|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Task. No.: | 3 | Points: | 5 | Turtlesim publisher |

|  |
| --- |
| Objectives:  1. Using ROS topics. 2. See topic lists. 3. See message types. |

|  |
| --- |
| Description: We saw the turtlesim application in ROS. The goal of this exercise is to program turtlesim using rospyPy to move it around the workspace. |

|  |  |
| --- | --- |
| Step | Action |
| 1 | Start roscore. |
| 2 | Run turtlesim node in another terminal. |
| 3 | Show list of topics which is publishing by turtlesim\_node.  $ tostopic list /rosout /rosout_agg /turtle1/cmd_vel /turtle1/color_sensor /turtle1/pose |
| 4 | To move the turtle inside the turtlesim application, publish the linear and angular velocity to the */turtle1/cmd\_vel* topic. |
| 5 | Check the type of the */turtle1/cmd\_vel* topic by using the following command.  $ rostopic type /turtle1/cmd_vel geometry_msgs/Twist |
| 6 | See the *geometry\_msgs/Twist* definition with the following command.  $ rosmsg show geometry_msgs/Twist  The twist message has two subsections: linear velocity and angular velocity. |
| 7 | Use *rostopic* with the following command to publishe the linear.x = 0.1 velocity to the turtlesim node. You don’t need to enter the complete command. Just type *rostopic pub /turtle1/ cmd\_vel*, and use the Tab key to autocomplete other fields.  $ rostopic pub /turtlel/cmdvel geometry_msgs/Twist "linear: x:0.1 y:0 z:0 angular: x:0 y: 0 z:0" |
| 8 | Create a new node called *move\_turtle* and publish a twist message to the *turtlesim* node. Save the below code as *move\_turtle.py*, and change the permission to executable.  #!/usr/bin/env python import rospy «Importing Twist message: Used to send velocity to Turtlesim from geometryjnsgs.msg import Twist «Handling command line arguments import sys «Function to move turtle: Linear and angular velocities are arguments def move_turtle(lin_vel,ang_vel): rospy.init_node('move_turtle', anonymous=False) #The /turtlel/cmdvel is the topic in which we have to send Twist messages pub = rospy.Publisher('/turtlel/cmd_vel', Twist, queue_size=10) rate = rospy.Rate(10) « 10hz «Creating Twist message instance vel = Twist() while not rospy.is_shutdown():  «Adding linear and angular velocity to the message vel.linear.x = lin_vel vel.linear.y = 0 vel.linear.z = 0 vel.angular.x = 0 vel.angular.y = 0 vel.angular.z = ang_vel rospy.loginfo("Linear Vel = %f: Angular Vel = %f",lin_vel,ang_vel) #Publishing Twist message pub.publish(vel) rate.sleep() if_name_  == '_main_' : try: #Providing linear and angular velocity through command line move_turtle(float(sys.argv[1]),float(sys.argv[2])) except rospy.ROSInterruptException:  pass |
| 9 | Run the *roscore* and *turtlebot* nodes.  $roscore  $ rosrun turtlesim turtlesim_node |
| 10 | Run the *move\_turtle.py* node along with the command-line arguments, which are 0.2 and 0.1. That is, linear velocity = 0.2 m/s and angular velocity = 0.1 rad/s.  $ rosrun hellow_world move_turtle.py 0.2 0.1 |
|  |  |

 This is shared under a Creative Commons Attribution-NonCommercial-ShareAlike 4.0 International Public License