A PRACTICAL INTRODUCTION TO ROS

1. INITIAL CONFIGURATION

OWN LAPTOP WITH UBUNTU 16.04

ROS KINETIC INSTALLATION

Copy the following commands sequentially in your terminal (CTRL+ALT+t):

```
sudo sh -c 'echo "deb http://packages.ros.org/ros/ubuntu $(lsb_release -sc) main" > /etc/apt/sources.list.d/ros-latest.list'
```

```
sudo apt-key adv --keyserver hkp://ha.pool.sks-keyservers.net:80 --recv-key 421C365BD9FF1F717815A3895523BAEEB01FA116
```

```
sudo apt-get update
```

```
sudo apt-get install ros-kinetic-desktop-full
```

```
sudo rosdep init
```

```
rosdep update
```

```
echo "source /opt/ros/kinetic/setup.bash" >> ~/.bashrc
```

ROS NODES TO BE INSTALLED

Copy the following commands sequentially in your terminal (CTRL+ALT+t):

```
sudo apt-get install ros-kinetic-turtlebot3-*
```

```
sudo apt-get install ros-kinetic-open-cr-module
```

```
sudo apt-get install ros-kinetic-dynamixel-*
```

```
sudo apt-get install ros-kinetic-dwa-local-planner
```

ADDITIONAL TOOLS TO INSTAL

```
sudo apt-get install git
```

```
sudo apt-get install ssh
```

VM IMAGE

Download and install Virtualbox: https://www.virtualbox.org/wiki/Downloads

Configure the Virtual Machine:

1. Press **NEW**
2. Operating system: **LINUX**
   Version: **Ubuntu (64bit)**
3. Memory Size: **7000Mb**
4. Hard drive: **Use an existing virtual hard drive file** -> Select the image provided.
5. In Settings -> Display -> **Enable 3D Acceleration**
6. In Settings -> Network -> Enable a second Adapter
   Attached to: **Bridged adapter**
   Name: Your wireless adapter

Open the VM and once Ubuntu request the password write: **EPD_2018**

Once Ubuntu is up and running, open a terminal (CTRL+ALT+t) and install the following ROS node:

sudo apt-get install ros-kinetic-dwa-local-planner

Change your machine name (all images will have the same one and it will be a problem when running ROS across multiple machines):

Open a terminal (CTRL+ALT+t) and modify the following files:

sudo gedit /etc/hosts

sudo gedit /etc/hostname

Restart the VM and you are ready for the ROS exercises ☺.

2. EXERCISES

BEGINNER LEVEL

1. The talker and the listener

Write two simple ROS nodes, a talker which will broadcast a simple message (“HELLO ERNI”) and a listener which will subscribe to the message published by the talker.

- Create a ROS workspace
- Create an empty ROS package
- Create both c++ files and configure the package.xml and CMakeLists.txt
- Build the ROS workspace
- Run both nodes
- Use rosnode and rostopic commands to confirm that everything is working as expected
- Create a launch file that run both nodes and the roscore at the same time
- Run the launch file and confirm again that everything is working as expected with rosnode and rostopic commands
- Use rosbag commands to save the message published by the talker.
- Run the rosbag recorded and the listener and check again with rosnode and rostopic that the listener is receiving the message.

2. The talker and the listener in a distributed system

In pairs, configure a distributed system in which one member will be the talker and the other one will be the listener.

- Check with **ping** command that you see the machine of your partner (Use **ifconfig** command to check the IP)
- Add a line in /etc/hosts with the IP and hostname of our partner.
- Add the following line in the **.bashrc** file:
  export ROS_MASTER_URI=http://PARTNER_HOSTNAME:11311/
- run the following command in the terminal to refresh the .bashrc file:
  source ~/.bashrc
- One run the talker and the other the listener.
- Check with rosnode and rostopic that you have communication.

3. The message chain

Create three custom messages (is up to you which information you want to send). The first one should be a simple one using primitive types (http://wiki.ros.org/msg). The second one should be composed by the first one and in the same way; the third one should be composed by the second one. Example:

First message:

ERNIan.msg

- string name
- string level
- string project

Second message:

BUL.msg

- string name
- int32 unitSize
- ERNIan[] unitMembers

Third message:

ERNI.msg

- int32 numberOfBUL
- BUL[] BULList

In the same ROS package, create four nodes:

- The first one will be a publisher of the most complex message. The node should create, fill and publish the message.
- The second node should subscribe to the message published by the first node, extract the data and publish the second message.
- The third node should subscribe to the message published by the second node, extract the data and publish the third message (the simplest one).
- The fourth node just subscribes to the message published by the third node and prints the data received.

Create a package containing the messages definition and the four nodes, compile it and run all nodes. Use the explained commands, rosnode, rostopic, etc. to analyse the communication and the rqtgraph tool to see the chain created.

4. The message collector

Create a message containing three primitive types (which ones is up to you). For example:

Custom message:

ERNIan.msg

- string name
- string level
- string project

Then create four nodes:

- Based on my example, the first node will publish a std_msgs/String with the name of the ERNIan.
- Based on my example, the second node will publish a std_msgs/String with the level of the ERNIan.
- Based on my example, the third node will publish a std_msgs/String with the project of the ERNIan.
- The fourth node will subscribe to the three messages published, will create and fill the ERNIan message with the information gathered and will publish the message.
- The three previous nodes should be subscribed to the complete message published by the fourth node.

Create a package containing the messages definition and the four nodes, compile it and run all nodes. Use the explained commands, rosnode, rostopic, etc. to analyse the communication and the rqtgraph tool to see the diagram.

INTERMEDIATE LEVEL

1. The Turtlebot3 simulation

Use Gazebo to simulate the robot, its sensors and a scenario. Launch all the nodes needed to perform 2D mapping and autonomous navigation:

2D Mapping

- Before launching the nodes, you should export the robot model in the terminal or add the command in bashrc to have the model charged always:
  export TURTLEBOT3_MODEL=burger
- Run Gazebo simulation (already prepared in turtlebot3_gazebo package which is already installed).if the simulation is going really slow, deactivate the simulation gui by modifying the launch file:
  roslaunch turtlebot3_gazebo turtlebot3_world.launch
- Run the turtlebot bring up node to setup all the transformations between the robot and its sensors:
  roslaunch turtlebot3_bringup turtlebot3_remote.launch
- Run the mapping node. Gmapping is a mapping algorithm widely used in robotics, but there are other mapping algorithms available in the same package turtlebot3_slam. Explore the rest also:
  roslaunch turtlebot3_slam turtlebot3_gmapping.launch
- Run rviz to have a visualization tool of what is going on. You should configure it to show the robot, the sensor data and the map that is being created.
  rosrun rviz rviz
- Run the teleop node to control the robot with the keyboard:
  roslaunch turtlebot3_teleop turtlebot3_teleop_key.launch
- Move the robot and enjoy the experience of creating a 2D map ☺.
- When the map is completed save it locally:
  rosrun map_server map_saver -f MAP_NAME
- Before closing all nodes, use rqtgraph, rosnode, rostopic, etc. and analyse the complexity of the system.

Autonomous navigation

- Launch again the simulation and the robot bringup nodes:

roslaunch turtlebot3_gazebo turtlebot3_world.launch
roslaunch turtlebot3_bringup turtlebot3_remote.launch

- Run the autonomous navigation node. You should configure the path to the saved map in advance:
  roslaunch turtlebot3_navigation turtlebot3_navigation.launch
- Run rviz again, configure it and use the available commands in the tool to send the robot to the desired position in the map.
  rosrun rviz rviz
- Enjoy the experience of navigating autonomously ☺

PRACTISING WITH THE REAL ROBOT

First, you should assembly the robot with the instructions provided (there are videos in youtube if needed). Then reproduce the simulation done before with the real robot. Things to have in mind:

- We should enter in the raspberry pi with **ssh** and launch there the bringup and navigation nodes.
- Run the rviz and configure it in your own laptop.

Explore all the possibilities that we already have with this robot: 2D mapping, autonomous navigation, frontier, exploration, etc.

Save a rosbag when performing the 2D mapping with the data of the sensors so we can run other mapping algorithms later on without running again the robot.