Mini Project

Aim: To make a Robot that rotates around the circumference of a circle in clockwise and Anti-Clockwise direction.

Theory:

What is ROS:

ROS is an open-source, meta-operating system for your robot. It provides the services you would expect from an operating system, including hardware abstraction, low-level device control, implementation of commonly-used functionality, message-passing between processes, and package management. It also provides tools and libraries for obtaining, building, writing, and running code across multiple computers. ROS is similar in some respects to 'robot frameworks,' such as Player, YARP, Orocos, CARMEN, Orca, MOOS, and Microsoft Robotics Studio.

The ROS runtime "graph" is a peer-to-peer network of processes (potentially distributed across machines) that are loosely coupled using the ROS communication infrastructure. ROS implements several different styles of communication, including synchronous RPC-style communication over services, asynchronous streaming of data over topics, and storage of data on a Parameter Server. These are explained in greater detail in our Conceptual Overview.

ROS is not a real-time framework, though it is possible to integrate ROS with real-time code. The Willow Garage PR2 robot uses a system called pr2_etherCAT, which transports ROS messages in and out of a real-time process. ROS also has seamless integration with the Orocos Real-time Toolkit.

Working:

- The robot is programmed in a way that it performs clockwise and anti-clockwise movements around a circular path.
- There are two motions possible for a robot: Linear and Angular.
- Linear motion is a motion along a straight line and angular motion is the rotational motion around an imaginary axis.
- In order to perform circular motion we have to provide both motions to the robot, ie: Linear and Angular.
- After providing both the motions, the robot travels along a circular path.
- Now to program clockwise and anti-clockwise movements the robots angular and linear values are reversed at small intervals.
- For eq: if linear value x = 0.5 and angular value z = 0.5 for the first interval it will be linear value x = -0.5 and angular value z = -0.5 in the next interval and so on.
- This motion will be stopped as soon as the shut down command is given.

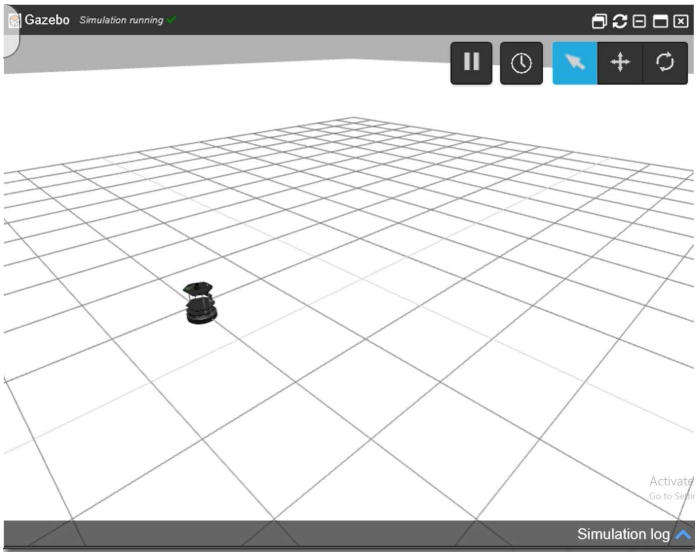
Steps:

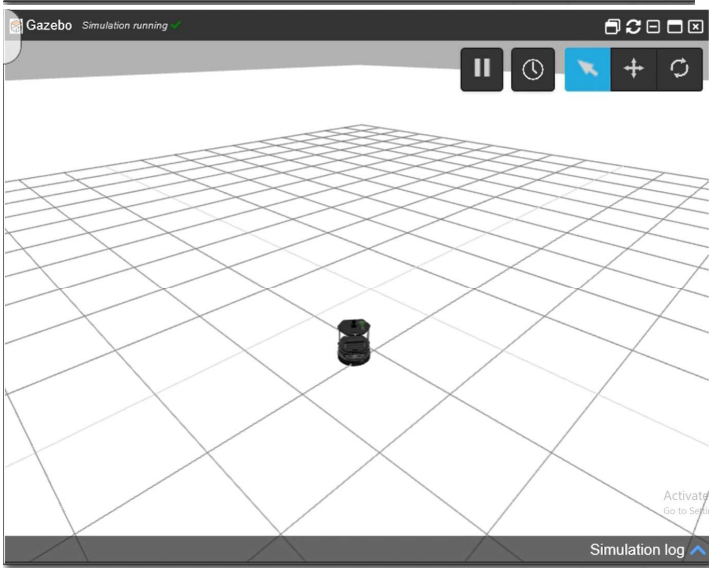
1. Create a new ROS Project in ROS Development Studio.
2. Open the Project.
3. All the project is to be designed in the Catkin Workspace, so to enter the Catkin Workspace write the command “cd catkin_ws/src” in the shell.
4. Here the project can be made, In order to make a new package write the command “catkin_create_pkg [package_name] rospy” in the shell. Assume the package name to be “move_bot”.
5. Now create a new python file under “catkin_ws/src/move_bot/src” for writing the python code. Assume the file name to be “move_rob.py”.
6. Create a simulation with empty world and a “turtlebot2” robot which can be viewed in the Gazebo Terminal.
7. In order to run the simulation we need to source the setup bash using the command “source devel setup.bash”.
8. Now write the code into the python file that you created.
9. Run the package using the rosrun command “rosrun <package> <node> <paramter>”. In our case it is “rosrun move_bot move_rob.py”
10. You can see the output in the Gazebo Terminal.

Code:

```
#! /usr/bin/env python
import rospy
from geometry_msgs.msg import Twist
from time import sleep

rospy.init_node('move_bot')
publisher = rospy.Publisher('/cmd_vel',Twist,queue_size=1)
rr = True

while not rospy.is_shutdown():
   msg = Twist()
   msg.angular.z = 0.5 if rr else -0.5
   publisher.publish(msg)
   msg.linear.x = 0.5 if rr else -0.5
   publisher.publish(msg)

   rospy.loginfo("Other")
   rr = not rr
   sleep(15)
```

Output:





Conclusion: Successfully programmed a Robot to perform specific movements using python script in ROS Development Studio.