Robotnik

RB-1 MODULAR MOBILE MANIPULATOR

System Software and Architecture Manual

Version 2.0

RBTNK-DOC-161020A

Robotnik Automation, S.L.L.

Contents

1 Software Architecture

This manual describes the RB-1 software architecture.

The RB-1 software architecture is based on ROS (Robot Operating System www.ros.org).

The second chapter gives a brief description of the ROS open source architecture. The third chapter describes the implementation of the ROS architecture in the RB-1 robot. The different robot software components are described then.

2 ROS Architecture

ROS is an open-source meta-operating system for your robot that provides inter-process message passing services (IPC) in a network.

ROS is also an integrated framework for robots that provides:

- Hardware abstraction layer
- Low level device control
- Robot common functionality (simulation, vision, kinematics, navigation, etc.)
- IPC
- Package and stack management

ROS provides libraries and tools to easy the development of robot software in a multi-computer system.

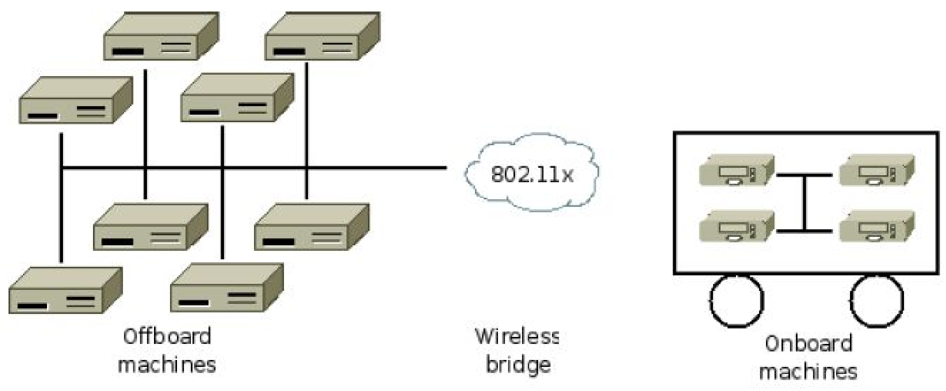


Figure 1 - ROS typical network configuration

ROS offers a framework to solve common research and development needs of robot systems:

- Cooperation of different research groups
- Proven functionality
- Easy and robust access to robotics hardware

One of the main objectives of ROS is the code reusability. This objective is fulfilled by a large and growing community of developers that share their results worldwide, and by the inclusion of other robot frameworks (ROS integrates Player/Stage, Orocos, etc.) and other open-source components (like Gazebo or Openrave).

ROS integrates additional development tools like rviz (simulation of complete robots and environments with maps), rqt_graph (visualization of node interconnection), rosbag (extreme useful data logging utility), etc.

For detailed systems descriptions, tutorials, and a really important number of stacks and packages, please visit www.ros.org.

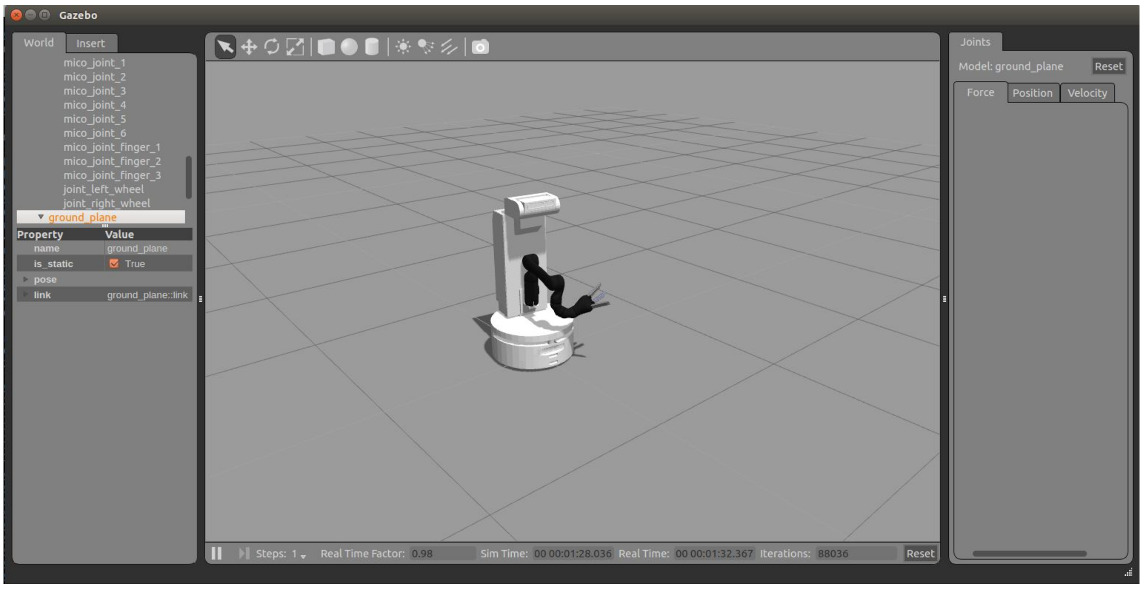


Figure 2 – ROS gazebo robot simulation

3 RB-1 Robot Architecture in ROS

RB-1 - ROS architecture is the result of the cooperative operation of several nodes. The robot has ROS Indigo installed, and robot specific software is provided in 6 stacks (metapackages):

- **rb1_robot**: includes all the packages required for the real robot control

- **rb1_sim**: allows the simulation of the robot and its sensors in Gazebo.

- **rb1_common**: packages shared between the robot and simulation stacks, i.e. robot description, navigation, etc.

- **rb1_base_robot**: includes all the packages required for the real robot base control.

- **rb1_base_sim**: allows the simulation of the robot base and its sensors in Gazebo.

- **rb1_base_common**: packages shared between the robot and simulation stacks, i.e. robot description, navigation, etc.

Additional documentation about the contained packages can be found in the README.md of the repos:

https://github.com/RobotnikAutomation/rb1_sim
https://github.com/RobotnikAutomation/rb1_robot
https://github.com/RobotnikAutomation/rb1_common

https://github.com/RobotnikAutomation/rb1_base_sim
https://github.com/RobotnikAutomation/rb1_base_robot
https://github.com/RobotnikAutomation/rb1_base_common

The simulated robot publishes almost the same data as the real robot and accepts the same commands thus allowing to easy test programs in simulation and directly testing on the real robots.

The default workspace of the robot is located at:

```
> /home/rb1/catkin_ws/
```

In addition the robot includes additional packages depending on the installed components (installed via apt-get or from sources). The source code packages installed locally is located at:

```
> /home/rb1/sources
```

Local packages are linked symbolically from the workspace.

A number of components used by the robot can be found in the gitHub repo:

https://github.com/RobotnikAutomation

Necessary packages are:

- **robotnik_msgs**

Simple package that contains standard services and messages commonly used in mobile robots.

https://github.com/RobotnikAutomation/robotnik_msgs

- **robotnik_sensors**

A package that contains the URDF files that describe the sensors that are mounted in the robot. These are used for simulation, but also for the robot description, that is used for visualization or packages as MoveIt!.

https://github.com/RobotnikAutomation/robotnik_sensors

Other **optional components** that may be installed in your robot and that can be downloaded from the gitHub. The most common are:

- **kinova-ros**

Package that implements the control of the Kinova JACO and MICO arms.

- **robotnik_trajectory_suite**

Package that interfaces between kinova-ros and MoveIt!

- **orbbec_camera**

Package that implements a device driver for Orbbec Astrea RGBD cameras.

- **hokuyo_node/urg_node**

Driver to read the scan data from Hokuyo laser devices.

- **rly_816**

Driver to access to the IO device RLY 816.

4 Network Configuration

RB-1 is equipped with a router that provides connectivity through ethernet and wireless. The network configuration is as follows:

IP Range: 192.168.0.X
Netmask: 255.255.255.0
Gateway: 192.168.0.1

DHCP: 192.168.0.[50-100]

Wifi SSID: RB1-161020A (A..Z)
Wifi Password: R0b0tn1K (R and K capital letters)

RB-1 router and CPU have the following configuration:

RB-1 Router:
User/Password: root / R0b0tn1K (R and K capital letters)
Router IP Address: 192.168.0.1

RB-1 CPU:
User/Password: rb1 / R0b0tn1K (R and K capital letters)
RB-1 IP Address: 192.168.0.200

The easiest way to access the RB-1 Modular Mobile Manipulator is to use a Secure Shell (SSH) client.

First, you need to connect to the RB-1 Network using the RB-1 Router SSID and password. Then, open your preferred SSH client and connect using the RB-1 CPU IP Address, user and password. If you use the OpenSSH client:

```
user@remote:~$ ssh rb1@192.168.0.200
```

5 Robot startup

5.1 Start-up sequence

Take a look to the back panel of the robot:

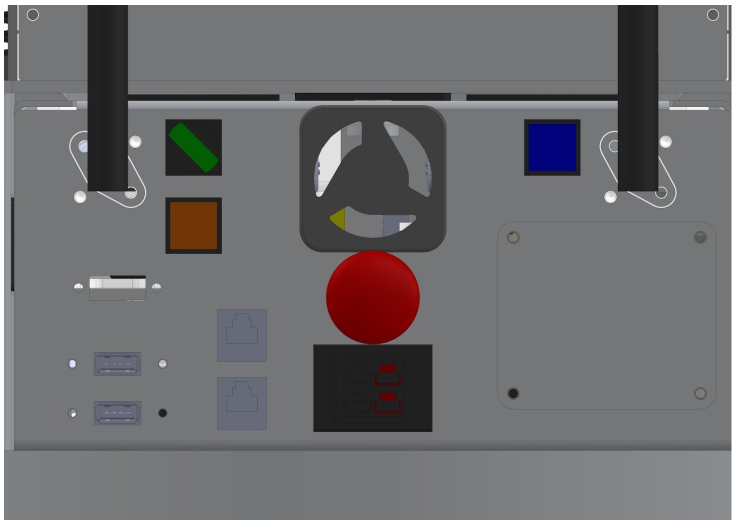
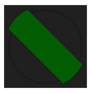
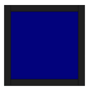
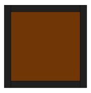
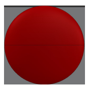


Figure 3. RB-1 back panel.

The following elements are the main components of the back panel necessary to start-up the robot:

	**ON/OFF switch**
	**CPU power button**
	**Motors reset button**
	**Emergency button**

The general ON/OFF SWITCH (green) must be activated for giving energy to all the elements of the system. Press the CPU POWER BUTTON (blue) button to turn ON the computer, the blue button will light up. At this moment the PC (Linux) starts-up and loads all the necessary files for booting.

After booting, it it is possible to connect to the system in a remote way or connect to the robot manually.

To move the robot, the EMERGENCY BUTTON (red) must be pulled out and the RESET BUTTON (orange) must be pressed once.

NOTE: Remember that the robot is able to reach high speeds, use the higher speeds only in open areas.

5.2 PS3 Dualshock Controller

This Game-pad has a smooth and precise control. The Bluetooth receiver is connected to an USB port at the computer.

To control the movements of the Robot and its devices with this controller, please follow the next instructions:

1. Switch on the Robot
2. Switch on the computer pressing the blue button. (The computer indicator gets illuminated)
3. Wait a minute until the computer is started.
4. Press the Motors reset button (orange).
5. Power on the Gamepad pressing the Start button.
6. The four red leds will flash until the connection is established with the Bluetooth receiver, and then, only the led 1 should remain on. If the controller can not reach this state, check the computer and the Bluetooth receiver and restart the robot. If the problem is not solved, see next section about pairing.
7. Pressing the Dead man button you should be able to move the robot.
8. If the led 1 blinks while the others are off you need to charge the controller. You can connect it to any USB port with the provided cable.

By default, the pad has three operation modes, to move independently the base, the torso and the arm of the robot. To enable each operation mode, the corresponding Deadman Button must be pressed:

R1	Base Deadman Button
R2	Torso Deadman Button

- L1 Arm Deadman Button

NOTE: If the Bluetooth connection is lost, the robot will detect this situation and will STOP for safety.

5.3 PS4 Dualshock Controller

This Game-pad has a smooth and precise control. The Bluetooth receiver is connected to an USB port at the computer.

To control the movements of the Robot and its devices with this controller, please follow the next instructions:

1. Switch on the Robot
2. Switch on the computer pressing the blue button. (The computer indicator gets illuminated)
3. Wait a minute until the computer is started.
4. Press the Motors reset button (orange).
5. Power on the Gamepad pressing the Share + PS button for 5 seconds, until the pad's front light starts to blink.
6. Wait a few seconda more until the pad's front light turns fixed.

By default, the pad has three operation modes, to move independently the base, the torso and the arm of the robot. To enable each operation mode, the corresponding Deadman Button must be pressed:

● R1	Base Deadman Button
● R2	Torso Deadman Button
● L1	Arm Deadman Button

NOTE: If the Bluetooth connection is lost, the robot will detect this situation and will STOP for safety.

5.4 Base pad

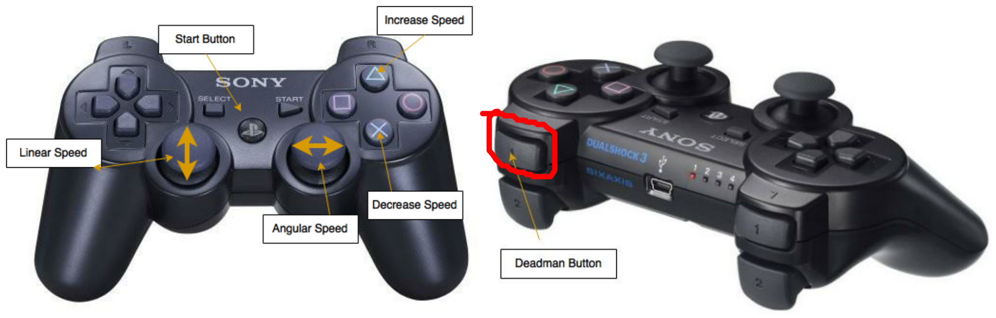


Figure 4 – PS3 Base Pad

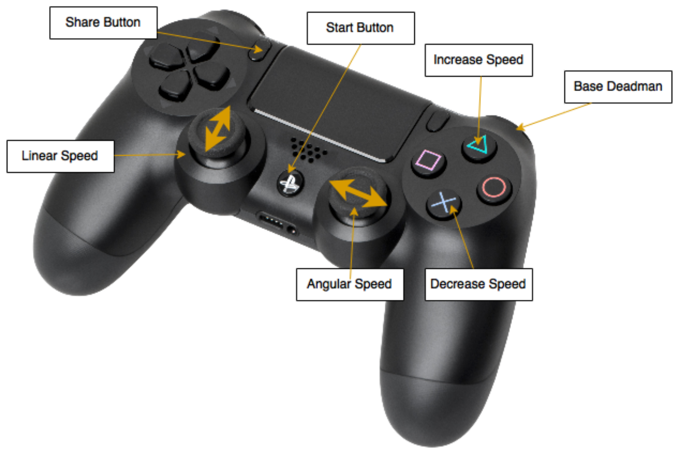


Figure 5 - PS4 Base Pad

The Base Deadman button (default R1) must be pressed to move the base.

Left and right joysticks control the linear (forward/backward movement) and the angular (rotational) speeds.

Triangle and Cross buttons control the speed of the base, which by default is set to 10% of the maximum speed.

5.5 Torso pad

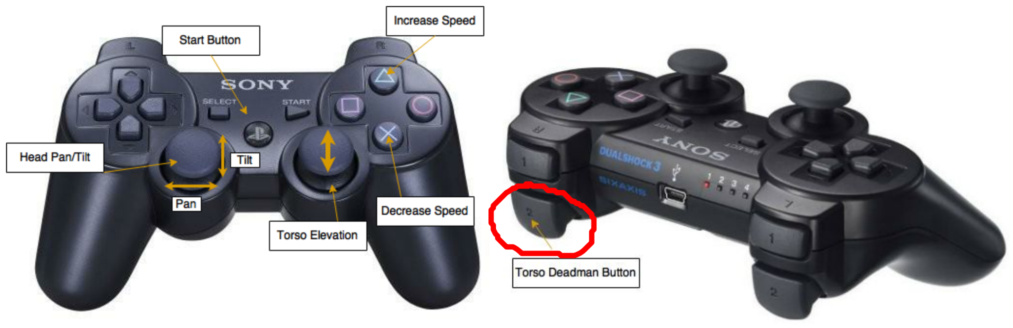


Figure 6 – PS3 Torso Pad

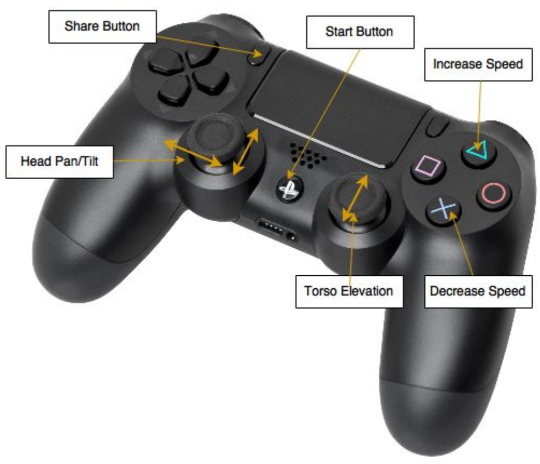


Figure 7 - PS4 Torso Pad

The Torso Deadman button (default R2) must be pressed to move the torso and the pan/tilt head.

Left joystick controls the Pan/Tilt head, while right joystick controls the elevation of the torso.

Triangle and Cross buttons control the speed of the base, which by default is set to 10% of the maximum speed.

5.6 Arm Pad

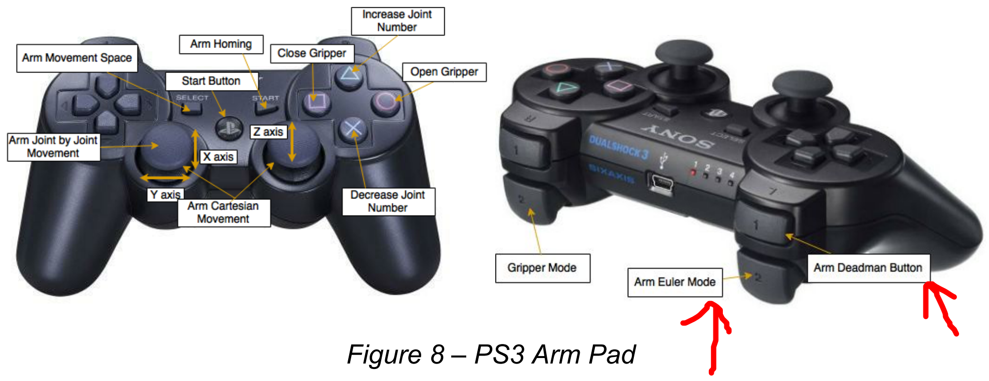


Figure 8 – PS3 Arm Pad

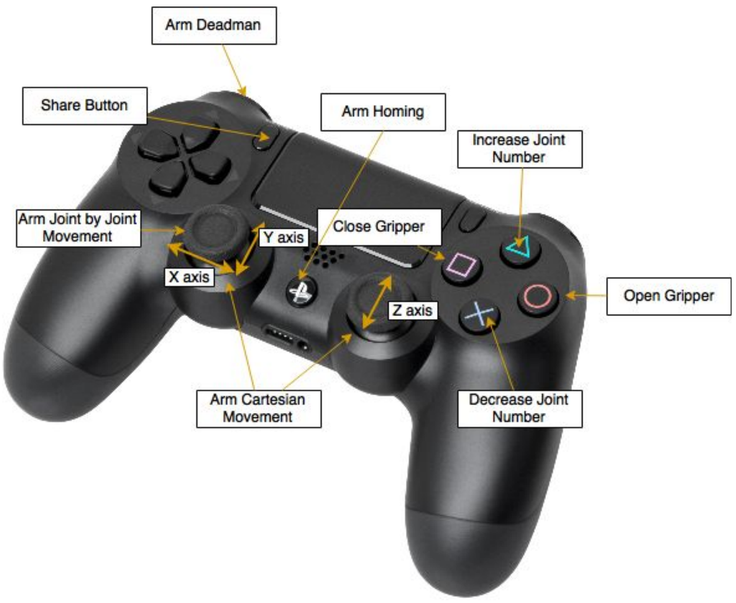


Figure 9 – PS4 Arm Pad

The Arm Deadman button (default L1) must be pressed to move the arm.

The Arm Movement Space (default Select) button selects the arm space movement: cartesian space or joint space.

In cartesian space movement, the Arm Euler Mode (default L2) selects to move the arm linearly (x, y, z) or rotationally (yaw, pitch, roll). In linear movement, the left joystick controls the X and Y axes position, while the right joystick controls the Z axis position. In orientation mode, the left joystick controls the rotation around the X and Y axes, while the right joystick controls the rotation around the Z axis.

In joint space movement, the left joystick controls the position of the current joint. Press Increase (default Triangle) and Decrease (default Cross) joint number button to change the current joint number.

The Arm Homing (default start) moves the arm to its home position. Prior to control the gripper the arm must me moved to its home position.

Open and Close finger buttons (default Circle and Square) close the fingers of the gripper. Gripper Finger Mode (default R2) select to control two or three fingers in the gripper.

It is easy to change the buttons associated to each function. Check the .yaml files that contain the button assignments, located at:

```
> ~/catkin_ws/src/rb1_base_common/rb1_base_pad/launch/ps3.yaml

> ~/catkin_ws/src/rb1_common/rb1_torso_pad/launch/ps3.yaml

> ~/catkin_ws/src/kinova-ros/kinova_pad/launch/ps3.yaml
```

You can know the number of each button with the command:

```
rb1@rb1:~/$ jstest /dev/input/js0
```

5.7 Pairing the Dualshock controller with the Bluetooth device.

If you have received one Dualshock together with your robot, it should be already paired and you don't need to do this process.

There can be only one PS3 joystick associated with the robot's Bluetooth device. If you want to associate another PS3 controller or the usual one has lost its association you have to pair it again with the following procedure:

1. Shutdown the robot CPU (for example, by pressing once the CPU POWER BUTTON (blue)).
2. Plug the usb cable to the PS3 joystick and to the robot USB port.
3. Start the robot CPU by pressing the CPU POWER BUTTON (blue). During the boot up process the PS3 pad and the robot will be automatically paired.
4. Unplug the USB cable from the joystick and robot. Wait a minute until the boot up process is finished.
5. Power on the Gamepad by pressing the Start button. Now you are ready to move the robot.

The robot will try to pair a pad at each boot up process. This procedure is specified in the robot root `.bashrc` file, as shown in the section 7 (Scripts and Start Configuration) of this manual.

5.8 How to charge the Gamepad battery

If you see that all the Gamepad leds are powered off, probably you will need to charge the gamepad battery. You can connect it to any USB port with the provided cable (with the CPU powered on).

6 Remote PC

6.1 Software installation and PC configuration

Some metapackages of the software have to be present in the remote computer in order to operate the robot remotely. In order to test the different components, the most useful tools are rviz (ROS Visualization tool) and rqt. In order to use these **in a remote machine**, we will need to:

1. Install ROS. To do that, follow the instructions at: http://wiki.ros.org/ROS/Installation. If your are not familiar with ROS, a good starting point is the Tutorials section of the official ROS Documentation: http://wiki.ros.org/ROS/Tutorials

2. Create a `catkin_workspace`, following the instructions at: http://www.ros.org/wiki/catkin/Tutorials/create_a_workspace

3. Install `rb1_common` and `rb1_sim` stacks in the remote computer (see github repositories). The second is not necessary, but is needed to simulate the robot.

4. Compile the stacks.

The recommended procedure to install the RB-1 packages in a remote PC is as follows:

1. Download the sources to a local folder called `sources` locate at the `$HOME` directory:

```
user@remote:~$ mkdir ~/sources
user@remote:~$ cd ~/sources
user@remote:~/sources/$ git clone \
            https://github.com/RobotnikAutomation/rb1_common
user@remote:~/sources/$ git clone \
            https://github.com/RobotnikAutomation/rb1_sim
```

2. Create a catkin workspace at the `$HOME` directory:

```
user@remote:~$ mkdir ~/catkin_ws/src -p
user@remote:~$ cd ~/catkin_ws/src
user@remote:~/catkin_ws/src$ catkin_init_workspace
```

3. Even though the workspace is empty (there are no packages in the 'src' folder, just a single CMakeLists.txt link) you can still "build" the workspace:

```
user@remote:~/catkin_ws/src$ cd ~/catkin_ws/
```

```
user@remote:~/catkin_ws/$ catkin_make
```

4. Create symbolic links in the workspace folder to the `rb1_common` and the `rb1_sim` folders at sources:

```
user@remote:~/catkin_ws/$ cd ~/catkin_ws/src
user@remote:~/catkin_ws/src/$ ln -sf ~/sources/rb1_common
user@remote:~/catkin_ws/src/$ ln -sf ~/sources/rb1_sim
```

5. And rebuild the workspace:

```
user@remote:~/catkin_ws/src/$ cd ~/catkin_ws
user@remote:~/catkin_ws/$ catkin_make
```

Once ROS is installed in your machine, you will need to setup your network in order to connect to the robot. A detailed guide can be found at http://wiki.ros.org/ROS/NetworkSetup. To connect your computer to the RB-1, you must follow the next steps:

Add the following line to the `/etc/hosts` file in your computer:

```
user@remote:~$ sudo vim.tiny /etc/hosts

# add the following line
192.168.0.200   rb1
```

Start the RB-1 robot, connect your computer to its wifi network, open a terminal and type:

```
user@remote:~$ export ROS_MASTER_URI=http://rb1:11311

user@remote:~$ rostopic list
```

And you'll see a list of ROS topics where ROS nodes are interchanging information.

- Add your computer hostname to the /etc/hosts file in the RB-1 computer:

```
user@remote:~$ ssh rb1@rb1
rb1@rb1:~$ sudo vim.tiny /etc/hosts
# add your hostname and ip address there
```

If everything worked you should be able to see all the topics published by the robot and the services offered by the robot controller:

```
rb1@rb1:~$ rostopic list
...
rb1@rb1:~$ rosservice list
...
```

You can view the values published by the nodes with rostopic echo, e.g.:

```
user@remote:~$ rostopic echo /rb1_control/odom
```

At this stage you can have a first look at the robot with

```
user@remote:~$ rqt
```

and

```
user@remote:~$ rosrun rviz rviz
```

6.2 Component testing

Instructions about how to launch and test each of the components are documented in the README.md file of each component and can be accessed via gitHub.

6.3 Robot simulation

Instructions for the robot simulation can be found in the README.md file of the rb1_sim repository.

7 Scripts and Start Configuration (in the robot)

The bootup process usually does not need to be changed by the user, but it is explained to simplify customization.

The robot control programs are launched during the boot-up process. This is achieved by starting three terminals with autologin, each one of those runs a different process. Each terminal is configured by a file locate at:

```
> /etc/init/ttyX.conf
```

where X is a number between 1 and 3. `TTY1` is configured to login as `root` user, while `TTY2` and `TTY3` are configured to login as `rb1` user.

Then, the program run by each terminal is defined by the `.bashrc` file located at the `$HOME` directory of each user:

```
> /root/.bashrc

> /home/rb1/.bashrc
```

7.1 /etc/init/ttyX.conf

Three terminals with autolog are initiated during boot

`> /etc/init/tty1.conf` calls mingetty and autologs as `root` in `TTY1` terminal. Its content is:

```
exec /sbin/mingetty --autologin root tty1
```

`> /etc/init/tty2.conf` calls mingetty and autologs as `rb1` in `TTY2` terminal. Its content is:

```
exec /sbin/mingetty --autologin rb1 tty2
```

`> /etc/init/tty3.conf` calls mingetty and autologs as `rb1` in `TTY3` terminal. Its content is:

```
exec /sbin/mingetty --autologin rb1 tty3
```

7.2 .bashrc

This is a start script, which every user has in its home directory. It’s content is executed each time the user logins on the system.

When the `root` user logs in the `TTY1` terminal, the PS3 Pad pairing tool is run. To do that, the following code has been added to the `/root/.bashrc` file:

```
> /root/.bashrc

#### BOOT ####
    echo "ROBOTNIK RB-1"
    Terminal=`tty`
    case $Terminal in
       "/dev/tty1") sleep 25;
      sixpair;
      sixad --start;;
    esac
```

When the `rb1` user logs in the `TTY2` terminal, a roscore instance is run. When the `rb1` user logs in the `TTY3` terminal, the `rb1_complete.launch` file from the `rb1_bringup` package is run, which brings up all the components of the robot. To do that, the following code has been added to the `/home/rb1/.bashrc` file:

```
> /home/rb1/.bashrc

#### BOOT ####
    echo "ROBOTNIK RB-1"
    Terminal=`tty`
    case $Terminal in
       "/dev/tty2") roscore;;
       "/dev/tty3") sleep 20;
       roslaunch rb1_bringup rb1_complete.launch;;
    esac
```

Some robots use alternative joystick drivers instead of the *sixpair*.

Note that the sleep value has been adjusted. If the server starts before the dynamic devices have been recognized, the server will exit with fault and the user will need to connect to the robot to start the server manually.