Lab1 - Introduction to Turtlebot

1 Introduction

This lab is designed to do a first approach to the Turtlebot 2 1 and to get use to the ROS network configuration. This robot will be used in the upcoming lab sessions. You can find all the information related to this robot in `http://wiki.ros.org/Robots/TurtleBot`. Furthermore, this lab session is also an introduction to the ros launch system known as launch files.

Figure 1: Turtlebot 2 with a Kubuki base

2 Pre-lab

Upgrade to hydro your ROS distribution (if you are not using it already) following the instructions from `http://wiki.ros.org/hydro/Installation/Ubuntu`. We recommend to install the `ros-hydro-desktop-full` package. Furthermore, you need to install all the required Turtlebot packages by using `sudo apt-get install ros-hydro-turtlebot*`. It is also recommended to install chrony `sudo apt-get install chrony` and synchronize the computer `sudo ntpdate ntp.ubuntu.com`.

After installing the hydro version and all the Turtlebot 2 packages follow this instructions:

- Run *roslaunch turtlebot_gazebo turtlebot_empty_world.launch*. This package is the one that starts the robot simulation within the Gazebo simulator.
- Add the Willow garage model to the gazebo simulator (place it so the robot is inside the building). You will have the model available after a while in the insert tab.
- Run *roslaunch turtlebot_teleop keyboard_teleop.launch* in a new terminal. Drive the robot around using the keys indicated in the terminal.
- Run *roslaunch turtlebot_gazebo gmapping_demo.launch* and *roslaunch turtlebot_rviz_launchers view_navigation.launch* in two new terminals. Drive the robot around and build a map. Save a screen shot of Rviz where a map of at least two rooms can be seen (the visualizer, not the simulator) and submit it through the Moodle before the day of the lab.

3 Lab work

Most of the work asked here is based on the Turtlebot tutorials from `http://wiki.ros.org/Robots/TurtleBot`. For the robots use *turtlebot* as user name and *ros* as password.

3.1 Network configuration

In ROS, two main variables have to be set up before start working: the `ROS_MASTER_URI`, which tells the system which robot/computer will host the roscore, and the `ROS_HOSTNAME`, which hosts the pc's IP or name. Pay special attention to this two variables if the messages are not properly sent or received over the network. If this happends that normally means that ether the configuration is wrong or that the name in either of one variable can not be resolved to an IP address.

In order to set up the network, connect to the appropriate wifi according to the robot (see table 1). Then, configure your network according to the IPs from table 1. For the configuration, three steps are done. First add the Turtlebot to the hosts file in */etc/hosts*:

```
$ sudo sh -c 'echo TURLEBOT_IP ROBOT_NAME >> /etc/hosts'
```

Use the TURTLEBOT_IP and ROBOT_NAME from the following table:

Robot Name	WiFi	Password	Robot's IP
turtlebot1	turtlebot1	turtlebot1	10.42.0.1
turtlebot2	turtlebot2	turtlebot2	10.42.0.1
turtlebot3	WLAN_24	C001D201E7A24	192.168.1.3
turtlebot4	WLAN_24	C001D201E7A24	192.168.1.4
turtlebot5	WLAN_24	C001D201E7A24	192.168.1.5

Table 1: Network configuration for Turtlebots.

and your computer's IP for YOUR_IP. Then, set up your `ROS_MASTER_URI`:

```
$ echo export ROS_MASTER_URI=http://ROBOT_NAME:11311 >> /home/$(whoami)/.bashrc
```

finally set up your `ROS_HOSTNAME`:

```
$ echo export ROS_HOSTNAME=YOUR_IP >> /home/$(whoami)/.bashrc
```

For all this changes to take effect you have to reload your `.bashrc` file:

```
$ source /home/$(whoami)/.bashrc
```

To check if the connection is properly done, first we open an ssh connection to the robot:

```
$ ssh turtlebot@ROBOT_NAME
```

and there, we run a roscore. Then run in your pc's terminal:

```
$ rostopic list
```

This command should return a list of topics if a the ROS master has a core running.

3.2 Bringup

In order to start the robot, ssh into it's computer and run:

```
$ roslaunch turtlebot_bringup minimal.launch
```

and, in your pc's terminal run:

```
$ roslaunch turtlebot_dashboard turtlebot_dashboard.launch
```

This will launch a graphical interface to check the robot's status. Here you can check, among other things, the batteries of both the robot and its pc.
To start the Kinect on the robot run:

```
$ roslaunch turtlebot_bringup 3dsensor.launch
```

and, in a new terminal:

```
$ roslaunch turtlebot_rviz_launchers view_robot.launch
```

Note that if you want to visualize the data on your workstation the process will be very slow due to the amount of information to send through network. It is highly recommended to run this two commands directly on the turtlebot's pc, not via ssh.

3.3 Teleoperation

For the keyboard teleoperation ssh into the turtlebot and run:

```
$ roslaunch turtlebot_teleop keyboard_teleop.launch
```

3.4 Navigation and mapping

This robot has already built up packages which allow to build maps using SLAM and also to do autonomous navigation on a given map. First, we are going to use the mapping node to obtain a small map. After bringing up the turtlebot and setting up the keyboard teleoperation, run (in a new terminal):

```
$ roslaunch turtlebot_navigation gmapping_demo.launch
```

In order to visualize the map and the robot's position run:

```
$ roslaunch turtlebot_rviz_launchers view_navigation.launch
```

Drive the robot around to build a small map and capture a screen shot for the report. To save the map use, on the robot's pc:

```
$ rosrun map_server map_saver -f /tmp/my_map
```

In order to navigate the robot autonomously on that map launch (on the robot):

```
$ roslaunch turtlebot_navigation amcl_demo.launch map_file:=/tmp/my_map.yaml
```

and close the teleoperation. If you have closed Rviz on your workstation use the same command as for the mapping to open it again. Now click on "2D Pose Estimate" and click where the robot is within the map (click for the position, and point for the direction). Now tell the robot where to go using "2D Nav Goal". Capture the screen while the robot is computing the trajectory for the report.

3.5 Follower

The last tutorial is a funny function for Turtlebot: the follower package. Run (on the robot):

```
$ roslaunch turtlebot_follower follower.launch
```

Now stand in front of the robot and move, it should follow you if you are not moving fast.

3.6 Turtlebot driver

Along with the lab documentation you can find a ROS package called `turtlebot_driver`. Copy it to your ROS hydro workspace (the one you set up on the last lab session). This package contains the script where the lab programming will be done. Please modify the driver.py (`turtlebot_driver/src/driver.py`) in order to be able to make the Turtlebot go to a given position. For this purpose, please modify the constructor of the driver class where the definition of the subscriber, publisher and velocity message are missing. You will have to modify also the functions `read_position`, which stores the position received on a message to the class variables, and `compute_velocity` where the velocity message is created.

To launch the node run:

```
$ roslaunch turtlebot_driver driver.launch
```

If no arguments are given to the `driver.lanch`, the goal point is $(0, 0, 0)$ for (x, y, θ). If you want this points to change use:

```
$ roslaunch turtlebot_driver driver.launch x:=x y:=y theta:=θ
```

Furthermore, it is possible to launch this node in a virtual environment. For this purpose first you have to set up your masters to your workstation to avoid problems with the real robot:

```
$ export ROS_MASTER_URI=http://localhost:11311
```

now, the terminal where you have run this command uses your workstation as ROS master. To launch the simulator run:

```
$ roslaunch turtlebot_driver driver.launch sim:=true
```

Please note that the this parameter can be combined with the previous ones to set up a goal location.

4 Optional

Modify you `turtlebot_driver.py` in order to be able to read from a txt file (given as input for the launch file) a whole trajectory instead of just going to one point. In this case you will have to modify also the function `load_goals` and `next_goal`. The trajectory file will have the following structure:

x_0,y_0,θ_0

x_1,y_1,θ_1

. . .

x_n,y_n,θ_n

and will be passed as an argument called `file` to the launch file. Please find an example of a trajectory in `others/list.txt`

5 Lab report

Write a brief report (maximum 2-3 pages) explaining your solution and problems faced. Include the final `driver.py` file. If you have done the optional part, please submit the new launch file also.