



## MILESTONE 3

### Team 5

Kirolos Thabet Fouad  
Omar Eltoutongy

19P6754  
19P1060

## Table of Contents:

1. Introduction: .....	3
2. Trajectory Planning Node: .....	4
3. RQT Graph: .....	5
4. Environment From Gazebo: .....	6
5. Simulation Video Link: .....	8

## Table of Figures:

Figure 1: Open manipulator robot .....	3
Figure 2: Code Part2 .....	4
Figure 3: Code Part1 .....	4
Figure 4: Code Part3 .....	5
Figure 5: RQT Graph (all) .....	5
Figure 6: Gazebo Environment .....	6
Figure 7: RQT Graph (active) .....	6
Figure 8: Robot reached in Gazebo .....	7
Figure 9: Robot is moving in Gazebo .....	7

## 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.

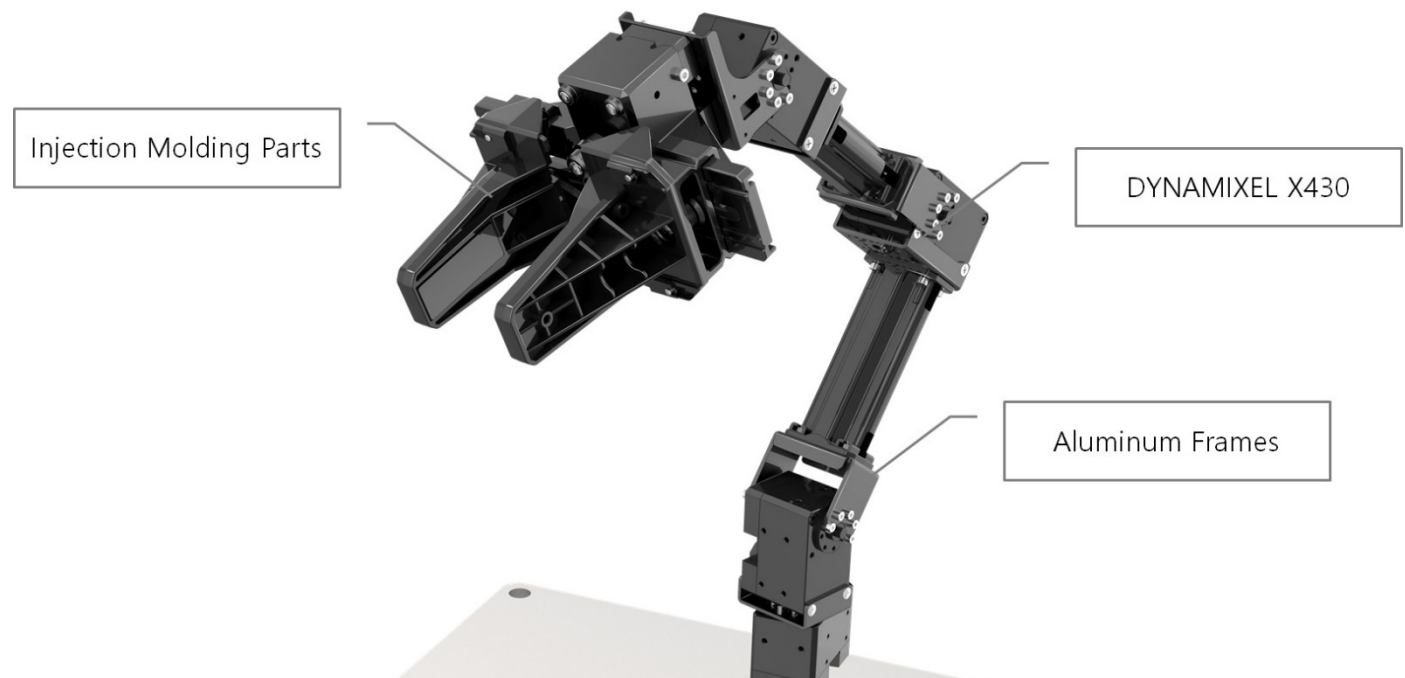


Figure 1: Open manipulator robot

## 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.

```

1  import rospy
2  from std_msgs.msg import Float64, Bool
3  import numpy as np
4
5  def gripper():
6      gripper_pos_pub.publish(Float64(0.1))
7      gripper_grip_pub.publish(Bool(True))
8
9  def release():
10     gripper_pos_pub.publish(Float64(0.1))
11     gripper_grip_pub.publish(Bool(False))
12
13  def third_order_trajectory_planning(tf, thetai, thetalf):
14     # Calculate third order polynomial coefficients
15     a = np.array([[1, 0, 0, 0],
16                  [0, 1, 0, 0],
17                  [0, 0, 1, 0],
18                  [0, 0, 0, 1]])
19     b = np.array([[-tf, thetai, thetalf, 0],
20                  [0, 0, 0, 0],
21                  [0, 0, 0, 0],
22                  [0, 0, 0, 0]])
23
24     poly_coeffs = np.linalg.solve(a, b).reshape(-1).tolist()
25     c0, c1, c2, c3 = poly_coeffs[0], poly_coeffs[1], poly_coeffs[2], poly_coeffs[3]
26     return c0, c1, c2, c3
27
28  if __name__ == '__main__':
29
30     # Used angles
31     my_first_angle = 0.2
32     my_second_angle = 0.185
33     my_third_angle = 0.15
34
35     # Initialize ROS node
36     rospy.init_node("trajectory_planner_node")
37
38     # Gripper publishers
39
40     # The 'latch=True' parameter in the 'rospy.Publisher' function ensures that the gazebo receives the last message
41     # and if not ready, the keep it alive until it is ready.
42     gripper_pos_pub = rospy.Publisher("/gripper_position/command", Float64, queue_size=10, latch=True)
43     gripper_grip_pub = rospy.Publisher("/gripper_attach_cmd", Bool, queue_size=10, latch=True)
44
45     # Create the publishers that'll move the robot's joints
46     joint1_pub = rospy.Publisher("/joint1_position/command", Float64, queue_size=10)

```

Figure 3: Code Part 1

```

47     joint4_pub = rospy.Publisher("/joint4_position/command", Float64, queue_size=10)
48
49     # Trajectory Plan 1
50     c0, c1, c2, c3 = third_order_trajectory_planning(tf=10, thetai=0, thetalf=my_first_angle)
51     rospy.loginfo("Calculated the coefficients based on 3rd order polynomial joint space trajectory planning for joint 1")
52     rospy.loginfo("c0: {}, c1: {}, c2: {}, c3: {}".format(c0, c1, c2, c3))
53
54     c4, c5, c6, c7 = third_order_trajectory_planning(tf=10, thetai=0, thetalf=my_second_angle)
55     rospy.loginfo("Calculated the coefficients based on 3rd order polynomial joint space trajectory planning for joint 2")
56     rospy.loginfo("c0: {}, c1: {}, c2: {}, c3: {}".format(c4, c5, c6, c7))
57
58     c16, c17, c18, c19 = third_order_trajectory_planning(tf=10, thetai=0, thetalf=my_third_angle)
59     rospy.loginfo("Calculated the coefficients based on 3rd order polynomial joint space trajectory planning for joint 4")
60     rospy.loginfo("c0: {}, c1: {}, c2: {}, c3: {}".format(c16, c17, c18, c19))
61
62     # Specify dt parameter and calculate discrete time vector
63     dt = 0.01 # in seconds
64     time_vector = np.linspace(start=0, stop=10, num=int((5-0)/dt))
65     for t in time_vector:
66         joint1_angle = c0 + c1 * t + c2 * (t**2) + c3 * (t**3)
67         joint2_angle = c4 + c5 * t + c6 * (t**2) + c7 * (t**3)
68         joint4_angle = c16 + c17 * t + c18 * (t**2) + c19 * (t**3)
69         print("Time: {}, Command Joint1 Angle: {}".format(t, joint1_angle))
70         print("Time: {}, Command Joint2 Angle: {}".format(t, joint2_angle))
71         print("Time: {}, Command Joint4 Angle: {}".format(t, joint4_angle))
72         joint1_pub.publish(Float64(joint1_angle))
73         joint2_pub.publish(Float64(joint2_angle))
74         joint4_pub.publish(Float64(joint4_angle))
75         rospy.sleep(dt)
76
77     # Grip before moving
78     gripper()
79     rospy.sleep(2.0)
80     rospy.logwarn("Trajectory successfully executed, terminating...")
81     rospy.logwarn("Object is gripped")
82
83     # Trajectory Plan 2
84     c8, c9, c10, c11 = third_order_trajectory_planning(tf=25, thetai=my_first_angle, thetalf=np.pi)
85     rospy.loginfo("Calculated the coefficients based on 3rd order polynomial joint space trajectory planning for joint 1")
86     rospy.loginfo("c0: {}, c1: {}, c2: {}, c3: {}".format(c8, c9, c10, c11))
87
88     c12, c13, c14, c15 = third_order_trajectory_planning(tf=25, thetai=my_second_angle, thetalf=0.2)
89     rospy.loginfo("Calculated the coefficients based on 3rd order polynomial joint space trajectory planning for joint 2")
90     rospy.loginfo("c0: {}, c1: {}, c2: {}, c3: {}".format(c12, c13, c14, c15))

```

Figure 2: Code Part 2

```

88
89
90 c12, c13, c14, c15 = third_order_trajectory_planning(tf=25, thetai=my_second_angle, thetalf=0.2)
91 rospy.loginfo("Calculated the coefficients based on 3rd order polynomial joint space trajectory planning for joint 2")
92 rospy.loginfo("c0: {}, c1: {}, c2: {}, c3: {}".format(c12, c13, c14, c15))
93
94 c20, c21, c22, c23 = third_order_trajectory_planning(tf=25, thetai=my_third_angle, thetalf=0)
95 rospy.loginfo("Calculated the coefficients based on 3rd order polynomial joint space trajectory planning for joint 4")
96 rospy.loginfo("c0: {}, c1: {}, c2: {}, c3: {}".format(c20, c21, c22, c23))
97
98 #Specify dt parameter and calculate discrete time vector
99 dt = 0.01 # in seconds
100 time_vector = np.linspace(start=10, stop=25, num=int((5-0)/dt))
101 for t in time_vector:
102     joint1_angle = c8 + c9 * t + c10 * (t**2) + c11 * (t**3)
103     joint2_angle = c12 + c13 * t + c14 * (t**2) + c15 * (t**3)
104     joint4_angle = c20 + c21 * t + c22 * (t**2) + c23 * (t**3)
105     print("Time: {}, Command Joint1 Angle: {}".format(t, joint1_angle))
106     print("Time: {}, Command Joint2 Angle: {}".format(t, joint2_angle))
107     print("Time: {}, Command Joint4 Angle: {}".format(t, joint4_angle))
108     joint1_pub.publish(Float64(joint1_angle))
109     joint2_pub.publish(Float64(joint2_angle))
110     joint4_pub.publish(Float64(joint4_angle))
111     rospy.sleep(dt)
112
113 # Release upon arrival
114 release()
115
116 rospy.logwarn("Trajectory successfully executed, terminating...")
117 rospy.logwarn("Object is released")
118
119 # Sleep sometime for stabilization before termination
120 rospy.sleep(2.0)

```

Figure 4: Code Part3

### 3. RQT Graph:

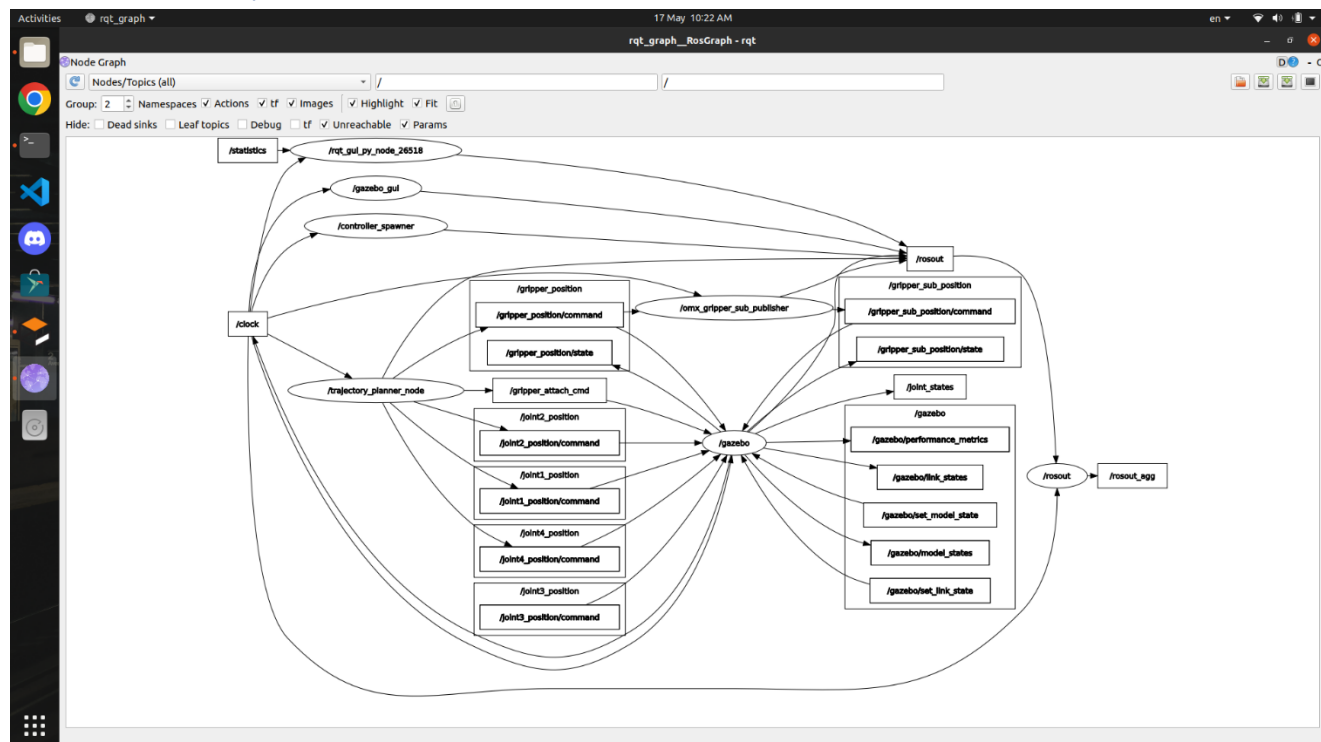


Figure 5: RQT Graph (all)



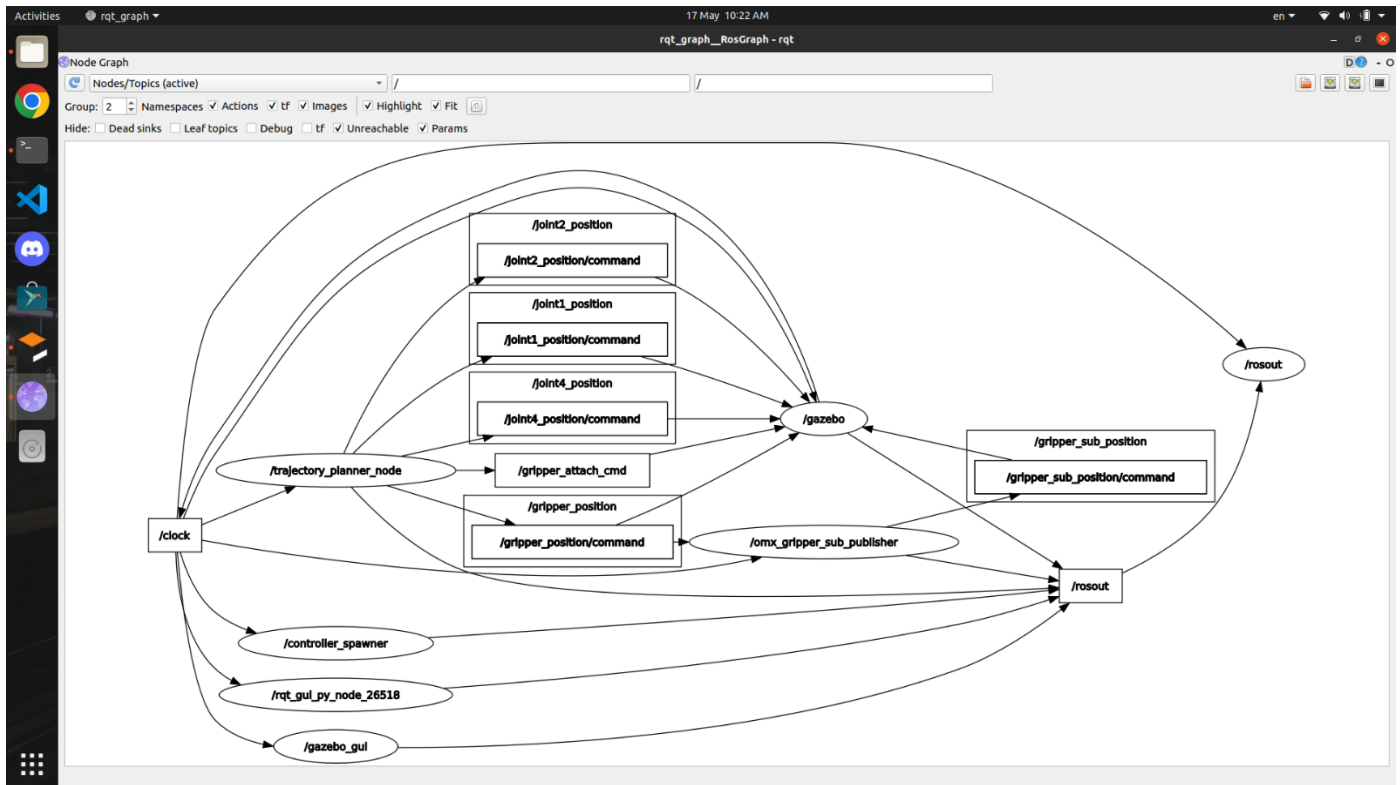


Figure 7: RQT Graph (active)

#### 4. Environment From Gazebo:

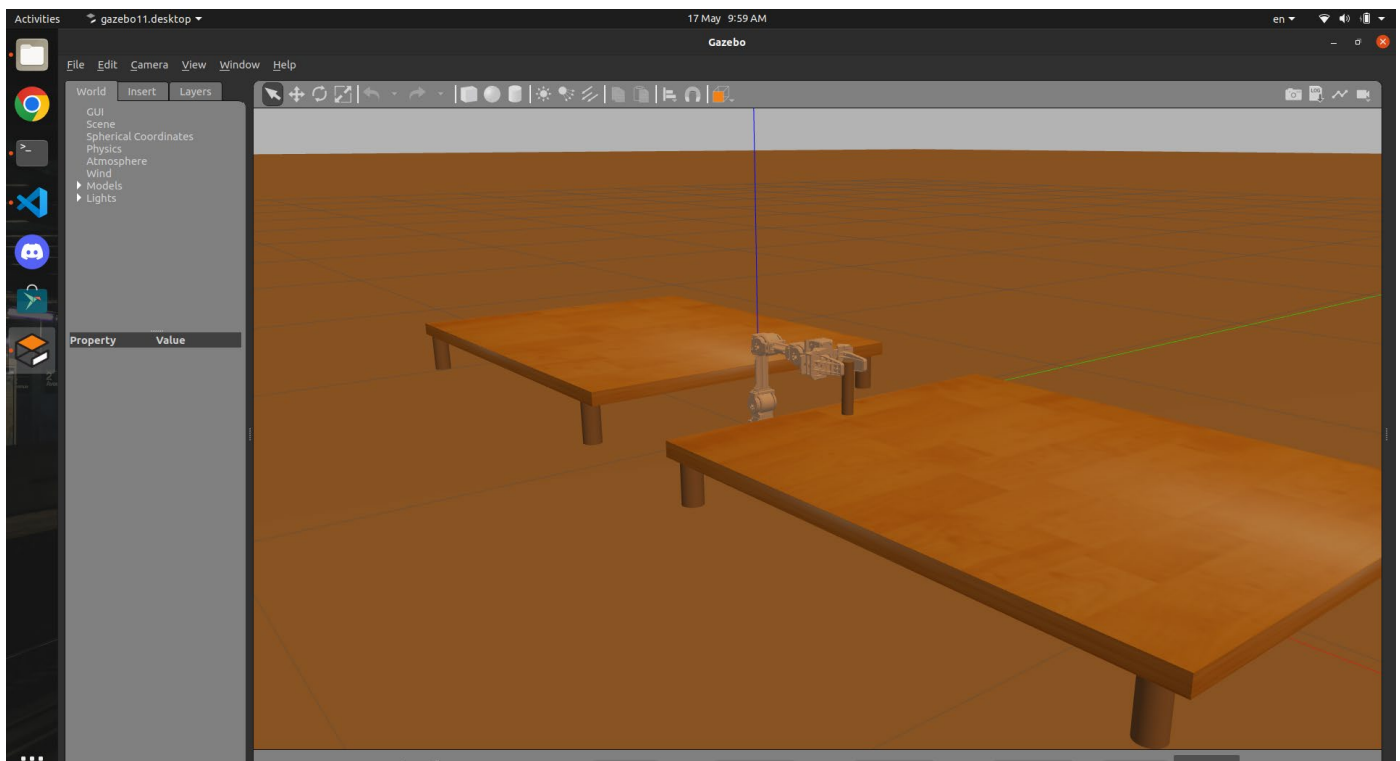


Figure 6: Gazebo Environment

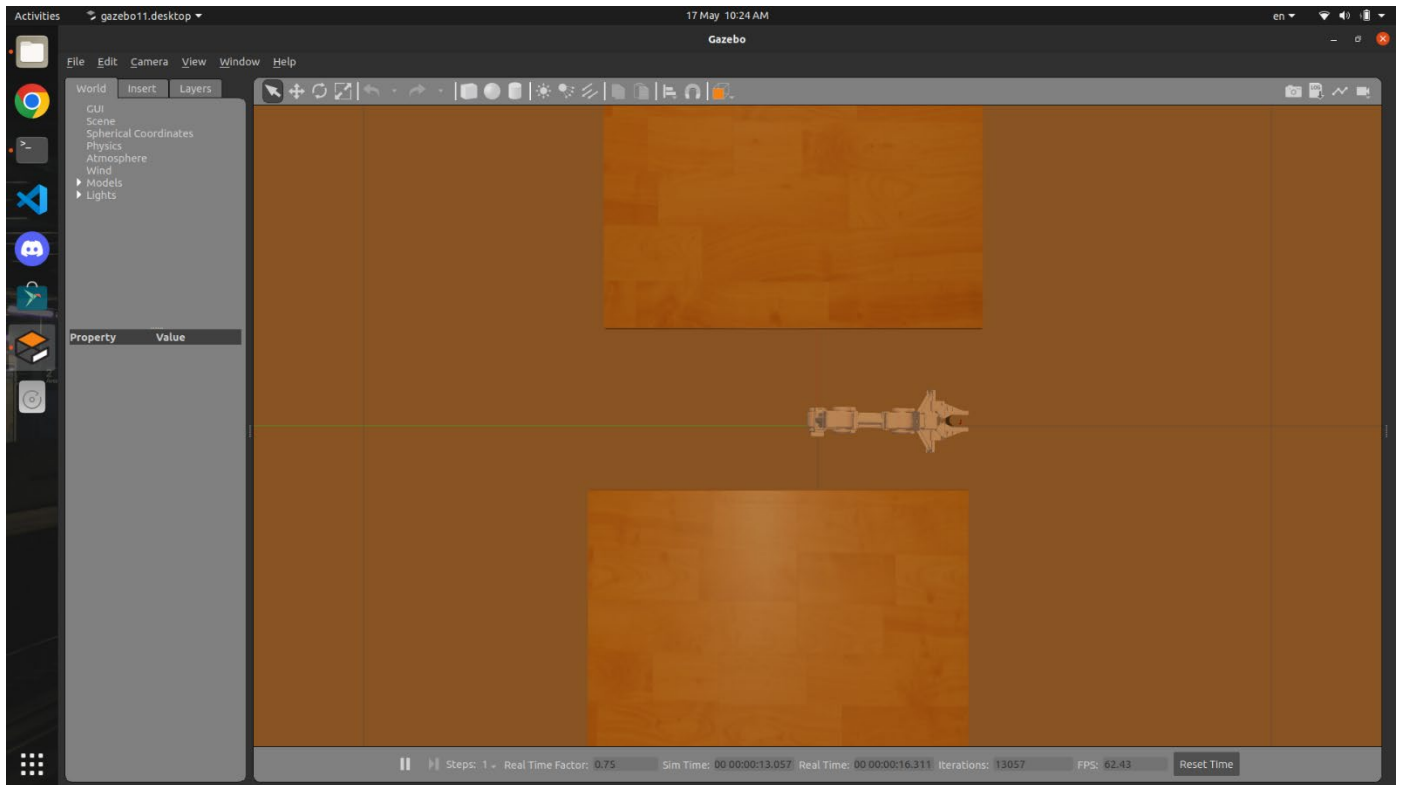


Figure 9: Robot is moving in Gazebo.

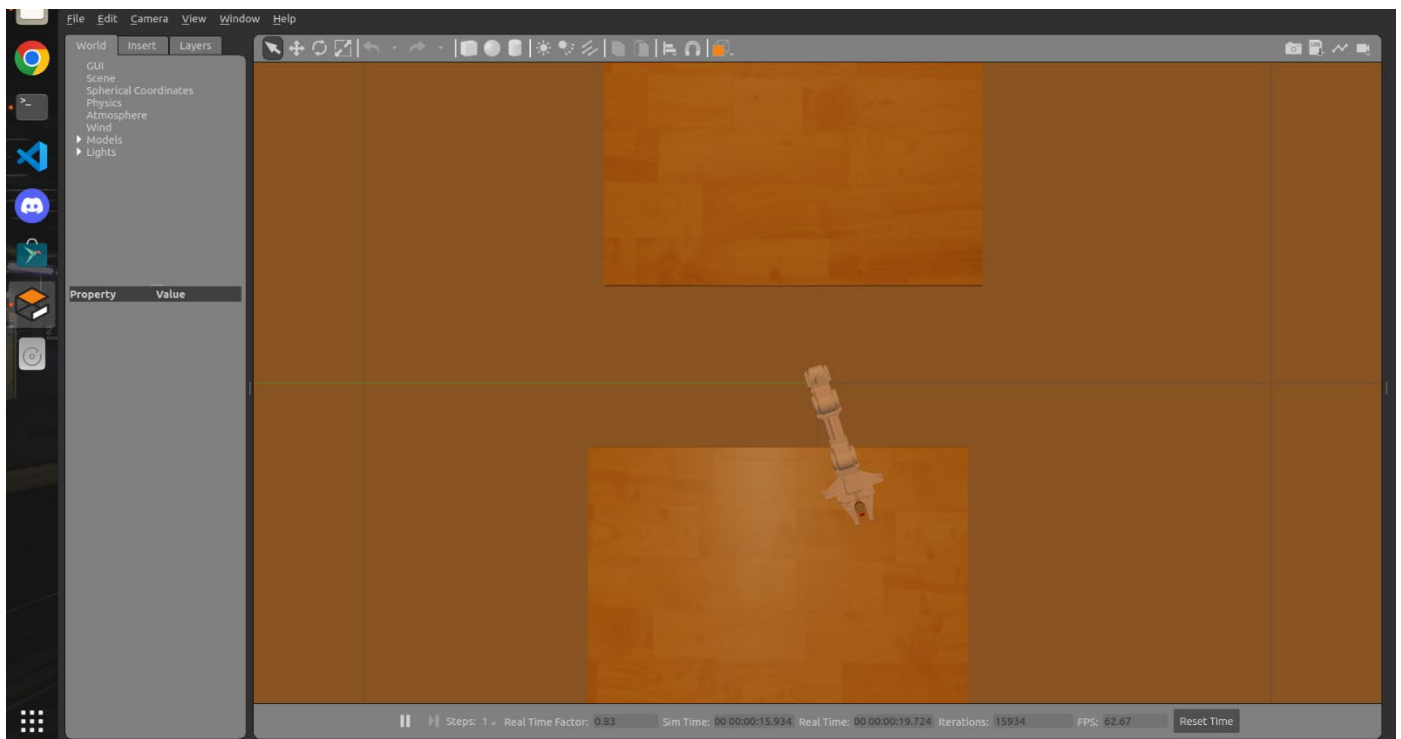


Figure 8: Robot reached in Gazebo.

## 5. Simulation Video Link:

You can watch the video of our project [here](#).