SOFAR Continuous Evaluation 2 – Programming a node

The test consists in programming a node that controls a differential drive robot to track a circular path.

Provided files/nodes

- The capture_key package, which you already know, publishes keystrokes to topic /key_typed.

- The ecn_turtle_control package, which contains three types of nodes :
  - ecn_turtle_path_publisher, which publishes:
    - The desired path of the robot (cicrle centered at the origin, with a radius from 1 to 4 m) for visualization under rviz. To see the path under rviz, add a « path » object.
    - The current circle radius, each time it is modified.
  - ecn_turtle_robot_publisher, which publishes a marker arrow to visualize the current position/orientation of the robot in rviz under the shape of an arrow. To visualize it, add a « marker » object in rviz.
  - ecn_turtle_robot_simulator, which subscribes to robot pose and circle radius and publishes the lateral and heading error of the robot with respect to the desired path.

- The « ecn_turtle_control.launch » file. It starts all the aforementioned nodes in such a way that they are correctly connected. It also defines a few global parameters of the application, namely initial circle radius and initial robot position and heading. With the established connection, it is possible to change the circle radius in steps of 0.5 m using the '+' and '-' keys.

Initial Step

- Retrieve the file « ce2.zip » using the link given on the whiteboard.
- Copy it to the « ros/src » folder of your account.
- Decompress it there. You should have a folders called « ce2 », which contains two packages: « capture_key », and « ecn_turtle_control ».

Your tasks

1. You have to program the node which controls the robot to track the desired path. It should do so correctly when the radius of the desired path changes. Name the source file « ecn_turtle_control_node.cpp », which is the name indicated in the « CmakeLists.txt » file. The use of the node skeleton is recommended.

2. Program a node to control the robot at 50 Hz to track the path. In order to do that, you need to determine what the input and output topics of the node should be (name and type). Once you've done that, call the teacher. The teacher will record what you did but will not tell you whether it is correct or not.
   As far as the control is concerned :
   - The translational velocity is a constant.
   - The rotational velocity obeys the following equation (simplified form of the Lyapunov based controller of your Mobile Robot control class) :
     $\omega = \frac{v_{ref}}{R} + K_y.v_{ref}.y_e + K_\theta.\theta_e$
     where R is the circle radius.

3. The control node will use parameters. The translational velocity of the robot is constant and equal to the value given by the control node parameter « vref ». Other parameters of the node are the gains K_y and K_θ.
   Note: In no way is the tuning of the gains the purpose of this exercise. You can use vref=1.0, Ky=5.0 and K_θ=1.0 in your tests.

4. Complete the launch file in order to start the control node in addition to the already existing nodes. Run the launch file and check that it works. For this, under rviz:
   - Add the visualization of the path.
   - Add the visualization of the robot marker.
   - Modify the circle radius during execution to check that the robot is indeed in closed loop.

Advices/information

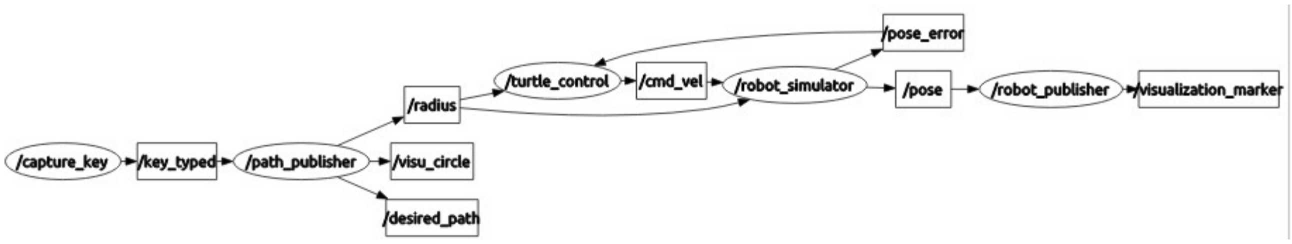
- To get a pass grade, you need to have a functional control node (the robot tracks the path), with the corresponding launch file, even if it uses hard-coded constants instead of parameters. In this case, initialize the radius to the the value given in the launch file.

- You should try to complete the pass grade task before attempting any improvement. Once you get to that level, call the teacher who will record your status.

- Code quality will be taken into account in the grading.

Advices/information

To help you understand the role of the nodes, here is the rqt_graph of the running application without the control node. The control node has to be added to the graph. I suggest you draw



/pose_error
/turtle_control
/cmd_vel
/robot_simulator
/pose
/robot_publisher
/visualization_marker
/radius
/capture_key
/key_typed
/path_publisher
/visu_circle
/desired_path