Turtlesim_gotogoal_Kcontrol 03/22/23

ROS1 Example

Contents

1. Terminal 1 ROSCORE.....2

2. Terminal 2 Turtlesim.....2

3. Terminal 3 cd Desktop Run python script.....3

4. Check the Result.....3

`Turtlesim_gotogoal_Kcontrol_py` 4

1. Terminal 1 ROSCORE
harman@harman-VirtualBox:~$ roscore
... logging to /home/harman/.ros/log/2e9aee08-8e23-11ec-ac71-080027ed3098/roslaunch-harman-VirtualBox-7008.log
Checking log directory for disk usage. This may take awhile.
Press Ctrl-C to interrupt
Done checking log file disk usage. Usage is <1GB.

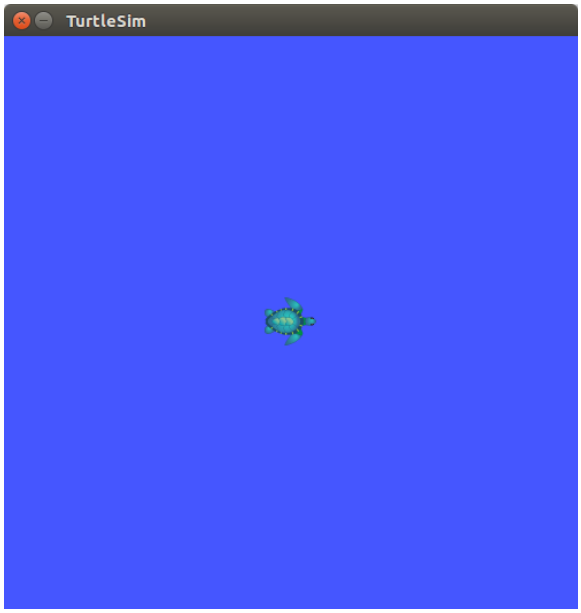
started roslaunch server http://harman-VirtualBox:36377/
ros_comm version 1.12.17

SUMMARY
========

PARAMETERS
*** /rosdistro: kinetic**
*** /rosversion: 1.12.17**

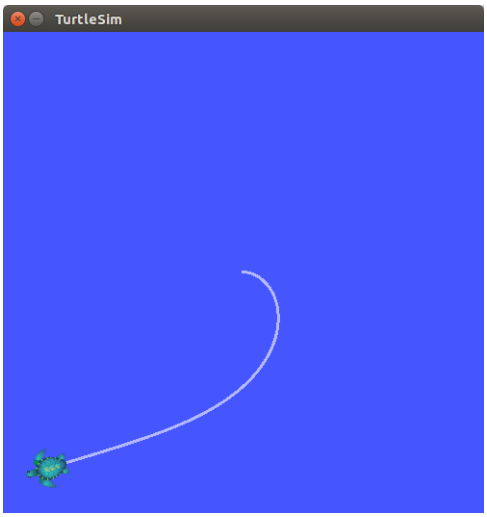
2. Terminal 2 Turtlesim
harman@harman-VirtualBox:~$ rosrun turtlesim turtlesim_node
[INFO] [1644904286.781127068]: Starting turtlesim with node name /turtlesim
[INFO] [1644904286.786197180]: Spawning turtle [turtle1] at x=[5.544445], y=[5.544445], theta=[0.000000]

(Set Always on Top)



3. Terminal 3 cd Desktop   Run python script

```
harman@harman-VirtualBox:~$ cd Desktop/
harman@harman-VirtualBox:~/Desktop$ python Turtlesim_gotogoal_Kcontrol.py
Set your x goal:1
Set your y goal:1
Set your tolerance:.01
('velocity =', 3.213376056424147)
('Angular Velocity =', -4.71238898038469)
('The answer is', 4)
('x =', 5.5444)
('y =', 5.5444)
('velocity =', 3.213376056424147)
('Angular Velocity =', -4.71238898038469)
('The answer is', 4)
('x =', 5.8398)
('y =', 5.4646)
……..
('velocity =', 0.005220153254455264)
('Angular Velocity =', -0.003374688347163257)
('The answer is', 4)
('x =', 1.0095)
('y =', 1.0029)
^Z
[1]+  Stopped                 python Turtlesim_gotogoal_Kcontrol.py
```



4. Check the Result

```
harman@harman-VirtualBox:~/Desktop$ rostopic echo /turtle1/pose -n1
x: 1.00947260857
y: 1.00285518169
theta: -2.84877228737
linear_velocity: 0.0
angular_velocity: 0.0
```

Turtlesim_gotogoal_Kcontrol_py

```
#!/usr/bin/env python            Turtlesim_gotogoal_Kcontrol.py   tur-
tlesim_cleaner/src/gotogoal.py GitHub

import rospy
from geometry_msgs.msg  import Twist
from turtlesim.msg import Pose
from math import pow,atan2,sqrt

class turtlebot():

    def __init__(self):
        #Creating our node,publisher and subscriber
        rospy.init_node('turtlebot_controller', anonymous=True)
        self.velocity_publisher = rospy.Publisher('/turtle1/cmd_vel',
Twist, queue_size=10)
        self.pose_subscriber = rospy.Subscriber('/turtle1/pose', Pose,
self.callback)
        self.pose = Pose()
        self.rate = rospy.Rate(10)

    #Callback function implementing the pose value received
    def callback(self, data):
        self.pose = data
        self.pose.x = round(self.pose.x, 4)
        self.pose.y = round(self.pose.y, 4)

    def get_distance(self, goal_x, goal_y):
        distance = sqrt(pow((goal_x - self.pose.x), 2) + pow((goal_y -
self.pose.y), 2))
        return distance

    def move2goal(self):
        goal_pose = Pose()
        goal_pose.x = input("Set your x goal:")
        goal_pose.y = input("Set your y goal:")
        distance_tolerance = input("Set your tolerance:")
        vel_msg = Twist()


        while sqrt(pow((goal_pose.x - self.pose.x), 2) + pow((goal_pose.y -
self.pose.y), 2)) >= distance_tolerance:

            #Porportional Controller    Reset Constants    to (0.5,2)
            #linear velocity in the x-axis:
            vel_msg.linear.x = .5 * sqrt(pow((goal_pose.x - self.pose.x),
2) + pow((goal_pose.y - self.pose.y), 2))
            vel_msg.linear.y = 0
            vel_msg.linear.z = 0

            #angular velocity in the z-axis:
            vel_msg.angular.x = 0
```

```
            vel_msg.angular.y = 0
            vel_msg.angular.z = 2 * (atan2(goal_pose.y - self.pose.y,
goal_pose.x - self.pose.x) - self.pose.theta)

            #Publishing our vel_msg
            self.velocity_publisher.publish(vel_msg)
            self.rate.sleep()

            print("velocity =", vel_msg.linear.x)
            print("Angular Velocity =", vel_msg.angular.z)
            print("The answer is", 2*2)
            print("x =", self.pose.x)
            print("y =", self.pose.y)

#Stopping our robot after the movement is over
        vel_msg.linear.x = 0
        vel_msg.angular.z = 0
        self.velocity_publisher.publish(vel_msg)

        rospy.spin()

if __name__== '__main__':
    try:
        #Testing our function
        x = turtlebot()
        x.move2goal()

    except rospy.ROSInterruptException:
        pass


#RESULT:
 #  harman@D104-45931:~/Desktop$ python gotogoal.py
# Set your x goal:1.0
# Set your y goal:1.0
# Set your tolerance:0.5
```