1. Ambiente operativo (server università)

Ubuntu 8.04.4 LTS

DISTRIB_ID=Ubuntu

DISTRIB_RELEASE=8.04

DISTRIB_CODENAME=hardy

DISTRIB_DESCRIPTION="Ubuntu 8.04.4 LTS"

Linux jobe 2.6.24-28-server #1 SMP Wed Nov 24 09:30:54 UTC 2010 x86_64 GNU/Linux

2. Installazione di Ros

2.1. Source Install

You will first need to setup your `sources.list` file to accept Debian packages from the ROS server.

Ubuntu 8.04 (Hardy)

```
sudo sh -c 'echo "deb http://code.ros.org/packages/ros/ubuntu hardy main" > /etc/apt/sources.list.d/ros-latest.list'
```

NOTE: After installing ROS on Hardy, some additional libraries will need to be installed. Make sure to follow the steps in Install Additional ROS Dependencies below.

Set up your keys

```
wget http://packages.ros.org/ros.key -O - | sudo apt-key add -
```

Installation

Make sure you have re-indexed the ROS.org server:

```
sudo apt-get update
```

There are many different libraries and tools in ROS. We provided four default configurations to get you started. You can also install ROS stacks individually.

PR2: ROS plus PR2-specific stacks, including PR2 simulator.

```
sudo apt-get install ros-boxturtle-pr2
```

Note: **You will get a prompt about hddtemp**: you can safely answer no to the prompt if you are not installing on an actual PR2. To avoid getting the prompt, you can set the debconf selection ahead of time:

```
echo "hddtemp hddtemp/daemon boolean false" | sudo debconf-set-selections
```

Install Additional ROS Dependencies (Hardy Only)

Not all of the dependencies for ROS on Ubuntu Hardy are available via apt, so you will need to compile and install boost and log4cxx into `/opt/ros`. This can be done very easily using rosdep:

```
. /opt/ros/boxturtle/setup.sh
```

```
rosdep install ros
```

Environment Setup

It's convenient if the ROS environment variables are automatically added to your bash session every time a new shell is launched:

```
echo "source /opt/ros/boxturtle/setup.sh" >> ~/.bashrc
```

```
. ~/.bashrc
```

3. Installazione del tool di installazione

Prerequisites

To install the tutorials you will need at least boxturtle-base installed before running the following, with it setup in your environment. Go back to boxturtle/Installation if you have not installed boxturtle-base.

Environment

If you are using binary installs.

```
. /opt/ros/boxturtle/setup.sh
```

Install tool

```
wget --no-check-certificate http://ros.org/rosinstall -O ~/rosinstall
```

```
chmod 755 ~/rosinstall
```

4. Completamento dell'installazione

Creazione di un nuovo package

Now we're going to go into your home or project directory and create our `bosch-ros-pkg` package.

Note that your installation of ROS is likely write-protected; in any case, it is unwise to modify the base installation without specific reasons. Instead you should create a new path in your **home** directory and prepend it to your ROS_PACKAGE_PATH as outlined below, and create additional packages in there. Prepending a path to ROS_PACKAGE_PATH causes all rosbash functions, such as roscd, to search through that path before moving on to the later paths, searching the default installation last. If you have trouble, it is useful to look at the ROS_PACKAGE_PATH documentation.

```
cd ~/
```

```
mkdir tesi_ros
```

```
export ROS_PACKAGE_PATH=~/tesi_ros:$ROS_PACKAGE_PATH
```

Note that the *export* line above must be run each time you open a new terminal (unless you edit your .bashrc file to do so automatically).

Modifica del file .bashrc

```
sudo gedit ~/.bashrc
```

alla fine del file devono esserci:

```
source /opt/ros/cturtle/setup.sh
export ROS_PACKAGE_PATH=/home/dani/tesi_ros:$ROS_PACKAGE_PATH
```

Installazione del package explore_stage

Now go into the ~/tesi_ros directory then create your package:

```
cd ~/tesi_ros
```

```
mkdir bosch-ros-pkg
```

```
cd bosch-ros-pkg
```

Percorso svn package da scaricare:

https://bosch-ros-pkg.svn.sourceforge.net/svnroot/bosch-ros-pkg/trunk/stacks/exploration/explore_stage

Istruzione da eseguire:

```
svn co https://bosch-ros-pkg.svn.sourceforge.net/svnroot/bosch-ros-pkg/trunk/
```

Rendere il package raggiungibile

Now lets make sure that ROS can find your new package. It is often useful to call *rospack profile* after making changes to your path so that new directories will be found:

```
rospack profile
```

```
rospack find explore_stage
```

```
YOUR_PACKAGE_PATH/explore_stage
```

If this fails, it means ROS can't find your new package, which may be an issue with your `ROS_PACKAGE_PATH`. Please consult the installation instructions for setup from SVN or from binaries, depending how you installed ROS. If you've created or added a package that's outside of the existing package paths, you will need to amend your ROS_PACKAGE_PATH environment variable to include that new location.

Posizionarsi in explore_stage:

```
roscd explore_stage
```

e controllare che contenga i file necessari:

```
ls
```

Dovresti visualizzare:

```
config explore.launch explore_slam.xml explore.xml move.xml
explore explore_slam.launch explore.vcg manifest.xml
```

Posizionarsi nel package explore:

```
roscd explore
```

E lanciare il make:

```
rosmake exploration
```

Installazione di rx-tools e di Rviz

Once all the system dependencies are installed, we can build our package that we just created. `rosmake` is just like the `make` command, but it does some special ROS magic. When you type `rosmake beginner_tutorials`, it builds the `beginner_tutorials` package, plus every package that it depends on, in the correct order. Since we listed rospy, roscpp, and std_msgs as dependencies when creating our ROS package, these packages (and their dependiencies, and so on) will be built by `rosmake` as well.

Utilizzo di rosdep install e di rosmake

```
rosdep install rxtools
```

```
rosmake rxtools
```

```
rosmake rviz
```

```
roscd stage
```

Lanciare RVIZ per visualizzare l'esplorazione:

```
rosrun rviz rviz -d $(rospack find stage)/rviz/stage.vcg
```

Dopo aver lanciato RVIZ, aggiungere la sottoscrizione ai topic MAP e MARKERS per vedere la mappa esplorata, i goal e il tracciato del percorso seguito.

Lanciare l'applicazione

Posizionarsi in explore_stage:

```
roscd explore_stage
```

E lanciare l'applicazione:

```
roslaunch explore_slam.launch
```

In caso di errori con il lancio di stageros (SEGFAULT), modificare la seguente riga nel file explore_slam.launch, aggiungendo il parametro "-g" (che avvia stageros senza la visualizzazione grafica):

```
<node pkg="stage" type="stageros" name="stage" args="-g $(find
bosch_worlds)/maze-noisy.world" respawn="false" output="screen"/>
```