**Masters in Medical Imaging and Applications - University of Girona**

COMPUTER AIDED SURGERY AND MEDICAL ROBOTICS

---

# Lab 1 Report

## Introduction to ROS

**Session of October 2019**

---

***Authors:***

Tewodros W. AREGA

***Supervisor:***

Narcís PALOMERAS

## 1 Turtlesim

ROS provides a simple turtle simulator called *turtlesim*. When we execute the *turtlesim_node*, it displays a *turtlebot*. At first, it doesn't move. But when we execute *turtle_teleop_key*, it creates a node that publishes velocity commands to the *turtlebot*.

Using the ROS commands, we have analyzed that:

- The name of the topic which sends velocity commands to *turtle_node* is: */turtle1/cmd_vel*
- The type of message the topic is publishing: *geometry_msgs/Twist*
- The name of the topic where the *turtlebot* position is published at:*/turtle1/pose*
- The type of message the topic is publishing: *turtlesim_msgs/Pose*

The nodes and topic graph is visualized in the figure 1

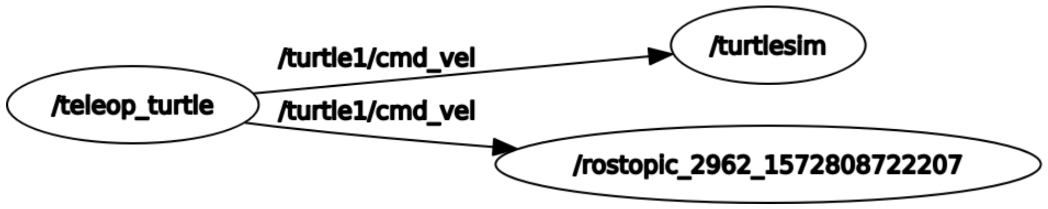

Figure 1: *ROS graph for nodes and topics of turtlesim*

## 2 Waypoint Controller

The *turtle_waypoint* is a node that will publish velocity commands to the *turtlebot* to make it go to a desired point. To create the node, first we created a package called *roslabs* in *catkin* workspace. We put the python node named *turtle_waypoint* inside the package.

The turtlebot is similar to a robot as it has a pose, publisher and subscriber. For this lab, first we created a publisher which will publish velocity commands to the topic */turtle1/cmd_vel*. The topic's message type is *geometry_msgs/Twist*. Then we defined a subscriber that will subscribe to the topic */turtle1/pose* , a topic where the actual *turtlesim* position is published.

In case the desired points aren't given while running the node, it is handled by checking the ROS parameter server for parameters and head to that point. But the parameters has to be set before. We used *rospy's* 'has_param' function to check whether parameters are set or not. If they are set, we used 'get_param' function to get the parameters from ROS parameter server.

When the subscriber receives a message a callback function is called to save the turtle's current position and orientation. We used the a dot operator to access the value of *x,y* and *theta* from the Pose message.

Finally to publish the velocity command to the *turtlebot*, first we calculate the euclidean distance between the current point and the desired point of the turtle.

In order for the turtle to move, the linear speed will consist of a constant multiplied by the position error(euclidean distance between the turtle and the desired point) and the angular speed will depend on the arctangent of the distance in the y-axis by the distance in the x-axis multiplied by a constant.

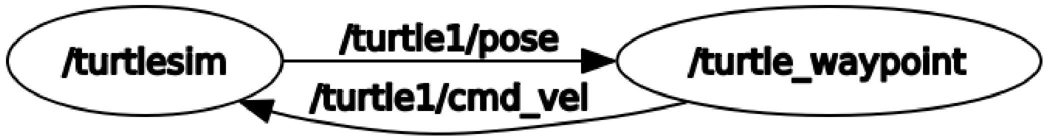

Figure 2: *ROS graph for nodes turtlesim and turtle_waypoint*

We have noticed two problems in this turtle. The first one is if we make the tolerance very small, the turtle would go crazy. The other problem is if we make the desired position very large, the turtle will also go everywhere.