



Department of
Computer Engineering

به نام خدا



Amirkabir University of Technology
(Tehran Polytechnic)

دانشگاه صنعتی امیرکبیر
دانشکده مهندسی کامپیووتر
اصول علم ربات

تمرین سری سوم

محمدسپهر توکلی کرمانی	نام و نام خانوادگی
۹۸۳۱۱۱۱	شماره دانشجویی
۱۴۰۱/۳/۲۶	تاریخ ارسال گزارش

فهرست گزارش سوالات

۳.....	سوال ۱ – سوال اول
۳.....	سوال ۲ – سوال دوم
۳.....	سوال ۳ – سوال سوم
۳.....	سوال ۴ – سوال چهارم
۴.....	سناریو ۱ - الف
۷.....	سناریو ۱ - ب
۸.....	سناریو ۱ - ج
۱۰.....	سناریو ۲ - الف
۱۲.....	سناریو ۲ - ب
۱۵.....	سناریو ۲ - ج

سوال ۱ – سوال اول

LIDAR و GPS دو نوع از سنسور های active هستند. camera و Inclinometer نیز دو نوع از سنسور های passive هستند. GPS سیگنال های خود را از ۲۴ ماهواره موجود می گیرد و یک انرژی تولید گرده است و بازخورد آنرا اندازه می گیرد ، پس این سنسور یک سنسور active است.

سوال ۲ – سوال دوم

برای مکان یابی در فضای داخلی GPS مناسب نیست زیرا امواج ماهواره ای را نمی توان به شکل مناسب و قابل استفاده دریافت کرد و خطأ در مسیر یابی بالا خواهد بود .

سوال ۳ – سوال سوم

در ناو بری کور از قطب نما برای پیدا کردن heading ربات حول محور Z و از شیب سنج برای پیدا کردن میزان چرخش در حول محور X استفاده می شود.

سوال ۴ – سوال چهارم

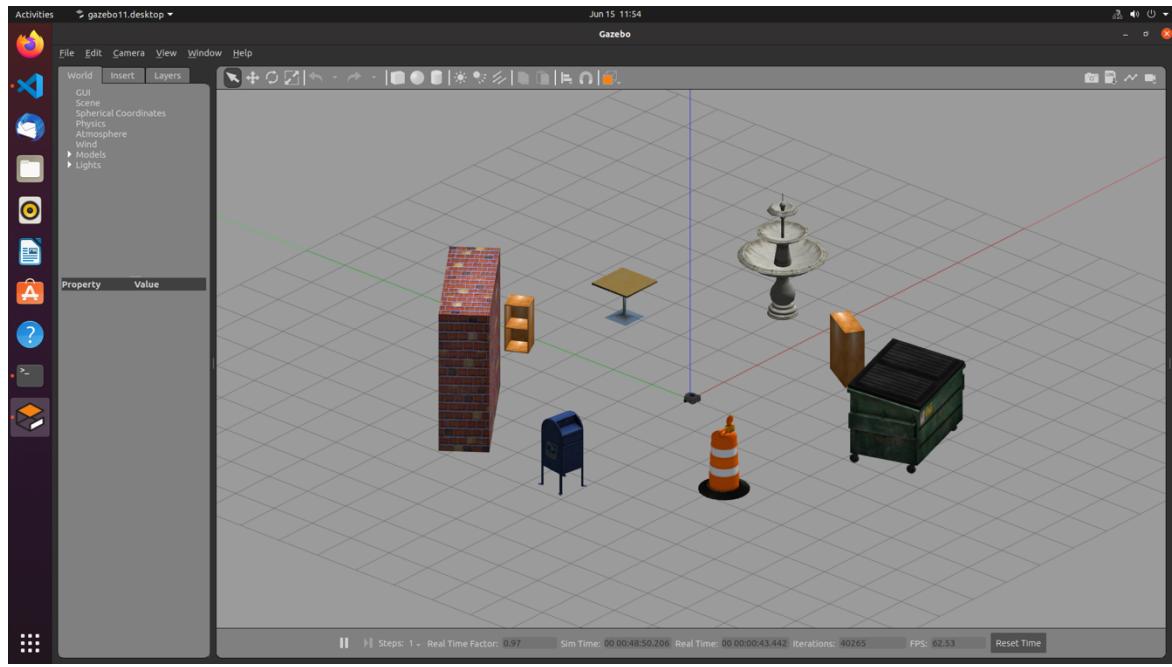
از قطب نما برای پیدا کردن heading ربات حول محور Z استفاده می شود. به این صورت که با خواندن جهت فعلی و جهت رو به رو در هنگام شروع ، می توان heading در جهت محور Z را به دست آورد.

بخش عملی

سناریو ۱ :
بخش (الف)

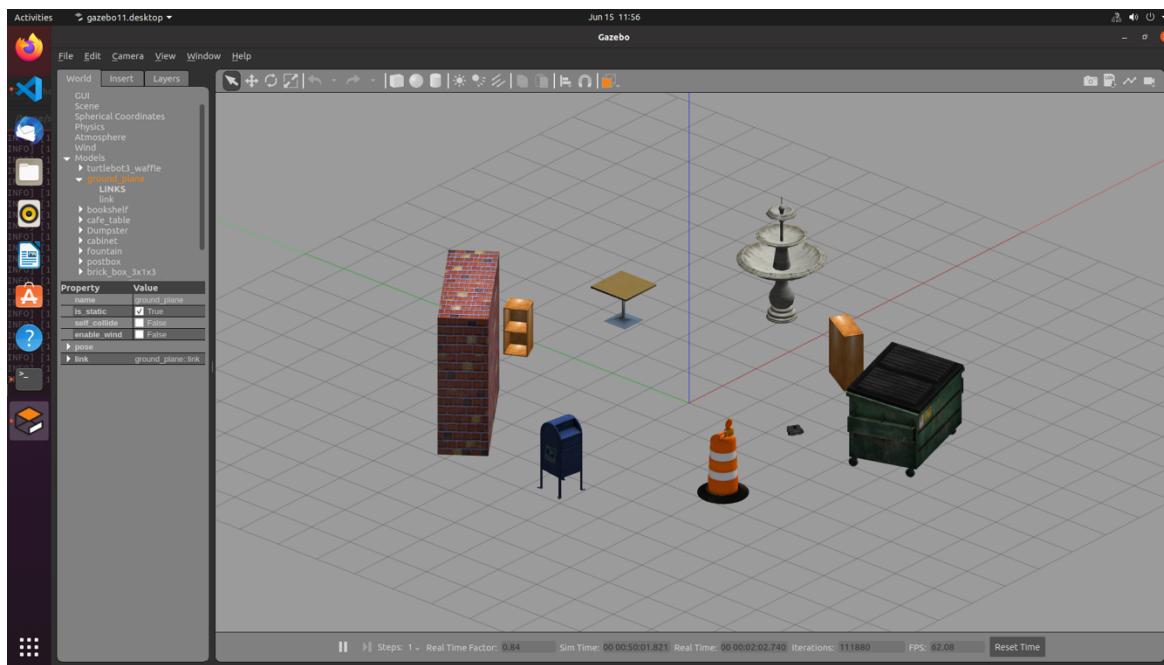
برای شروع مقادیر position را برای موانع در یک آرایه قرار داده و از آن در کد استفاده می کنیم و فاصله را تا نزدیک ترین مانع حساب کرده و آنرا در تاپیک ClosestObstacle قرار می دهیم :

```
1  #!/usr/bin/python3
2
3  import rospy
4  from nav_msgs.msg import Odometry
5  from distance_caculator.msg import Obst
6  import math
7
8  class ClosestObstacle:
9
10     def __init__(self) -> None:
11
12         rospy.init_node("closest_obstacle_node" , anonymous=True)
13
14         self.odom_subscriber = rospy.Subscriber("/odom" , Odometry , callback = self.read_distance)
15         self.obstacle_publisher = rospy.Publisher('/ClosestObstacle' , Obst , queue_size=10)
16
17         # obstacle details
18         self.obstacles = {
19             "bookshelf" : (2.64, -1.55),
20             "dumpster" : (1.23, -4.57),
21             "barrel" : (-2.51, -3.08),
22             "postbox" : (-4.47, -0.57),
23             "brick_box" : (-3.44, 2.75),
24             "cabinet" : (-0.45, 4.05),
25             "cafe_table": (1.91, 3.37),
26             "fountain" : (4.08, 1.14)
27         }
28
29     def read_distance(self,odom):
30         pose_x = odom.pose.pose.position.x
31         pose_y = odom.pose.pose.position.y
32
33         minDist = 10 ** 6
34
35         for obstacle,position in self.obstacles.items():
36             distance = math.sqrt(((pose_x - position[0]) ** 2) + ((pose_y - position[1]) ** 2))
37
38             if distance < minDist:
39                 closest = obstacle
40                 minDist = distance
41
42             obs = Obst()
43             obs.obstacle_name = closest
44             obs.distance = minDist
45
46             rospy.loginfo(f"Closest Obstacle Published: {closest} Position : {minDist}")
47
48             self.obstacle_publisher.publish(obs)
49
50     if __name__ == "__main__":
51         startNode = ClosestObstacle()
52         rospy.spin()
```



```
[INFO] [1655319308.979791, 2947.231000]: Closest Obstacle Published: bookshelf Distance : 3.062994304144607
[INFO] [1655319309.023761, 2947.274000]: Closest Obstacle Published: bookshelf Distance : 3.062995252039188
[INFO] [1655319309.025768, 2947.275000]: Closest Obstacle Published: bookshelf Distance : 3.0629962286858734
[INFO] [1655319309.067800, 2947.319000]: Closest Obstacle Published: bookshelf Distance : 3.0629971766348585
[INFO] [1655319309.083858, 2947.334000]: Closest Obstacle Published: bookshelf Distance : 3.062998124610639
[INFO] [1655319309.123706, 2947.375000]: Closest Obstacle Published: bookshelf Distance : 3.062999101340985
[INFO] [1655319309.167742, 2947.420000]: Closest Obstacle Published: bookshelf Distance : 3.06300004937117
[INFO] [1655319309.211808, 2947.464000]: Closest Obstacle Published: bookshelf Distance : 3.063000997428152
[INFO] [1655319309.255789, 2947.507000]: Closest Obstacle Published: bookshelf Distance : 3.0630019742421597
[INFO] [1655319309.257035, 2947.508000]: Closest Obstacle Published: bookshelf Distance : 3.063002922353547
[INFO] [1655319309.299865, 2947.545000]: Closest Obstacle Published: bookshelf Distance : 3.063003870491732
[INFO] [1655319309.322548, 2947.568000]: Closest Obstacle Published: bookshelf Distance : 3.0630048473894043
[INFO] [1655319309.363787, 2947.608000]: Closest Obstacle Published: bookshelf Distance : 3.0630057955820047
[INFO] [1655319309.407931, 2947.652000]: Closest Obstacle Published: bookshelf Distance : 3.0630067438021564
[INFO] [1655319309.451828, 2947.693000]: Closest Obstacle Published: bookshelf Distance : 3.063007720784536
[INFO] [1655319309.497096, 2947.733000]: Closest Obstacle Published: bookshelf Distance : 3.0630086690594513
[INFO] [1655319309.539778, 2947.772000]: Closest Obstacle Published: bookshelf Distance : 3.0630096173612578
[INFO] [1655319309.541424, 2947.773000]: Closest Obstacle Published: bookshelf Distance : 3.063010594427606
[INFO] [1655319309.583743, 2947.815000]: Closest Obstacle Published: bookshelf Distance : 3.063011542784006
[INFO] [1655319309.602960, 2947.834000]: Closest Obstacle Published: bookshelf Distance : 3.063012491167296
[INFO] [1655319309.643749, 2947.874000]: Closest Obstacle Published: bookshelf Distance : 3.0630134683175974
[INFO] [1655319309.687722, 2947.915000]: Closest Obstacle Published: bookshelf Distance : 3.06301441675554816
[INFO] [1655319309.731775, 2947.955000]: Closest Obstacle Published: bookshelf Distance : 3.0630153652202567
```

برای حرکت دادن ربات از teleop استفاده می کنیم و خروجی ترمینال نیز به شرح زیر است :



```
[INFO] [1655319379.335884, 3009.271000]: Closest Obstacle Published: dumpster Distance : 2.2760314353360314
[INFO] [1655319379.337791, 3009.271000]: Closest Obstacle Published: dumpster Distance : 2.2760381683600928
[INFO] [1655319379.379906, 3009.307000]: Closest Obstacle Published: dumpster Distance : 2.276044698039276
[INFO] [1655319379.412138, 3009.334000]: Closest Obstacle Published: dumpster Distance : 2.2760512224422027
[INFO] [1655319379.459853, 3009.378000]: Closest Obstacle Published: dumpster Distance : 2.2760579395443825
[INFO] [1655319379.503939, 3009.422000]: Closest Obstacle Published: dumpster Distance : 2.2760644536834236
[INFO] [1655319379.547967, 3009.467000]: Closest Obstacle Published: dumpster Distance : 2.276070962717071
[INFO] [1655319379.595894, 3009.512000]: Closest Obstacle Published: dumpster Distance : 2.276077663482566
[INFO] [1655319379.597781, 3009.512000]: Closest Obstacle Published: dumpster Distance : 2.276084162202159
[INFO] [1655319379.644198, 3009.558000]: Closest Obstacle Published: dumpster Distance : 2.276090655832226
[INFO] [1655319379.659955, 3009.563000]: Closest Obstacle Published: dumpster Distance : 2.27609734101543
[INFO] [1655319379.702493, 3009.601000]: Closest Obstacle Published: dumpster Distance : 2.276103824613804
[INFO] [1655319379.747962, 3009.638000]: Closest Obstacle Published: dumpster Distance : 2.2761103029145655
[INFO] [1655319379.791938, 3009.669000]: Closest Obstacle Published: dumpster Distance : 2.27611697275433
[INFO] [1655319379.836348, 3009.706000]: Closest Obstacle Published: dumpster Distance : 2.276123441458499
[INFO] [1655319379.883929, 3009.749000]: Closest Obstacle Published: dumpster Distance : 2.276129905184466
[INFO] [1655319379.928082, 3009.790000]: Closest Obstacle Published: dumpster Distance : 2.2761365595387413
[INFO] [1655319379.972018, 3009.826000]: Closest Obstacle Published: dumpster Distance : 2.2761430134037712
[INFO] [1655319380.019929, 3009.870000]: Closest Obstacle Published: dumpster Distance : 2.2761494626944745
[INFO] [1655319380.021697, 3009.871000]: Closest Obstacle Published: dumpster Distance : 2.2761561020787338
[INFO] [1655319380.063944, 3009.901000]: Closest Obstacle Published: dumpster Distance : 2.2761625419067744
[INFO] [1655319380.104470, 3009.934000]: Closest Obstacle Published: dumpster Distance : 2.27616897652799
[INFO] [1655319380.148031, 3009.971000]: Closest Obstacle Published: dumpster Distance : 2.2761756014497148
```

بخش ب)

در این بخش از سرویس ها استفاده می کنیم و عملیات مرحله الف را اینبار به جای msg با سرویس ها انجام می دهیم ، برای این منظور ابتدا سرویس را با ورودی و خروجی زیر تعریف می کنیم :

```
1   string obstacle_name
2   ---
3   float64 distance
```

سپس آنرا در نود جدیدی وارد کرده و با استفاده از آن و دادن نام مانع به عنوان ورودی ، فاصله ربات تا آن مانع را به شکل float بر می گردانیم :

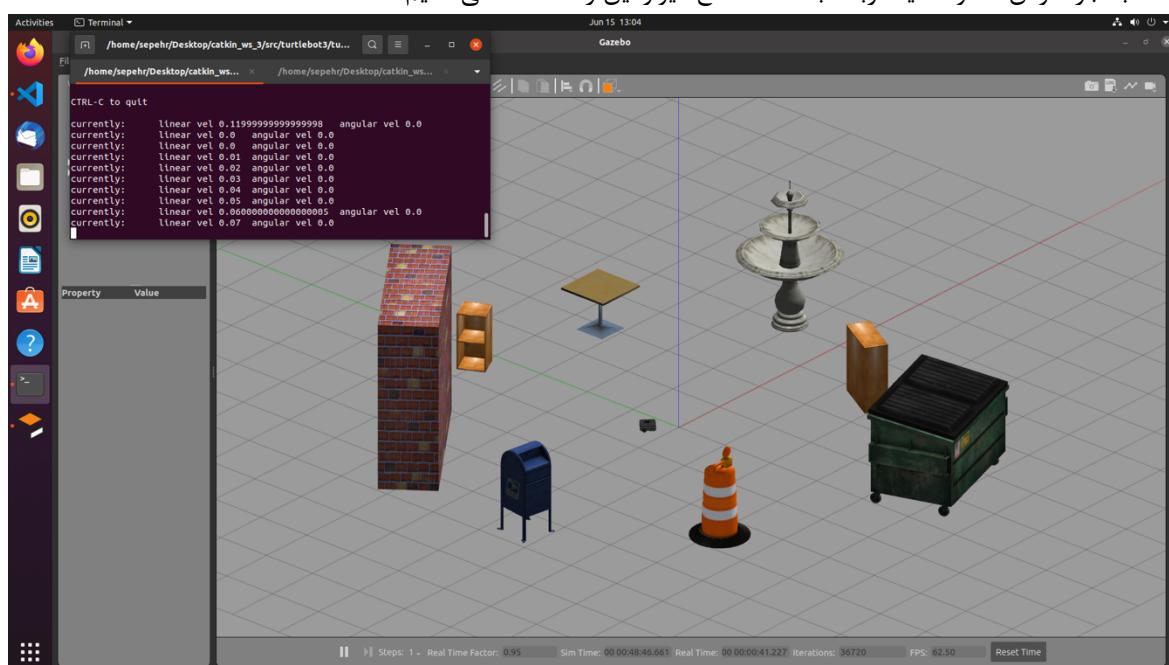
```
1  #!/usr/bin/python3
2
3  import rospy
4  from distance_caculator.msg import Obst
5  from distance_caculator.srv import GetDistance
6
7  class ClosestObstacle:
8
9      def __init__(self) -> None:
10         rospy.init_node("closest_obstacle_node" , anonymous=True)
11         # prepare service
12         rospy.wait_for_service('get_distance')
13         self.getDistService = rospy.ServiceProxy('get_distance', GetDistance)
14         self.obstacle_publisher = rospy.Publisher('/ClosestObstacle' , Obst , queue_size=10)
15
16         # obstacle details
17         self.obstacles={
18             "bookshelf" : (2.64, -1.55),
19             "dumpster" : (1.23, -4.57),
20             "barrel" : (-2.51, -3.08),
21             "postbox" : (-4.47, -0.57),
22             "brick_box" : (-3.44, 2.75),
23             "cabinet" : (-0.45, 4.05),
24             "cafe_table": (1.91, 3.37),
25             "fountain" : (4.08, 1.14)
26         }
27
28     def calculate_distance(self):
29         while True :
30             minDist = 10 ** 6
31             for obstacle in self.obstacles.keys():
32                 res = self.getDistService(obstacle)
33                 currentDist = res.distance
34                 if currentDist < minDist:
35                     closest = obstacle
36                     minDist = currentDist
37
38             obs = Obst()
39             obs.obstacle_name = closest
40             obs.distance = minDist
41
42             rospy.loginfo(f"Closest Obstacle Published: {closest} Distance : {minDist}")
43             self.obstacle_publisher.publish(obs)
44
45     if __name__ == "__main__":
46         startNode = ClosestObstacle()
47         startNode.calculate_distance()
48         rospy.spin()
```

بخش ج)

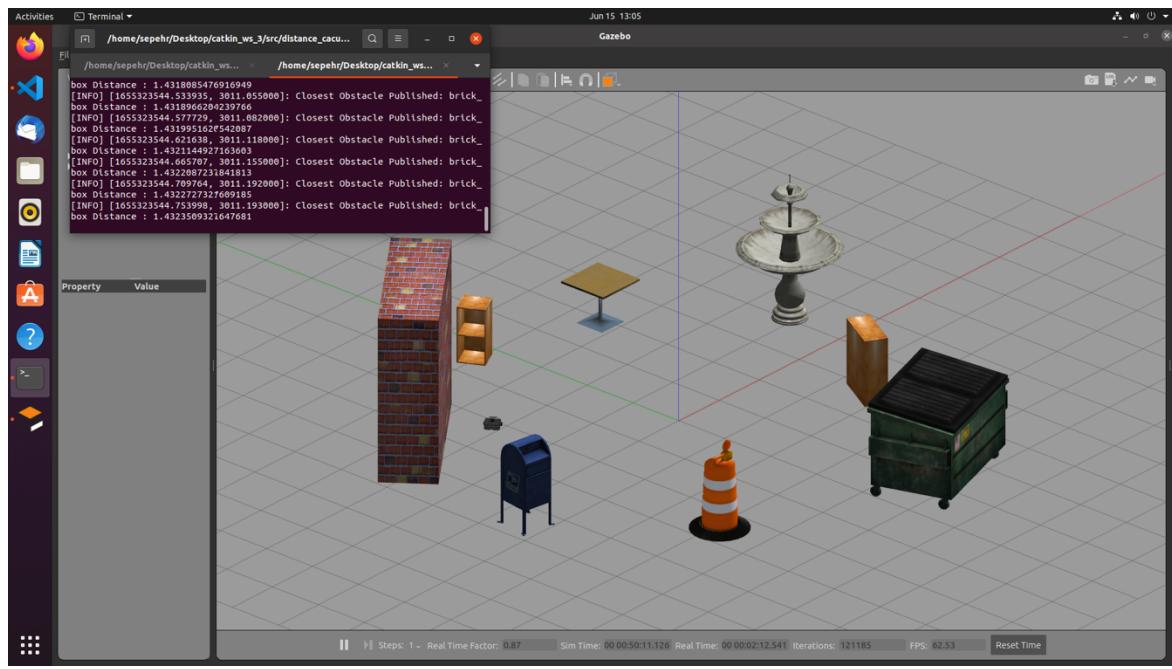
در این بخش با استفاده از laserscan در فاصله ۱.۵ متری موانع ربات شروع به چرخش مطابق زاویه خواسته شده می کند :

```
1  #!/usr/bin/python3
2  import rospy
3  from nav_msgs.msg import Odometry
4  from distance_caculator.msg import Obst
5  from math import radians,pi
6  from sensor_msgs.msg import LaserScan
7  import numpy as np
8  import tf
9  from geometry_msgs.msg import Twist
10
11
12 class AvoidObstacleNode:
13
14     def __init__(self) -> None:
15
16         rospy.init_node("avoid_obstacle_node", anonymous=True)
17         self.cmd_publisher = rospy.Publisher('/cmd_vel', Twist, queue_size=10)
18         self.angular_speed=0.1
19         self.obstacles={
20             "bookshelf" : (2.64, -1.55),
21             "dumpster" : (1.23, -4.57),
22             "barrel" : (-2.51, -3.08),
23             "postbox" : (-4.47, -0.57),
24             "brick_box" : (-3.44, 2.75),
25             "cabinet" : (-0.45, 4.05),
26             "cafe_table": (1.91, 3.37),
27             "fountain" : (4.08, 1.14)
28         }
29
30
31     > def angular_error(self, diff):-
32
33     > def get_heading(self):-
34
35     > def rotate(self,remaining):-
36
37     > def check_distance(self):-
38
39
40     if __name__ == "__main__":
41         startNode = AvoidObstacleNode()
42         startNode.check_distance()
```

حالا با اجرا کردن کد و هدایت ربات به سمت مانع دیوار این را مشاهده می کنیم :



هرگاه فاصله کمتر از ۱.۵ شود ربات شروع به چرخش به راست می کند :



این عملیات یعنی خواندن از تاپیک laserscan و اعمال چرخش لازم در این بخش از کد انجام می شود:

```

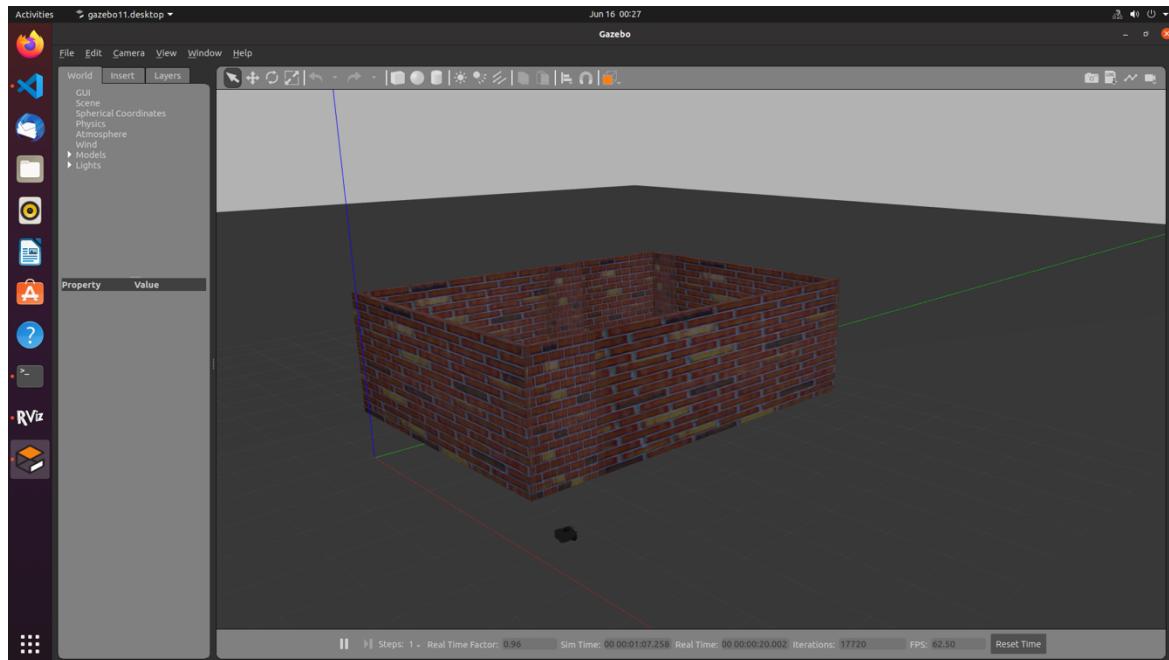
msg = rospy.wait_for_message("/scan", LaserScan, timeout=1)
ranges=np.array(msg.ranges)
min_ind=np.argmin(ranges)

remaining = self.angular_error(radians(180-min_ind))
self.rotate(remaining)

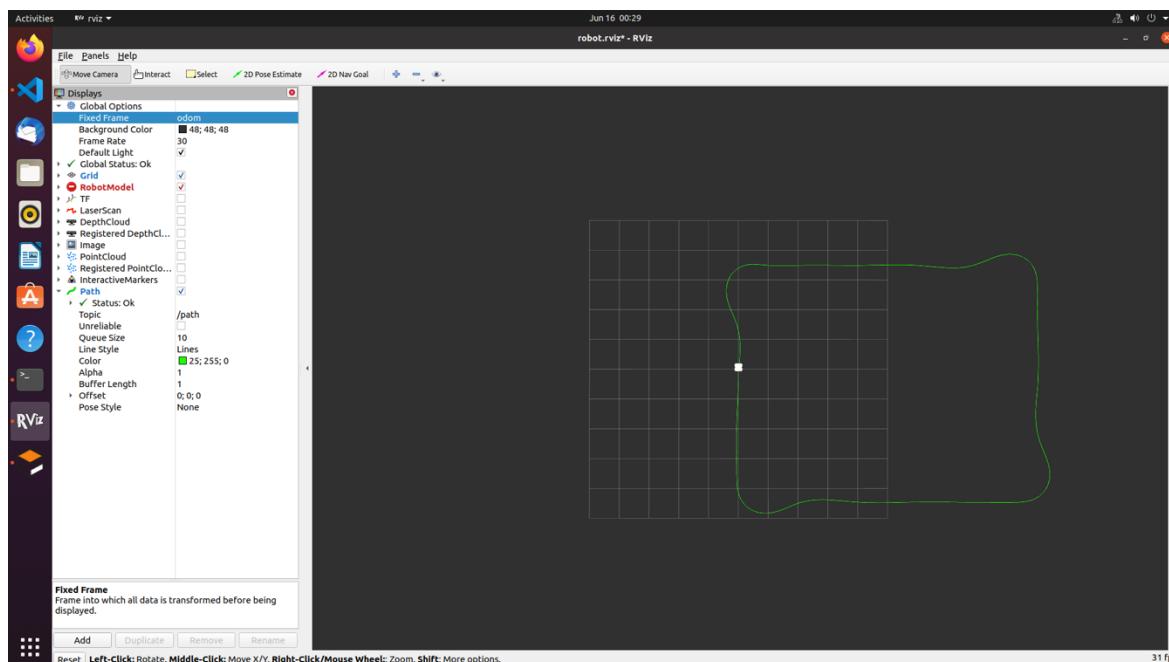
```

سناریو ۲ : بخش (الف)

ابتدا با استفاده از فایل `include` و `launch` کردن محیط مورد نظر را اجرا می کیم و ربات را در نقطه صفر و صفر قرار می دهیم و بنابر کد نوشته شده ربات دیوار را دنبال می کند :



مسیر rviz نیز به شکل زیر است :



کد نوشته شده برای این بخش نیز به شرح زیر است که در آن پس از تعریف مقادیر مناسب برای PID کنترولر ، در تابع fallow_wall نزدیکی به دیوار و دنبال کردن آن چک می شود. کمترین فاصله از دیوار را هم با استفاده از آخرین پیام منتشر شده روی تاپیک laserscan پیدا می کیم :

```

1  #!/usr/bin/python3
2
3  import rospy
4  from geometry_msgs.msg import Twist
5  from sensor_msgs.msg import LaserScan
6
7  class PIDController():
8      def __init__(self):
9          rospy.init_node('fallow_wall', anonymous=False)
10
11         self.ki_angle = 0.1
12         self.kp_angle = 0.7
13         self.kd_angle = 8
14
15         self.kd_distance = 0.005
16         self.ki_distance = 0.3
17         self.kp_distance = 1
18
19         self.cmd_vel = rospy.Publisher('/cmd_vel', Twist, queue_size=10)
20
21
22
23     def distance_from_wall(self):
24         laser_data = rospy.wait_for_message("/scan", LaserScan)
25         rng = laser_data.ranges[:180]
26         return min(rng)
27
28
29     def follow_wall(self):
30         d = self.distance_from_wall()
31         sum_i_theta = 0
32         prev_theta_error = 0
33
34         move_cmd = Twist()
35         move_cmd.angular.z = 0
36         move_cmd.linear.x = self.ki_distance
37
38
39         while not rospy.is_shutdown():
40             self.cmd_vel.publish(move_cmd)
41
42             err = d - self.kp_distance
43             sum_i_theta += err * self.kd_distance
44
45             P = self.kp_angle * err
46             I = self.ki_angle * sum_i_theta
47             D = self.kd_angle * (err - prev_theta_error)
48
49             move_cmd.angular.z = P + I + D
50             prev_theta_error = err
51             move_cmd.linear.x = self.ki_distance
52             d = self.distance_from_wall()
53
54     if __name__ == '__main__':
55         start = PIDController()
56         start.follow_wall()
57

```

فایل لانج :

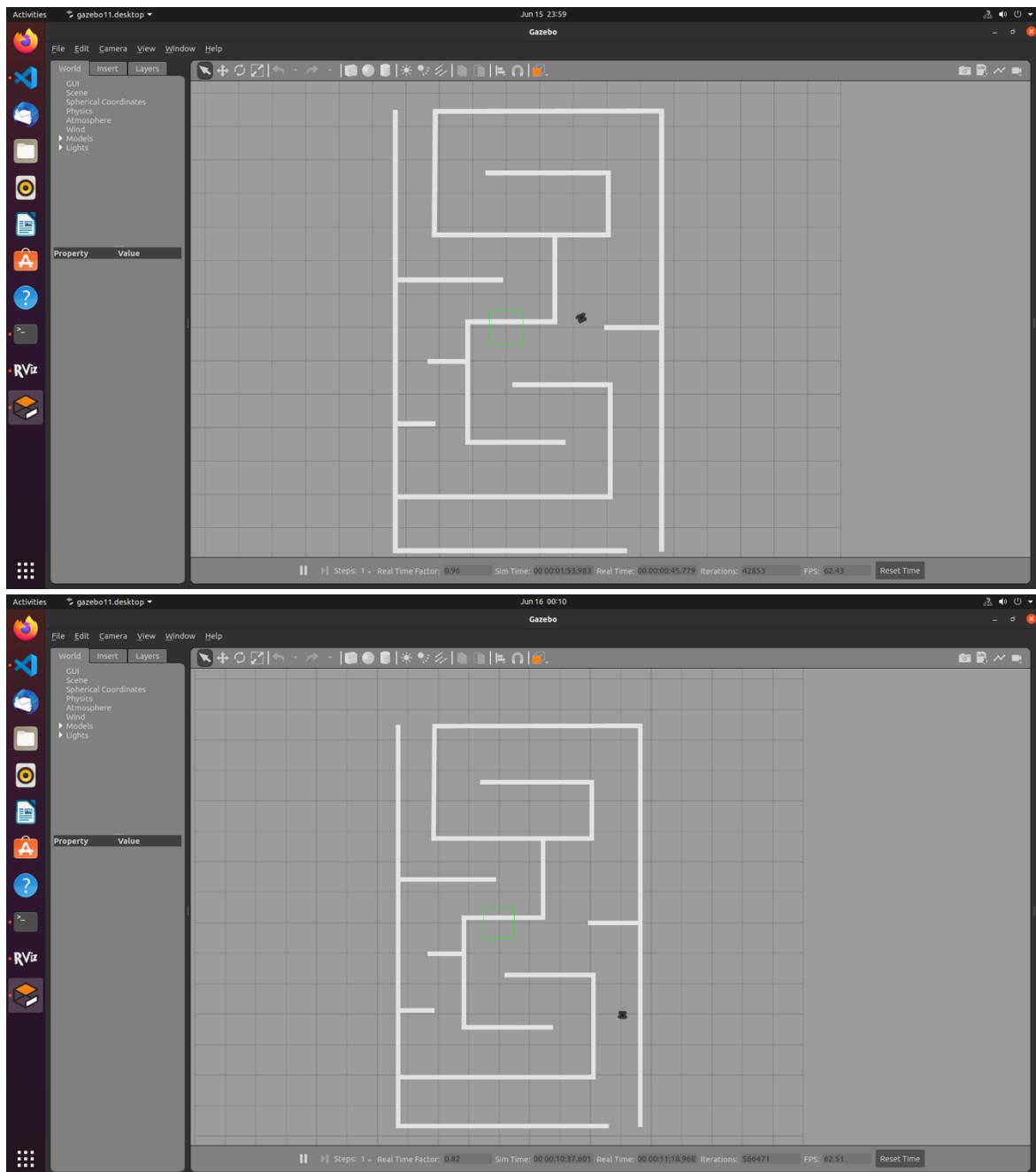
```

1 <launch>
2   <node pkg="wall_following" type="follow_wall.py" name="follow_wall" output="screen" ></node>
3   <arg name="model" default="$(env TURTLEBOT3_MODEL)" doc="model type [burger, waffle, waffle_pi]" />
4   <arg name="x_pos" default="0.0"/>
5   <arg name="y_pos" default="0.0"/>
6   <arg name="z_pos" default="0.0"/>
7
8   <include file="$(find gazebo_ros)/launch/empty_world.launch">
9     <arg name="world_name" value="$(find wall_following)/worlds/square.world"/>
10    <arg name="paused" value="false"/>
11    <arg name="use_sim_time" value="true"/>
12    <arg name="gui" value="true"/>
13    <arg name="headless" value="false"/>
14    <arg name="debug" value="false"/>
15  </include>
16  <node pkg="wall_following" type="monitor.py" name="monitor" output="screen"></node>
17
18  <param name="robot_description" command="$(find xacro)/xacro --inorder $(find turtlebot3_description)/urdf/turtlebot3_$(arg model).urdf.xacro" />
19  <include file="$(find turtlebot3viz_launchers)/launch/view_robot.launch" />
20
21  <node pkg="gazebo_ros" type="spawn_model" name="spawn_urdf" args="-urdf -model turtlebot3_$(arg model) -x $(arg x_pos) -y $(arg y_pos) -z $(arg z_pos) -param robot_description" />
22 </launch>

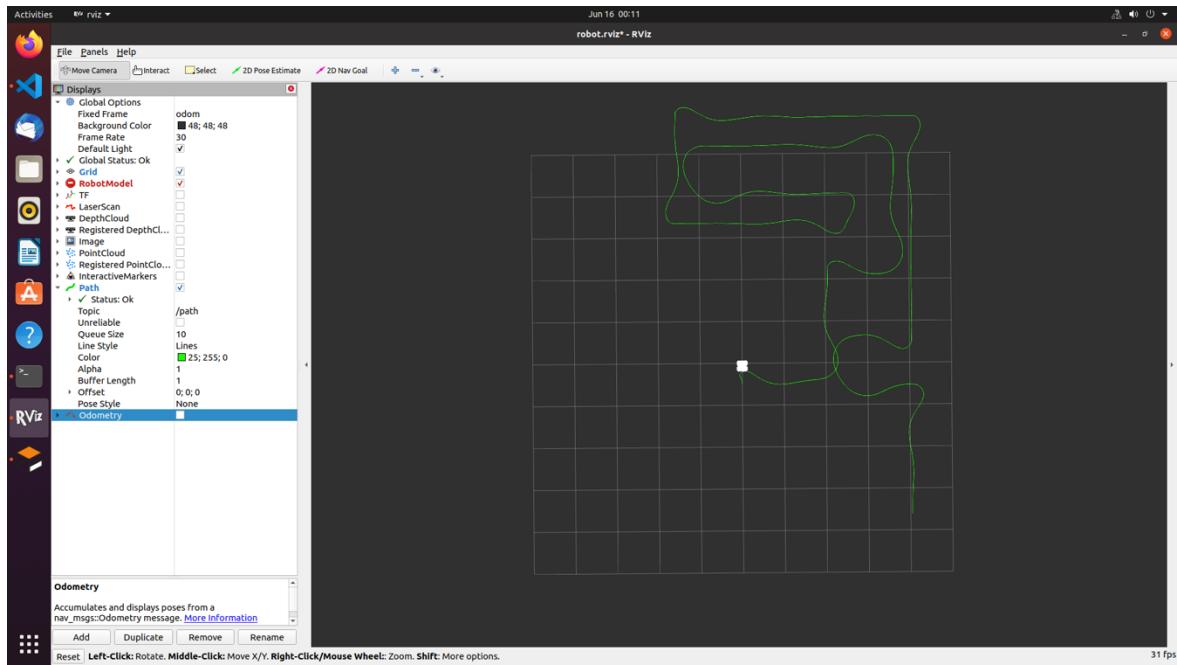
```

بخش ب)

برای این بخش نیز مانند بخش قبلی محیط maze را لود کرده و ربات را در نقطه گفته شده می گذاریم و ربات شروع به دنبال کردن دیوار های maze می کند:



مسیر rviz نیز به شکل زیر است :

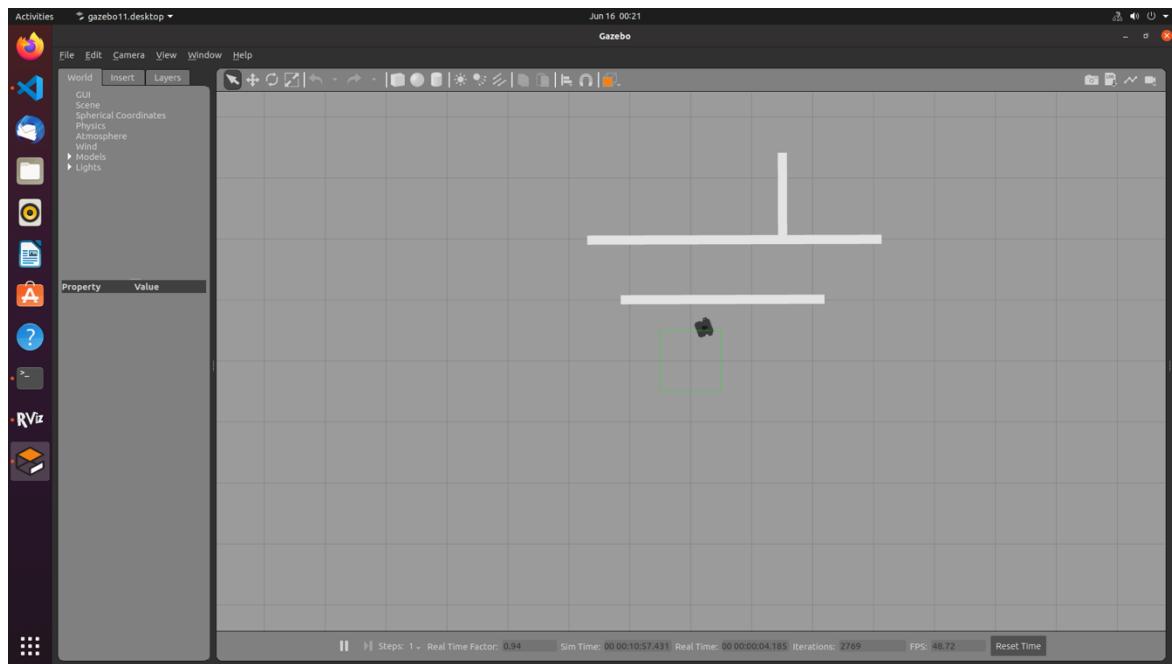


کد نوشته شده نیز به شرح زیر است و در آن مانند بخش قبلی کنترولر PID را پیاده کردیم و در هر لحظه فاصله ربات را تا نزدیکترین دیوار در نظر می گیریم. ربات شروع به حرکت کرده تا جایی که از heading maze خارج شود :

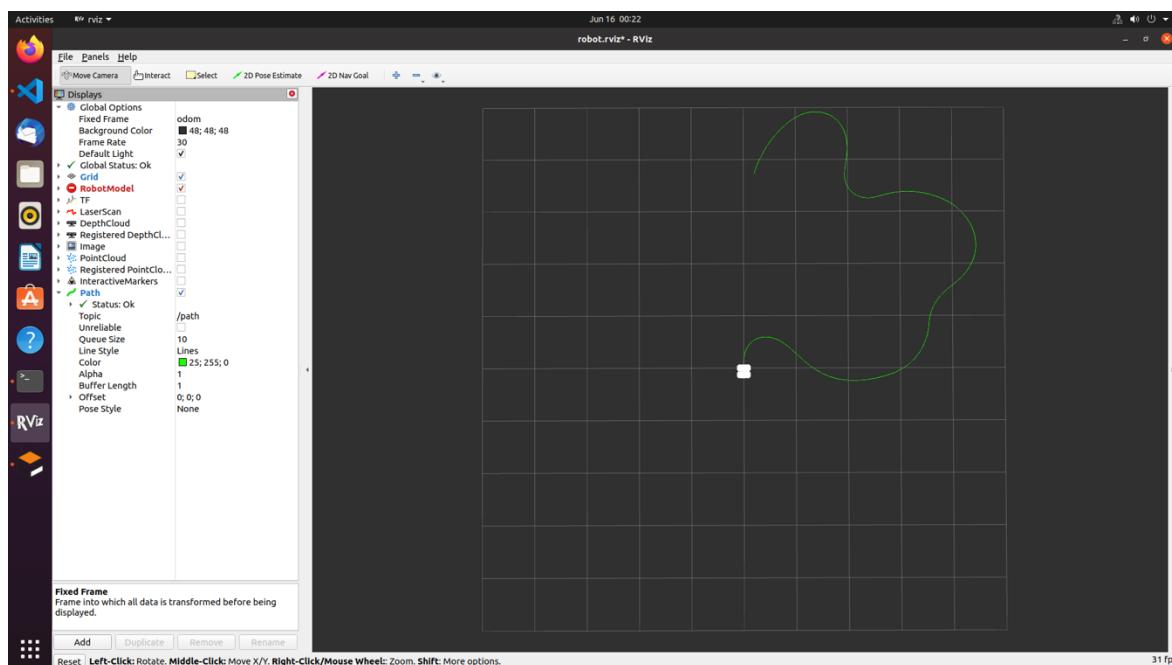
```
1  #!/usr/bin/python3
2
3  import rospy
4  from geometry_msgs.msg import Twist
5  from sensor_msgs.msg import LaserScan
6
7  class PIDController():
8      def __init__(self):
9          rospy.init_node('maze_fallow', anonymous=False)
10
11         self.ki_angle = 0.1
12         self.kp_angle = 0.6
13         self.kd_angle = 7
14
15         self.kd_distance = 0.005
16         self.ki_distance = 0.09
17         self.kp_distance = 0.5
18
19         self.cmd_vel = rospy.Publisher('/cmd_vel', Twist, queue_size=10)
20
21     def distance_from_wall(self):
22         laser_data = rospy.wait_for_message("/scan", LaserScan)
23         rng = laser_data.ranges[5:180]
24         return min(rng)
25
26     def front_distance_from_wall(self):
27         laser_data = rospy.wait_for_message("/scan", LaserScan)
28         rng = laser_data.ranges[0:5]
29         return min(rng)
30
31     def follow_wall(self):
32
33         d = self.distance_from_wall()
34         sum_i_theta = 0
35         prev_theta_error = 0
36
37         move_cmd = Twist()
38         move_cmd.angular.z = 0
39         move_cmd.linear.x = self.ki_distance
40
41
42         while not rospy.is_shutdown():
43             front_d = self.front_distance_from_wall()
44             if front_d <= self.kp_distance :
45                 front_d = self.front_distance_from_wall()
46                 twist = Twist()
47                 twist.angular.z = -0.2
48                 self.cmd_vel.publish(twist)
49
50             else:
51                 self.cmd_vel.publish(move_cmd)
52
53                 err = d - self.kp_distance
54                 sum_i_theta += err * self.kd_distance
55
56                 P = self.kp_angle * err
57                 I = self.ki_angle * sum_i_theta
58                 D = self.kd_angle * (err - prev_theta_error)
59
60                 prev_theta_error = err
61                 move_cmd.angular.z = P + I + D
62                 move_cmd.linear.x = self.ki_distance
63
64                 d = self.distance_from_wall()
65
66     if __name__ == '__main__':
67         start = PIDController()
68         start.follow_wall()
69
70
```

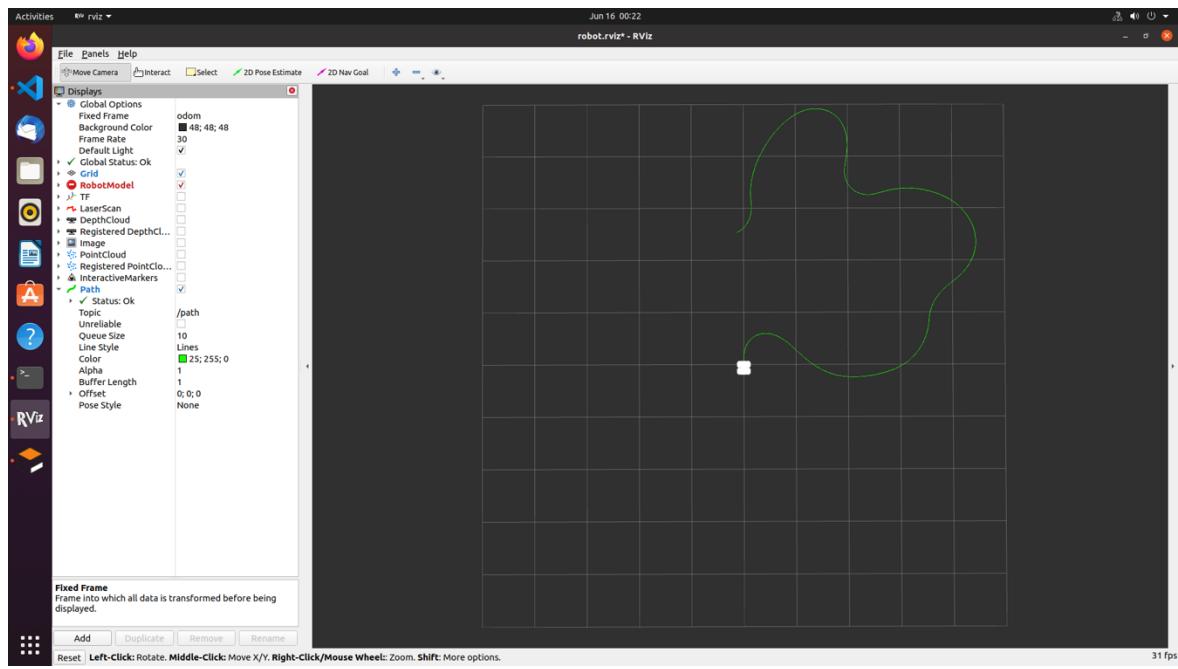
بخش ج)

در این قسمت ربات به سمت هدف مشخص شده حرکت می کند ولی اگر در این بین دیوار وجود داشته باشد آنرا دنبال می کند تا بتواند یک مسیر مستقیم بدون دیوار به سمت هدف پیدا کند :



مسیر rviz نیز به شکل زیر است :





کد نیز به شکل زیر است و در آن نقطه مقصد در نظر گرفته شده است و فاصله تا آن (تابع `distance_from_goal`) را چک می کنیم تا به نقطه هدف برسیم. با اعمال کنترولر PID در تابع `follow_wall` به دنبال کردن دیوار و هم زمان نزدیک شدن به هدف می پردازیم.

```

4   import tf
5   from geometry_msgs.msg import Twist
6   from sensor_msgs.msg import LaserScan
7   from nav_msgs.msg import Odometry
8   from math import atan2
9   from cmath import sqrt
10
11  class PIDController():
12      def __init__(self):
13          rospy.init_node('goal_fallow', anonymous=False)
14
15          self.goal_x = 3
16          self.goal_y = -1
17
18          self.currentX = 0
19          self.currentY = 0
20
21          self.ki_angle = 0.08
22          self.kp_angle = 0.4
23          self.kd_angle = 8
24
25          self.kd_distance = 0.005
26          self.ki_distance = 0.02
27          self.kp_distance = 0.9
28
29          self.cmd_vel = rospy.Publisher('/cmd_vel', Twist, queue_size=5)
30
31
32
33      def distance_from_wall(self):
34          laser_data = rospy.wait_for_message("/scan", LaserScan)
35          rng = laser_data.ranges[10:190]
36          return min(rng)
37
38      def front_distance_from_wall(self):
39          laser_data = rospy.wait_for_message("/scan", LaserScan)
40          rng1 = laser_data.ranges[0:10]
41          rng2 = laser_data.ranges[350:359]
42          return min(min(rng1),min(rng2))
43
44      def distance_from_goal(self):...
45
46      def angle_to_goal(self):...
47
48      def get_heading(self):...
49
50      def move_to_goal(self):...
51
52      def follow_wall(self):...
53
54
55      def run(self):
56          goal_d = self.distance_from_goal()
57          while not rospy.is_shutdown():
58              goal_d = self.distance_from_goal()
59              if goal_d.real > 0.35:
60                  d = self.distance_from_wall()
61                  if d > self.kp_distance + 0.2:
62                      self.move_to_goal()
63                  else:
64                      self.follow_wall()
65
66              else:
67                  move_cmd = Twist()
68                  move_cmd.linear.x = 0.0
69                  move_cmd.angular.z = 0.0
70                  break
71
72
73      if __name__ == '__main__':
74          start = PIDController()
75          start.run()
76
77
78
79
80
81
82
83
84
85
86
87
88
89
90
91
92
93
94
95
96
97
98
99
100
101
102
103
104
105
106
107
108
109
110
111
112
113
114
115
116
117
118
119
120
121
122
123
124
125
126
127
128
129
130

```