FEUP Universidade do Porto Faculdade de Engenharia

MEEC **Autonomous Systems** **2023/2024**

Laboratory Work #7 – Implementation of basic movement routines for controlling a Real Omnidirectional Robot's (GotoXY, FollowLine and FollowCircle)

Part 1 - Simulation

1. First, to use the ROS package developed for this lab work, you need to use a virtual machine running Ubuntu 20.04 or have Ubuntu natively installed on your computer. To configure a virtual machine on your computer, follow the guide presented in this link:

   https://drive.google.com/file/d/1p-gmWfnHX9WiVgQw524mA5Gr7VZM3Pcs/view?usp=sharing

   Run the virtual machine with the Ubuntu OS using password "12345". Do not perform any upgrade.

2. Regarding one omnidirectional robot, assume that it can be controlled by setting the desired angular velocity (W), forward linear velocity (V), and lateral linear velocity (Vn). Given the true robot pose provided by the simulator (xodo, yodo, thetaodo) and the desired end pose (xf, yf, thetaf) implement the method GoToXY, present in the file "SARosNavController.cpp" in the folder "/home/catkin_ws/src/autonomous_systems/sa_ros_nav_controller/src/sa_ros_nav_controller", that commands the robot to a final pose with a certain tolerance for angular and distance error. Implement this method using two state machines, one for linear motion and another for angular motion and test it properly.

   Use the Visual Studio Code to edit the code files.

   Some reminders while using the ROS package:

   - To compile the code, go the "catkin_ws" directory and run the command "catkin_make":

     ```
     $ cd ~/catkin_ws
     $ catkin_make
     ```

   - To run the ROS nodes, execute the following command:

     ```
     $ roslaunch sa_robot_ros_nav_conf wake_up_simulated_robot.launch
     ```

   - To stop the ROS nodes, press Crtl + C in the terminal

   - To edit the code, open the navigation controller package:

     ```
     $ cd ~/catkin_ws/src/autonomous_systems/sa_ros_nav_controller/
     $ code .
     ```

   - To execute the control functions (GoToXY, FollowLine and FollowCircle), the ROS nodes must first be running in a terminal window. Then, in another window, execute one of the following instructions:

     - GoToXY**:** rosservice call /unnamed robot/gotoxy srv – xf(m) yf(m) thetaf(deg)

     Example: $ rosservice call /unnamed_robot/gotoxy_srv -- 0.5 0.5 90

     - FollowLine**:** rosservice call /unnamed robot/followline srv – xi(m) yi(m) xf(m) yf(m)

thetaf(deg)
Example: $ rosservice call /unnamed_robot/followline_srv -- 0 0 1.5 0 180

- FollowCircle: rosservice call /unnamed robot/followcircle srv – xc(m) yc(m) r(m) anglef(deg) thetaf(deg)

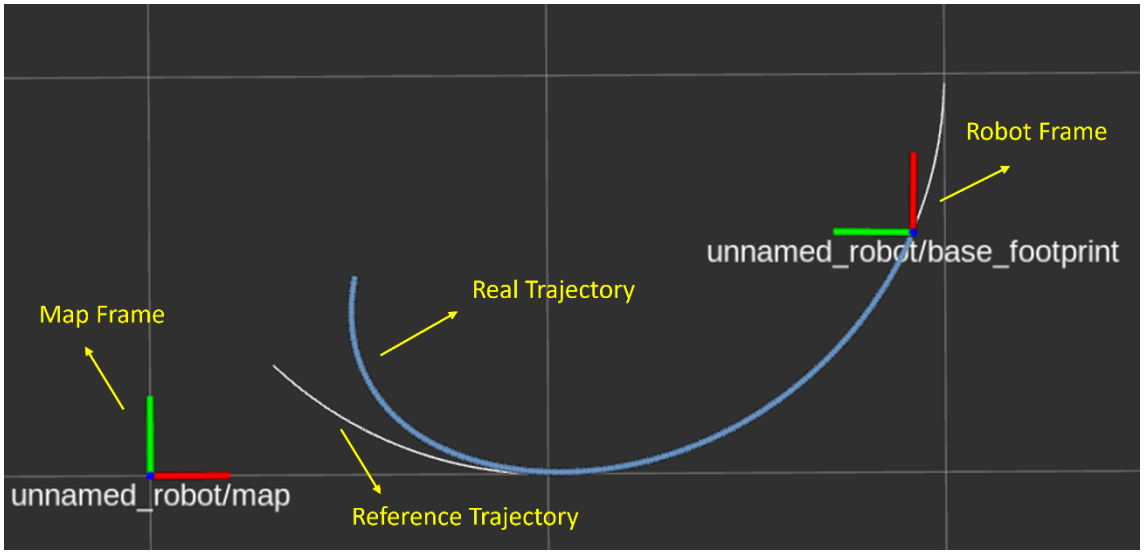
Example: $ rosservice call /unnamed_robot/followcircle_srv -- 1 1 1 -90 40

3. Given a straight line defined by the points (xi, yi) and (xf, yf), implement the method FollowLine, in the same file, that commands the robot in order to follow this line in the direction from point (xi, yi) to (xf, yf), stopping at this end point with a certain orientation.

   a) Start with a state machine like the one used in GoToXY replacing only in the state Go_Forward and in the state De_Accel_Lin the speed control equations. Take into account the distance from the robot to the line.

4. Given a circle defined by the coordinates of the center (xc, yc), the radius "r" and the final angle of the circle (anglef), implement the method FollowCircle, in the same file, that commands the robot in order to follow this line in the counterclockwise, stopping at this end point (xf, yf), with a certain orientation (thetaf).

   a) Start with a state machine like the one used in GoToXY and change the speed control equations. Take into account the distance from the robot to the circle.



Part 2 – Test with the real robot

1. Now, after testing and verifying the control algorithms in simulation, lets test the same algorithms in a real robot. First, install NoMachine on your computer so that you can connect to the robot's Raspberry Pi.

   https://www.nomachine.com

2. Now, connect to the wireless LAN with ssid TP-Link_28CD, and in access key 49871005.

3. Then, to connect to the robot, first turn on the switch presented in the robot. After that,

start the NoMachine application and connect to the Raspberry Pi, with ip address 172.16.1.XXX, where XXX is the robot number. If the system asks for a username and password use the following credentials.

User: sdpo-ratf
Password: 5dpo5dpo

4. After connecting to the Raspberry Pi, it is necessary to make small changes to some files in the "sa_catkin_ws" folder. If you use the Visual Studio Code, stop all the nodes. This editor is very heavy in terms of processing. Alternatively, you can use the File Manager and double click in the file name. It opens the file with another much lighter editor.

Probably you need to go to Raspberry_menu->Preferences->Keyboard and Mouse->Keyboard->Keyboard Layout then select Layout Portuguese and OK.

a) First, you need to change the code of the algorithms (GoToXY, FollowLine and FollowCircle) that are located in the path "sa_catkin_ws/src/sa_ros_nav_controller /src/sa_ros_nav_controller/SARosNavController.cpp". You can copy the file from part 1 and make some minor adjustments to the parameters.

b) Besides that, to use the EKF, you need to specify the location of the beacons, and also the initial position of the robot, in the file located in the path "sa_catkin_ws/src/sa_robot_ros_nav_conf/launch/localization/sdpo_ratf_ros_localization/sdpo_ratf_ros_localization.yaml".

```
1  map_frame_id: $(arg robot_id)/map
2  odom_frame_id: $(arg robot_id)/odom
3  base_frame_id: $(arg robot_id)/base_footprint
4  laser_frame_id: $(arg robot_id)/laser
5
6  beacons_diam: 0.09
7  beacons_valid_dist: 0.20
8  beacons: [ 0.900, 0.700,
9             -0.900, 0.700,
10             0.900,-0.700,
11            -0.900,-0.700 ]
12
13 ekf_pose_ini_x: 0.0
14 ekf_pose_ini_y: 0.0
15 ekf_pose_ini_th: 0.0
16
17 ekf_cov_ini_p_x: 0.1
18 ekf_cov_ini_p_y: 0.1
19 ekf_cov_ini_p_th: 0.1
20 ekf_cov_q_d: 0.1
21 ekf_cov_q_dn: 0.1
22 ekf_cov_q_dth: 0.05
23 ekf_cov_r_dist: 0.0001
24 ekf_cov_r_ang: 0.001
25
26 ekf_mode_ini: "Fusion"
27
28 publish_pose: True
```

Beacons Location

Initial Robot Pose

c) Finally, after changing these files compile the code with the following commands:

$ cd ~/sa_catkin_ws
$ catkin_make

5. The commands to start the ROS packages and to call the algorithms are very similar to the ones used in the Part 1.

   - To run the ROS nodes, execute the following command:

     $ roslaunch sa_robot_ros_nav_conf wake_up_almighty_ratf.launch

   - To stop the ROS nodes, press Crtl + C in the terminal

   - To display the Node Graph execute

     $ rqt_graph

   - To execute the control functions (GoToXY, FollowLine and FollowCircle), the ROS nodes must first be running in a terminal window. Then, in another window, execute one of the following instructions:

     - GoToXY**:** rosservice call /unnamed robot/gotoxy srv – xf(m) yf(m) thetaf(deg)

     Example: $ rosservice call /unnamed_robot/gotoxy_srv -- 0.5 0.5 90

     - FollowLine**:** rosservice call /unnamed robot/followline srv – xi(m) yi(m) xf(m) yf(m) thetaf(deg)

     Example: $ rosservice call /unnamed_robot/followline_srv -- 0 0 1.5 0 180

     - FollowCircle**:** rosservice call /unnamed robot/followcircle srv – xc(m) yc(m) r(m) anglef(deg) thetaf(deg)

     Example: $ rosservice call /unnamed_robot/followcircle_srv -- 1 1 1 -90 40

   - Finally, if for some reason you need to stop the robot, run one of the following commands in the terminal (they are the same):

     $ rosservice call /unnamed_robot/stop_controller_srv
     $ stop

Some notes while using NoMachine:

- If you need to move a window that is outside the screen press Alt + Mouse left button and drag the window.

  - If the keyboard layout is not matching yours then go to “Home > Preferences > Keyboard and Mouse > Keyboard > Keyboard Layout”. Then change to your keyboard layout and press OK