SUMMIT XL STEEL

MOBILE ROBOT

SOFTWARE MANUAL

Robotnik Automation,
Spain

RBTNK-DOC-SXLS0-190507AA

Contents

1. Software Architecture

This manual describes the SUMMIT XL STEEL software architecture.

The SUMMIT XL STEEL 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 SUMMIT XL STEEL 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 ease the development of robot software in a multi-computer system.

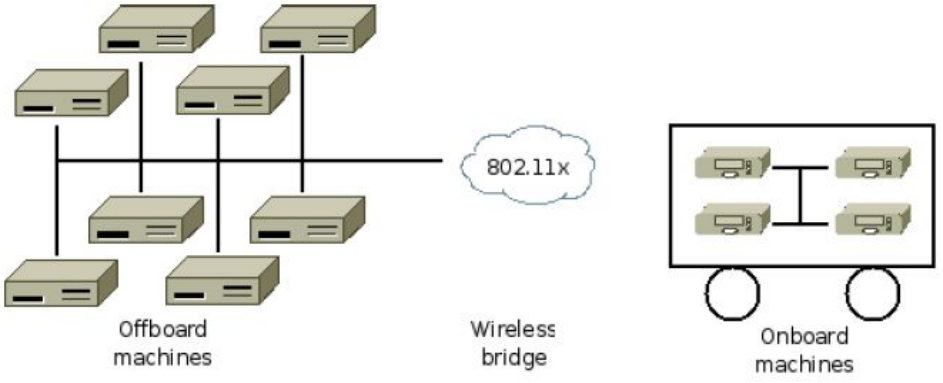


Figure 1 - ROS typical network configuration

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), rxgraph (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.

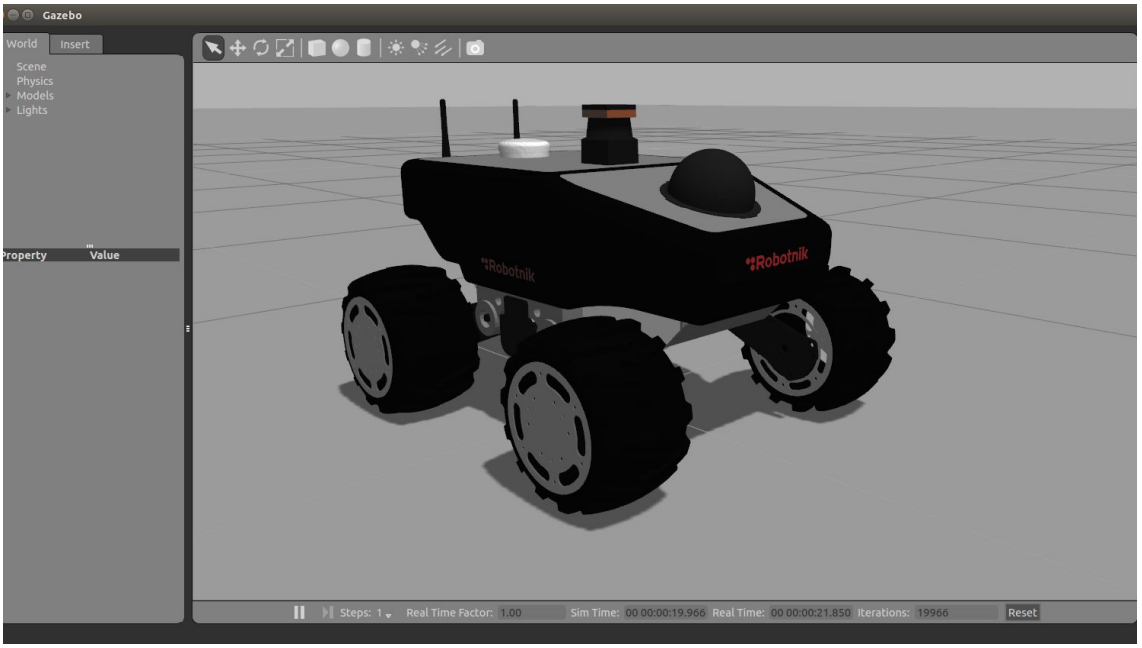


Figure 2 – ROS gazebo robot simulation

3. SUMMIT XL STEEL Robot Architecture in ROS

SUMMIT XL STEEL - ROS architecture is the result of the cooperative operation of several nodes. The robot has ROS Kinetic Desktop Full installed.

ROS SUMMIT XL STEEL specific software is provided in three stacks (metapackages):

- **summit_xl_sim**: allows the simulation of the robot and its sensors in Gazebo.
- **summit_xl_robot**: includes all the packages required for the robot control
- **summit_xl_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/summit_xl_sim
https://github.com/RobotnikAutomation/summit_xl_robot
https://github.com/RobotnikAutomation/summit_xl_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.

3.1 SUMMIT_XL_COMMON

- **summit_xl_description**

This package contains the robot models available in the SUMMIT XL STEEL robot.

- **summit_xl_control**

This package contains the ROS Control controllers used by the SUMMIT XL STEEL robot

- **summit_xl_localization**

This package contains several algorithms used in the localization of the SUMMIT XL STEEL robot, both at the local level (odometry) and global level (maps, gps).

- **summit_xl_navigation**

This package contains the configuration to use the SUMMIT XL STEEL robot with the ROS Navigation stack.

- **summit_xl_pad**

This package contains the ROS node to read from the gamepad controller.

3.2 SUMMIT_XL_ROBOT

- **summit_xl_bringup**

This package contains the launch files needed to run the SUMMIT XL STEEL robot.

- **summit_xl_controller**

This package contains the ROS Control controller developed for the SUMMIT XL STEEL robot.

- **summit_xl_web**

This package contains the web interface to the SUMMIT XL STEEL robot

3.3 SUMMIT_XL_SIM

- **summit_xl_sim_bringup**

This package contains the launch files needed to run the robot simulation.

- **summit_xl_gazebo**

This package contains the robot simulation using Gazebo.

3.4 Additional packages.

In addition the robot includes additional packages depending on the installed components (installed via apt-get or from sources). Packages installed from sources can be found in the `$HOME/sources` folder and are linked to the `$HOME/catkin_ws/src` folder.

Additional software components used by the robot can be found in the Robotnik Automation 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

SXLS0-190507AA needs also the following packages to access it's sensors:

- **nmea_navsat_driver**

This package provides a ROS interface for GPS devices that output compatible NMEA sentences.

```
> sudo apt-get install ros-kinetic-nmea-navsat_driver
```

- **axis camera**

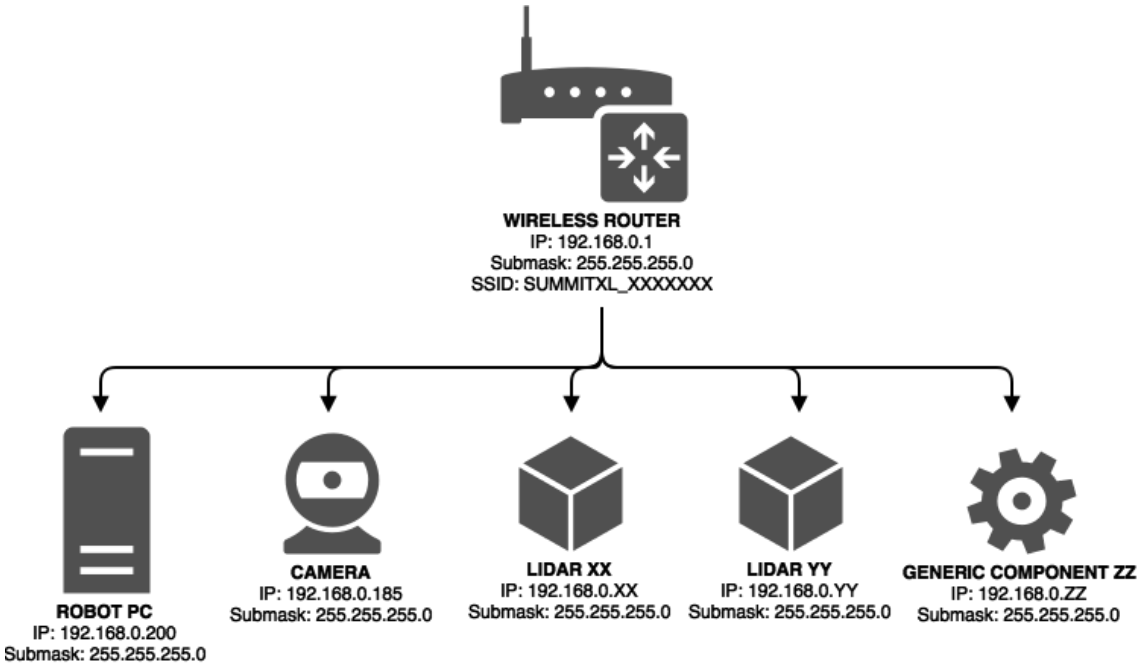
https://github.com/RobotnikAutomation/axis_camera

The following packages were installed as dependencies:

```
> sudo apt-get install -y ros-kinetic-transmission-interface
ros-kinetic-joint-state-controller ros-kinetic-navigation
ros-kinetic-ros-control ros-kinetic-ros-controllers
ros-kinetic-velocity-controllers ros-kinetic-control-toolbox
ros-kinetic-cmake-modules ros-kinetic-serial
```

```
ros-kinetic-joystick-drivers ros-kinetic-rosbridge-server
ros-kinetic-robot-localization ros-kinetic-twist-mux
ros-kinetic-rosbridge-suite ros-kinetic-imu-tools
ros-kinetic-mavros-* ros-kinetic-teb-local-planner
ros-kinetic-slam-gmapping
```

4. Network Configuration



You will find information in how to configure the network following the steps in http://www.ros.org/wiki/ROS/NetworkSetup

4.1 Ethernet

Connect to the router by plugging an Ethernet cable to the robot back pannel:

Summit - Static IP address 192.168.0.200
Subnet Mask 255.255.255.0
Gateway address: 192.168.0.1

The router is configured to give the connected computer an IP address via DHCP.

4.2 Wireless

The wifi router configuration by default is the following:

You can start connecting with

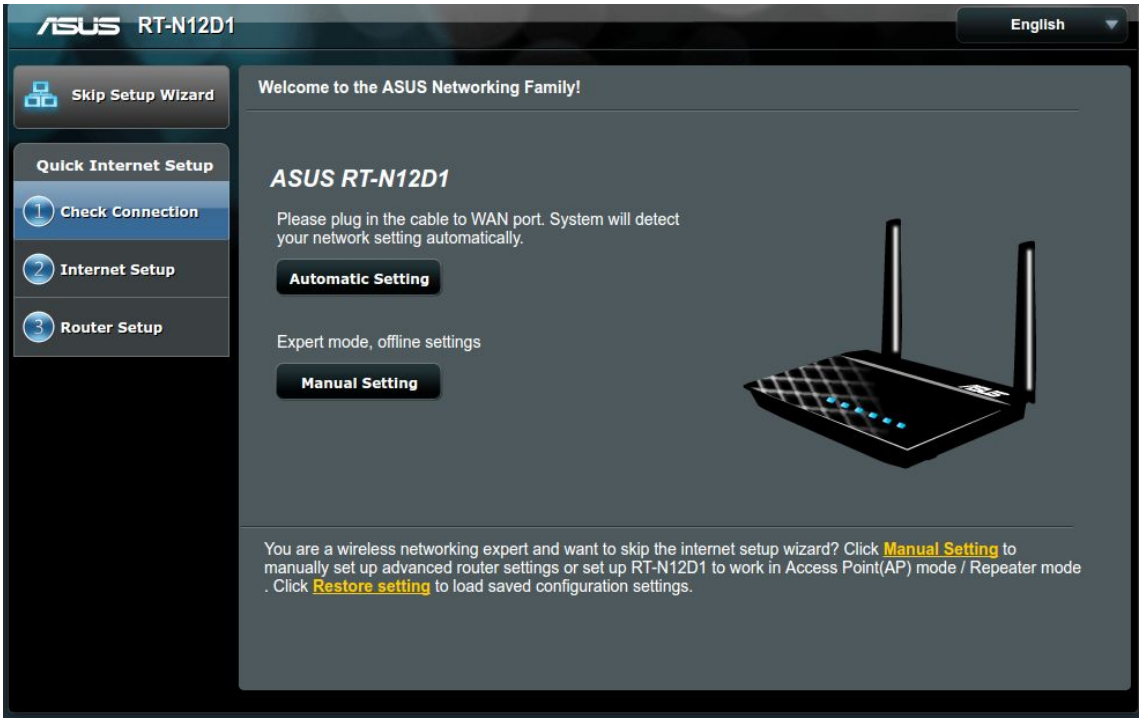
Wifi SSID: SXLS0-190507AA
Wifi Password: R0b0tn1K (R and K capital letters)

4.2.1 Configuring the router as a repeater

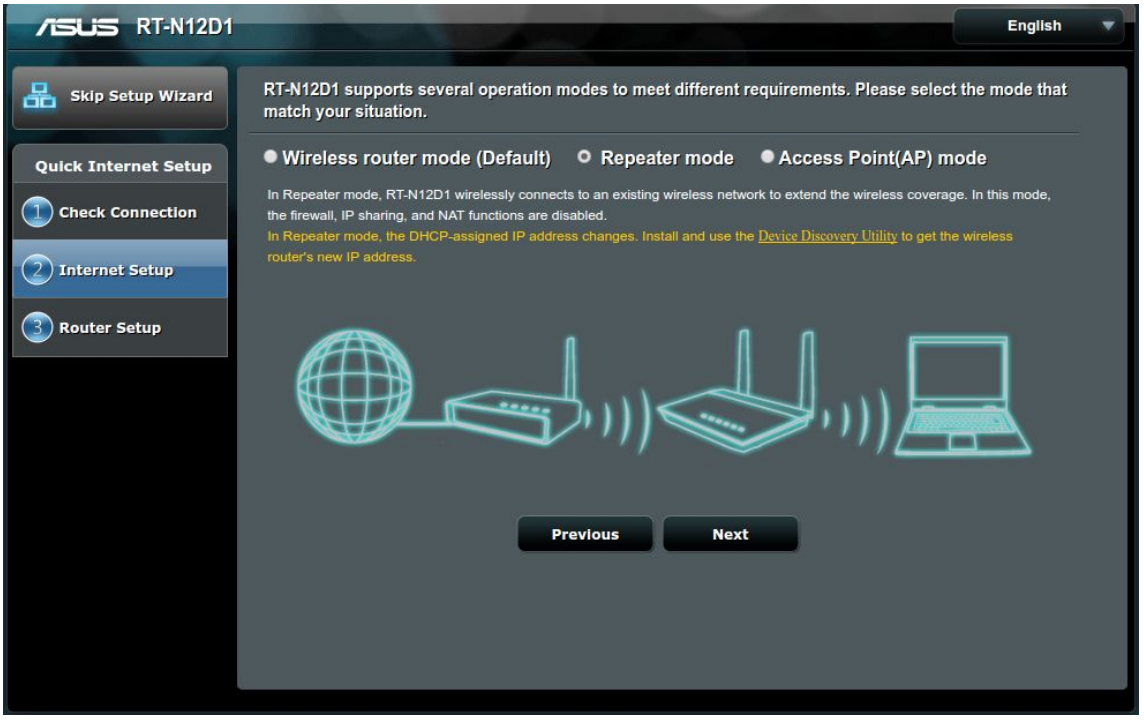
Asus RT-N12D1

The router can be configured as repeater, enabling the robot to connect to an external WiFi network and extending the range of it.

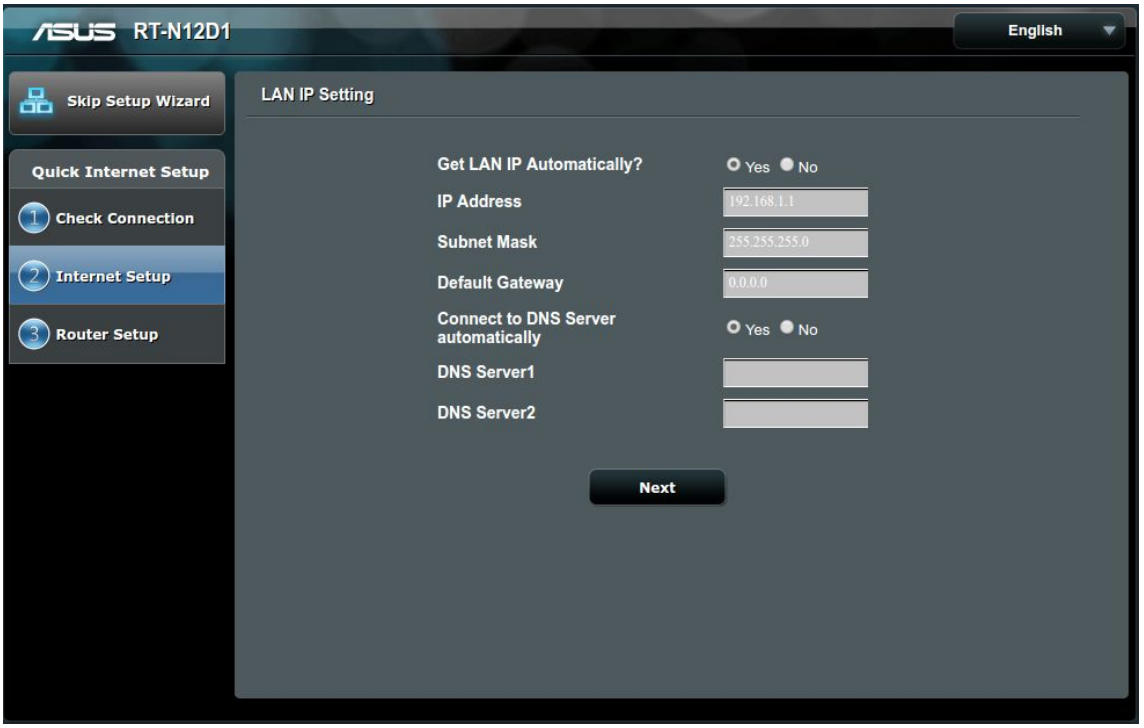
1. Acces to the router panel via a network browser (router.asu.com or the current ip address)
2. Reset the configuration to factory values or go to the Setup Wizard.
3. Check connection:
    a. Select Manual Setting



4. Login information setup
5. Internet Setup:
    a. Select “Repeater Mode”



6. Select the network you want to connect.
7. Wireless settings.
    a. Use the default settings
8. LAN IP Settings
    a. Set ‘Yes’ to get the LAN IP automatically.



9. Apply the changes and wait until the router applies the modifications.

NOTE: don’t forget to update the network configuration in the pc controller in

order to work with the new ip.

4.3 Connected devices

At the beginning of section 4, you could picture a generic diagram of the network with the IP and submask. In this section, you could also find the user and password for the default connected devices.

Router:
User/Password: admin / R0b0tn1K
Router IP Address : 192.168.0.1

Robot Computer:
User/Password: summit / R0b0tn1K
Robot IP Address : 192.168.0.200

UR 5 Control Unit:
User/Password: -
Robot IP Address : 192.168.0.X

5. Robot startup

5.1 Start-up sequence

Take a look to the back panel of the robot:

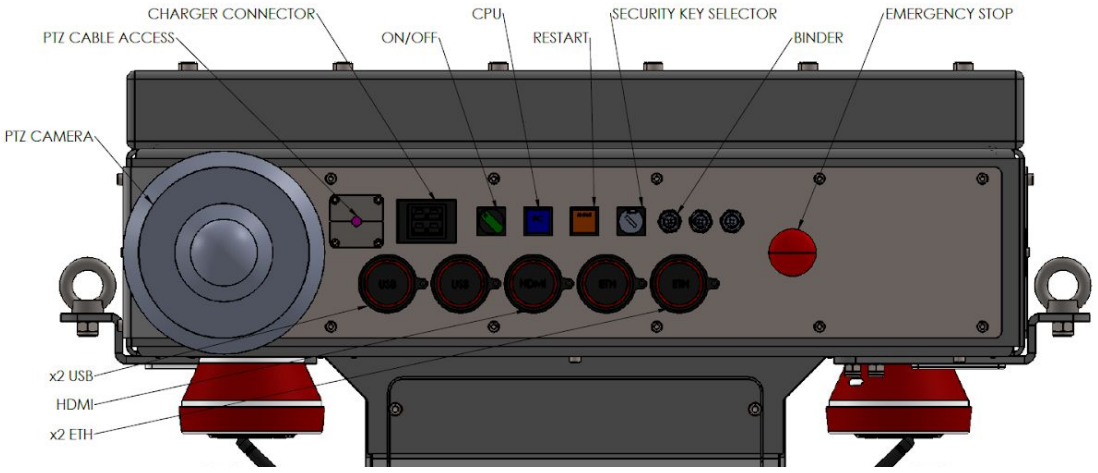


Figure 1. SUMMIT XL STEEL back panel

- **EMERGENCY STOP** will disable the drivers and stop the robot. CAUTION there is no rearm button, so the robot will keep moving when the Emergency Stop is released.
- General **ON/OFF** key: cuts the power of the whole robot. It has a green light indicator.
- **CPU POWER** blue indicator/switch: turns on and off the computer
- **RESTART:** orange indicator button restarts the power of robot.
- **CHARGER CONNECTOR**: to connect the provided battery charger
- Three **binder** connectors: 5,12 VDC BAT. Intended to power external devices and protected with fuse
- Two free **USB** 2.0 ports
- Two **Ethernet** ports WAN and LAN
- One **HDM**I port.

The general ON/OFF SWITCH (green) must be activated for giving energy to all the elements of the system. Then, press the CPU POWER BUTTON (blue) button to turn ON the computer, the blue button will light up. At this moment the PC (Linux) boots up and loads all the necessary programs to operate the robot.

To move the robot, the **EMERGENCY BUTTON** (red) must be pulled out.

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

5.2 Robot Remote Controller

This robot uses the PS4 DualShock as a remote controller. This gamepad has a smooth and precise control. It is connected to the computer through Bluetooth, with a Bluetooth receiver is connected to an USB port at the computer. The `ds4drv` software package is required to access properly to the PS4 DualShock Controller.

The gamepad has been already paired with the computer. To start the remote controlling just press the PS Button. In INDIGO, the share button and the start button must be pressed at the same time. By default, the configuration of the buttons is the following:

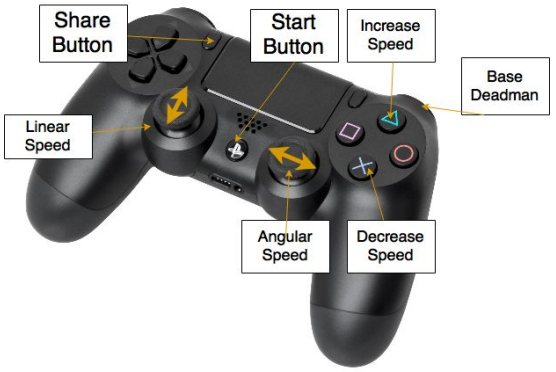


Figure 2 – Base Operation Mode

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

It is easy to change the buttons functions. Check the .yaml files that contain the button assignments located at the summit_xl_pad folder.

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

```
> jstest /dev/input/js0
```

5.3 Pairing the Dualshock controller with the Bluetooth

This part only applies to PS3 Dualshock Controllers. If you have received a 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:

-Shutdown the robot CPU (by pressing the CPU power button several seconds or through ssh:

```
$ > sudo shutdown –h now
```

-Plug the usb cable to the PS3 joystick and to the robot USB port

-Start the robot CPU pressing the CPU Power. Wait until the Axis camera has been initialized

-Unplug the USB cable from the joystick and robot

- Power on the Gamepad pressing the Start button.

The joystick is trying to be paired each time the robot starts. This procedure is written in the robot root .bashrc file and calls the following nodes when the robot starts:

```
$ > sudo sixpair

$ > sudo sixad -s (or sudo sixad --start)
```

5.4 How to charge the Gamepad battery

Connect the gamepad to a standard micro-usb charger.

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 the remote machine**, we will need to:

1. Install ROS
2. Install summit_xl_common and summit_xl_sim stacks in the remote computer (see gitHub repositories). The second is not necessary, but is useful if we want to simulate the robot.
3. Compile the stacks:

   Configure a catkin workspace if you don't have it:

```
> mkdir ~/catkin_ws/src
> cd ~/catkin_ws/src
> catkin_init_workspace
```

```
> mkdir -p ~/catkin_ws/src
> cd ~/catkin_ws/src
> catkin_init_workspace
```

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:

```
> cd ~/catkin_ws/
> catkin_make
```

The creation of catkin_workspaces is described in:
http://www.ros.org/wiki/catkin/Tutorials/create_a_workspace

After that copy or link the summit_xl_common and the summit_xl_sim folders to this workspace

```
> cd ~/sources
> git clone https://github.com/RobotnikAutomation/summit_xl_common
> git clone https://github.com/RobotnikAutomation/summit_xl_sim
> cd ~/catkin_ws/src
> ln -sf ~/sources/summit_xl_common
> ln -sf ~/sources/summit_xl_sim
```

And compile:

```
> cd ~/catkin_ws
> catkin_make
```

Once ROS is installed in your machine you will need to configure this machine so that ROS connects to the SUMMIT XL STEEL ROS_MASTER

Add the following line to your /etc/hosts file

```
192.168.0.200    SXLS0-190507AA
```

Start the SUMMIT XL STEEL robot, connect to its wifi network, open a terminal and type:

```
> export ROS_MASTER_URI=http://SXLS0-190507AA:11311

> rostopic list
```

Add your computer hostname in the rover /etc/hosts file

```
> ssh summit@SXLS0-190507AA
> sudo vim /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:

```
> rostopic list
…
> rosservice list
…
```

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

```
> rostopic echo /summit_xl_control/odom
```

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

```
> rqt
```

and

```
> 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 summit_xl_sim repository.

6.4 Robot diagnostics

The easiest way to check the robot status and get a fast diagnostic is by accessing its internal web server (package map_nav_manager).

Just write the robot address in the browser: http://192.168.0.200:9001

See the specific map_nav_manager guide for more information.

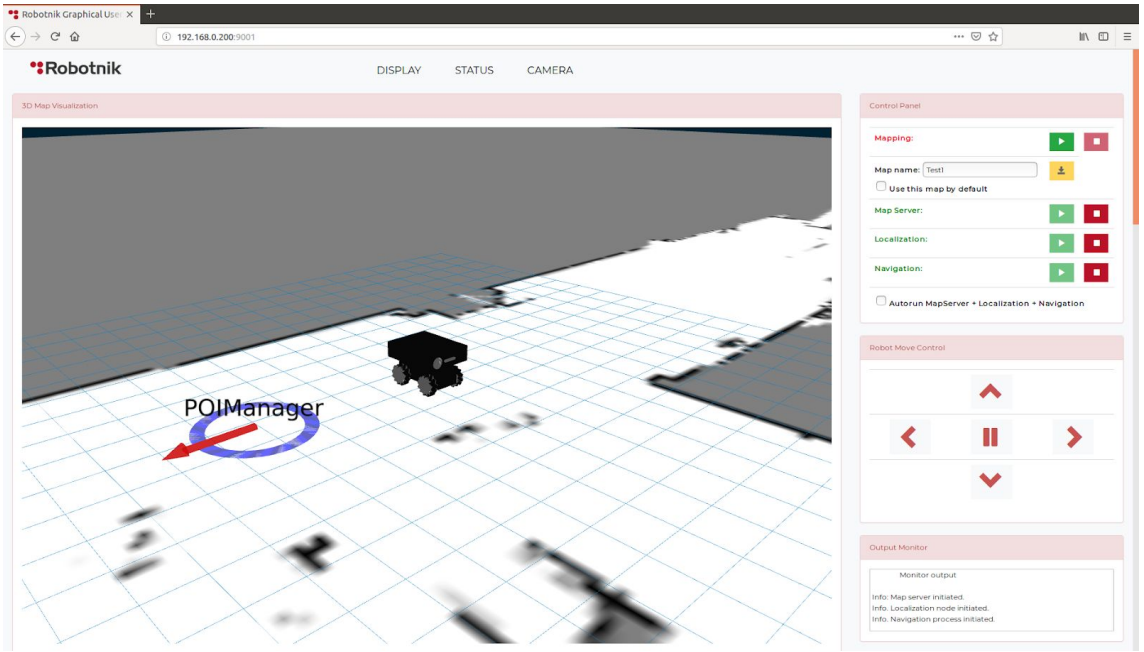


Figure 4 – Map Nav Manager

7. Scripts and Start Configuration (in the robot)

For the Start-up and the execution of the robot control programs, the SUMMIT XL STEEL is configured with some scripts. These scripts are called from several TTY that are launched during the boot up. This usually does not need to be changed by the user, but it is explained to simplify customization.

7.1 TTY

/etc/init/tty1.conf calls mingetty and autologs as root in tty1 terminal:

```
#exec /sbin/getty -8 38400 tty1
exec /sbin/mingetty --autologin root tty1
```

/etc/init/tty2.conf calls mingetty and autologs as jr2 user in tty2 terminal:

```
#exec /sbin/getty -8 38400 tty2
exec /sbin/mingetty --autologin jr2 tty2
```

/etc/init/tty3.conf calls mingetty and autologs as jr2 user in tty3 terminal:

```
#exec /sbin/getty -8 38400 tty3
exec /sbin/mingetty --autologin jr2 tty3
```

/etc/init/tty4.conf calls mingetty and autologs as jr2 user in tty4 terminal:

```
#exec /sbin/getty -8 38400 tty4
exec /sbin/mingetty --autologin jr2 tty4
```

7.2 .bashrc

This is a start script, which every user has in its home directory. You edit this file for the user you used in the before script `tty2` and `tty3`. For the user `summit`, you need to edit the file `$HOME/.bashrc`.

The following code has been added at the end of the file:

```
# SUMMIT XL STEEL
```

```
# AUTOBOOT
echo "ROBOTNIK SUMMIT XL STEEL"
      Terminal=`tty`
      case $Terminal in
      "/dev/tty2") roscore;;
      "/dev/tty3") sleep 20;
            roslaunch summit_xl_bringup summit_xls_complete.launch;;
      "/dev/tty4") sleep 20;
            cd /home/summit/catkin_ws/src/summit_xl_robot/summit_xl_web;
            python -m SimpleHTTPServer;;
      esac
```

For the user script `tty1`. For the user `root`, the file `/root/.bashrc` has been modified to include:

```
#### BOOT ####
    echo "ROBOTNIK SUMMIT XL STEEL"
    Terminal=`tty`
    case $Terminal in
       "/dev/tty1") sleep 25;
            ds4drv;;
    esac
```

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.