Lab 2
Control of the TurtleSim using ROS

February 1, 2022

Learning Objectives

1. Configure nodes to exchange information via topics within the publisher/subscriber framework of ROS.
2. Write custom publisher/subscriber nodes in ROS to listen to a topic and perform computations when new information is available on that topic.
3. Learn how to interact and control a small, mobile robot such as the Turtlebot in a simulated environment (TurtleSim).
4. Leverage pre-existing hardware drivers and other nodes in the ROS repository and integrate them together with custom-written nodes as part of a robot application.

Overview

The primary objective of this lab is to introduce you to Robot Operating System (ROS). Similar to other operating systems, ROS provides various services including hardware abstraction, low-level device control, implementation of commonly-used functionality, message-passing between processes, and package management[1]. In this lab, you will use the aforementioned services of ROS to program a robotic vehicle (i.e. a Turtlebot or TurtleSim) to respond to commands from custom scripts.

Nodes for interfacing with the velocity control already exist in ROS and will be leveraged in this lab for publishing and subscribing to ROS topics. Your node running on the simulated robot will subscribe to and publish topics to drive the robot accordingly. You will complete a number of ROS tutorials to gain familiarity with control of a mobile robot.

Lab Requirements

Create a velocity controller for driving the TurtleSim robot from a computer that has a keyboard connected to it. Use ROS as the framework that supports the message-passing between nodes and provides the driver for interfacing with the keyboard.

Below are some steps for helping you complete this lab. They are not intended to be a step-by-step set of instructions for the assignment, but instead, they are intended to assist you with various

[1]ROS Introduction, `http://wiki.ros.org/ROS/Introduction`

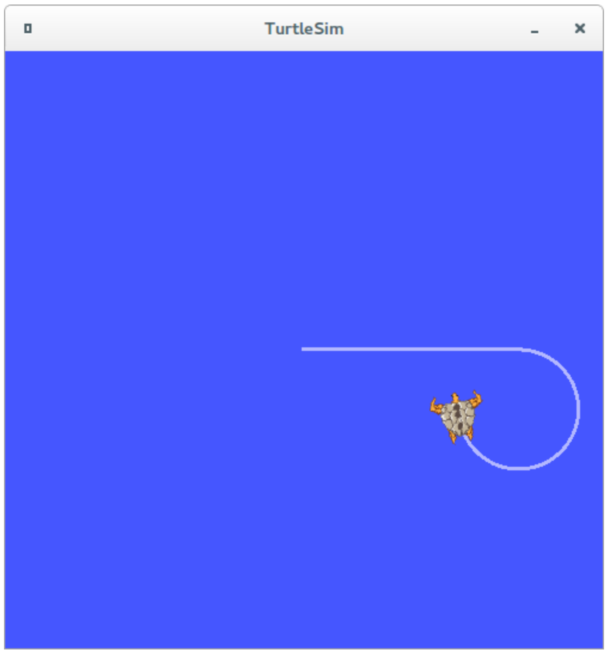

Figure 1: Example TurtleSim output drawing the letter ‘P’.

portions of the exercise.

1. Ensure your Ubuntu virtual machine is running and functioning properly.
2. Complete ROS tutorials 1-15. `http://wiki.ros.org/ROS/Tutorials`
3. Create a custom ROS package in your catkin work space titled lab2_LASTNAME. Replace LASTNAME with your last name.
4. Make a new directory called `scripts` in your package. The location of this folder should be:

   `~/catkin_ws/src/lab2_LASTNAME/scripts/`
5. In your `scripts` folder, create a simple publisher and subscriber to ensure you understand how to publish and subscribe to topics.

   `http://wiki.ros.org/ROS/Tutorials/WritingPublisherSubscriber%28python%29`
6. Create a new file called LASTNAME_letter.py in the `scripts` directory.
7. Develop a script to drive the TurtleSim and write the first letter of your first name (upper or lower case is acceptable). If my first name was Paul, the output could resemble as shown in Fig. 1.
8. Your custom script should subscribe to a keyboard message. i.e. once the letter ‘P’ is pressed, the robot should start drawing the letter.
9. Create a new folder called `launch` in your package. Build a launch file to execute all of your nodes and packages.

Deliverables

- Demonstrate the successful velocity control of the TurtleSim using a custom package and script in order to draw a letter. A keypress should initiate the drawing. Your script must both publish and subscribe over topics.
- The TurtleSim should respond smoothly and accurately to the script.
- The publisher node that you create must be incorporated in a custom package, which can be run using a launch file.
- All code (custom) for your project should be submitted as part of your lab report's appendix section, as well as uploaded to Canvas. List other packages used for the project.
- Sign the coversheet of your lab report to cite and document any sources you used to complete this report.