Lab0 - Introduction to ROS

1 The goal

The goal of this lab session is to get started with the software tools that will be used used during the lab sessions of Probabilistic Robotics, especially ROS. The Robot Operating System (ROS) is a software framework for robot software development. ROS provides libraries and tools to help software developers creating robot applications. It provides hardware abstraction, device drivers, libraries, visualizers, message-passing, package management and more. ROS will be used in the next lab exercises, and so, it is fundamental to understand all the basics given in this session.

2 Pre-lab: Installing Ubuntu and ROS

Currently the only officially supported platform for ROS is Ubuntu. Ubuntu is a computer OS (Operating System) based on Debian Linux. The computers in the Robotics Lab have a Ubuntu distribution installed in a virtual machine, but is highly recommended to have Ubuntu installed in your own machines, so you are able to work when you don't have access to the lab.

In order to install Ubuntu you can either:

1. (Recomended) Install Ubuntu in a partition alongside your other OS (hard disk install). Make sure that you install the distributions specified below:
   - Ubuntu 14.04 LTS (download iso).
   - ROS Indigo Desktop Full install (install instructions).
2. (not recommended) Install Ubuntu in a virtual machine inside your normal OS. In this case you can donwload a Virtualbox and follow the same installation instruction than the previous point, but this time on the virtual machine. Note that this kind of installation can lead to problems with graphics drivers in some or all labs.

3 Lab: ROS tutorials

Now that you have Ubuntu and ROS successfully installed, please follow the most fundamental ROS tutorials. Please have in mind that your ROS workspace, the place where you need to put all your packages is the catkin_ws folder inside the home directory.

- Creating a workspace for catkin.
  `http://wiki.ros.org/ROS/Tutorials/InstallingandConfiguringROSEnvironment`
- Navigating the ROS FileSystem.
  `http://wiki.ros.org/ROS/Tutorials/NavigatingTheFilesystem`
- Creating a ROS package.
  `http://wiki.ros.org/ROS/Tutorials/CreatingPackage`
- Understanding ROS Nodes.
  `http://wiki.ros.org/ROS/Tutorials/UnderstandingNodes`
- Understanding ROS topics.
  `http://wiki.ros.org/ROS/Tutorials/UnderstandingTopics`
- Understanding ROS services and Parameters.
  `http://wiki.ros.org/ROS/Tutorials/UnderstandingServicesParams`
- Writing a simple Publisher and subscriber (Pyhton).
  `http://wiki.ros.org/ROS/Tutorials/WritingServiceClient%28python%29`

- Examining the Simple Publisher and Subscriber.
  `http://wiki.ros.org/ROS/Tutorials/ExaminingPublisherSubscriber`

Please note that the ROS Cheat Sheet, which you can find in the files for this Lab, might be useful for you.

3.1 Turtlesim Waypoint Node

During the Lab session you will need to create a node to interact with the turtle simulator seen in Understanding ROS nodes tutorial. Run again turtlesim:

```
roscore
```

In a new terminal:

```
rosrun turtlesim turtlesim_node
```

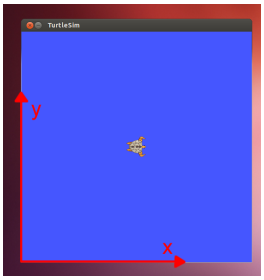

Figure 1: Turtlesim visualization.

And using the ROS tools you've seen in the tutorials (rostopic, rosmsg, rqt_graph) identify:

1. The name of the topic used to send velocity commands to the turtle and the type of these messages.
2. The name of the topic where the turtle position is published at, and the type of these messages.

Clone the repository of the Probabilistic labs in your workspace. This repository contains all the code for this labs. After download compile the code (if you are using a desktop machine from the lab skip this step and go to the next one to update the repository):

```
cd ~/catkin_ws/src
git clone https://bitbucket.org/gvallicrosa/probabilistic_labs.git
cd ~/catkin_ws
catkin_make
```

Remember before starting any lab to update the repos with:

```
cd ~/catkin_ws/src/probabilistic_labs
git pull
cd ~/catkin_ws
catkin_make
```

Run the node in the turtle_waypoint package:

```
rosrun lab0_ros turtle_waypoint.py
```

You will see that nothing happens. You need to modify the *turtle_waypoint.py* node inside the package in order to publish commands to the turtle to make it go to a desired point. This desired position will be specified when running the node through the command line like:

```
rosrun lab0_ros turtle_waypoint.py 1.2 5.0
```

being the desired point x: 1.2 and y: 5.0. If the point is not given when calling the node, the node will check the ROS parameter server for the parameters */default_x* and */default_y* and head to that point. In the case the parameters are not set the turtle won't move. You can set ROS parameters like:

```
rosparam set default_x 5
rosparam set default_y 10
```

4 Lab report

After the lab session: Write a brief report (max 2 pages) explaining your solution and problems faced. Include the final *turtle_waypoint.py* file.