ROS2 TurtleBot

Learning outcome

This exercise will help students to get familiar with topics and how to publish to them. This can be done by command-line and via a python script. Different motions will be targeted, such as, basic open-loop linear and rotational motion and closed-loop path following.

Grading

- Grade 3: task 1 + 2, successful demo and could not answer all questions
- Grade 4: task 1 + 2, successful demo and could answer main questions
- Grade 5: task 1 + 2 + 3, successful demo and could answer main questions

Exercises:

Task 1: TurtleBot simulation

The objective of this task is to get familiar with the turtlebot simulation and how to utilize it. Complete the following tasks:

1. Open 2 terminals and execute the following command:
   $ ros2 run turtlesim turtlesim_node
   $ ros2 run turtlesim turtle_teleop_key
2. Try out the tele-operation and investigate the topics and nodes with suitable ROS commands
3. Find out what topic and message are used for sending velocity commands to the TurtleBot
4. Use the **ros2 topic** command to move the robot in a circle and in a square path
5. Find a way to concatenate multiple commands to achieve consecutive motions

Task 2: TurtleBot control

The objective of this task is to achieve closed-loop control of a simulated mobile robot within Gazebo environment. The robot we are going to simulate for this task is the Waffle Pi of the Turtlebot3 family.

Preparations for Turtlebot3 Simulation:

1. Install dependencies. Execute parts: 3.1.3 & 3.1.4 from the following tutorial:
   (Follow only the Humble tab)
   https://emanual.robotis.com/docs/en/platform/turtlebot3/quick-start/

   Note: installation does not include creating a turtlebot workspace (turtlebot3_ws), which can be done by: **$ mkdir turtlebot3_ws && cd turtlebot3_ws && mkdir src**

2. Install Simulation package and run examples:
   (Follow only the Humble tab)
   https://emanual.robotis.com/docs/en/platform/turtlebot3/simulation/#gazebo-simulation

Task 2.1:

1. Launch **turtlebot3_world.launch** or **turtlebot3_house.launch** from the turtlebot3_gazebo package.
2. Investigate the topics/nodes then find the topic/message required for commanding the robot.
3. Now, python will be used for motion control. Run the provided script:

$ python3 {path to file}/ros2_goforward.py

4. Study the code, which contains explanations of the statements in the comments, and understand how it works.

Task 2.2:

We need to develop a ROS node in order to complete the following motion control tasks, by utilizing feedback from the state of the robot:

1. Go to a point [x, y] in the workspace. Add the goal location as argument to input on the command-line, such that new goals can be added online.
2. Go to a specific pose [x, y, theta] in the world space. Add the goal pose as argument to input on the command-line, such that new goals can be added online.
3. Follow a predefined path. Define a suitable path (e.g., circle, square or something more complex) and control the robot such that it follows it closely.

Take care of what the goal(s) of the robot are at different instances.
There can be many solutions. Present two.

4. Think of several ways how control could be improved. For example, by a better robot model, separating control actions, better controller library, etc.

Hint1- Modifying this code is a fast start (only in **rospy** syntax , but helpful):
http://wiki.ros.org/turtlesim/Tutorials/Go%20to%20Goal
Hint2- Targets (point, pose, path) require suitable type definitions
Hint3- For step 3, to define a path, one option is to teleoperate the robot and log the robot footsteps

Task 3:

1. Include suitable ROS info messages (with **rosmsg**) that indicate the status of the program. For example, when TurtleBot received a new goal pose or reached the goal.
2. Monitor the messages with **rqt_console**.
3. Collect all necessary data to evaluate the performance of your controller, e.g., with **rosbag** and plot the error with **rqt_plot**. Consider carefully what error is useful for evaluation.
4. Use a revision control system, such as https://gitlab.tuni.fi/

Please remember to attach your python scripts/ Git repository / links to Rosbags along with demo meeting invitation.

To arrange a demo contact: Eetu (eetu.airaksinen@tuni.fi)