

# Probabilistic Robotics

## Introduction to ROS- Lab Session

This document will guide you through the 1<sup>st</sup> practical work related with the Localization subject.

### 1. The goal

The goal of this Lab session is to get started with ROS, the robotics framework used in the next lab sessions.

Robot Operating System (ROS) is a software framework for robot software development. ROS provides libraries and tools to help software developers create 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, so it is fundamental to understand all the basics given in this session.

### 2. Work to do

#### 2.1 Pre-Lab: Installing Ubuntu and ROS

Currently the only official supported platform for ROS is Ubuntu. Ubuntu is a computer OS (Operating System) based on Debian Linux. As the computers in the Robotics Lab doesn't have Ubuntu installed, the firsts steps you need to do is install Ubuntu and ROS in your machines.

In order to install Ubuntu you can either:

- a) Install Ubuntu in a virtual machine inside your normal OS.
- b) Install Ubuntu in a partition alongside your other OS [**Advanced users**]. You can install Ubuntu and ROS by yourself following the instructions on their websites ([www.ubuntu.com](http://www.ubuntu.com); [www.ros.org](http://www.ros.org)), but please make sure you install the distributions specified below:

**Ubuntu 12.04.01 LTS and ROS Groovy.**

Let's have a deeper look at the first option.

## How to install Ubuntu Linux + ROS in a Virtual Machine?

### 1. Install VirtualBox:

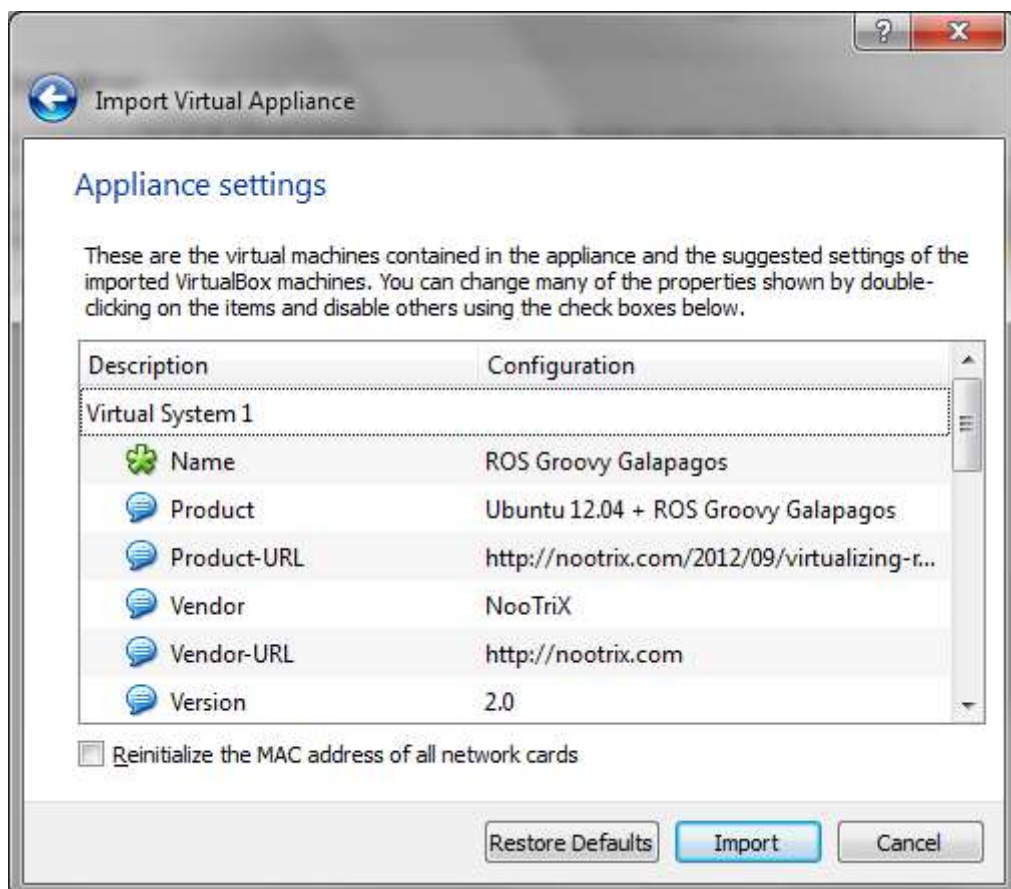
- Go to the Downloads section of the VirtualBox website:  
<https://www.virtualbox.org/wiki/Downloads>
- Choose the download link that matches your current OS.
- Execute the installer you've just download and follow the instructions.
- Download and install the VirtualBox Extension Pack you can find in the same page.

### 2. Download and install ROS VirtualMachine.

- Download an already packed version of Ubuntu + ROS for virtual machines.  
[3,3 GB]

<http://nootrix.com/wp-content/uploads/2013/01/rosGroovyGalapagos.ova>

- Open Virtual Box and select: File; Import Appliance  
Select the file you just downloaded. (ROS.ova)



- Click Import and Agree and wait until the importing finishes.

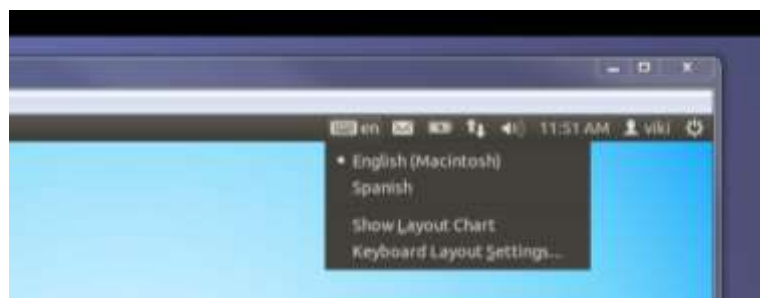
- Open again Virtual Box. In the settings menu you can choose all the system setting like the Base Memory, CPUs used, etc.


Set at least base memory to 2048 MB and CPUs to 2, to make Ubuntu run fluently.

- Select the ROS tab and click the Start Arrow!

Done! Ubuntu will start briefly.

- The default keyboard layout is in English (Macintosh). To change it, go to System Settings; Keyboard Layout and add your preferred layout to the list. After this you will have the ability to select the keyboard layout in the top right corner.



- The default users and passwords for the virtual machines are both: *viki*
- All the interaction with ROS is done though the terminal. To open a terminal in a new window. Click the  icon or press (Ctrl+ALT+T).

## 2.2 During the Lab: ROS Tutorials

Now that we 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 `ros_workspace` folder inside the home directory.

### ROS FUNDAMENTAL TUTORIALS [[www.ros.org/wiki/Tutorials](http://www.ros.org/wiki/Tutorials)]

---

**Navigating the ROS FileSystem**

<http://www.ros.org/wiki/ROS/Tutorials/NavigatingTheFilesystem>

---

**Creating a ROS package**

<http://www.ros.org/wiki/ROS/Tutorials/CreatingPackage>

---

**Understanding ROS Nodes**

<http://www.ros.org/wiki/ROS/Tutorials/UnderstandingNodes>

---

**Understanding ROS topics**

<http://www.ros.org/wiki/ROS/Tutorials/UnderstandingTopics>

---

**Understanding ROS services and Parameters**

<http://www.ros.org/wiki/ROS/Tutorials/UnderstandingServicesParams>

---

**Writing a simple Publisher and subscriber (Python)**

<http://www.ros.org/wiki/ROS/Tutorials/WritingPublisherSubscriber%28python%29>

---

**Examining the Simple Publisher and Subscriber**

<http://www.ros.org/wiki/ROS/Tutorials/ExaminingPublisherSubscriber>

---

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

### 2.3 During the Lab: 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 window:

```
# rosrun turtlesim turtlesim_node
```



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

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

Unzip the files for the lab from the course webpage in your catkin\_ws/src folder.

Make all ROS packages:

```
# cd ~/catkin_ws  
# catkin_make
```

Run the node in the turtle\_waypoint package:

```
# rosrun turtle_waypoint turtle_waypoint.py
```

You will see that nothing happens. You need to modify the turtle\_waypoint.py node inside the package in order to do the following:

The node publishes commands to the turtle in order to make the turtle go to a desired point. The desired position will be specified when running the node through the command line like:

```
roslaunch turtle_waypoint 1.2 5.0
```

(In this case the desired point is x: 1.2, 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 that 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
```

## 2.4 After the Lab

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