CSCE 4620/5620: Real-Time Operating Systems (Fall 2024)

Homework 4 (Due November 22)

Submit homework on Canvas

ROS programming assignment

1. Installing ROS.

- If you have access to a Linux machine computer, you can install and run ROS directly on the system. If this is possible for you, it's likely the best choice. You'll enjoy the convenience and enjoy the speed of the system relative to running ROS on a virtual machine.
- On a Windows system, you can install a virtual machine that runs Linux, and then run ROS on the virtual machine. The main disadvantage to this approach is that using the virtual machine incurs significant computational overhead, which can cause system response to be a bit slow. For this assignment, the virtual machine option is reasonable.

There are many web pages, tutorials, and books that provide guidelines for installing ROS. Here are two useful tutorials.

http://wiki.ros.org/ROS/Installation

Youtube series:

http://www.youtube.com/playlist?list=PLDC89965A56E6A8D6

[This step should already be done in class.]

Programming assignment: to create a path planning program using either C++ or python that works with turtlesim package in ROS. It requires writing both subscriber and publisher code. **Please learn about these concepts by reading https://cse.sc.edu/~jokane/agitr/ (Chapter 3) or watch the following two videos: https://www.youtube.com/watch?v=aL7zLnaEdAg https://www.youtube.com/watch?v=v9RgXcosuww**

You will control the motion of a robot using the turtlesim package. The simulated (turtle) robot lives on a 30x30 grid. The coordinates for the lower-left corner are (0, 0), and for the upper-right corner are (30,30). Initially, the turtle is positioned at (20, 20) heading west. The simulator is designed to prevent turtle from escaping the square workspace.

You will write a program that should do the following.

- When the program begins, it should prompt the user for an input goal position (xg, yg).

- Once the goal position is entered, your robot should move to the goal using a proportional control scheme:
    - Determine the desired heading to the goal (i.e., the heading angle from the robot's current position to the goal).
    - Determine the heading error.
    - Set the angular velocity of the robot to be proportional to the heading error.
    - Set the linear velocity of the robot to be proportional to the distance to the goal.

  These commands should be executed in a loop, until the robot reaches the goal position (within certain error tolerance). This is a control scheme. Feel free to design a more sophisticated controller.

Submission:

A zip file contains all the following information:

- A WORD/PDF file with a brief explanation of you design, two screenshots show the movement of the turtle after running your program.
- A readme file shows how to execute your code.
- Your code (C++/Python).

Making sure that based on your readme file, we should be able to compile and execute your code.