MCT 344 Industrial Robotics

Milestone 3

Amal Amr AbdelHamid Zewita

19P1504

Team 35

Table of Contents

List of Figures

1.0 TERMINAL CREATION

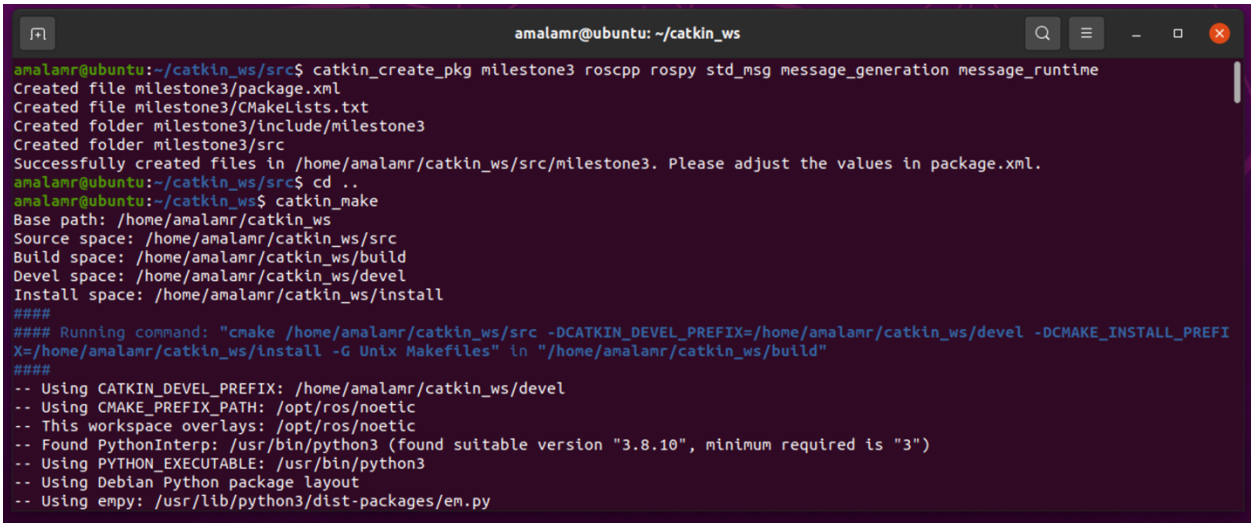

Figure 2: Creating Package

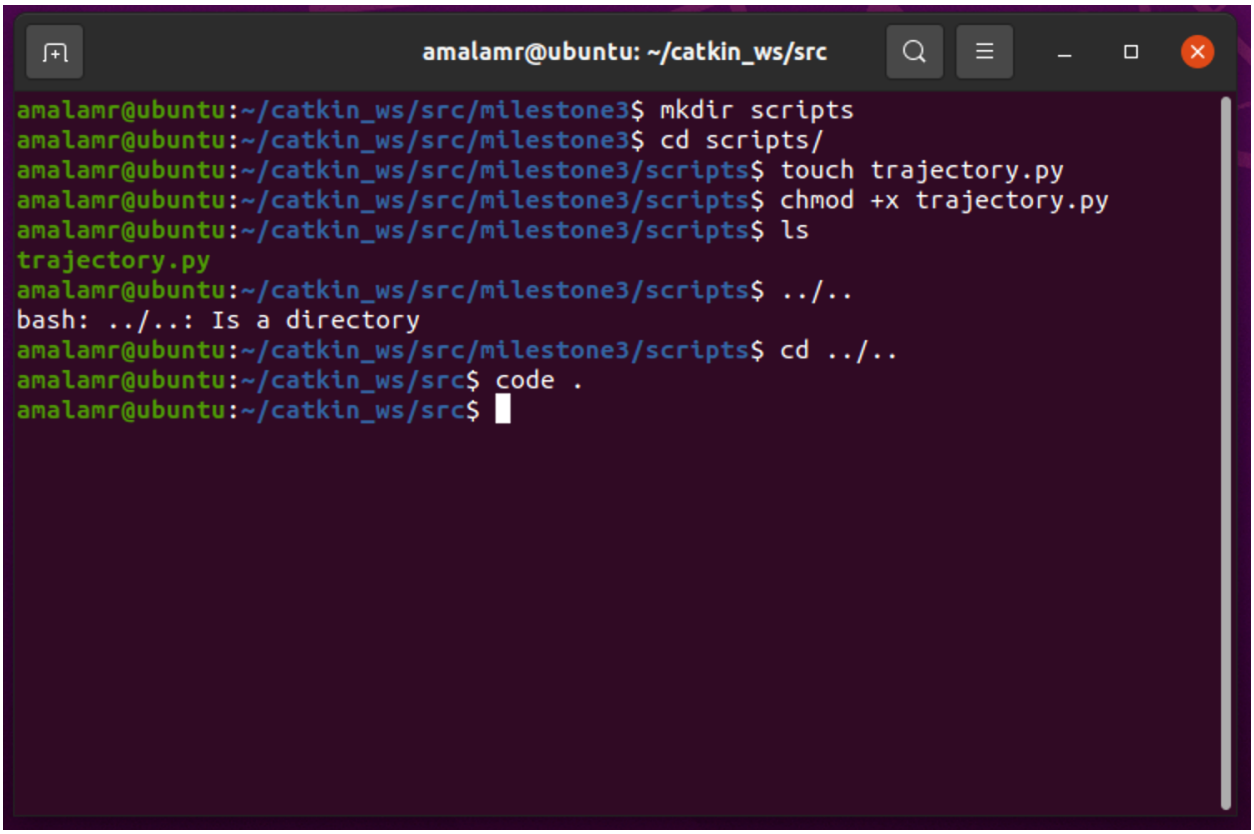

Figure 1: Code

2.0 GAZEBO

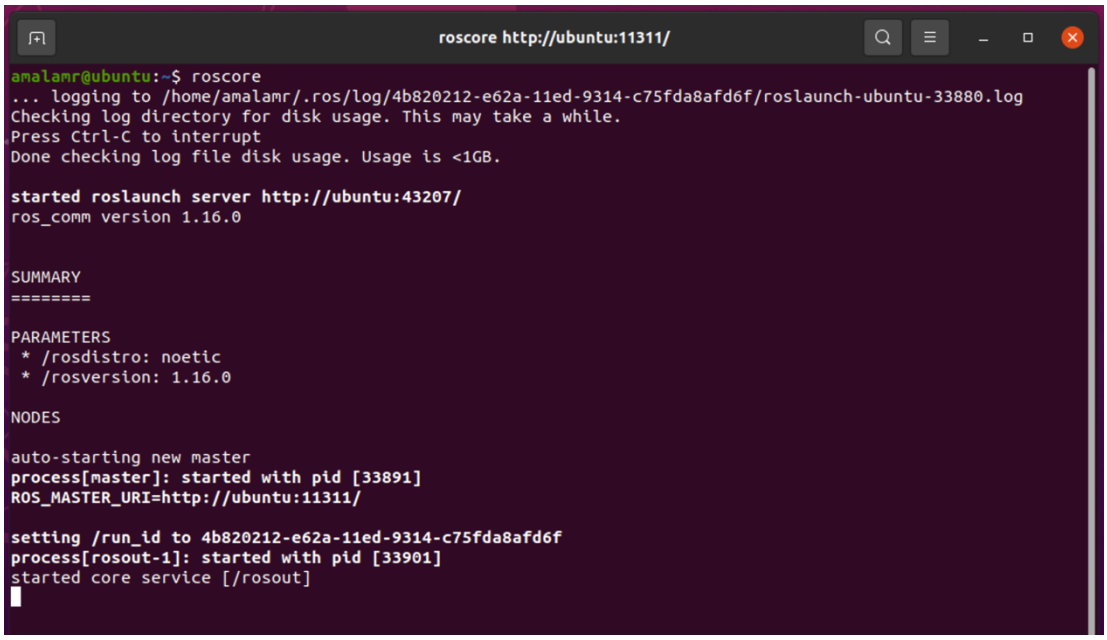

Figure 3: Running Roscore

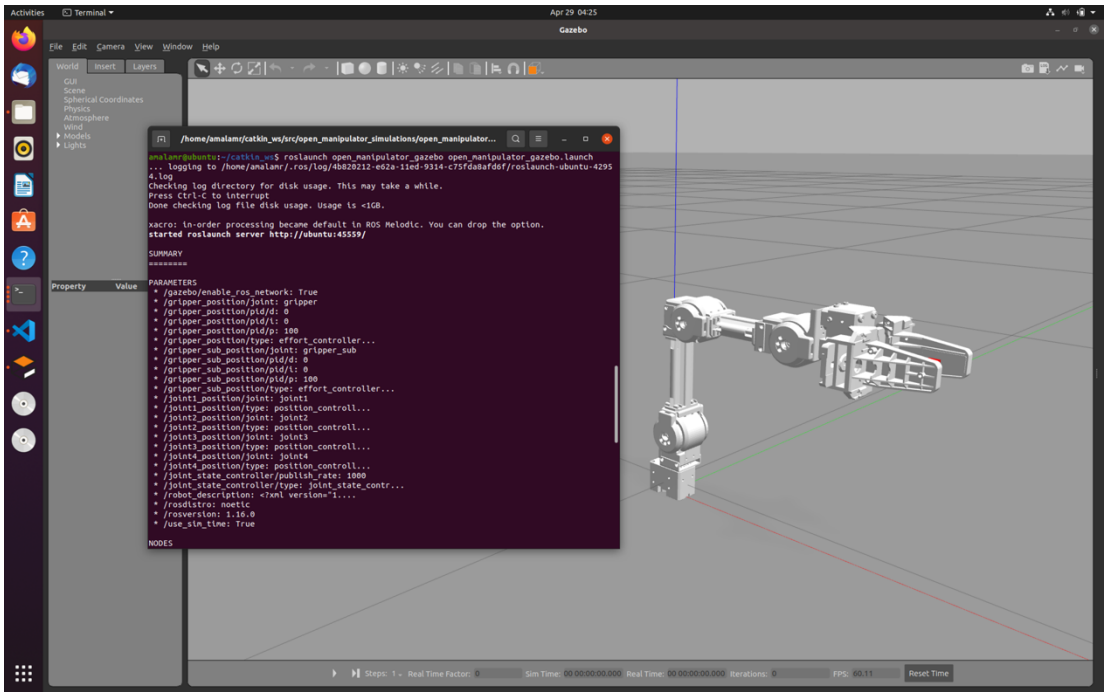

Figure 4: Launching Gazebo

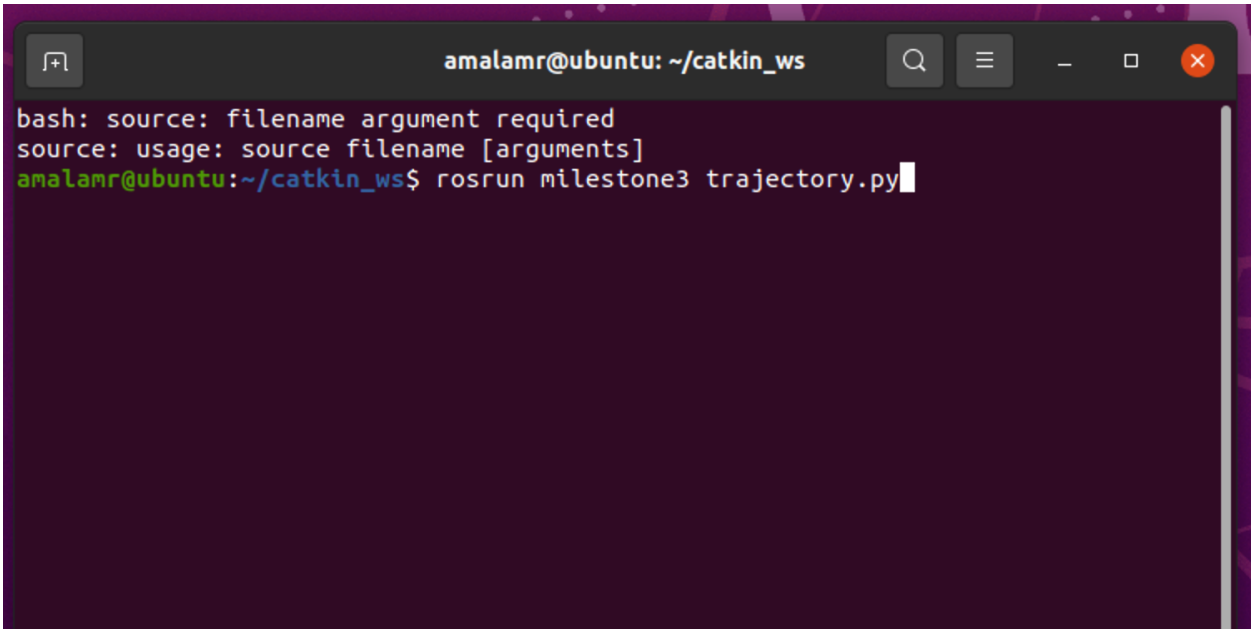

Figure 6: Running Trajectory Node

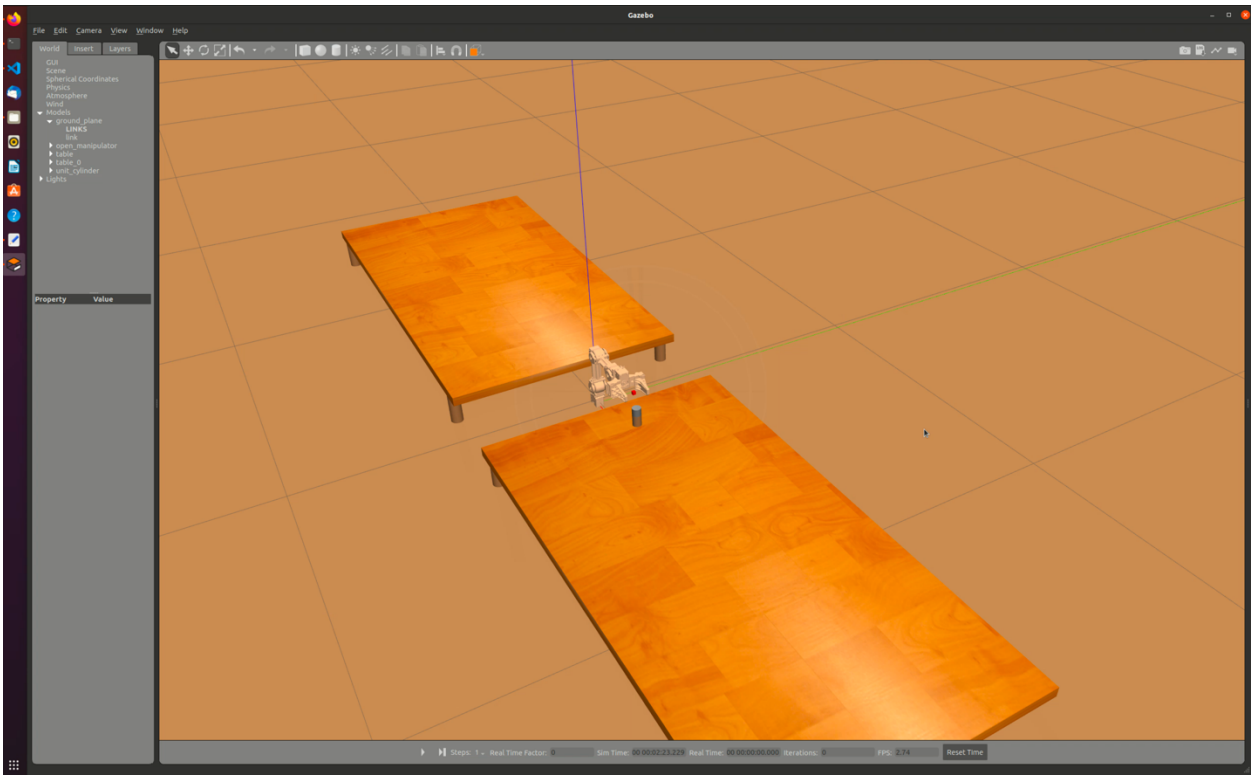

Figure 5: Environment

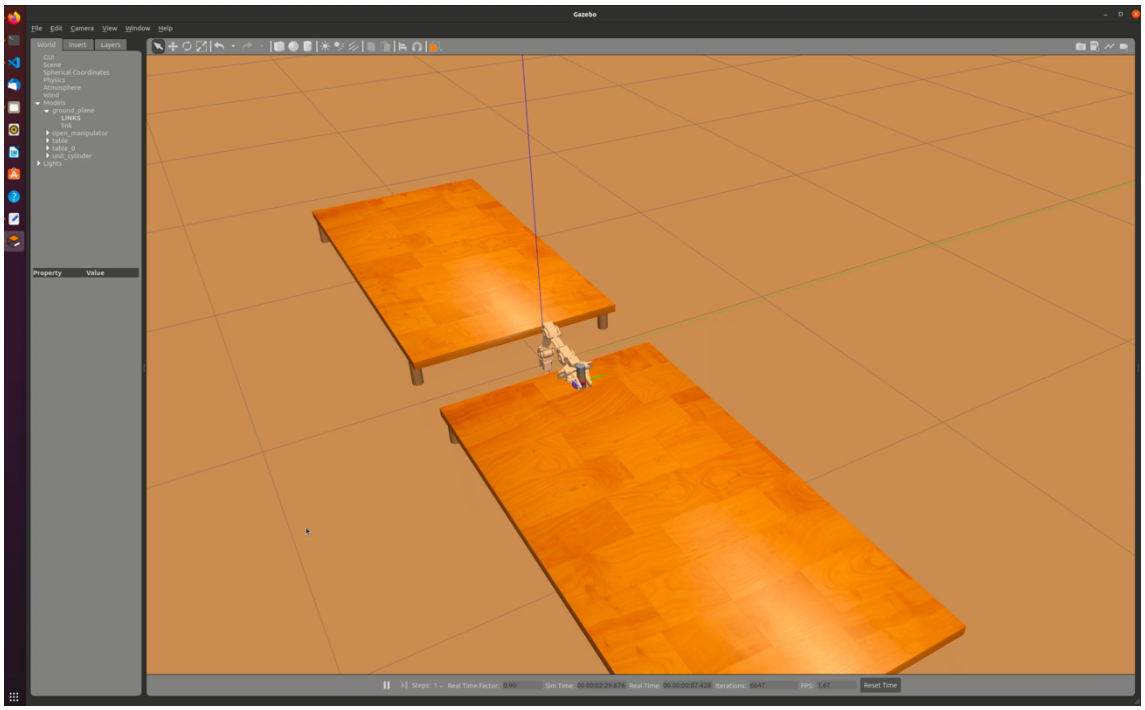

Figure 8: Position 2

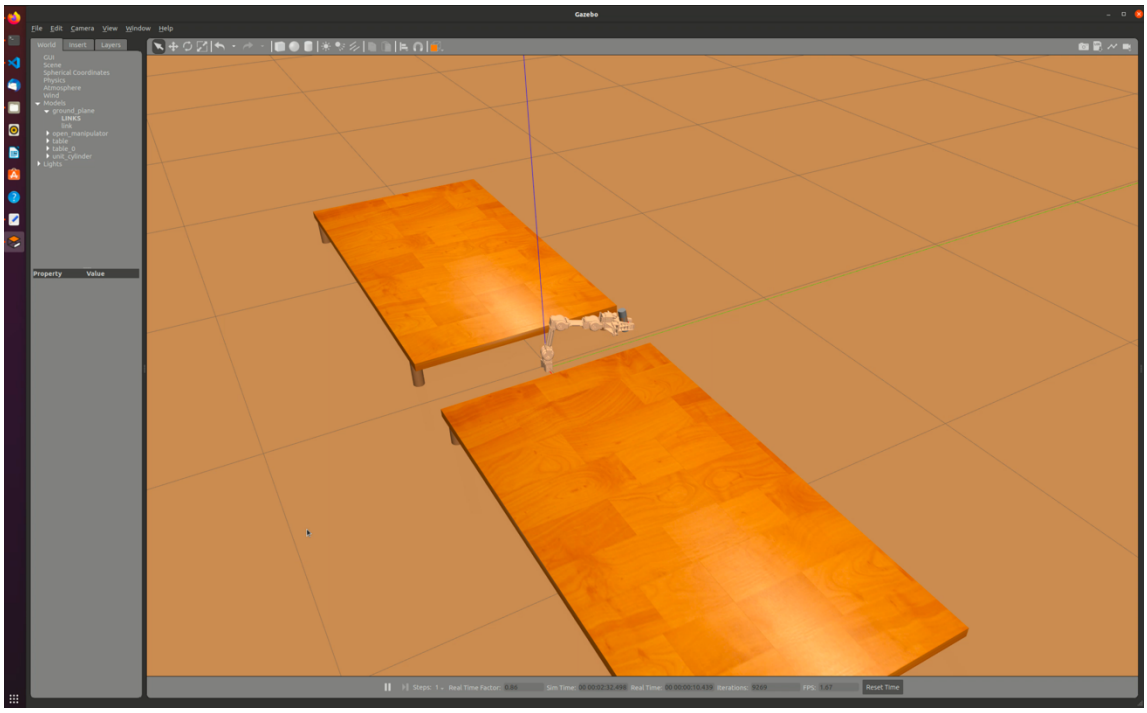

Figure 7: Position 3

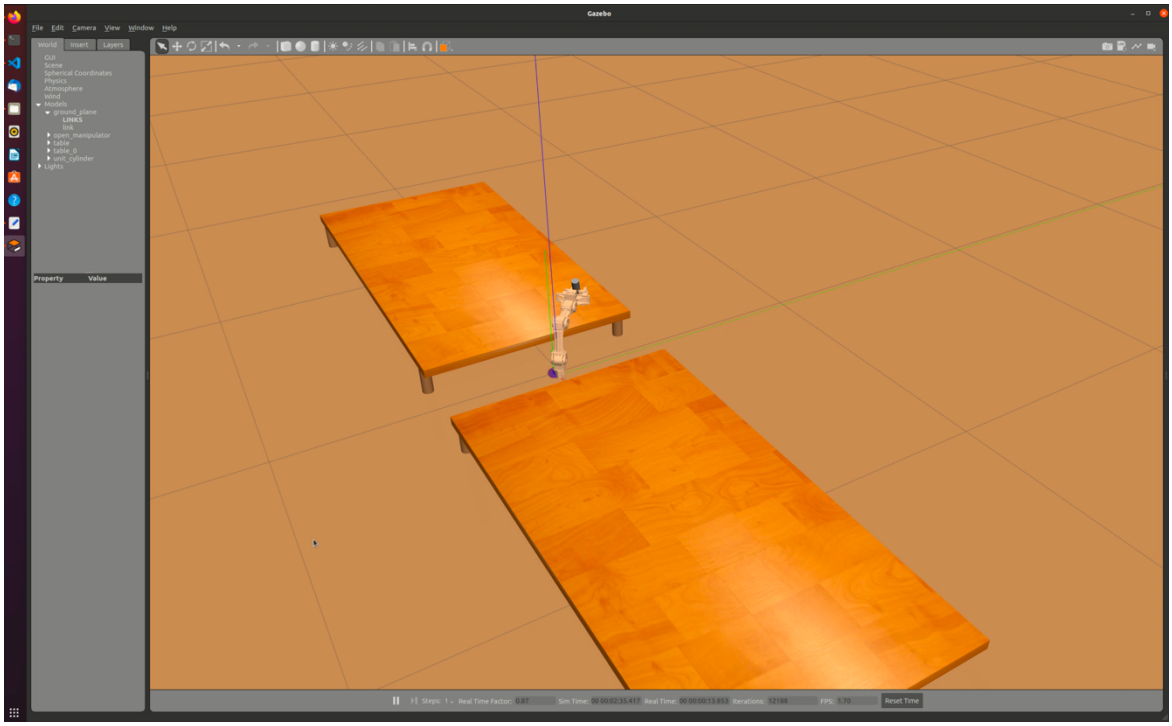

Figure 10: Position 4

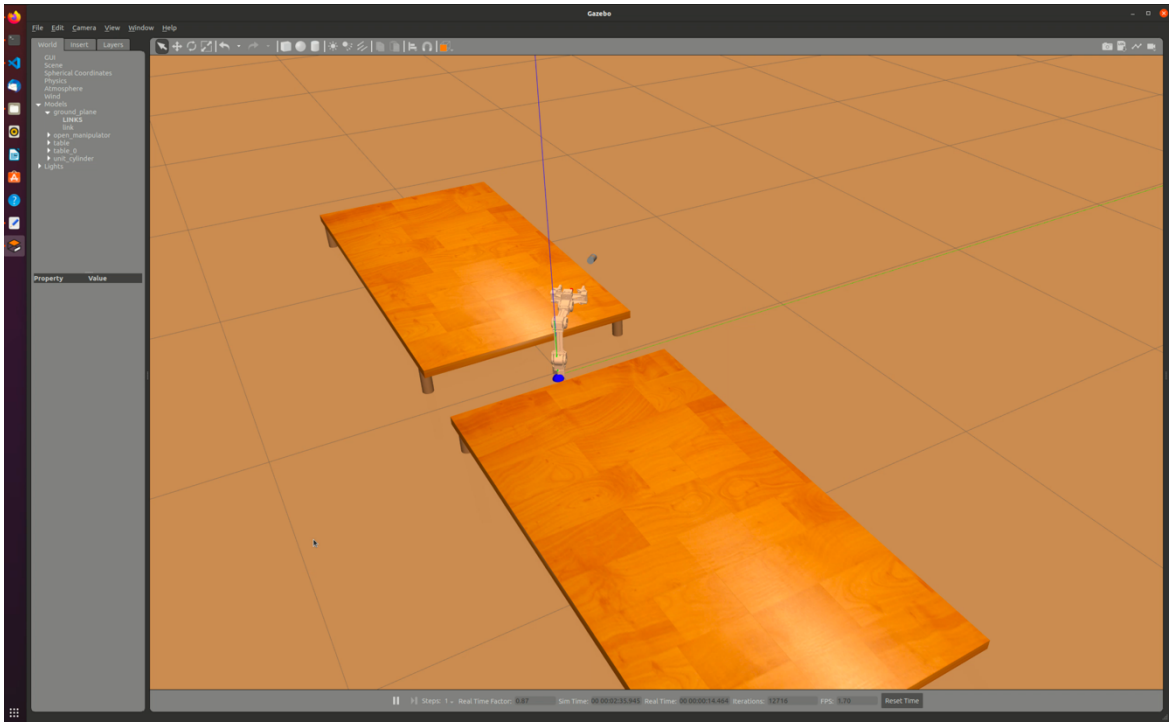

Figure 9: Gripper Releases

3.0 CODE

```
import rospy
from std_msgs.msg import Float64, Bool
import numpy as np

def grip():
    gripper_pos_pub.publish(Float64(-0.1))
    gripper_grip_pub.publish(Bool(True))

def release():
    gripper_pos_pub.publish(Float64(0.1))
    gripper_grip_pub.publish(Bool(False))

# function calculating the coefficients for 3rd order polynomial
def third_order_trajectory_planning(tf, thetai, thetaf):
    # Calculate third order polynomial coefficients
    a = np.array([[1,   0,      0,              0],
                  [1,   tf,     tf**2,      tf**3],
                  [0,   1,      0,              0],
                  [0,   1,      2*tf,   3*(tf**2)]])
    b = np.array([[thetai],
                  [thetaf],
                  [0],
                  [0]])

    poly_coeffs = np.linalg.solve(a, b).reshape(-1).tolist()
    c0, c1, c2, c3 = poly_coeffs[0], poly_coeffs[1], poly_coeffs[2],
poly_coeffs[3]
    return c0, c1, c2, c3

if __name__ == "__main__":

    # Initialize ROS node
    rospy.init_node("trajectory_planner_node")

    # Gripper publishers
    gripper_pos_pub = rospy.Publisher("/gripper_position/command",
Float64, queue_size= 10, latch=True)
    gripper_grip_pub = rospy.Publisher("/gripper_attach_cmd", Bool,
queue_size= 10, latch=True)

    # Create the publishers that'll move the robot's joints
```

```
    joint1_pub = rospy.Publisher("/joint1_position/command", Float64,
queue_size=10)
    joint2_pub = rospy.Publisher("/joint2_position/command", Float64,
queue_size=10)
    joint4_pub = rospy.Publisher("/joint3_position/command", Float64,
queue_size=10)

   # 1st TRAJECTORY_PLAN
    c0, c1, c2, c3 = third_order_trajectory_planning(tf=10, thetai=0,
thetaf=0)
    rospy.loginfo("Calculated the coefficients based on 3rd order
polynomial joint space trajectory planning for joint 1")
    rospy.loginfo("c0:{}, c1:{}, c2:{}, c3:{}".format(c0, c1, c2, c3))

    c4, c5, c6, c7 = third_order_trajectory_planning(tf=10, thetai=0,
thetaf=0.3)
    rospy.loginfo("Calculated the coefficients based on 3rd order
polynomial joint space trajectory planning for joint 2")
    rospy.loginfo("c0:{}, c1:{}, c2:{}, c3:{}".format(c4, c5, c6, c7))

    c16, c17, c18, c19 = third_order_trajectory_planning(tf=10, thetai=0,
thetaf=0)
    rospy.loginfo("Calculated the coefficients based on 3rd order
polynomial joint space trajectory planning for joint 4")
    rospy.loginfo("c0:{}, c1:{}, c2:{}, c3:{}".format(c16, c17, c18, c19))

    # Specify dt parameter and calculate discrete time vector
    dt = 0.01    # in seconds
    time_vector = np.linspace(start=0, stop=10, num=int((5-0)/dt))
    for t in time_vector:
        joint1_angle = c0 + c1 * t + c2 * (t**2) + c3 * (t**3)
        joint2_angle = c4 + c5 * t + c6 * (t**2) + c7 * (t**3)
        joint4_angle = c16 + c17 * t + c18 * (t**2) + c19 * (t**3)
        print("Time: {}, Command Joint1 Angle: {}:".format(t,
joint1_angle))
        print("Time: {}, Command Joint2 Angle: {}:".format(t,
joint2_angle))
        print("Time: {}, Command Joint4 Angle: {}:".format(t,
joint4_angle))
        joint1_pub.publish(Float64(1*joint1_angle))
        joint2_pub.publish(Float64(1*joint2_angle))
        joint4_pub.publish(Float64(1*joint4_angle))
        rospy.sleep(dt)
```

```
    # Grip before moving
    grip()
    rospy.sleep(2.0)
    rospy.logwarn("Trajectory successfuly executed, terminating...")
    rospy.logwarn("Object is gripped")

    # 2nd TRAJECTORY_PLAN
    c8, c9, c10, c11 = third_order_trajectory_planning(tf=35, thetai=0,
thetaf=np.pi)
    rospy.loginfo("Calculated the coefficients based on 3rd order
polynomial joint space trajectory planning for joint 1")
    rospy.loginfo("c0:{}, c1:{}, c2:{}, c3:{}".format(c8, c9, c10, c11))

    c12, c13, c14, c15 = third_order_trajectory_planning(tf=35,
thetai=0.3, thetaf=0.1)
    rospy.loginfo("Calculated the coefficients based on 3rd order
polynomial joint space trajectory planning for joint 2")
    rospy.loginfo("c0:{}, c1:{}, c2:{}, c3:{}".format(c12, c13, c14, c15))

    c20, c21, c22, c23 = third_order_trajectory_planning(tf=35, thetai=0,
thetaf=-0.2)
    rospy.loginfo("Calculated the coefficients based on 3rd order
polynomial joint space trajectory planning for joint 4")
    rospy.loginfo("c0:{}, c1:{}, c2:{}, c3:{}".format(c20, c21, c22, c23))

    #Specify dt parameter and calculate discrete time vector
    dt = 0.01    # in seconds
    time_vector = np.linspace(start=10, stop=25, num=int((5-0)/dt))
    for t in time_vector:
        joint1_angle = c8 + c9 * t + c10 * (t**2) + c11 * (t**3)
        joint2_angle = c12 + c13 * t + c14 * (t**2) + c15 * (t**3)
        joint4_angle = c20 + c21 * t + c22 * (t**2) + c23 * (t**3)
        print("Time: {}, Command Joint1 Angle: {}:".format(t,
joint1_angle))
        print("Time: {}, Command Joint2 Angle: {}:".format(t,
joint2_angle))
        print("Time: {}, Command Joint3 Angle: {}:".format(t,
joint4_angle))
        joint1_pub.publish(Float64(1*joint1_angle))
        joint2_pub.publish(Float64(1*joint2_angle))
        joint4_pub.publish(Float64(1*joint4_angle))
        rospy.sleep(dt)
```

```
    # Release upon arrival
    release()

    rospy.logwarn("Trajectory successfuly executed, terminating...")
    rospy.logwarn("Object is released")

    # Sleep sometime for stabilization before termination
    rospy.sleep(2.0)
```