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.

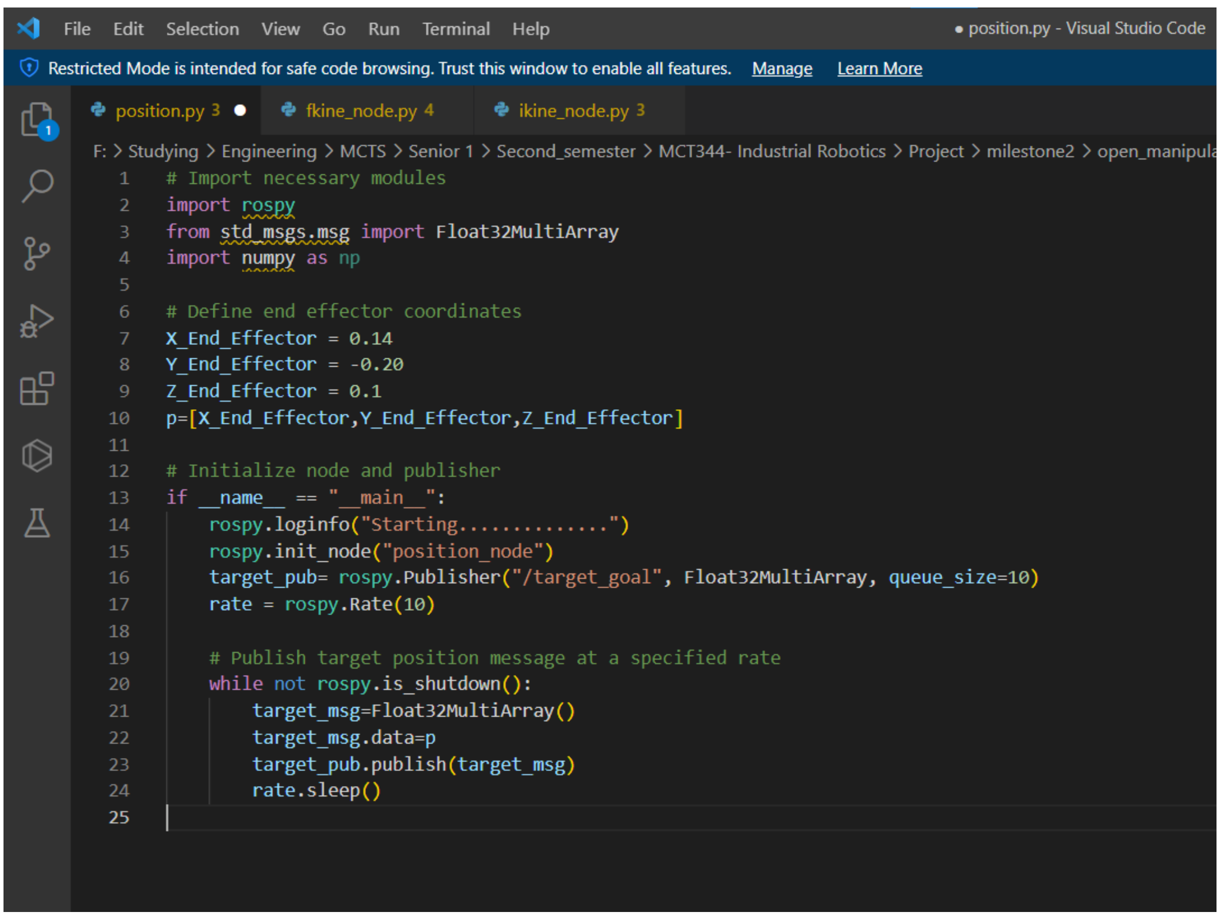
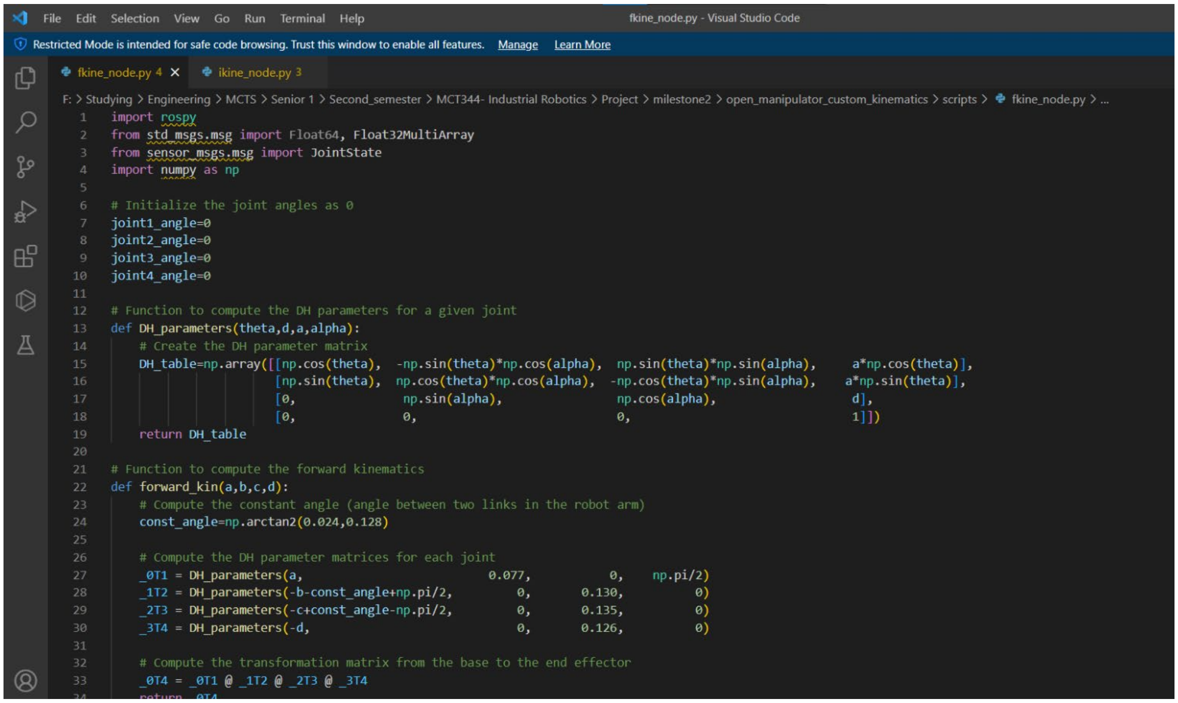


Figure 1: position node

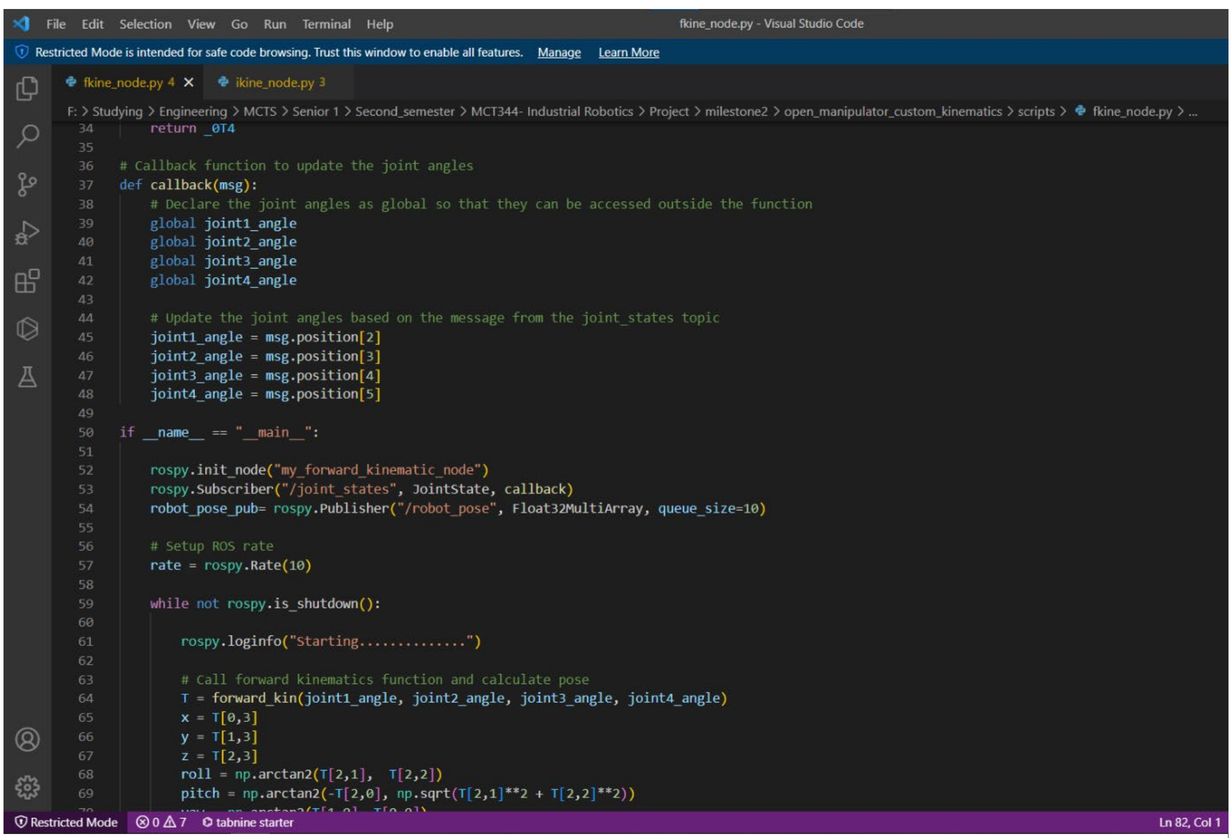
2. fkine node:

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.



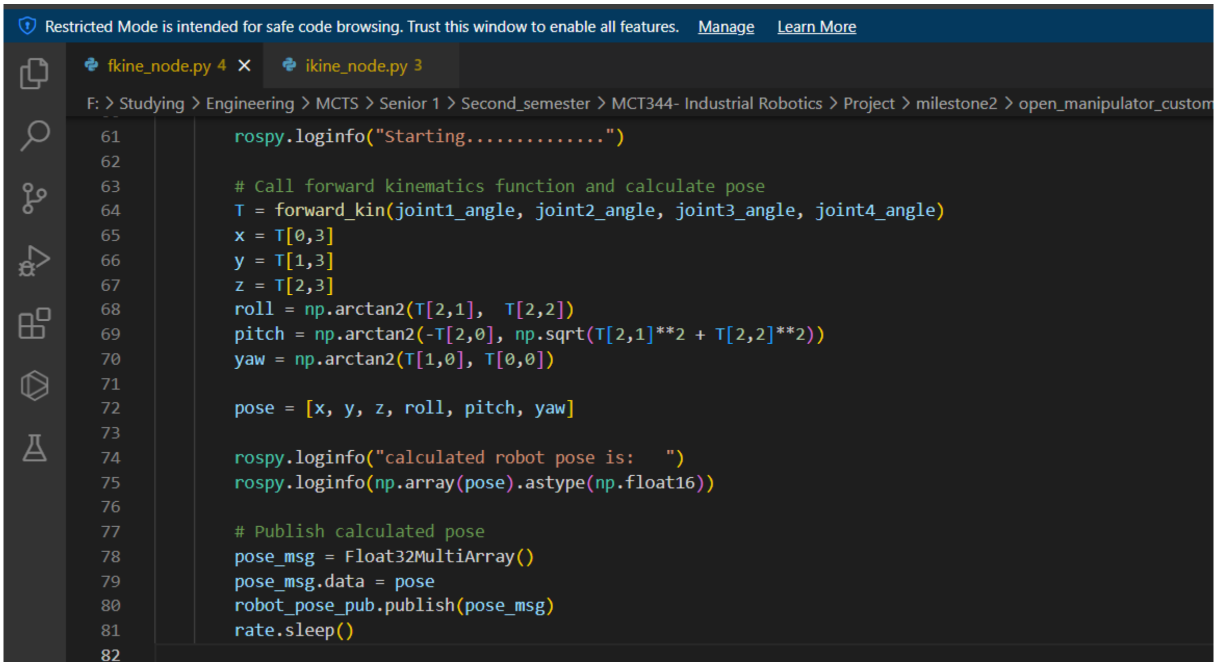
```python
import rospy
from std_msgs.msg import Float64, Float32MultiArray
from sensor_msgs.msg import JointState
import numpy as np

# Initialize the joint angles as 0
joint1_angle=0
joint2_angle=0
joint3_angle=0
joint4_angle=0

# Function to compute the DH parameters for a given joint
def DH_parameters(theta,d,a,alpha):
    # Create the DH parameter matrix
    DH_table=np.array([[np.cos(theta),  -np.sin(theta)*np.cos(alpha),  np.sin(theta)*np.sin(alpha),     a*np.cos(theta)],
                       [np.sin(theta),  np.cos(theta)*np.cos(alpha),  -np.cos(theta)*np.sin(alpha),    a*np.sin(theta)],
                       [0,               np.sin(alpha),                np.cos(alpha),                   d],
                       [0,               0,                            0,                               1]])
    return DH_table

# Function to compute the forward kinematics
def forward_kin(a,b,c,d):
    # Compute the constant angle (angle between two links in the robot arm)
    const_angle=np.arctan2(0.024,0.128)

    # Compute the DH parameter matrices for each joint
    _0T1 = DH_parameters(a,                          0.077,         0,    np.pi/2)
    _1T2 = DH_parameters(-b-const_angle+np.pi/2,         0,     0.130,          0)
    _2T3 = DH_parameters(-c+const_angle-np.pi/2,         0,     0.135,          0)
    _3T4 = DH_parameters(-d,                             0,     0.126,          0)

    # Compute the transformation matrix from the base to the end effector
    _0T4 = _0T1 @ _1T2 @ _2T3 @ _3T4
```

Figure 3: fkine node part1



```python
    return _0T4

# Callback function to update the joint angles
def callback(msg):
    # Declare the joint angles as global so that they can be accessed outside the function
    global joint1_angle
    global joint2_angle
    global joint3_angle
    global joint4_angle

    # Update the joint angles based on the message from the joint_states topic
    joint1_angle = msg.position[2]
    joint2_angle = msg.position[3]
    joint3_angle = msg.position[4]
    joint4_angle = msg.position[5]

if __name__ == "__main__":

    rospy.init_node("my_forward_kinematic_node")
    rospy.Subscriber("/joint_states", JointState, callback)
    robot_pose_pub= rospy.Publisher("/robot_pose", Float32MultiArray, queue_size=10)

    # Setup ROS rate
    rate = rospy.Rate(10)

    while not rospy.is_shutdown():

        rospy.loginfo("Starting.............")

        # Call forward kinematics function and calculate pose
        T = forward_kin(joint1_angle, joint2_angle, joint3_angle, joint4_angle)
        x = T[0,3]
        y = T[1,3]
        z = T[2,3]
        roll = np.arctan2(T[2,1],  T[2,2])
        pitch = np.arctan2(-T[2,0], np.sqrt(T[2,1]**2 + T[2,2]**2))
```

Figure 2: fkine node part2




```python
        rospy.loginfo("Starting.............")

        # Call forward kinematics function and calculate pose
        T = forward_kin(joint1_angle, joint2_angle, joint3_angle, joint4_angle)
        x = T[0,3]
        y = T[1,3]
        z = T[2,3]
        roll = np.arctan2(T[2,1],  T[2,2])
        pitch = np.arctan2(-T[2,0], np.sqrt(T[2,1]**2 + T[2,2]**2))
        yaw = np.arctan2(T[1,0], T[0,0])

        pose = [x, y, z, roll, pitch, yaw]

        rospy.loginfo("calculated robot pose is:   ")
        rospy.loginfo(np.array(pose).astype(np.float16))

        # Publish calculated pose
        pose_msg = Float32MultiArray()
        pose_msg.data = pose
        robot_pose_pub.publish(pose_msg)
        rate.sleep()
```

Figure 4: fkine node part3

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

```
kirolos@kirolos:~$ rostopic info /joint_states
Type: sensor_msgs/JointState

Publishers:
 * /gazebo (http://kirolos:37373/)

Subscribers:
 * /control_gui (http://kirolos:46419/)
 * /my_forward_kinematic_node (http://kirolos:42803/)


kirolos@kirolos:~$ rosmsg info sensor_msgs/JointState
std_msgs/Header header
  uint32 seq
  time stamp
  string frame_id
string[] name
float64[] position
float64[] velocity
```

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).

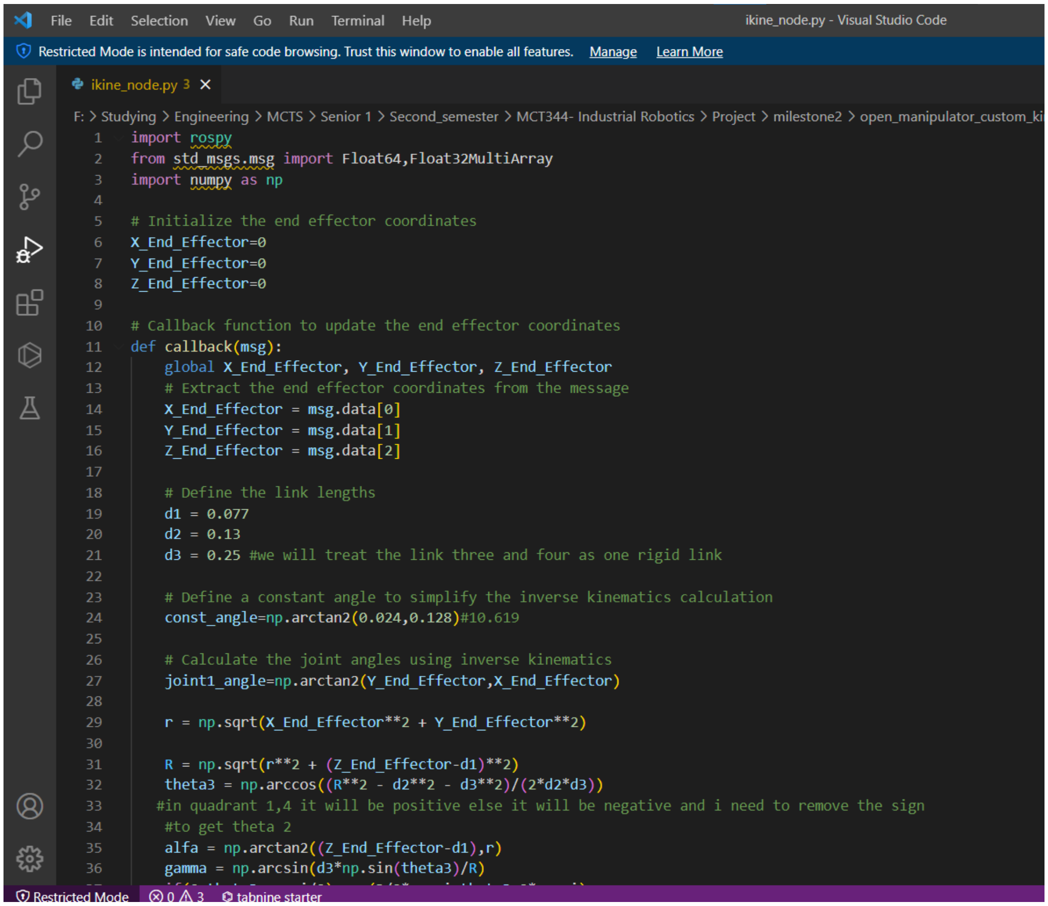


Figure 7: ikine node part 1

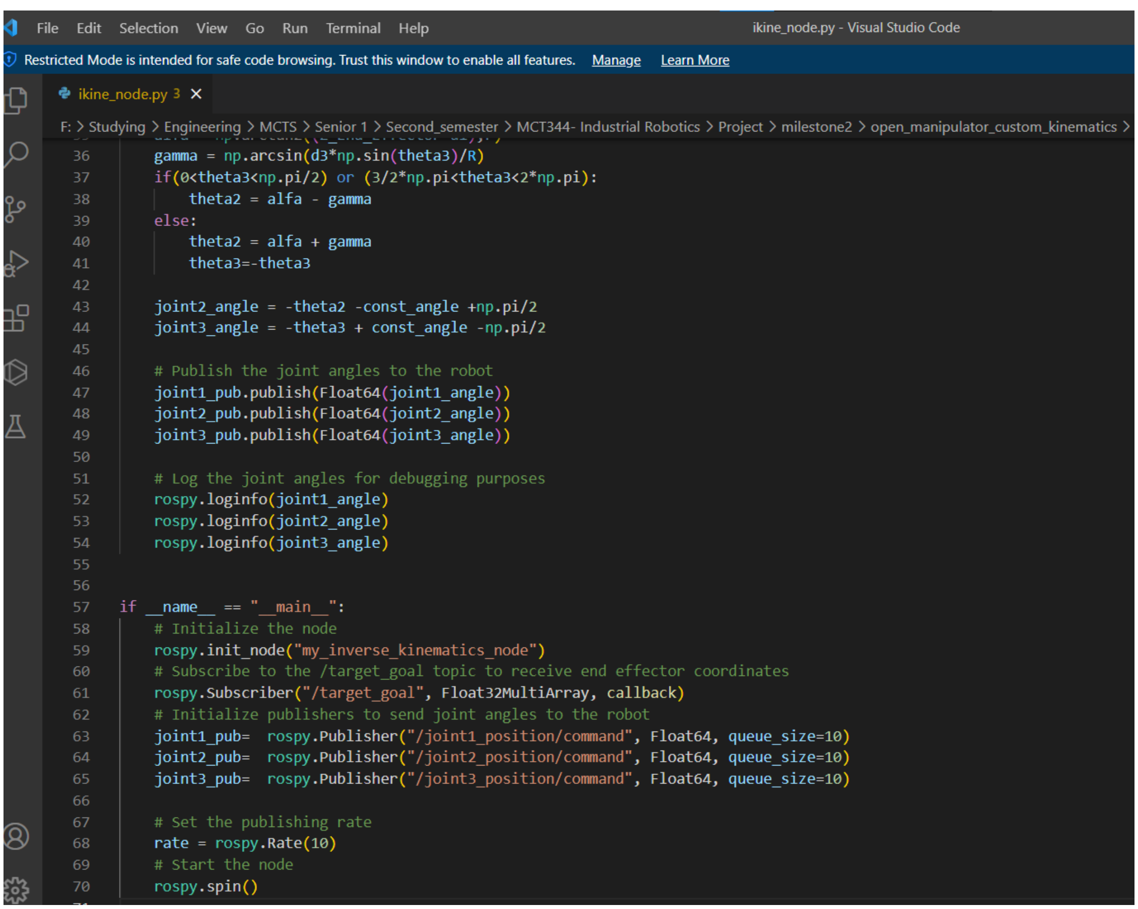


Figure 6: ikine node part 2

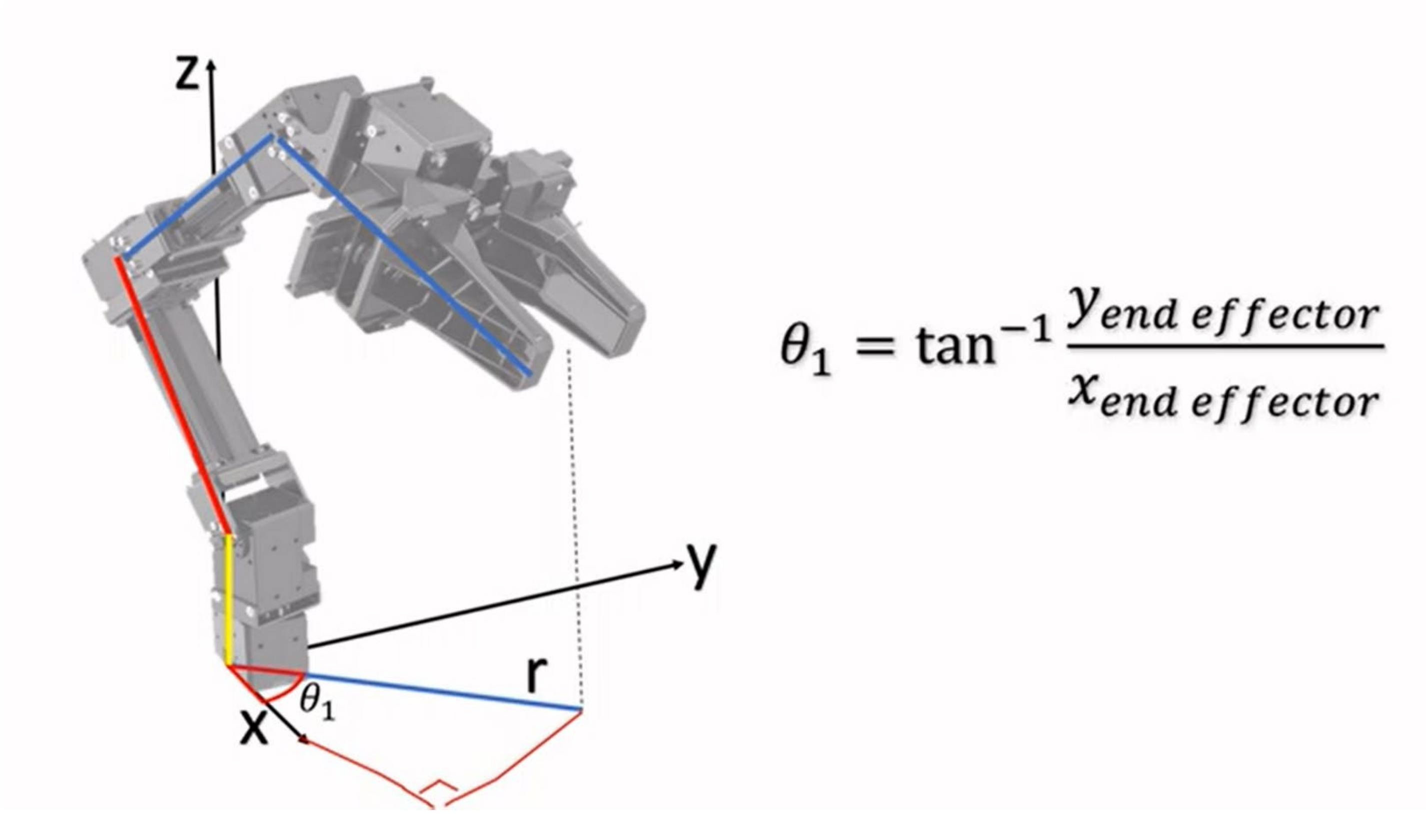


Figure 9: geometrical explain p1

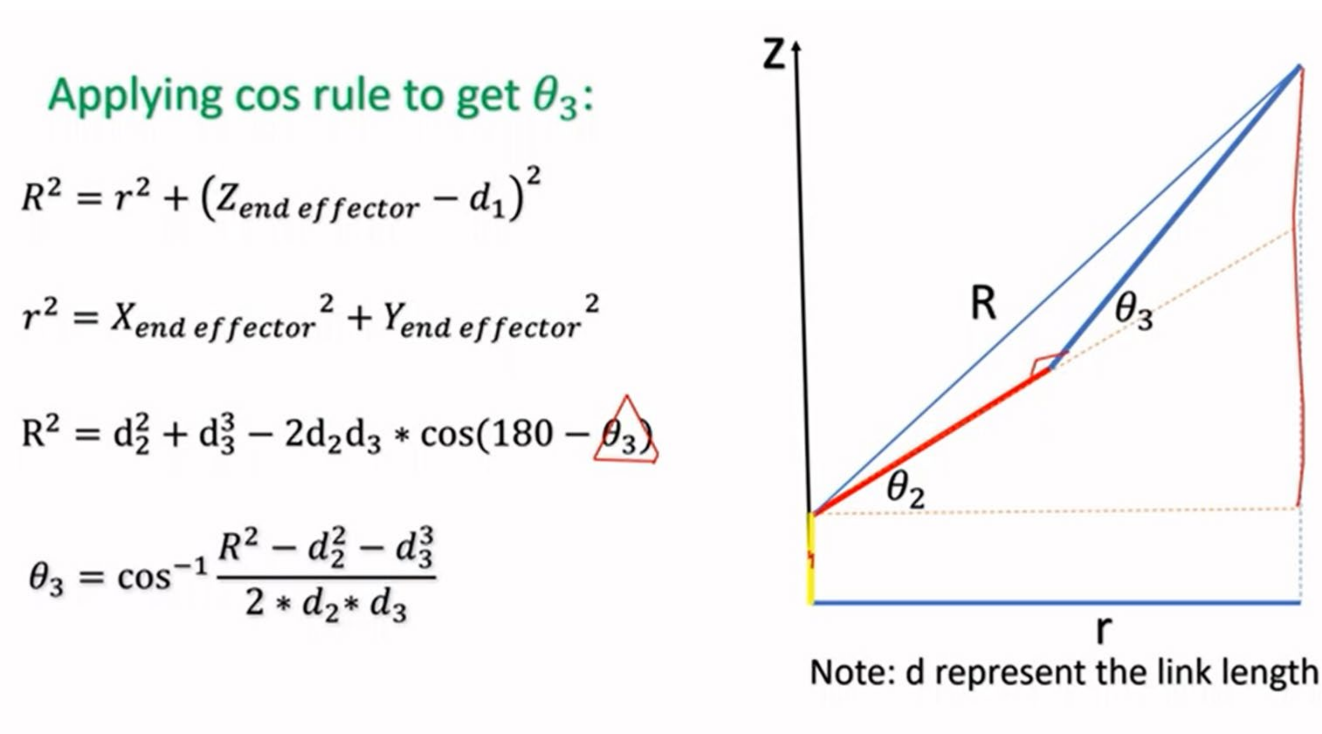


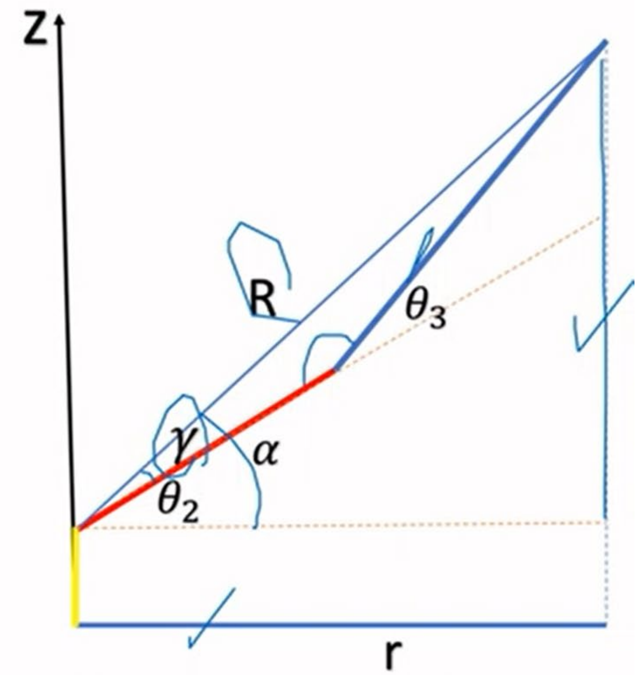
Figure 8: geometrical explain p2

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:

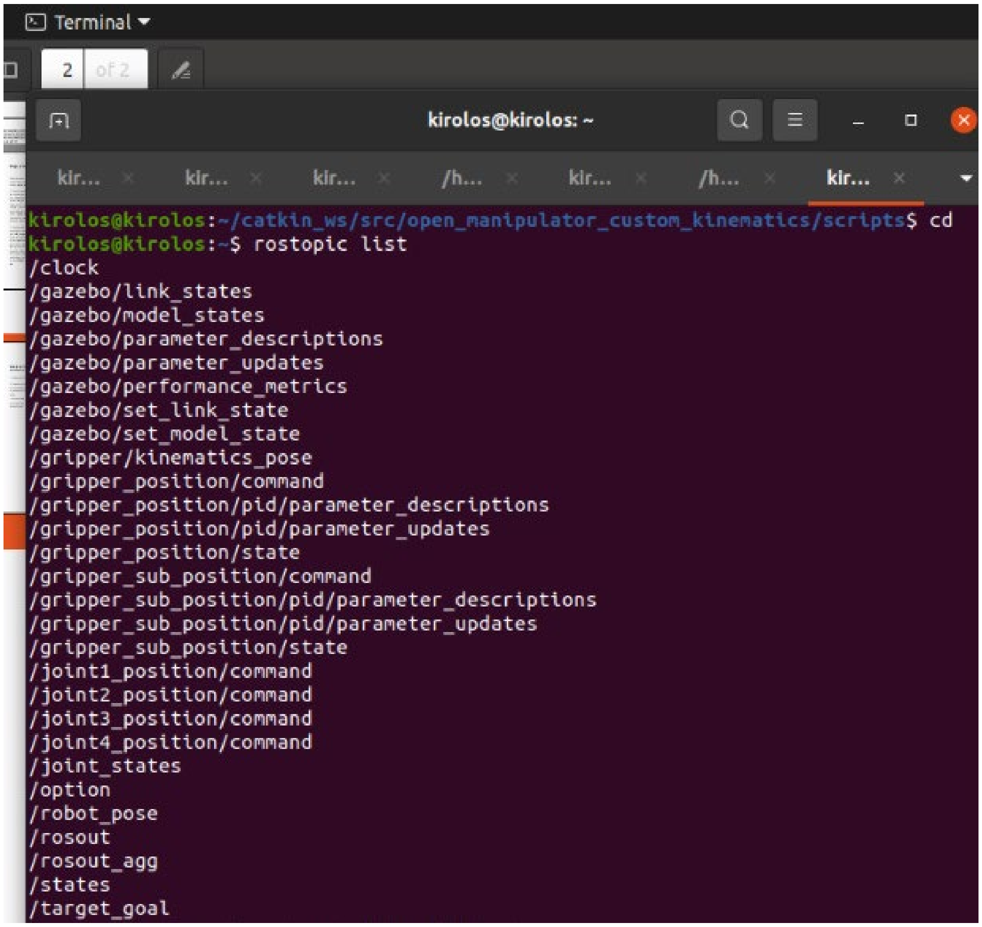


Figure 11: Rostopic list

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

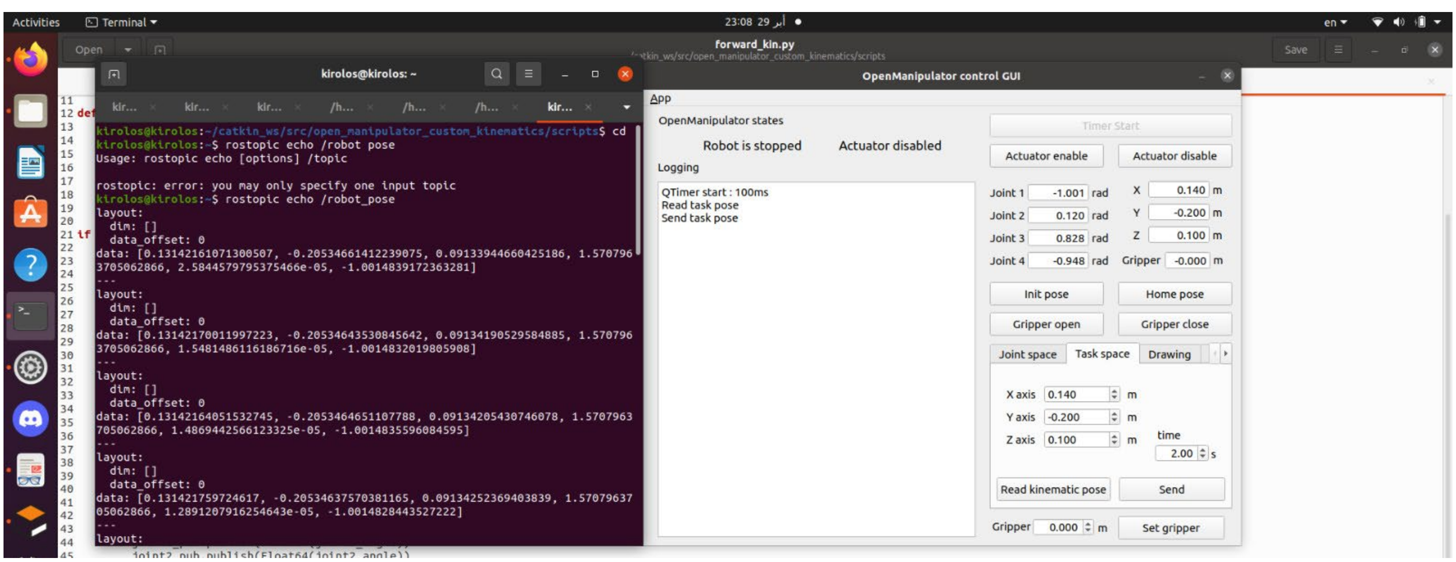


Figure 12: topic echoing robot pose

`

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

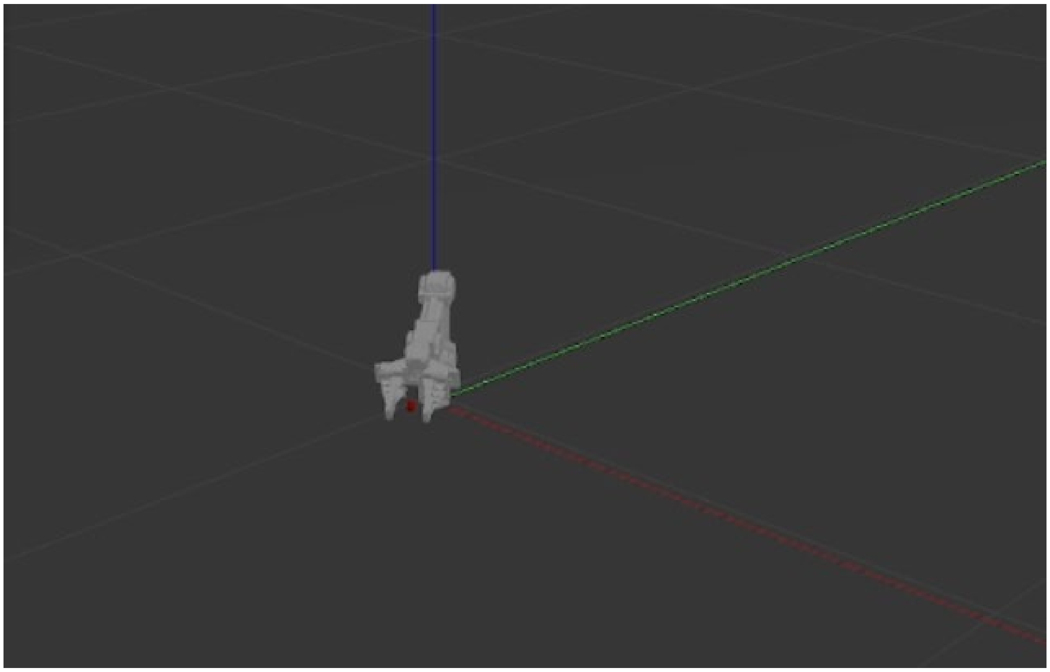


Figure 13: robot after published the required position

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

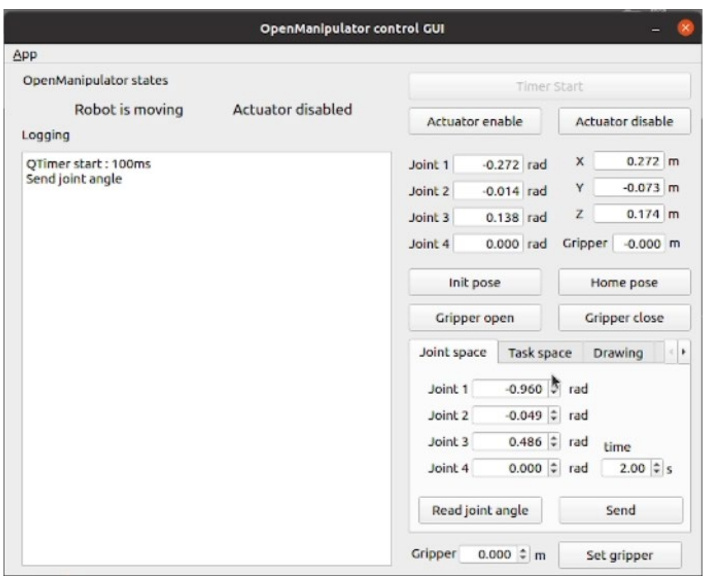


Figure 15: sending the calculated angles from the ikine node.

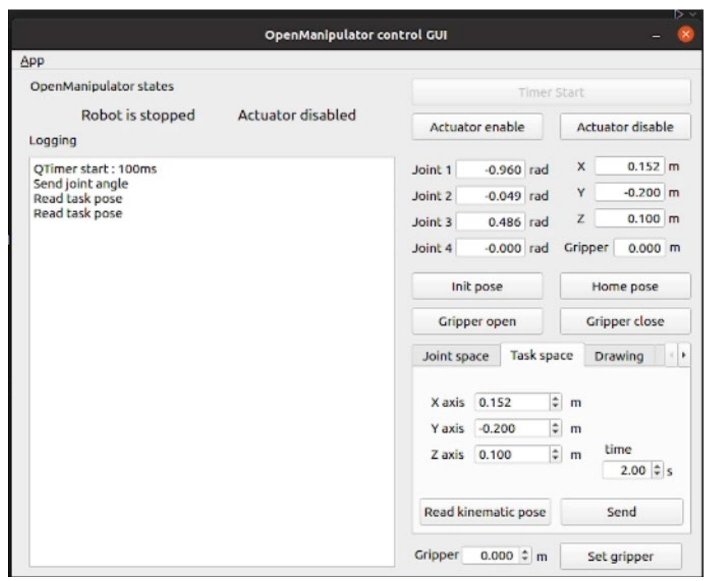


Figure 14:reading the position to compare with the target position node.

RQT Graph:

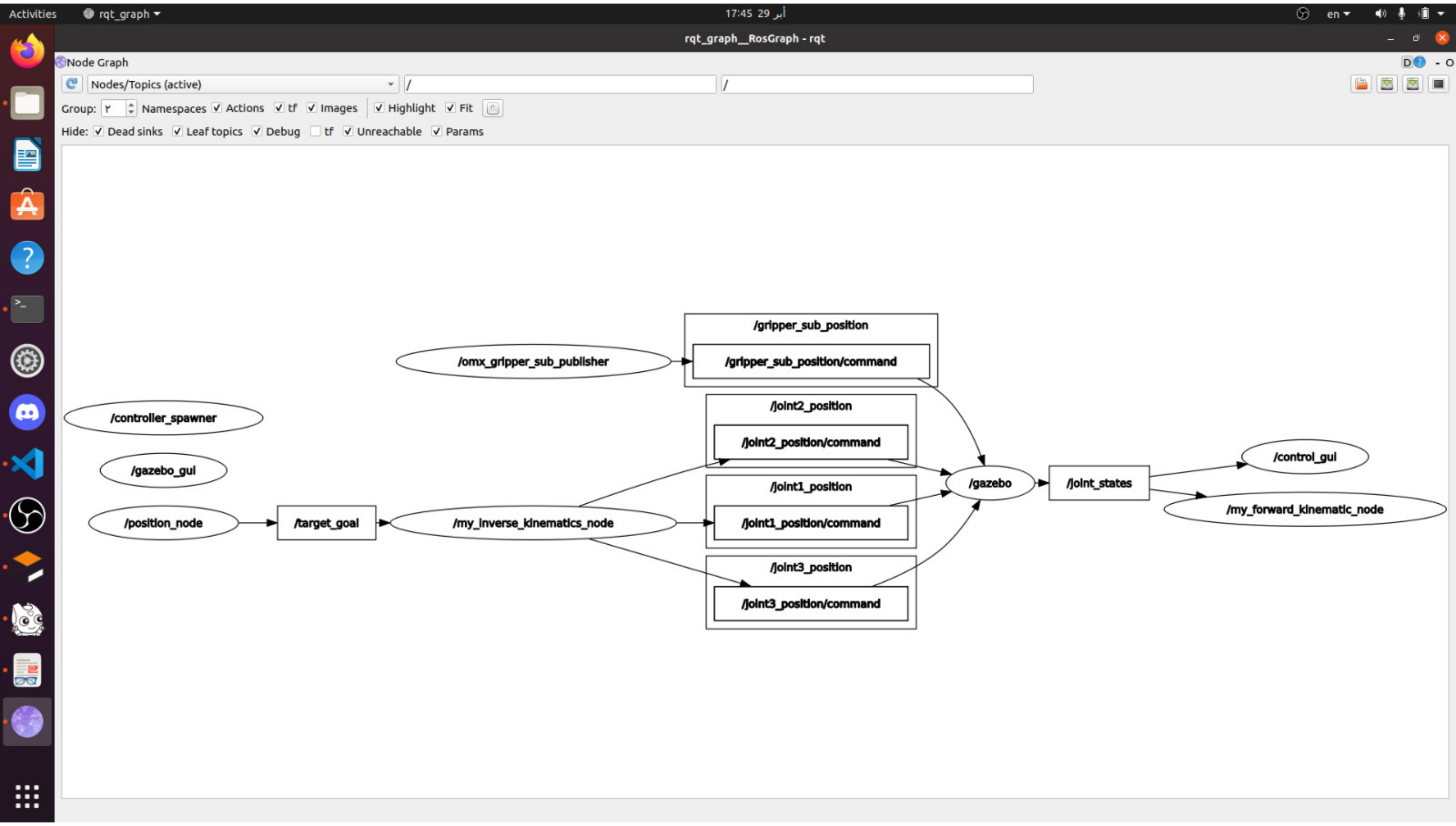


Figure 16: RQT graph

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)]()