CSCE 452/752 Robotics and Spatial Intelligence, Fall 2024

Project 1

Overview and purpose

This assignment asks you to install ROS, to explore its behavior, and to write a relatively simple program within the ROS system. The purpose is for you to gain some basic experience with ROS and to ensure that you have a working development environment for the other projects in the course.

Logistics

This assignment should be done *individually*.

Reference material

You will likely need to consult quite a few elements of the ROS documentation to complete this assignment. The notes from August 28 recommend some specific links to help you get started.

The ability to find, sift through, read, understand, and apply relevant documentation is an important skill, and this assignment is intended to give you some practice with that skill.

Your tasks

This project consists of several tasks that should be completed in sequence. One important element of the final submission will be a report in PDF format documenting your completion of these steps. Instructions below that mention “your report” are referring to this one single PDF.

Task 1: Install

Install ROS on the device of your choice. You must use **ROS2**. It is strongly recommended to select the **Humble** distribution of ROS2.

Recall that ROS support for Windows or MacOS environments can be limited, so you may want to consider using a Linux computer or dual-booting Linux with your usual operating system.

Nothing in particular is expected in the report from this task.

Task 2: Draw

Within your newly-installed ROS system, launch the `turtlesim` simulator along with the `turtle_teleop_key` node. Use the arrow keys to drive the robot around a little and draw a picture with the lines left as the turtle moves. Capture a screenshot of your creation and include it in your report.

This task helps to ensure that you've installed ROS correctly, that your environment can correctly display graphical programs, and that you can use multiple terminals to launch several ROS nodes at once.

Task 3: Explore

While `turtlesim` is still running, use the `ros2` command line tool to explore your ROS installation.

1. Find a list of all of the **topics** in the system.
2. Choose one of these topics to use for the following steps.
3. Find the name of your topic's data type. Find the number of publishers and the number of subscribers.
4. Find the details of your topic's data type. That is, find a list of fields included in that message type.
5. Find the total number of **interfaces** in your ROS installation, including message types, service types, and action types.
6. Find the number of **packages** in your ROS installation that start with the letter `s'.

Paste the commands you used and their outputs into your report.

Task 4: Implement

Write one program, using Python or another ROS-supported language of your choice, that does these two things at the same time:

- Publish messages on the topic `/turtle1/cmd_vel` that tell the turtle to move slowly forward with a slight turn to the left. Your program should publish the same message approximately once per second. If you are doing this correctly, the turtle should move smoothly in a circular pattern.
- Subscribe to message on the topic `/turtle1/color_sensor`. When a message published on that topic arrives, your program should print the message to the console.

To ensure that your program can earn full credit, consult the source code style guide. Be sure to create a **ROS workspace** and a **ROS package** within that workspace to contain the program you write. Include your program in this package, and configure your package to treat this program as an executable. Successfully build the workspace. Then use the command

```
ros2 run <package_name> <program_name>
```

to execute your program. Paste the commands you used build the package and run the program, along with their outputs, into your report. Also include a screenshot of the turtle drawing the circle in your report. Include the source code for your circle-drawing program in your final submission.

Task 5: Install

Download the `sim` package from this link and extract it into the `src` subdirectory in your ROS workspace.

ROS2 `sim` package

This package contains a basic version of a simple simulator that we will use for several of the projects in this course.

Build your workspace with this new package. Confirm that the new package is recognized by the ROS system by using the command

```
ros2 pkg prefix sim
```

to see the directory where the installed package resides. Paste the output of this command into your report.

Task 6: Visualize

The `sim` package includes an executable called `sim1`, which you can start using an appropriate `ros2 run` command. After starting this simulator, you will notice that, unlike `turtlesim`, `sim1` does not create any window to show you what's going on inside the simulation. Instead, you'll need to use the standard ROS visualization tool, called `rviz2`, which is part of the `rviz2` package.

While `sim1` is running, start `rviz2`. Configure `rviz2` to show the details of the `sim1` simulation. Specifically, you should:

- Use the Add button in the lower left to add a `RobotModel` display. This should show the shape of the robot, a blue disc along with a darker blue segment showing the direction the robot is facing. You may

need to configure the "Description Topic" setting of this display to point to the topic `/robot_description`.
- Use the Add button again to add a `MarkerArray` display. In the configuration for this display, you should set the topic to `/sim_markers`.

After `rviz` is configured, you should make the simulated robot move by publishing some messages of type `geometry_msgs/Twist` on the topic `/cmd_vel`. The easiest way to do this is probably to execute a very slightly modified version your program from Task 4. Alternatively, you could use `teleop_twist_keyboard` (a general version of the `turtlesim`-specific teleoperation node you used earlier) or even `ros2 topic pub` to send those messages. Just make sure that the simulated robot is moving. Notice the red trail, shown by the `MarkerArray` display, showing the path traced out by the robot in the past. In your report, include a screenshot of the `rviz2` window showing a `sim1` robot that has moved through some non-trivial path from its starting location. Both the robot (a blue disc) and the path (a thin red curve) should appear in the screenshot.

Finally, use the Save Config menu option in `rviz2` to save the configuration of displays to a file for future use, so you don't need to add the displays you want and fiddle with their settings the next time you use `rviz2`. In your final submission, include the `.rviz` file created by Save Config.

What to submit

You should submit a **single zip archive** containing everything listed below.

1. A PDF report showing the results of your exploration. Include the specific details (commands, outputs, screenshots, etc.) requested in the detailed description above. You may also discuss any unexpected challenges you encountered or unexpected observations you made.
2. From Task 4, the Python source code for your circle drawing program.
3. From Task 7, the `.rviz` file you saved.

[Submit via Canvas.]