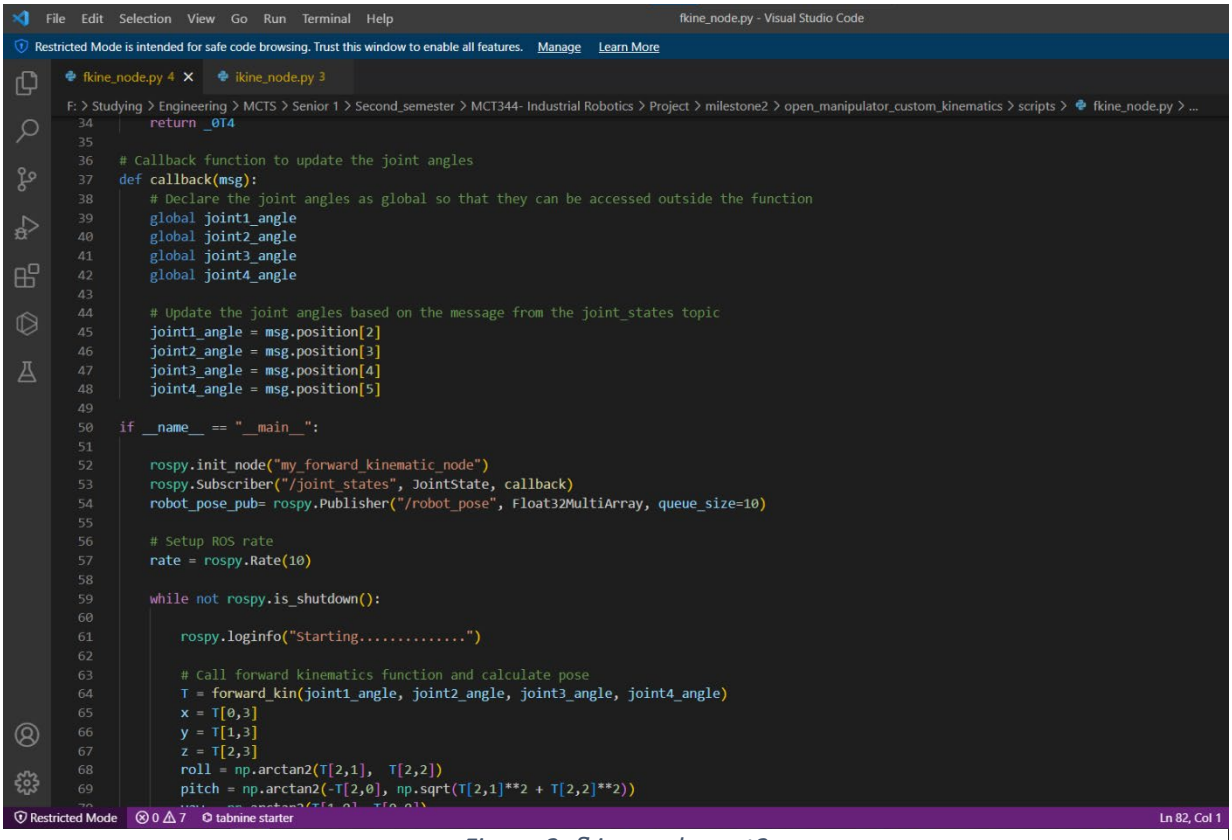


Figure 2: fkine node part2

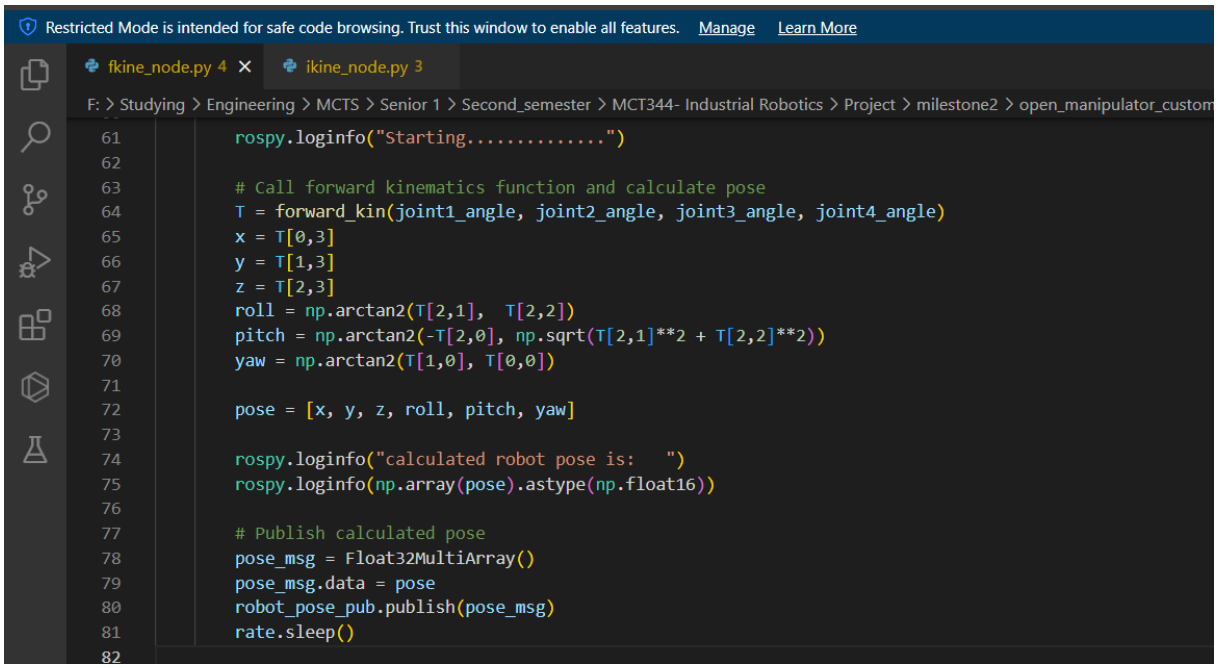


Figure 4: fkine node part3

To work with the joint states we had to know its details as following:

Figure 5: Joint_states details

3. ikine node:

this nodes subscribes to only one topic which is the target_goal and publishes to the 4 jointstate/command topic which are connected to gazebo to move the robot, its accuracy may be checked using the gui and we found that the maximum error in our calculations is about 0.02 , we can guess the problem which is the shifting in the axis in gazebo (small but considerable).

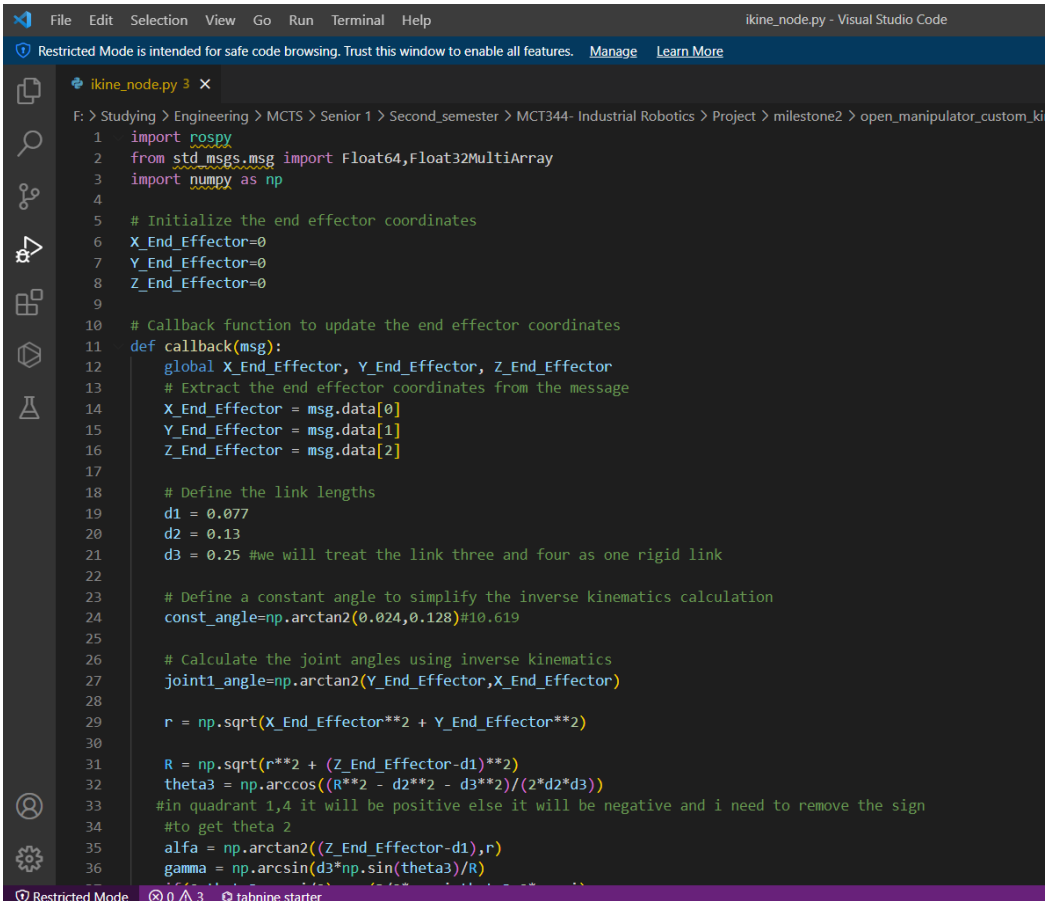


Figure 7: ikine node part 1

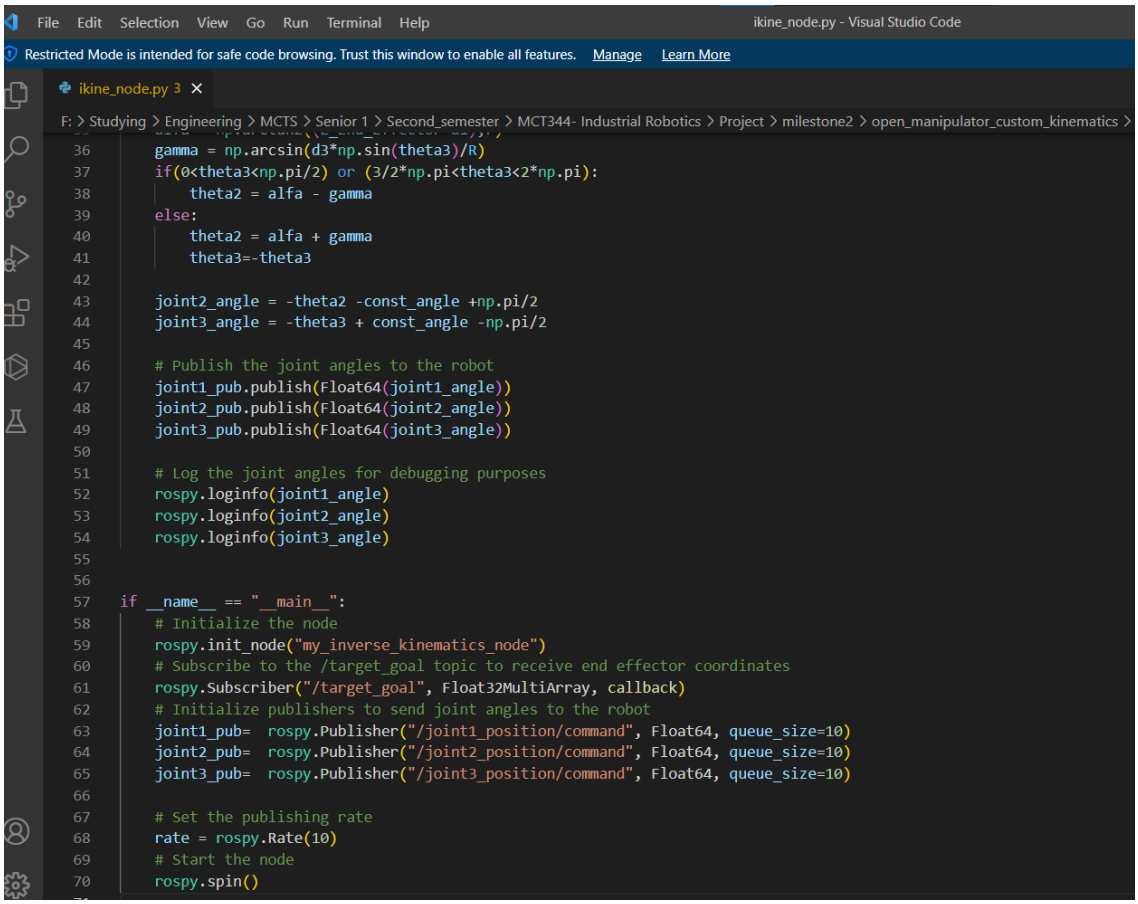
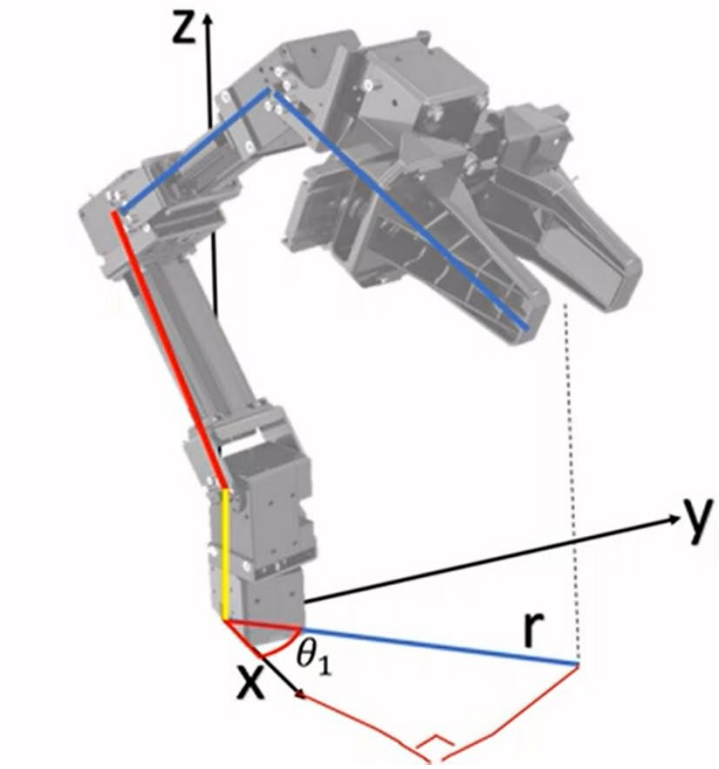


Figure 6: ikine node part 2



$$\theta_1 = \tan^{-1} \frac{y_{end\ effector}}{x_{end\ effector}}$$

Figure 9: geometrical explain p1

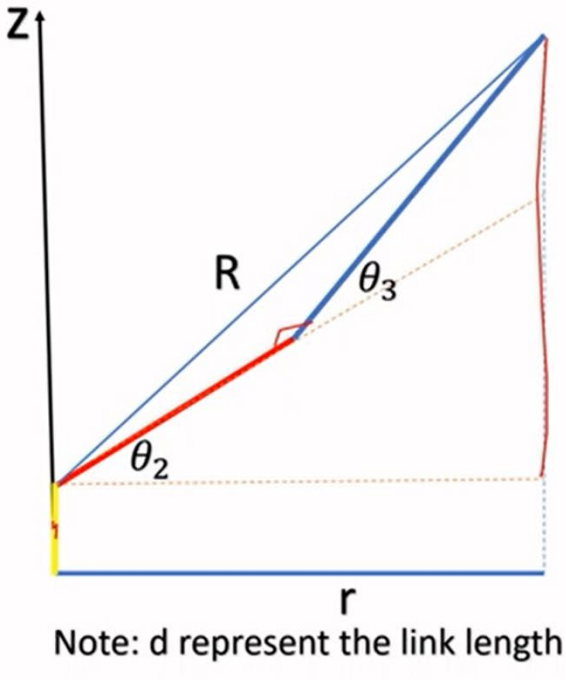
Applying cos rule to get θ_3:

$$R^2 = r^2 + \left(Z_{end\ effector} - d_1\right)^2$$

$$r^2 = X_{end\ effector}{}^2 + Y_{end\ effector}{}^2$$

$$R^2 = d_2^2 + d_3^3 - 2d_2d_3 * \cos(180 - \theta_3)$$

$$\theta_3 = \cos^{-1} \frac{R^2 - d_2^2 - d_3^3}{2 * d_2 * d_3}$$



Note: d represent the link length

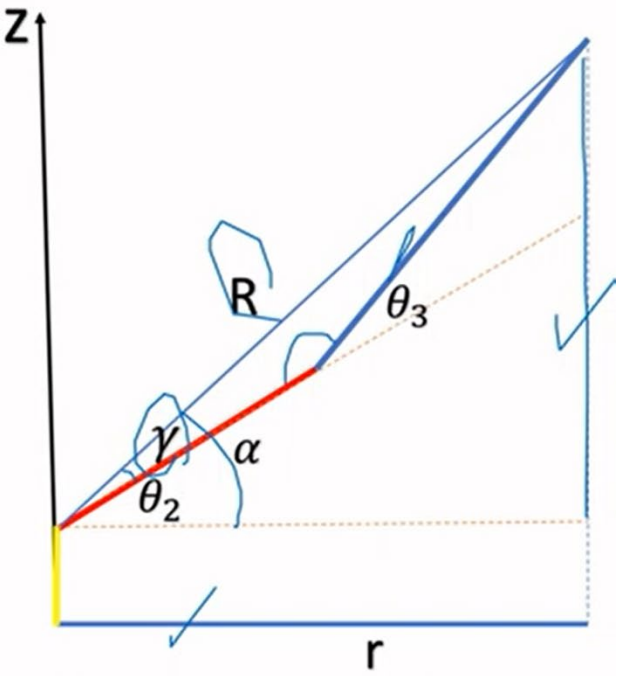
Figure 8: geometrical explain p2

Applying sin rule to get θ_2:

$$\alpha = \tan^{-1}\frac{(Z_{end\ effector} - d_1)}{r}$$

$$\gamma = \sin^{-1}\frac{d_3 \sin(\theta_3)}{R}$$

$$\therefore \theta_3 = \alpha - \gamma$$



Note: d represent the link length

Figure 10: geometrical explain p3

Screenshots of the outputs:

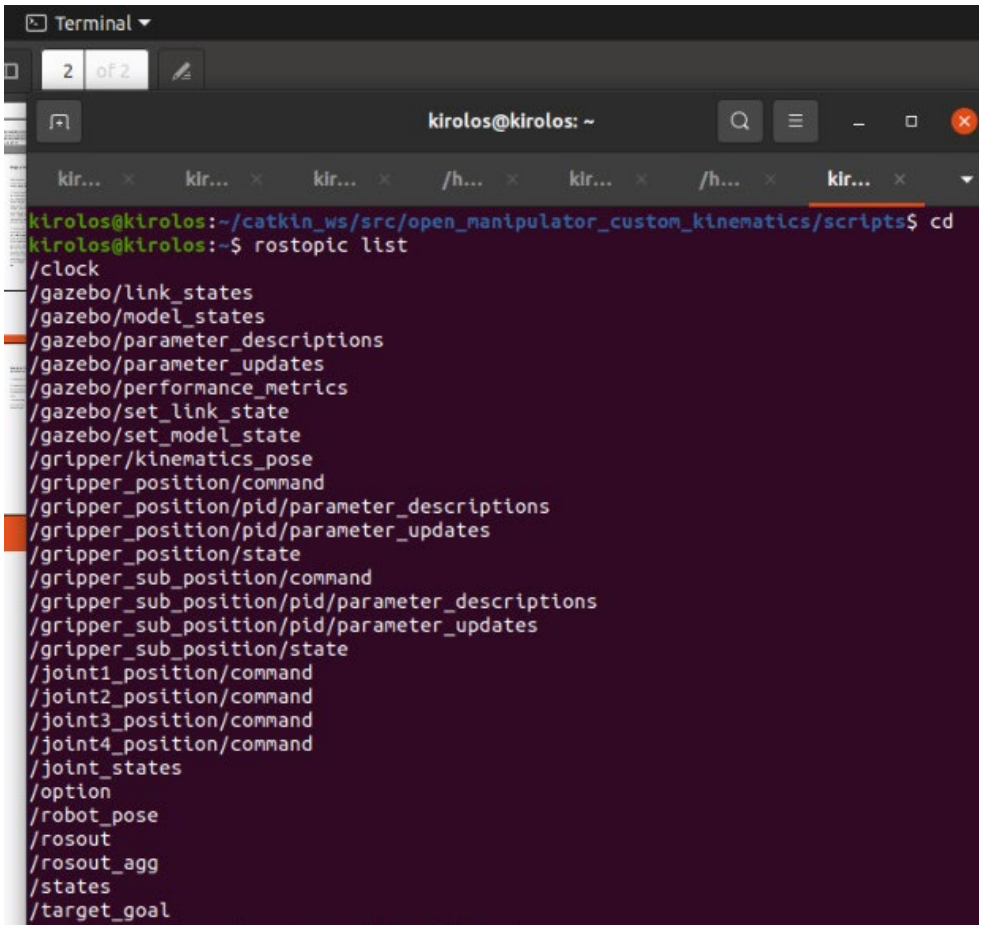


Figure 11: Rostopic list

a. screenshots of topic echoing of calculated pose after running forward kinematics nodes

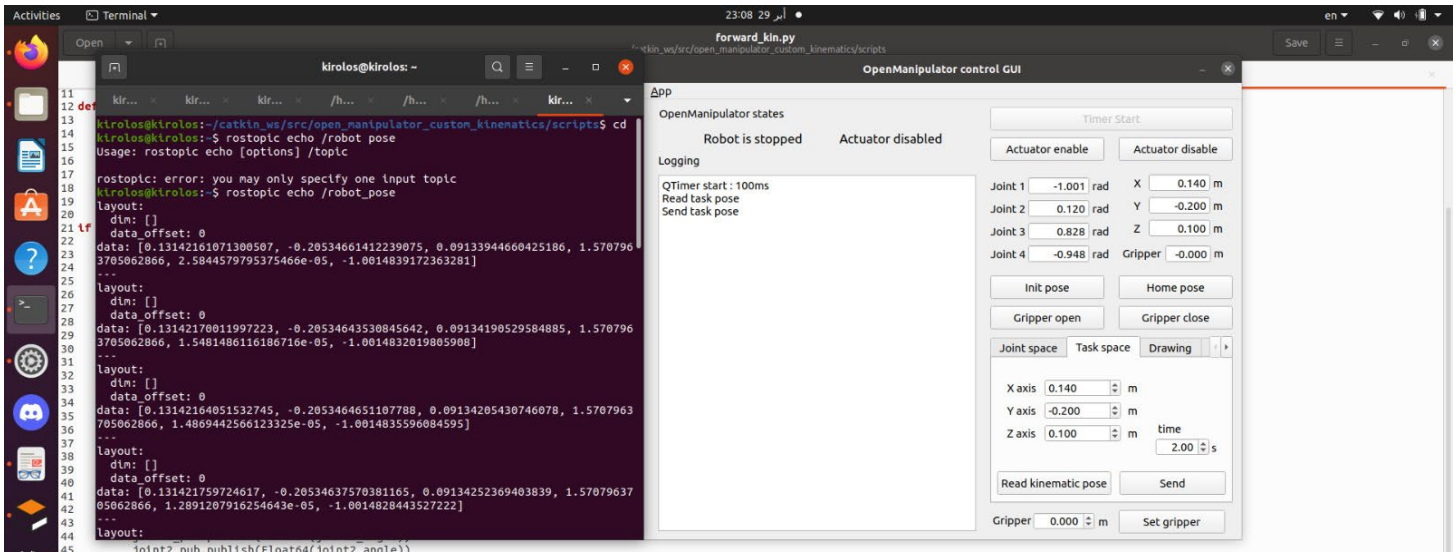


Figure 12: topic echoing robot pose

`

b. screenshots for the robot moved inside gazebo after running your inverse kinematics node.

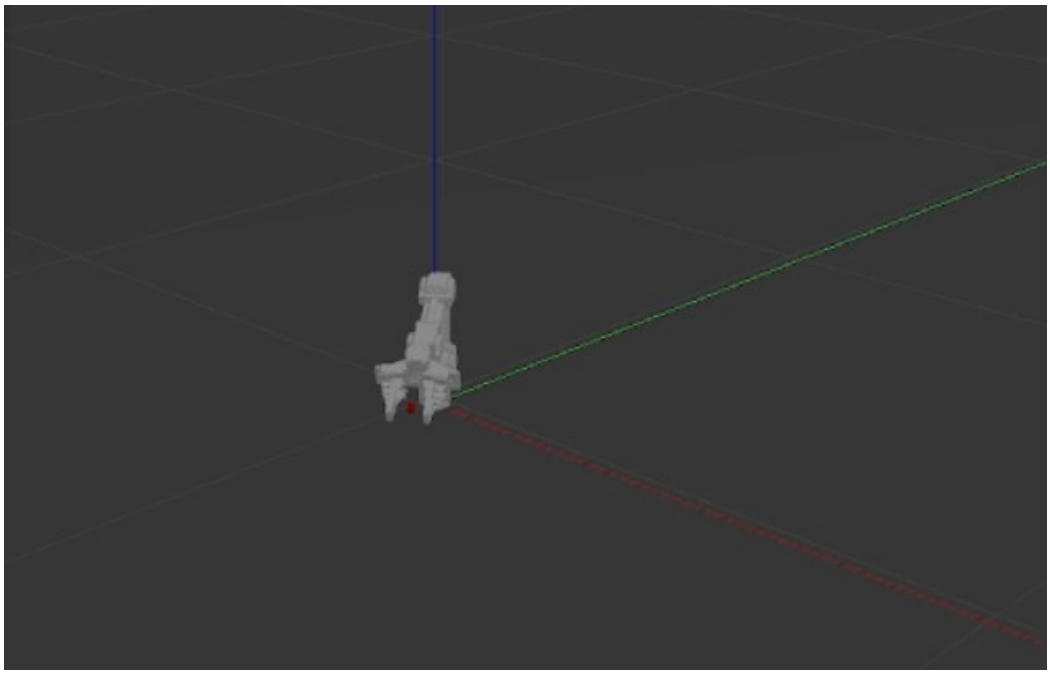


Figure 13: robot after published the required position

c. screenshots of moving the robot using GUI controller and verifying your calculations

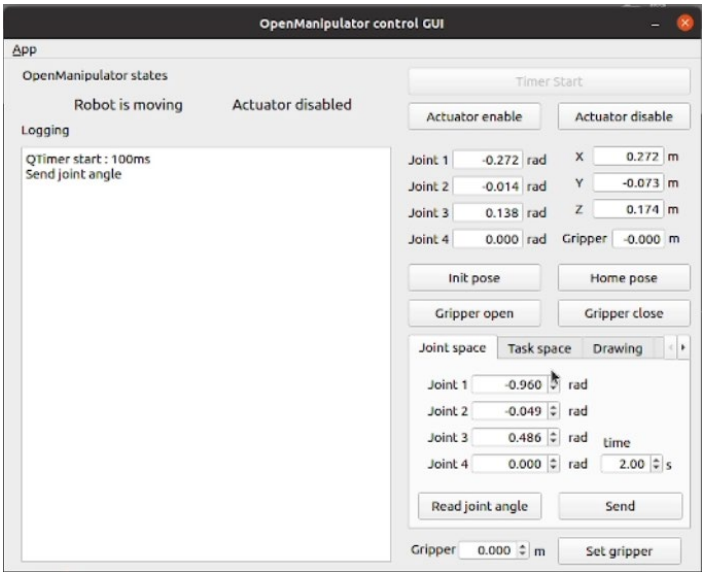


Figure 15: sending the calculated angles from the ikine node.

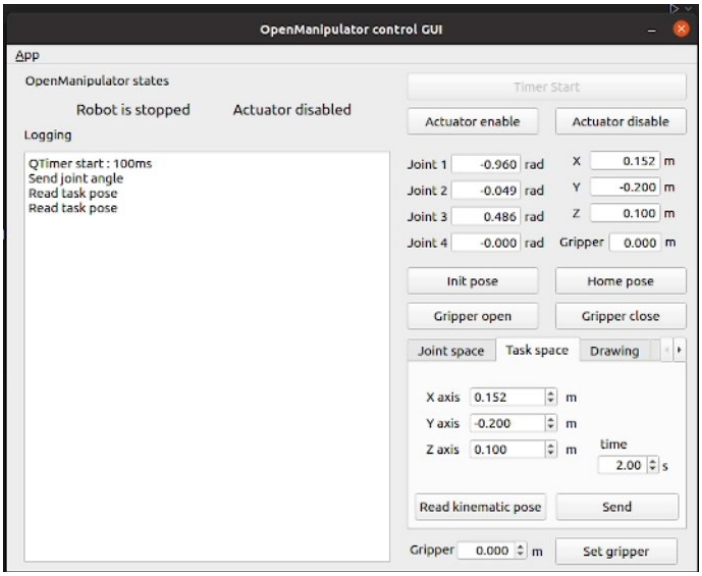


Figure 14:reading the position to compare with the target position node.

RQT Graph:

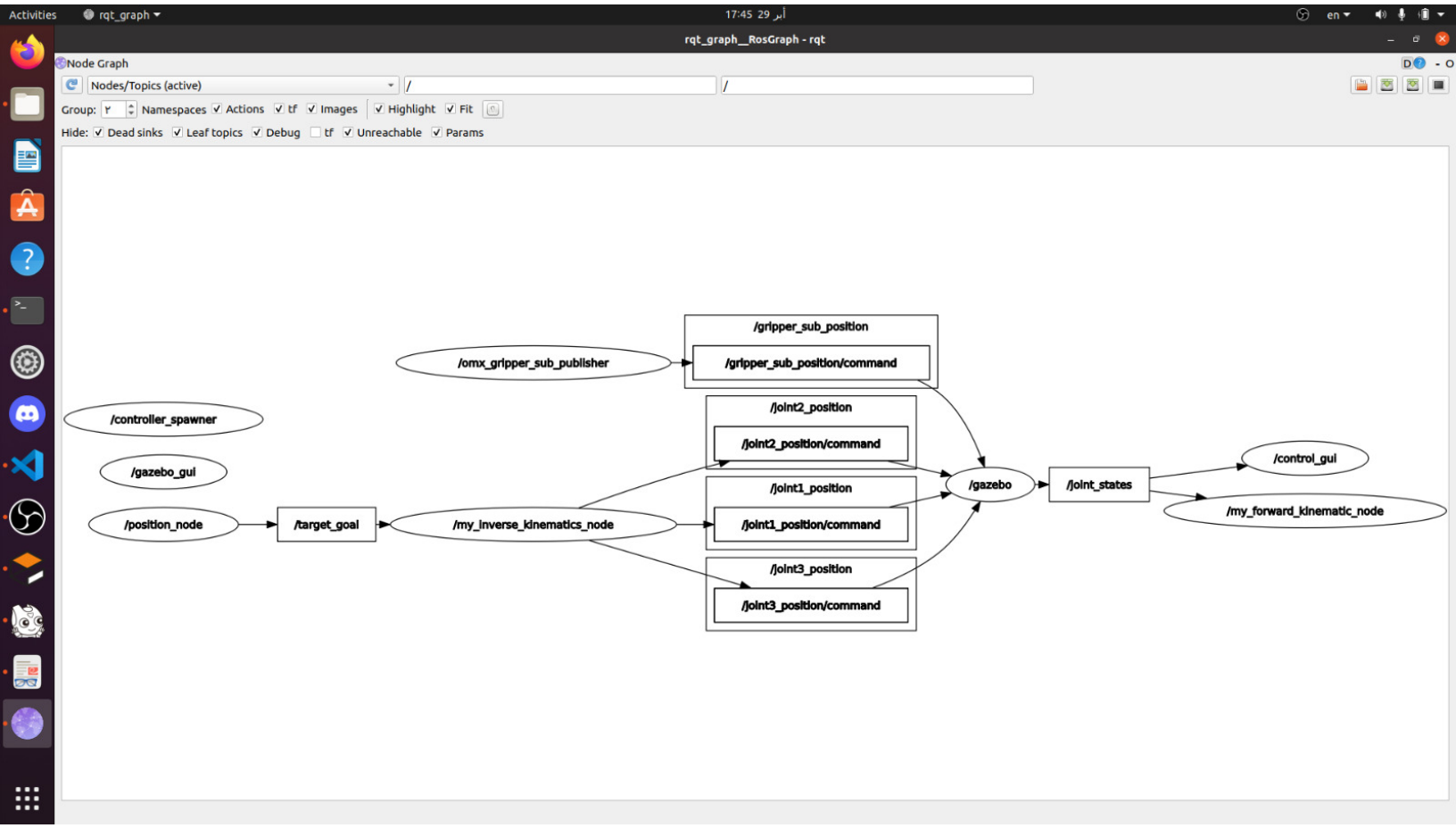


Figure 16: RQT graph

Links:

In this link, you will find our video for this milestone.

In this link, you will find our data for this milestone.

References:

OpenMANIPULATOR-X (robotis.com)

MILESTONE 3

Team 5

Kirolos Thabet Fouad	19P6754
Omar Eltoutongy	19P1060

Table of Contents:

Table of Figures:

1. Introduction:

In this report, a significant stage in our project that will showcase the team's progress and achievements. In this milestone, we will be focusing on developing an environment using GAZEBO, a powerful simulation tool, where our robot, tables, and a task object will coexist. The primary objective of this milestone is to demonstrate the robot's ability to execute a pick-and-place task, successfully moving the task object from one location to another, while ensuring collision-free operation.

By reaching this milestone, our team aims to showcase our proficiency in designing and implementing a functional simulation environment, as well as our ability to program the robot's movements effectively. This milestone is crucial as it tests the core capabilities of the robot in terms of navigation, perception, and manipulation, emphasizing the importance of seamless and safe interaction with its surroundings.

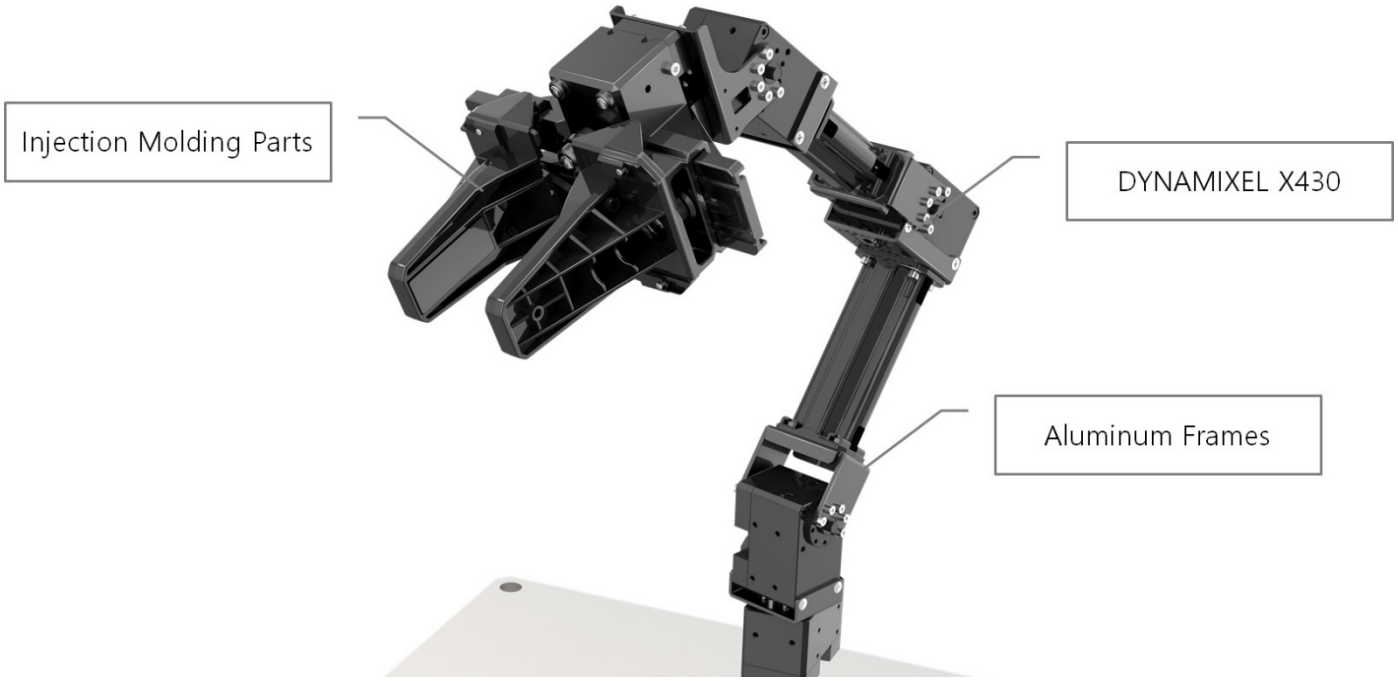


Figure 1:Open manipulator robot

2. Trajectory Planning Node:

This node moves the robot end effector to the required position at which the object needed to be moved is. The motion is performed by two joints (joint1 & joint 2& joint3). Then, the gripper closes to hold the object. After that, the robot moves the end effector to the desired release position. Finally, the gripper opens to release the object.

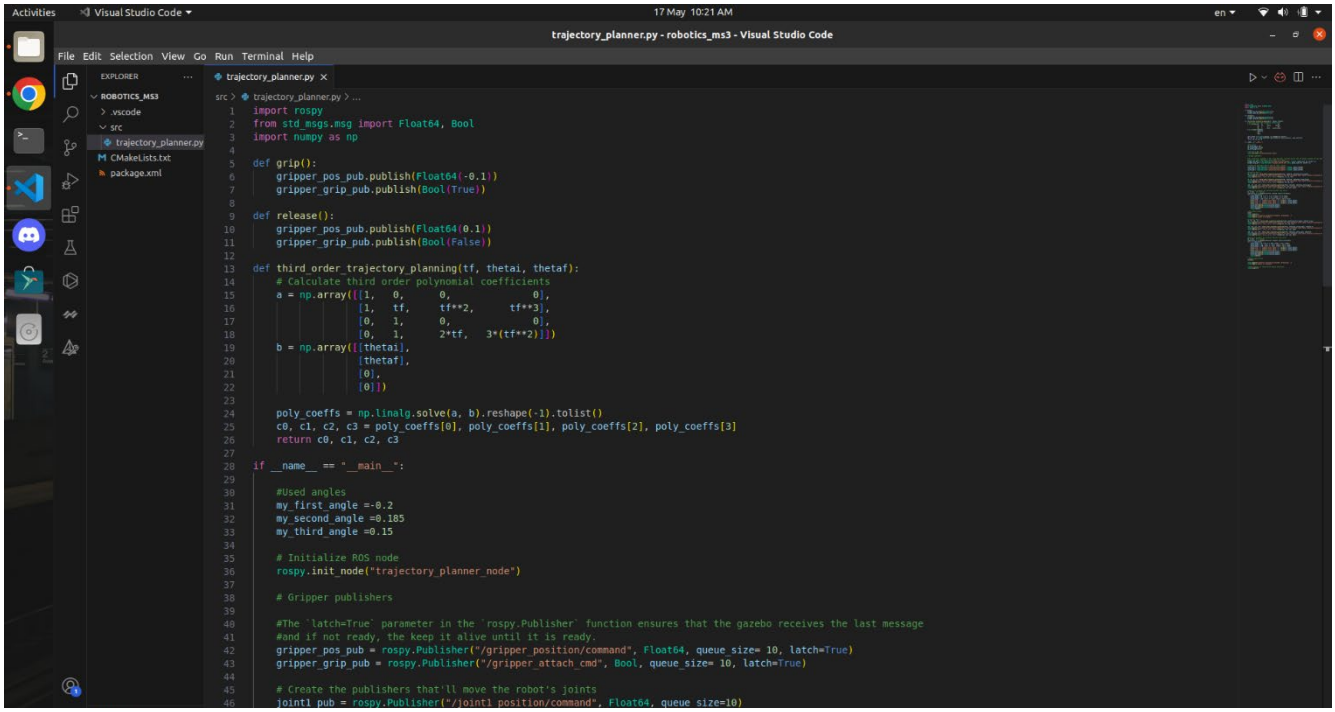


Figure 3: Code Part1

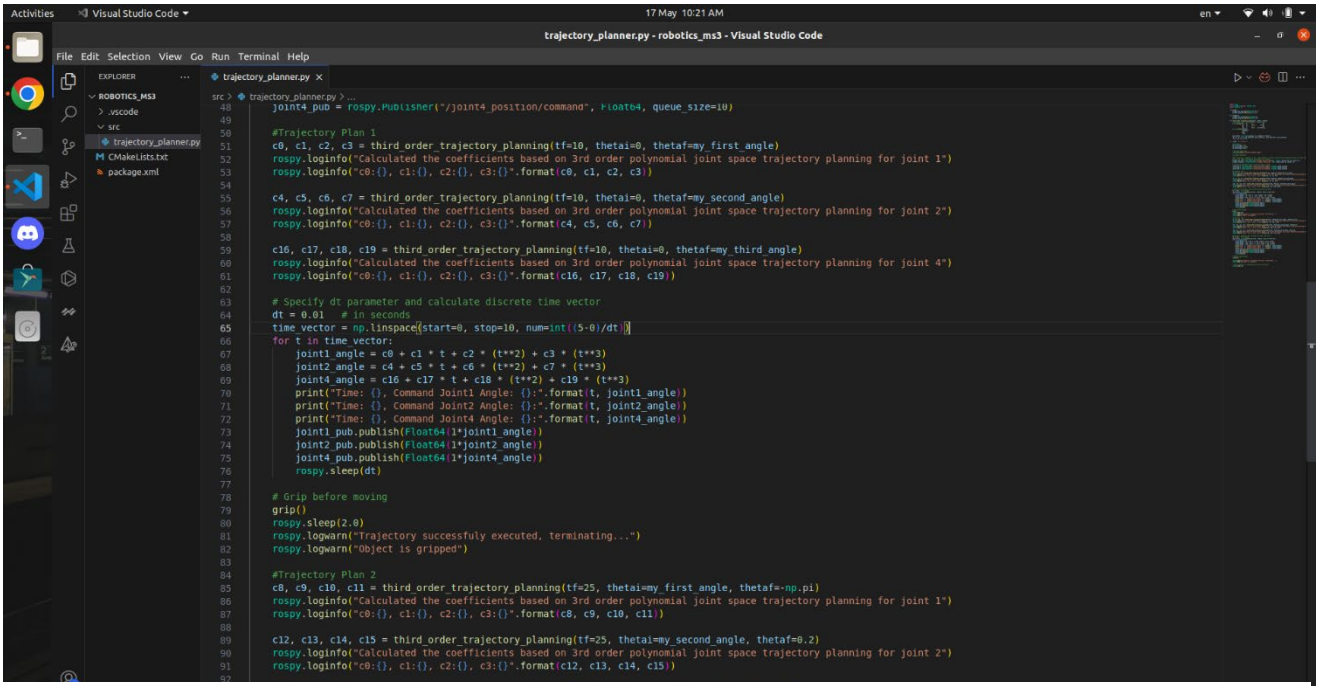


Figure 2: Code Part2

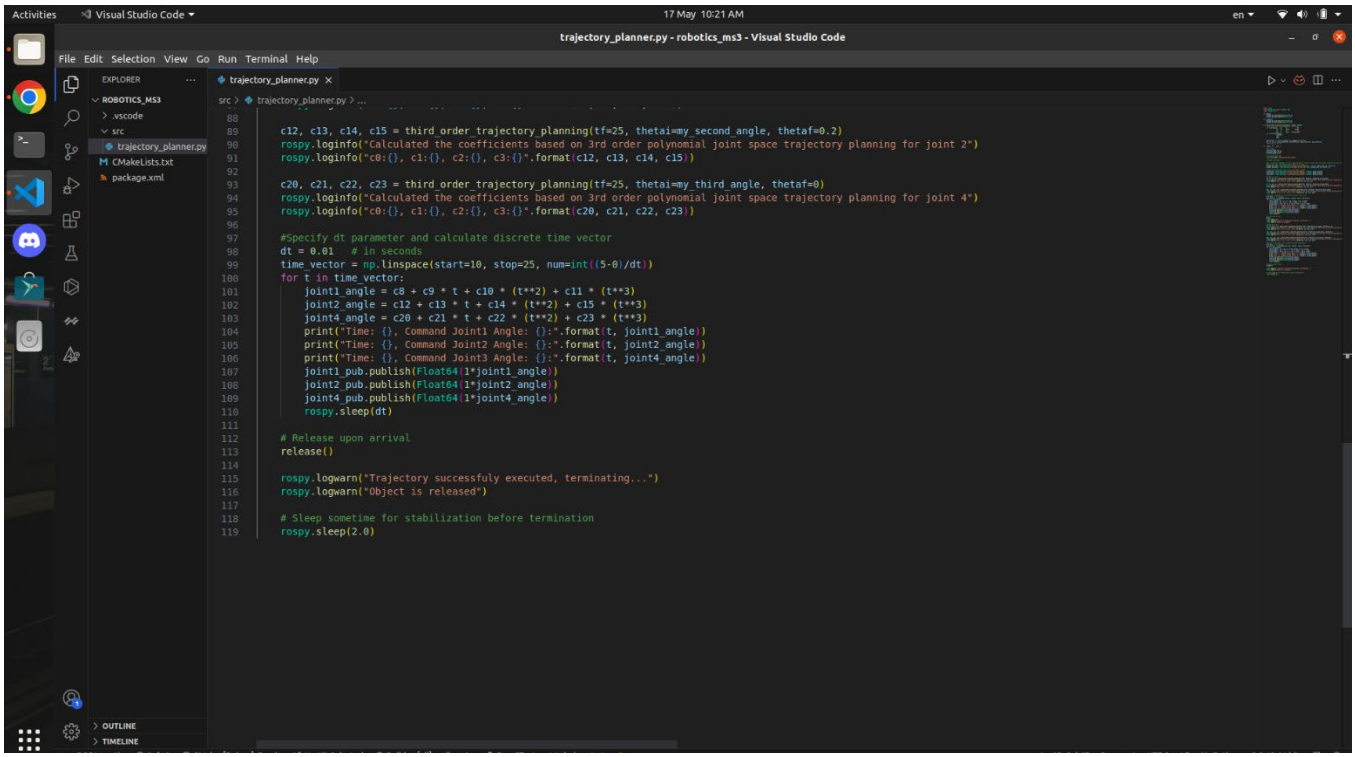


Figure 4: Code Part3

3. RQT Graph:

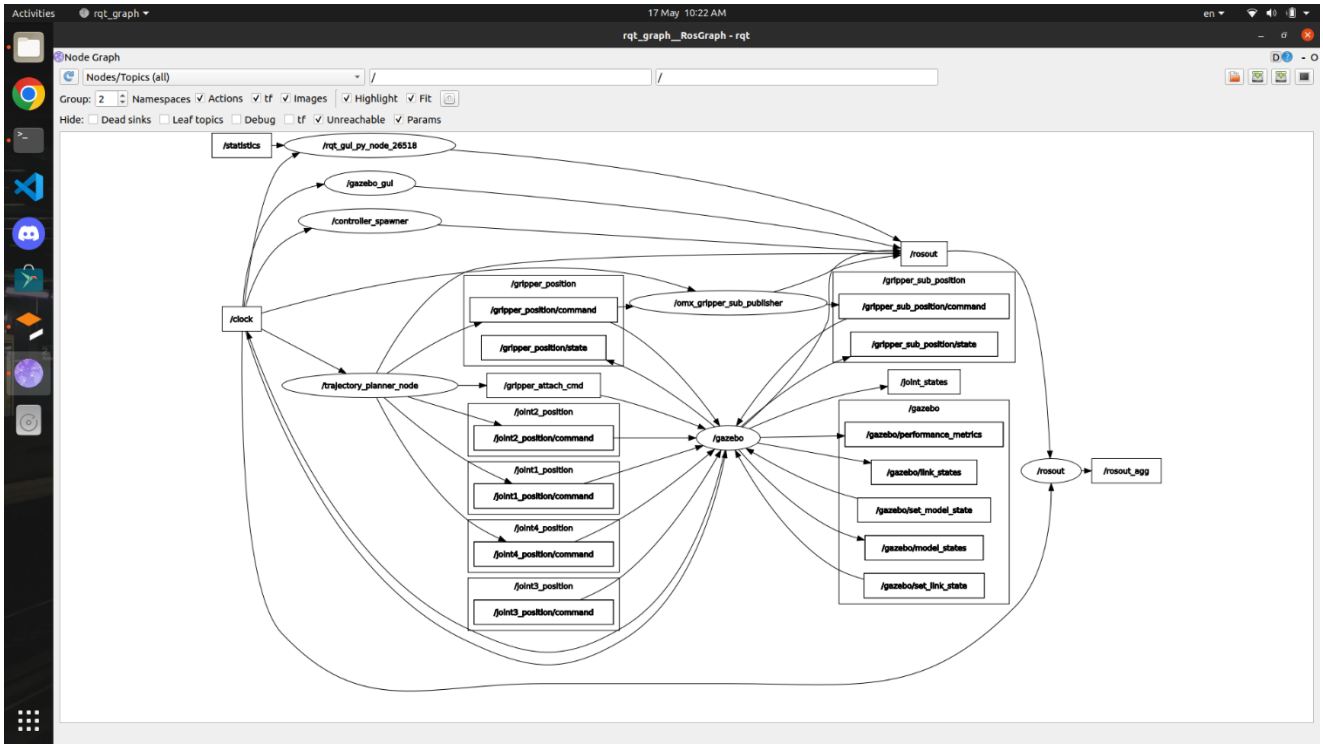


Figure 5: RQT Graph (all)

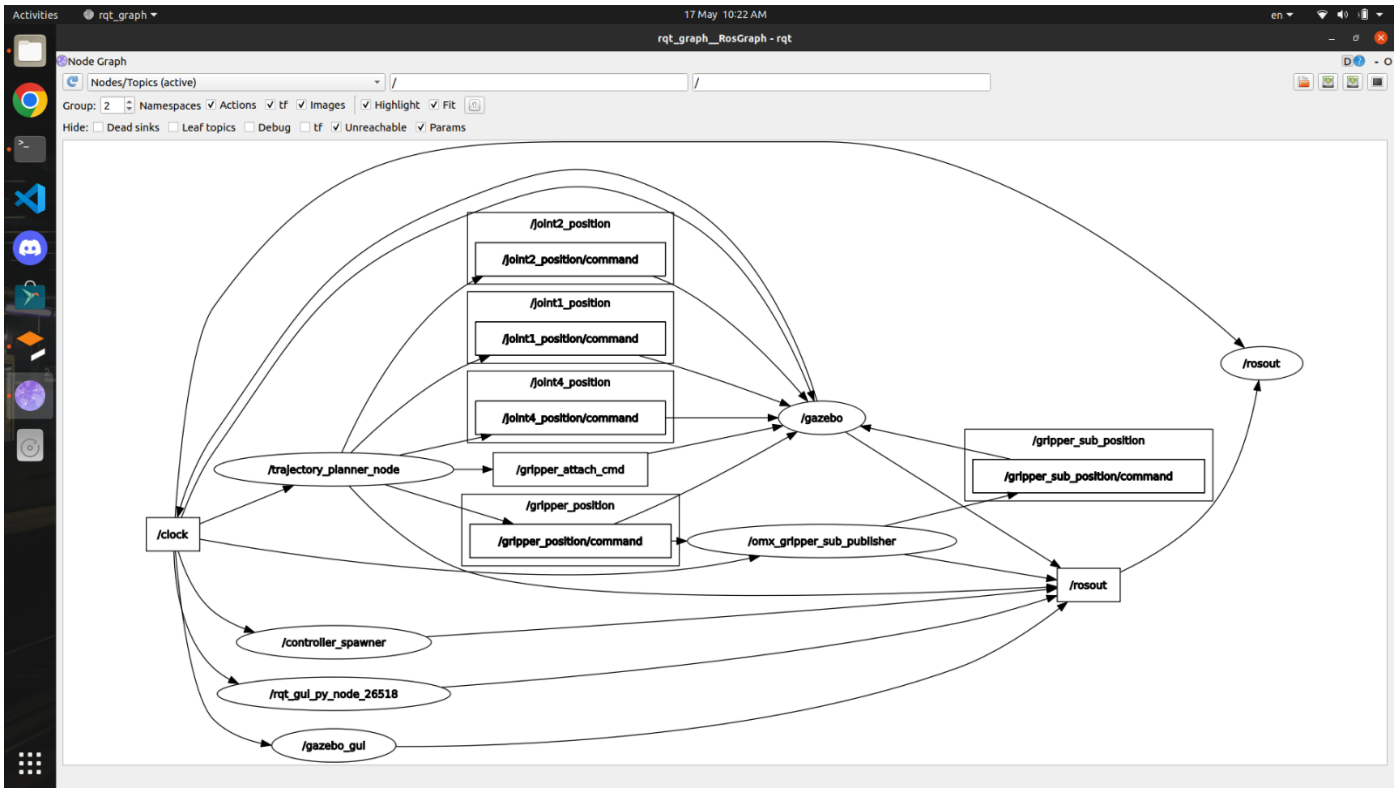


Figure 7: RQT Graph (active)

4. Environment From Gazebo:

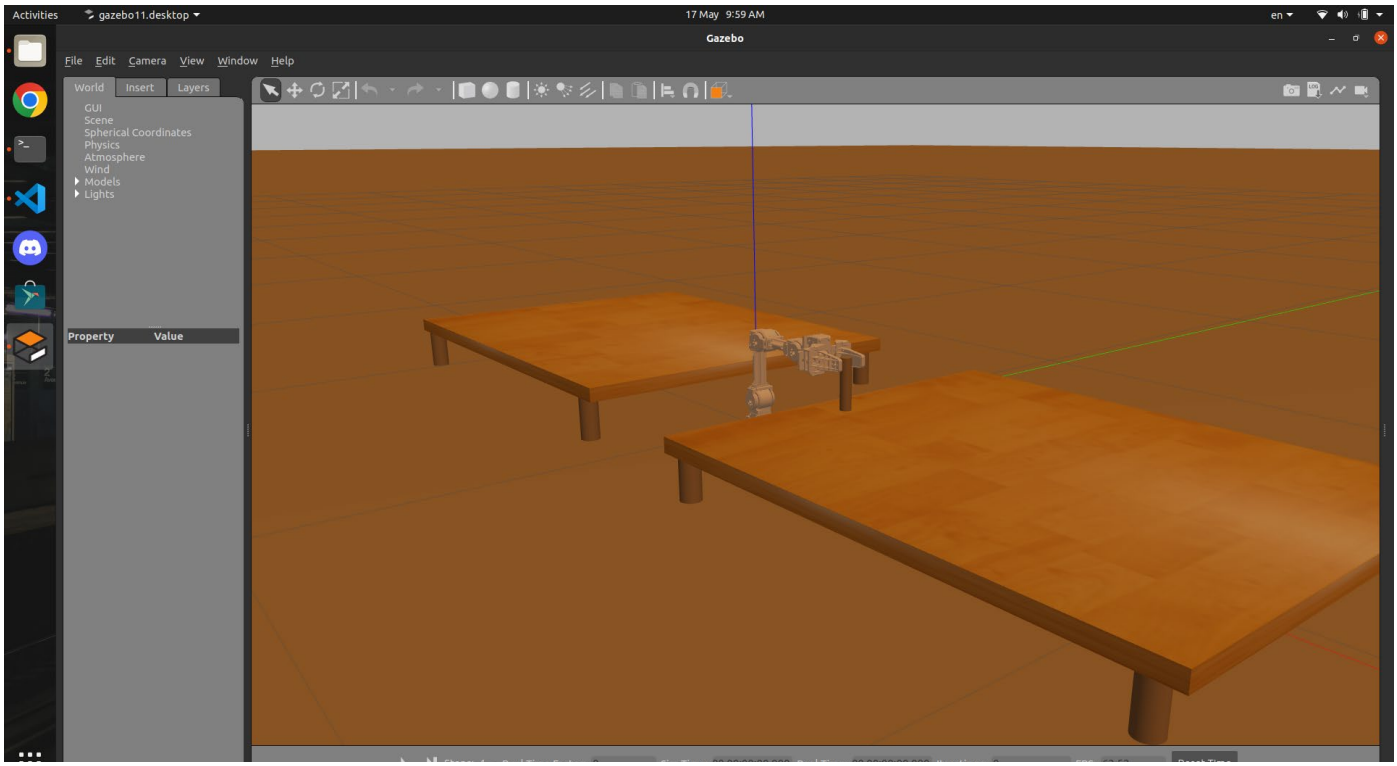


Figure 6: Gazebo Environment

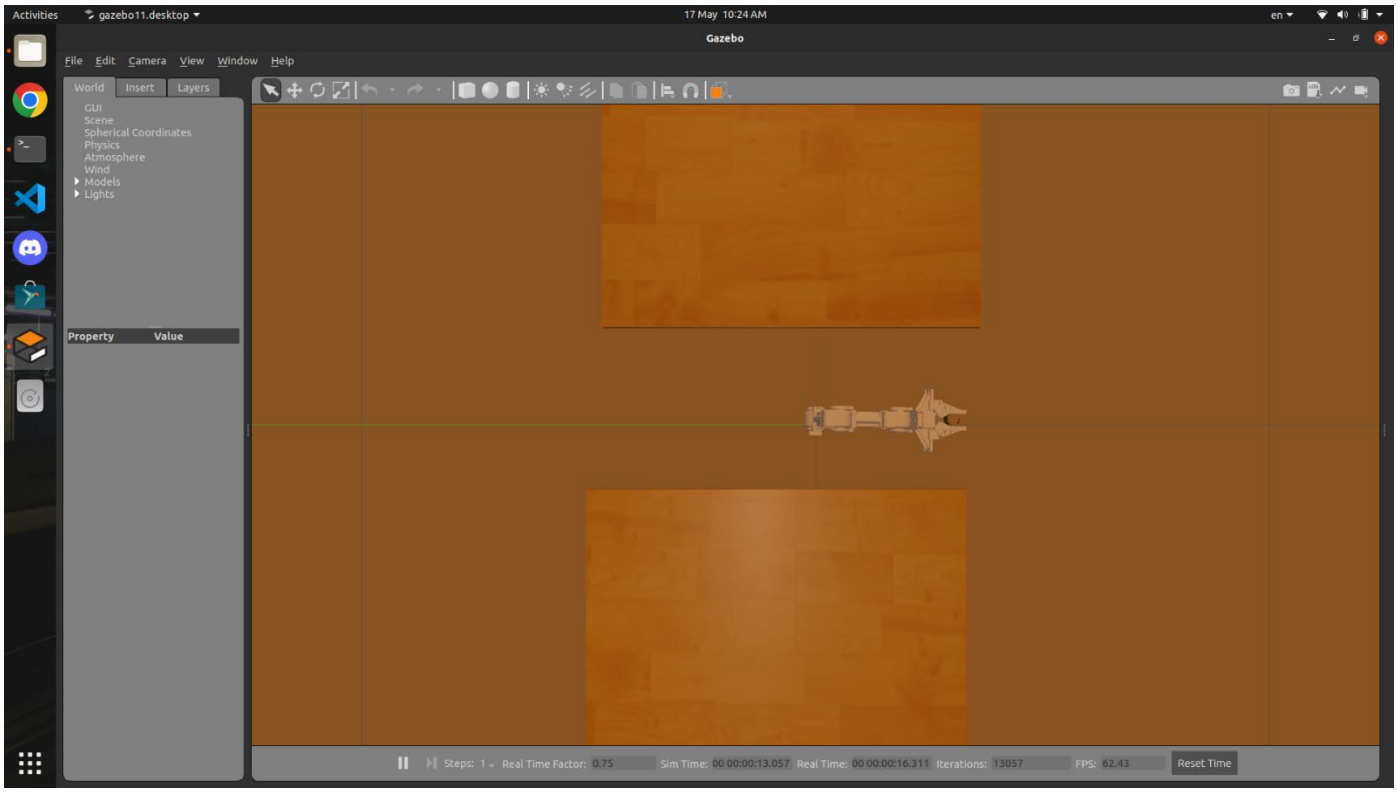


Figure 9:Robot is moving in Gazebo.

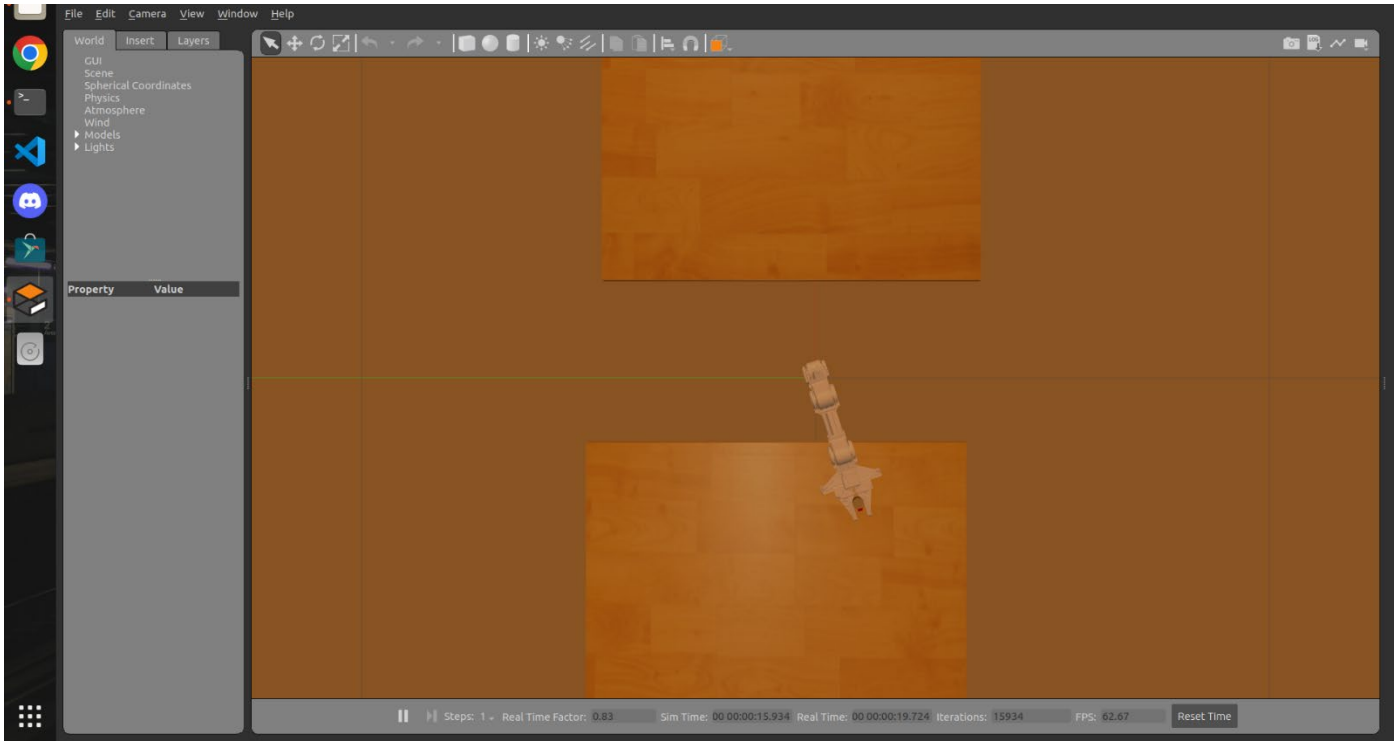


Figure 8: Robot reached in Gazebo.

5. Simulation Video Link:

You can watch the video of our project here.