MCT 344 Industrial Robotics

Milestone 2

Amal Amr AbdelHamid Zewita

19P1504

Team 35

Table of Contents

List of Figures

1.0 WORKSPACE CREATION

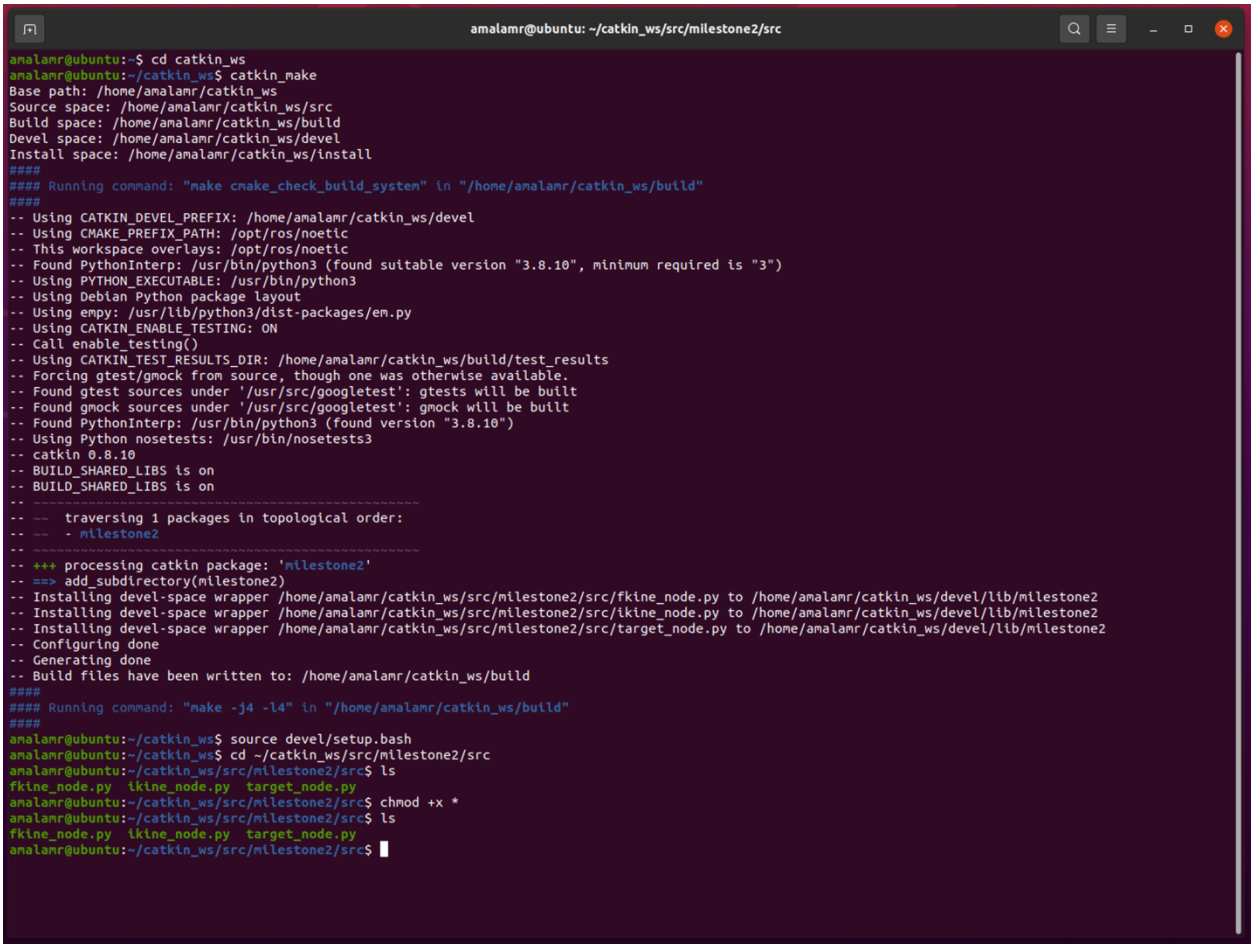

Figure 1: Workspace Creation

2.0 FORWARD KINEMATICS

2.1 Terminal & Gazebo Work

Figure 2: Running Roscore

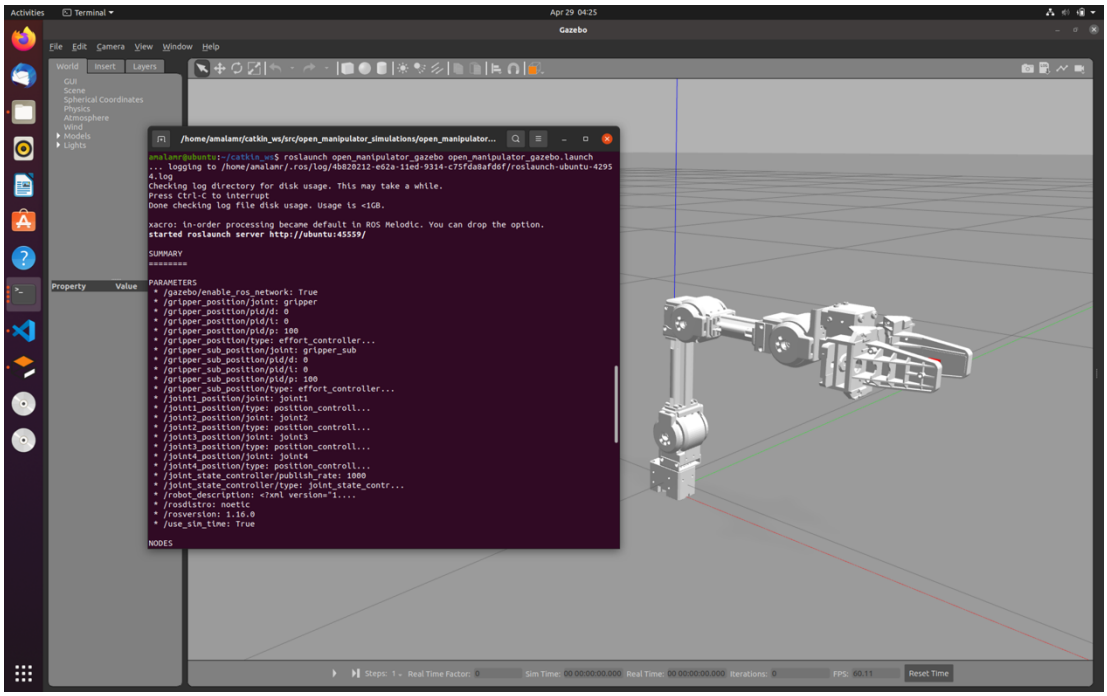

Figure 3: Launching Gazebo

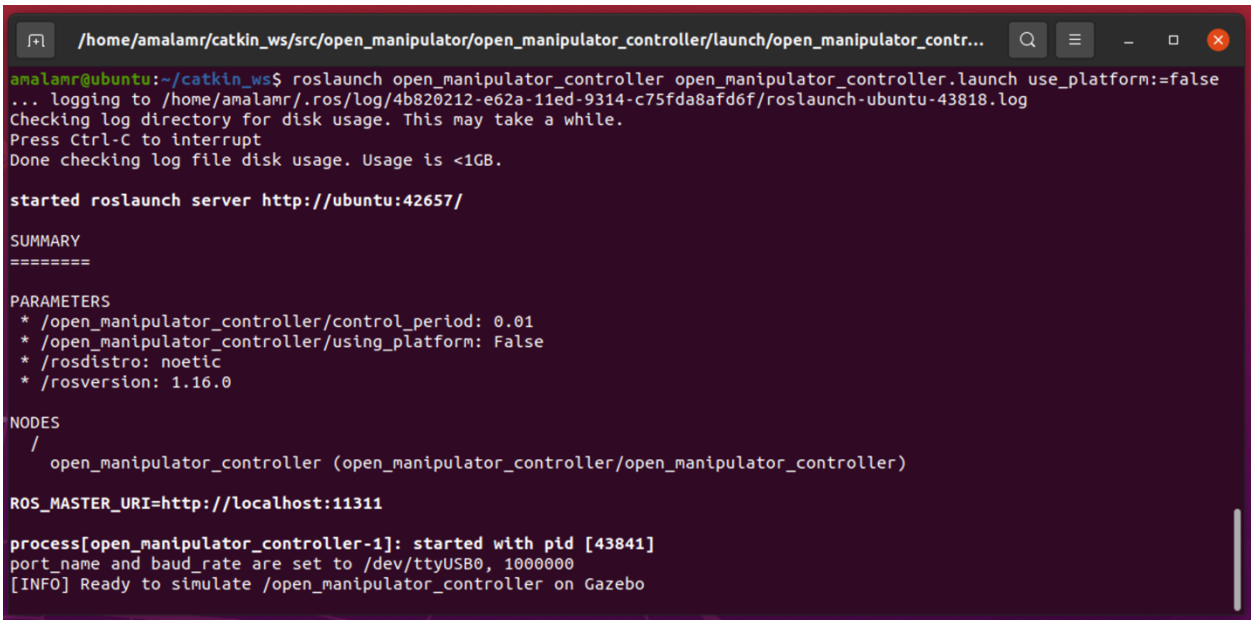

Figure 5: Launching GUI 1

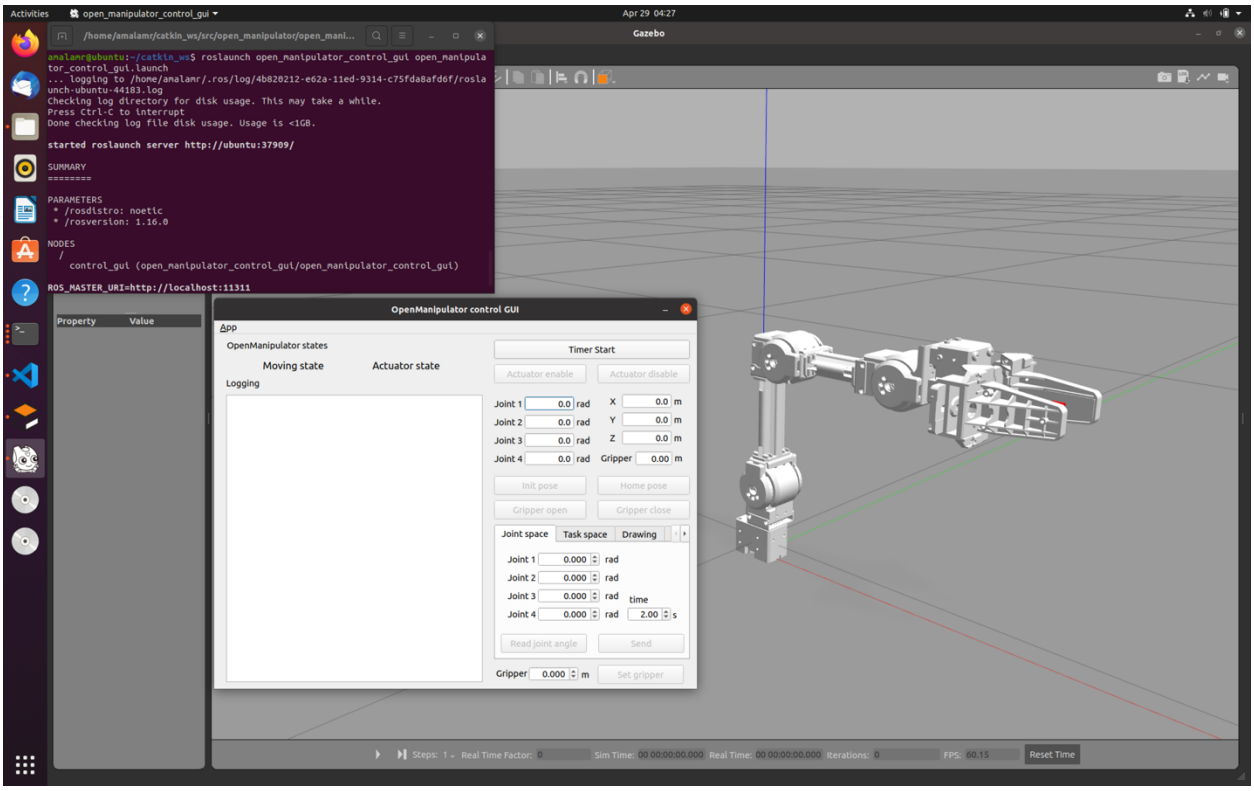

Figure 4: Launching GUI 2

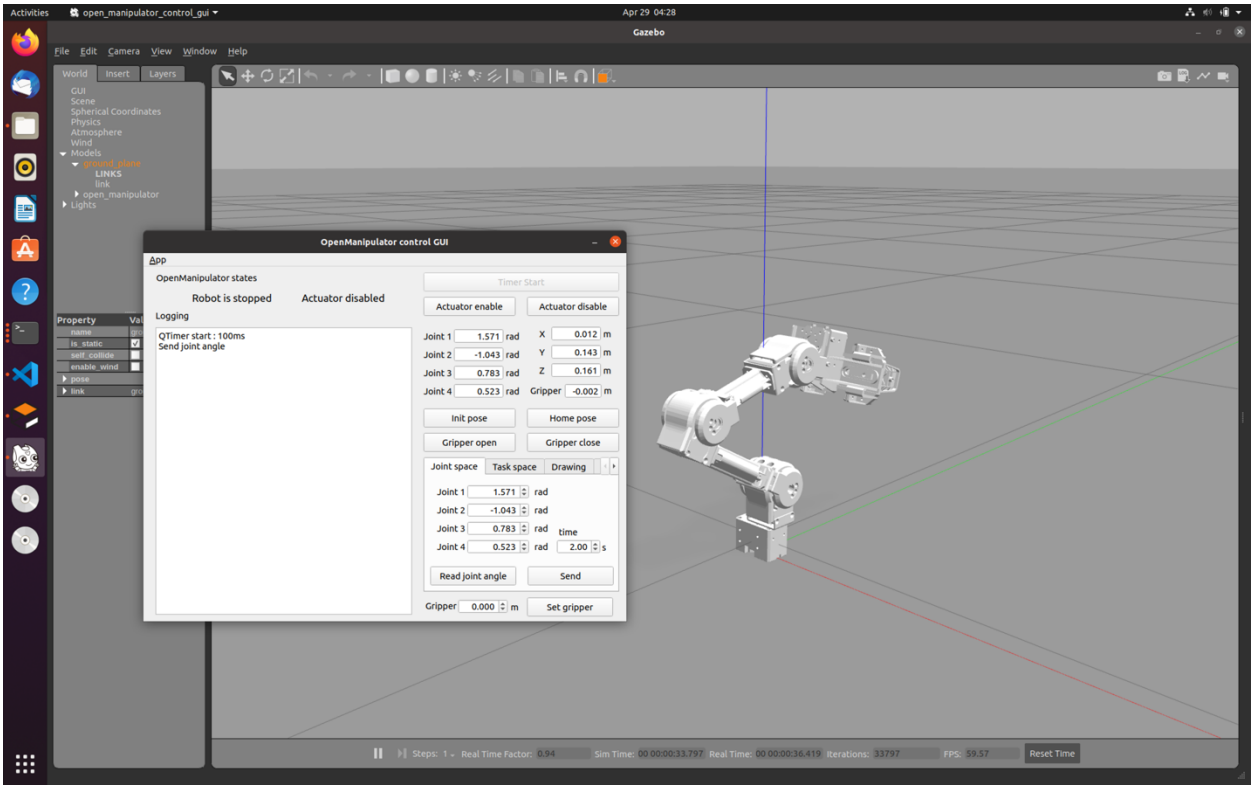

Figure 6: Moving Robot through GUI

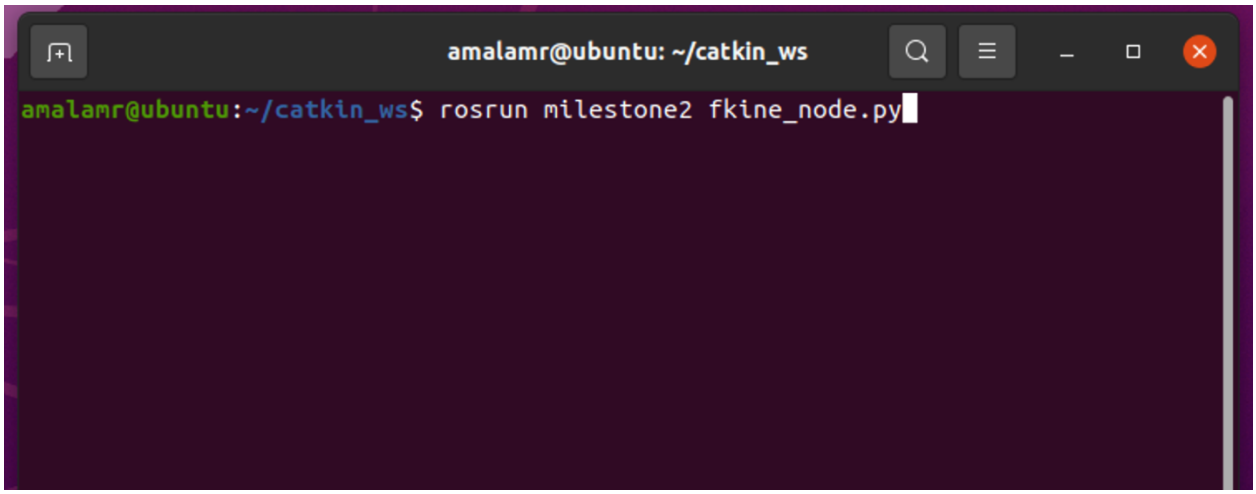

Figure 7: Running Forward kinematics node.

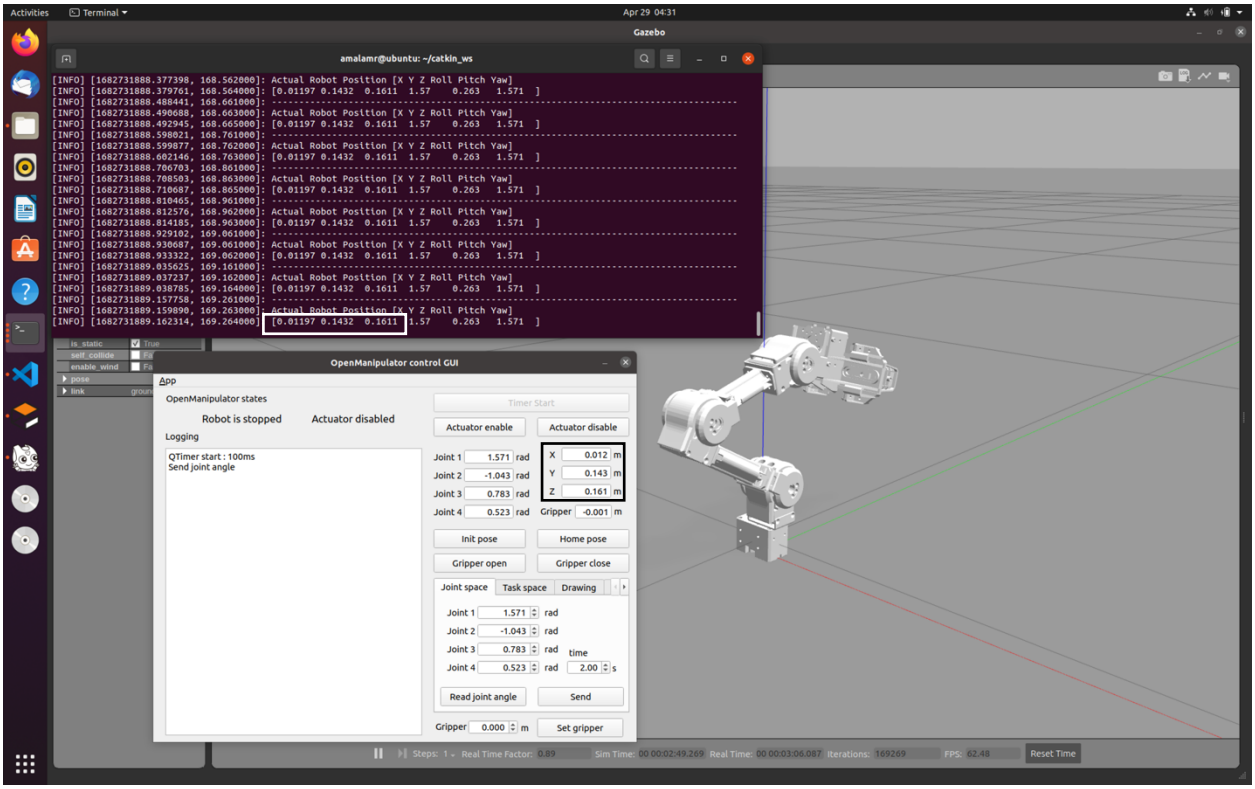

Figure 9: The Forward Kinematics Node Showing the same X, Y and Z as the GUI

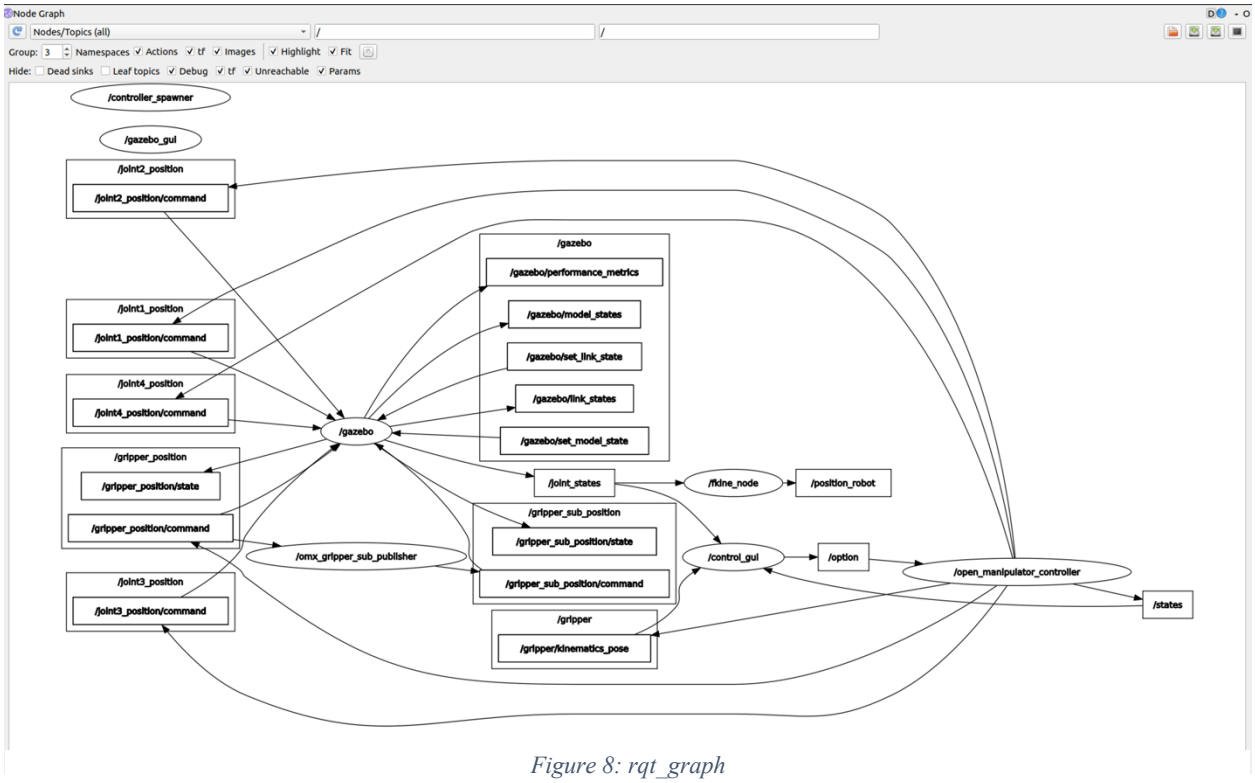

Figure 8: rqt_graph

2.2 Forward Kinematics Node Code

```
#!/usr/bin/env python3
import rospy
import numpy as np
from std_msgs.msg import Float32MultiArray
from sensor_msgs.msg import JointState

# DH parameter function which returns the DH matrix
def std_DH(theta, d, a, alpha):
    DH = 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

def joints_clbk(recived_msg):
    # angle and matrix as golabal variables
    global theta1, theta2, theta3, theta4, FinalMat
    # this will be called each time the topic receives
    theta1 = recived_msg.position[2]
    theta2 = recived_msg.position[3]
    theta3 = recived_msg.position[4]
    theta4 = recived_msg.position[5]

if __name__ == '__main__':

    # initalization the variable with 0
    theta0, theta1, theta2, theta3, theta4, FinalMat = 0, 0, 0, 0, 0, 0

    # create and intializing node
    rospy.init_node("fkine_node")

    # the translation matrix for the offset of the first joint with the
origin
    translation = np.array([ [1, 0, 0, 0.012],
                             [0, 1, 0,     0],
                             [0, 0, 1,     0],
```

```
                              [0, 0, 0,     1] ])

    # position publisher
    position_pub = rospy.Publisher("/position_robot", Float32MultiArray,
queue_size = 10)

    # actual joint angles
    rospy.Subscriber("/joint_states", JointState, joints_clbk)

    # the angle between 2nd and 3rd joint
    theta0 = np.arctan2(0.024, 0.128)

    rate = rospy.Rate(10)


    while not rospy.is_shutdown():

        # calcualting the DH matrices (theta considering initial position)
        T01 = std_DH (theta1,                          0.077,         0,
np.pi / 2)
        T12 = std_DH (-theta2 - theta0 + (np.pi / 2),     0,     0.130,
0)
        T23 = std_DH (-theta3 + theta0 - (np.pi / 2),     0,     0.124,
0)
        T3H = std_DH (-theta4,                            0,     0.126,
0)

        # the transformation matrix from the end effector to the origin
        FinalMat = translation @ T01 @ T12 @ T23 @ T3H

        # getting the cordination
        pos_msg = Float32MultiArray()
        x = FinalMat[0, 3]
        y = FinalMat[1, 3]
        z = FinalMat[2, 3]

        # getting the orientation
        roll  = np.arctan2( FinalMat[2, 1],
FinalMat[2, 2])
        pitch = np.arctan2(-FinalMat[2, 0], np.sqrt(FinalMat[2, 1] ** 2 +
FinalMat[2, 2] ** 2))
        yaw   = np.arctan2( FinalMat[1, 0],
FinalMat[0, 0])
```

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

        # publishing the message
        pos_msg.data = pose
        position_pub.publish(pos_msg)
        rospy.loginfo("-----------------------------------------------------
---------------------------------")
        rospy.loginfo("Actual Robot Position [X Y Z Roll Pitch Yaw]")
        rospy.loginfo(np.array(pose).astype(np.float16))
        rate.sleep()
```

3.0 INVERSE KINEMATICS

3.1 Manual Calculation

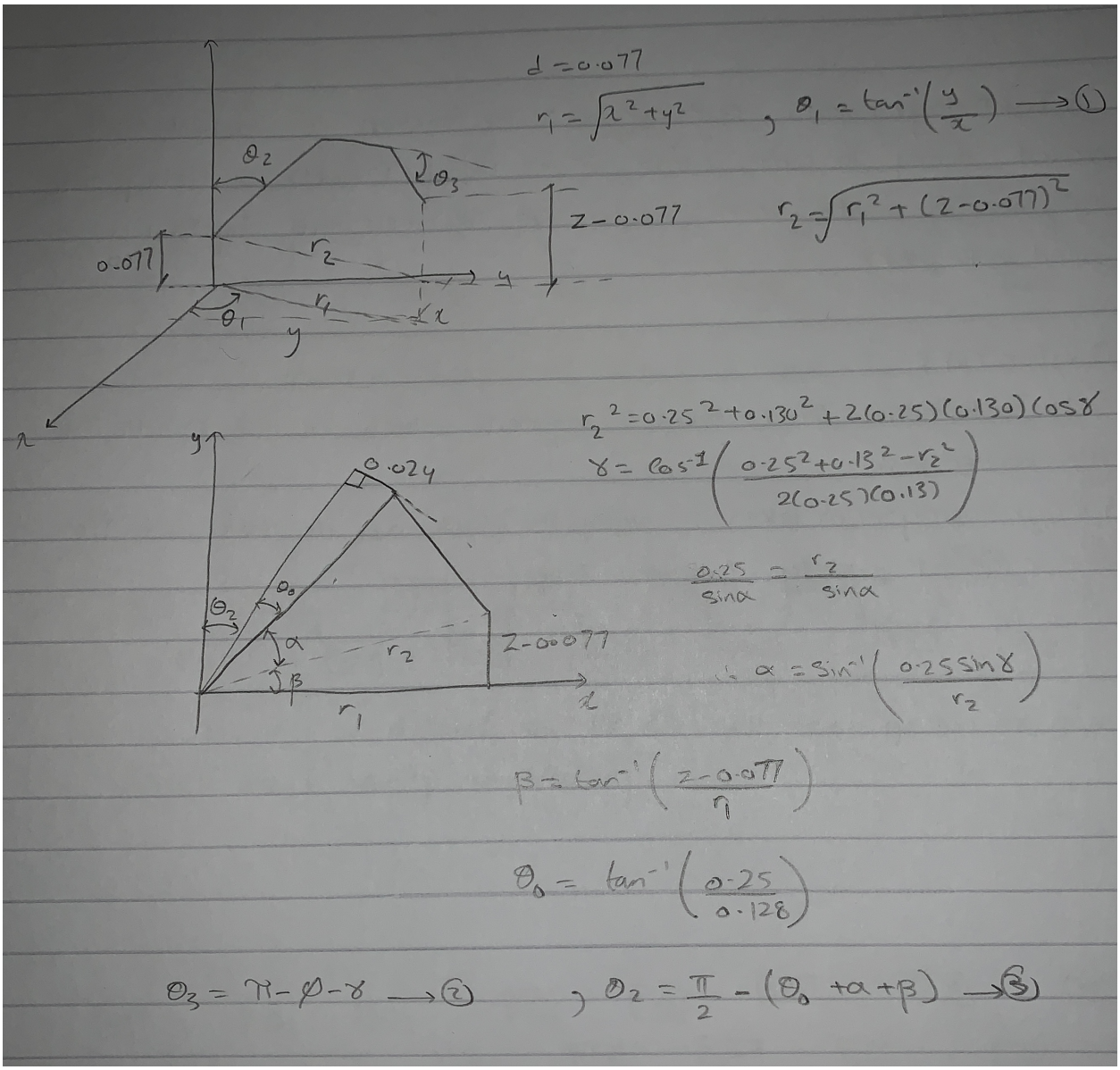

Figure 10: Geomatric Inverse Kinamtics Method

3.2 Terminal & Gazebo Work

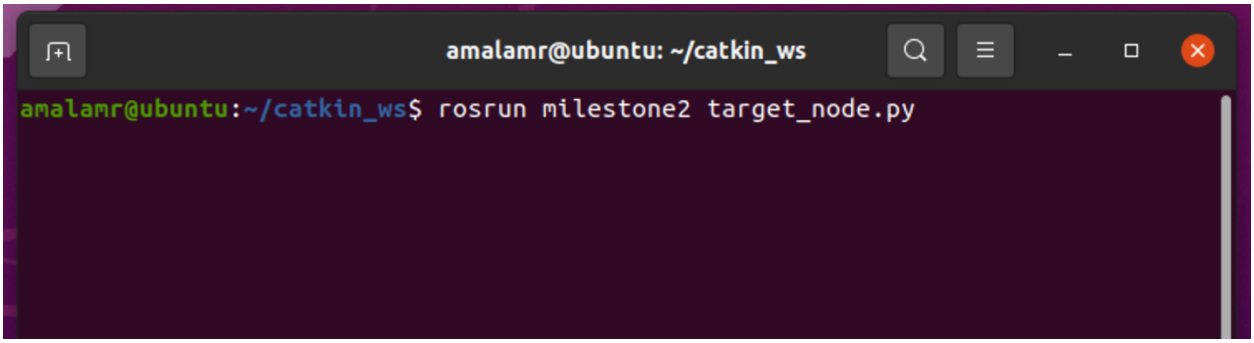

Figure 11: Running Target Node

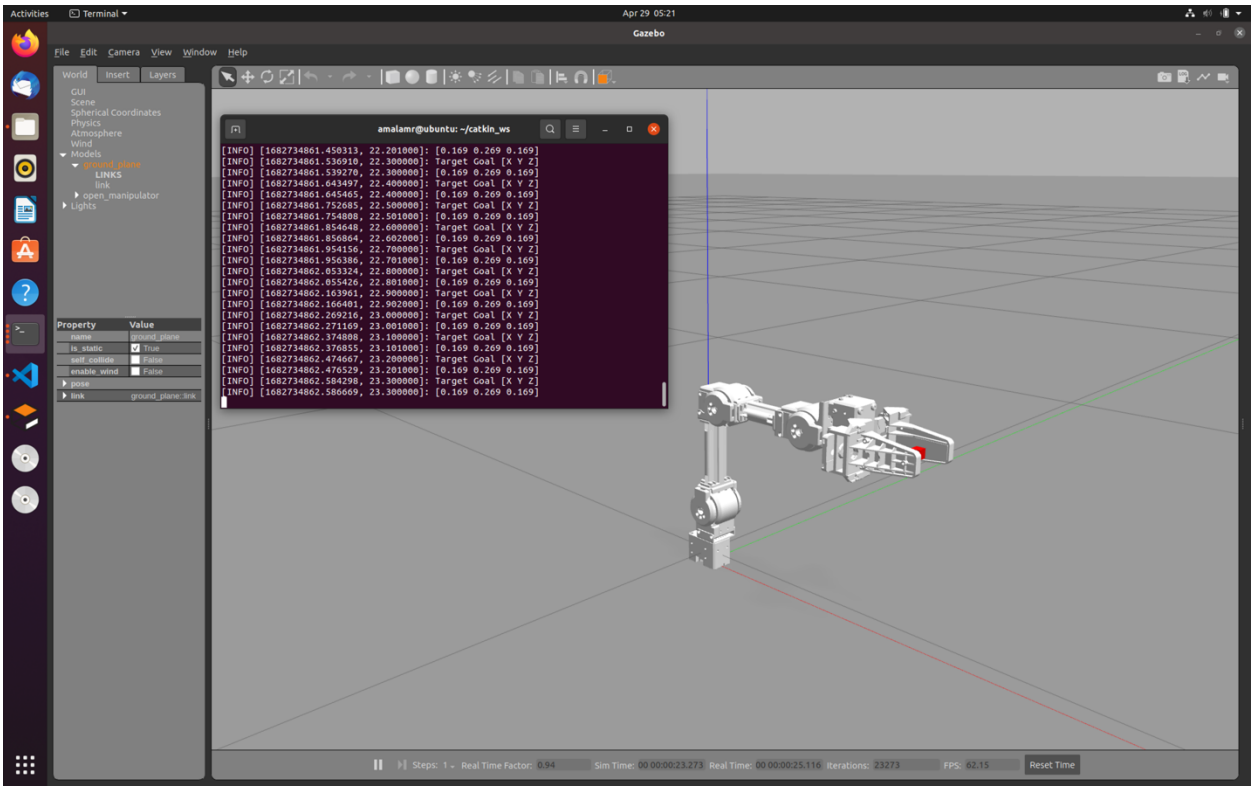

Figure 12: Target Node Running

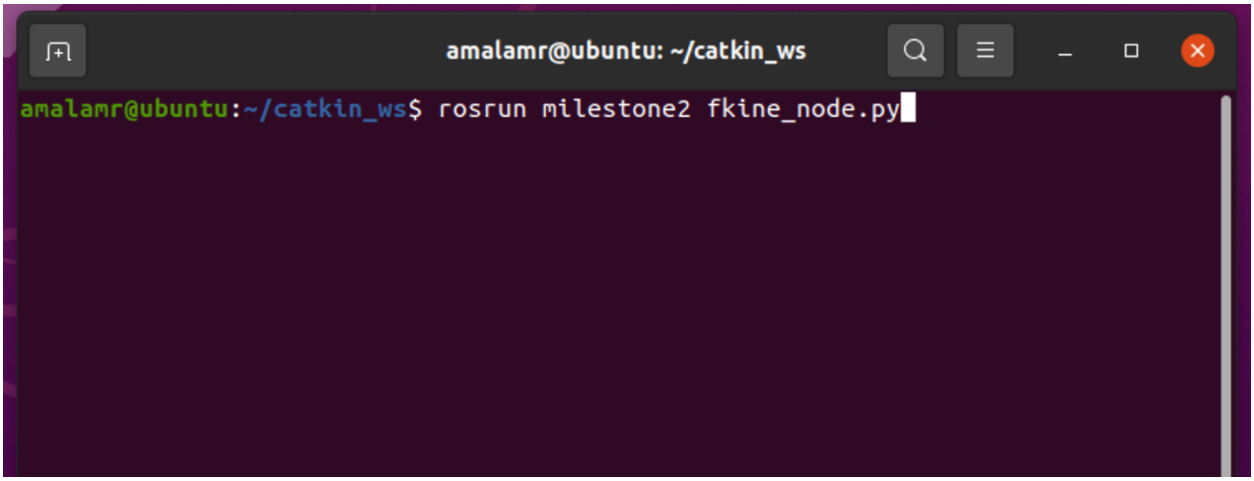

Figure 13: Running Inverse Kinematics Node

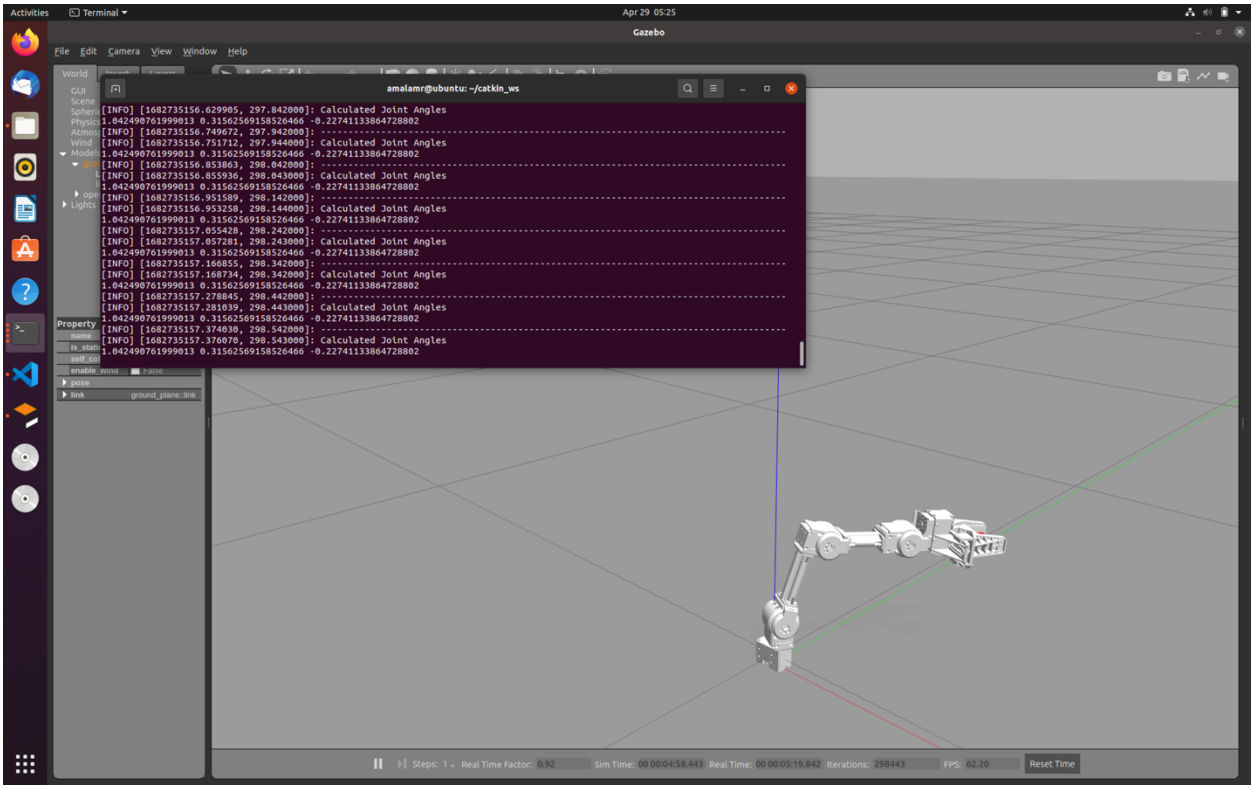

Figure 14: Inverse Kinematics Node Running

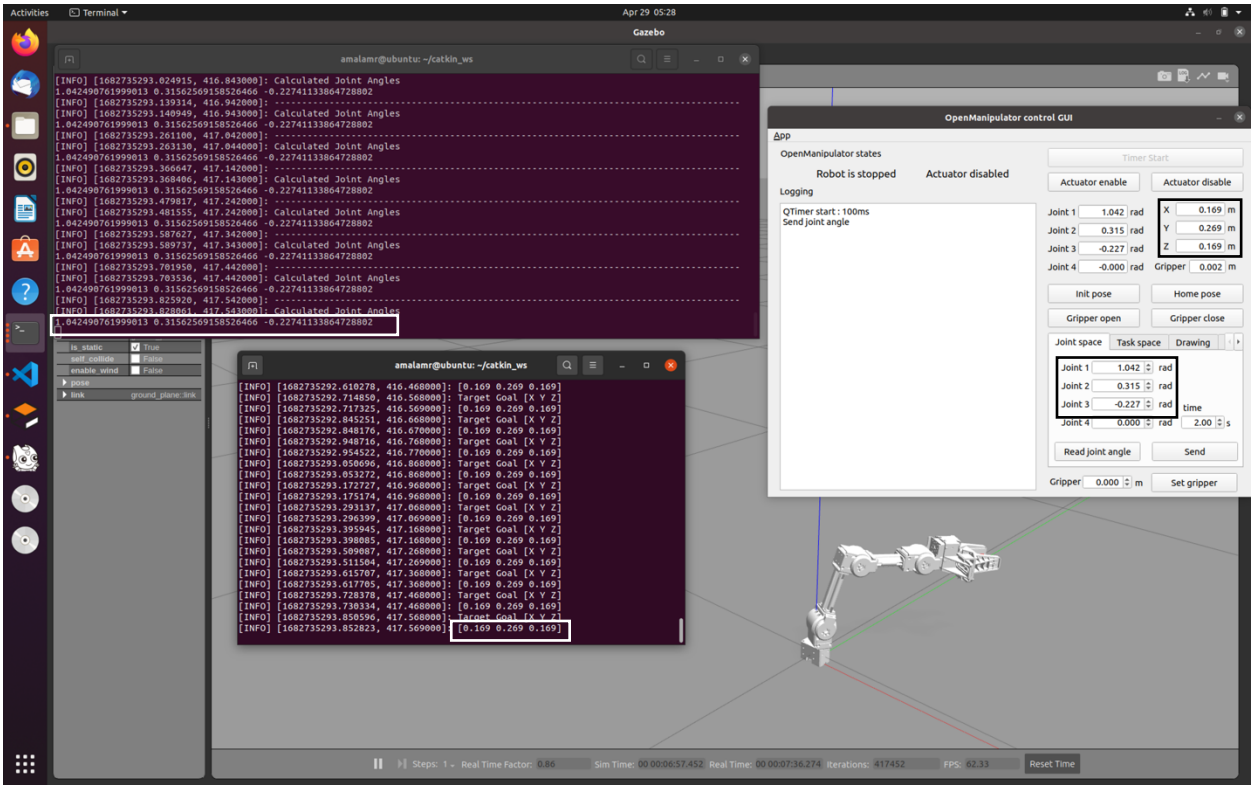

Figure 16: Using the GUI to verify that angles calculated by ikine_node

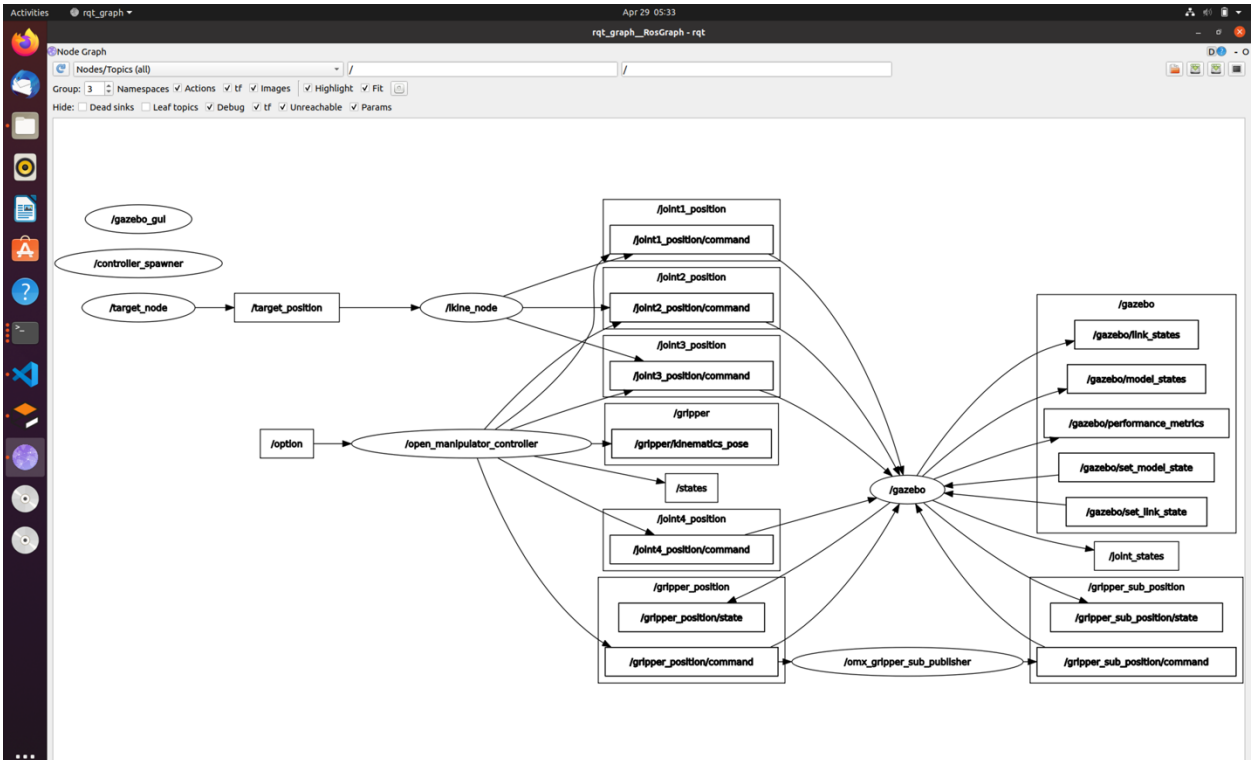

Figure 15: rqt_graph without GUI

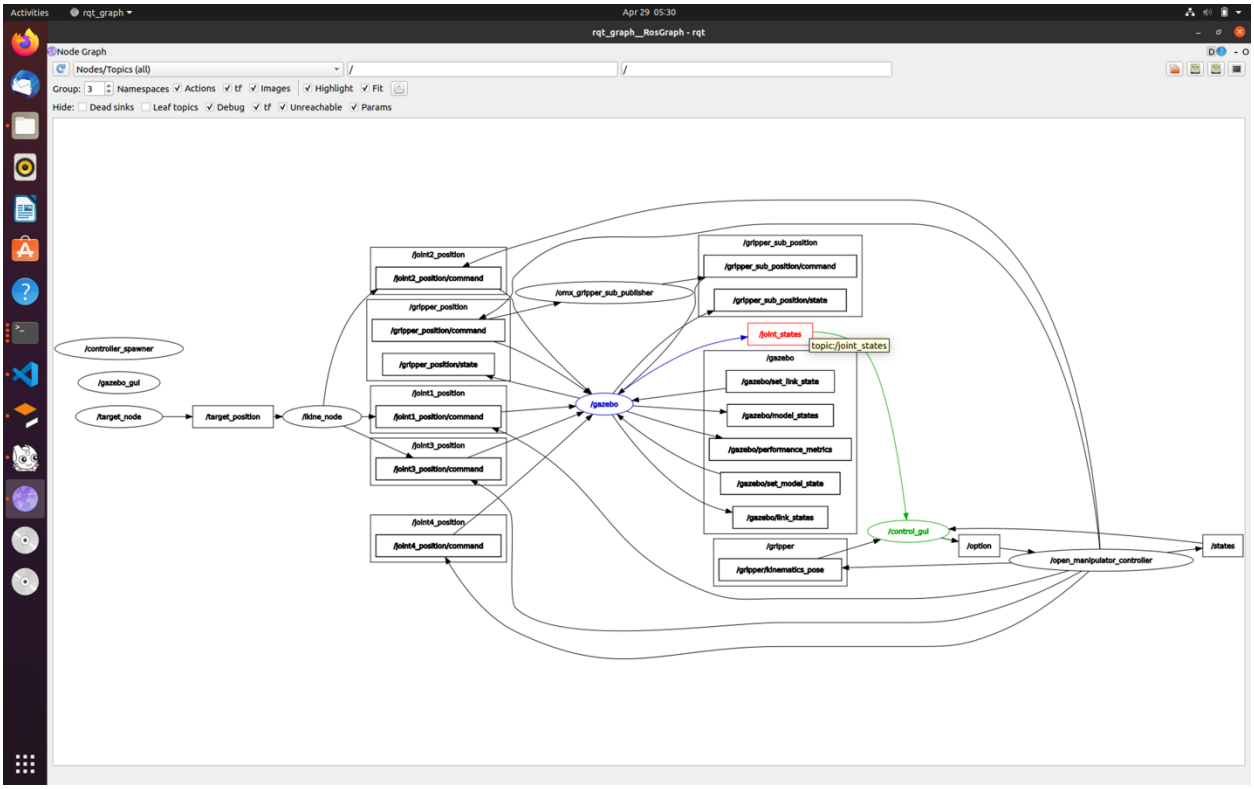

Figure 17: rqt_graph with GUI

3.3 Target Node Code

```
#!/usr/bin/env python3
import rospy
import numpy as np
from std_msgs.msg import Float32MultiArray

if __name__ == '__main__':

    # create and intializing node
    rospy.init_node("target_node")

    # position publisher
    target_pub = rospy.Publisher("/target_position", Float32MultiArray,
queue_size = 10)

    rate = rospy.Rate(10)

    while not rospy.is_shutdown():

        # getting the cordination
        target_msg = Float32MultiArray()

        # publishing the message
        target_msg.data = [0.169, 0.269, 0.169]
        target_pub.publish(target_msg)
        rospy.loginfo("Target Goal [X Y Z]")
        rospy.loginfo(np.array(target_msg.data).astype(np.float16))
        rate.sleep()
```

3.4 Inverse Kinematics Node Code

```
#!/usr/bin/env python3
import rospy
import numpy as np
from std_msgs.msg import Float64, Float32MultiArray

def target_clbk(recived_msg):

    global x_goal, y_goal, z_goal, theta0, theta1, theta2, theta3

    x_goal = recived_msg.data[0] - 0.012
    y_goal = recived_msg.data[1]
    z_goal = recived_msg.data[2]

    # inverse kinematics calculations
    r1 = np.sqrt(x_goal ** 2 + y_goal ** 2)
    r2 = np.sqrt(r1 ** 2 + (z_goal - d) ** 2)

    phi   = np.arctan(0.128 / 0.024)
    gamma = np.arccos((l3 ** 2 + 0.13 ** 2 - r2 ** 2) / (2 * l3 * l2))
    alpha = np.arcsin((l3 / r2) * np.sin(gamma))
    beta  = np.arctan((z_goal - d) / r1)

    # complying with our robot
    theta1 = np.arctan(y_goal / x_goal)
    theta2 = np.pi / 2 - (theta0 + alpha + beta)
    theta3 = np.pi - phi - gamma

if __name__ == '__main__':

    # create and intializing node
    rospy.init_node("ikine_node")

    # joint actuators
    joint1_pub = rospy.Publisher("/joint1_position/command", Float64,
queue_size = 10)
    joint2_pub = rospy.Publisher("/joint2_position/command", Float64,
queue_size = 10)
    joint3_pub = rospy.Publisher("/joint3_position/command", Float64,
queue_size = 10)
    rospy.Subscriber("/target_position", Float32MultiArray, target_clbk)
```

```
    x_goal, y_goal, z_goal, theta0, theta1, theta2, theta3 = 0, 0, 0, 0,
0, 0, 0

    # robot parameter
    theta0 = np.arctan(0.024 / 0.128)
    d = 0.077
    l2 = 0.13
    l3 = 0.25

    rate = rospy.Rate(10)

    while not rospy.is_shutdown():

        # moving the robot
        joint1_pub.publish( Float64(theta1) )
        joint2_pub.publish( Float64(theta2) )
        joint3_pub.publish( Float64(theta3) )

        # publishing the message
        rospy.loginfo("-----------------------------------------------------
---------------------------------")
        rospy.loginfo("Calculated Joint Angles")
        print(theta1, theta2, theta3)
        rate.sleep()
```

4.0 VIDEO LINK

https://drive.google.com/drive/folders/12GzNeXPt04xlqZa32DG2jnj5VDyOArYm?usp=sharing