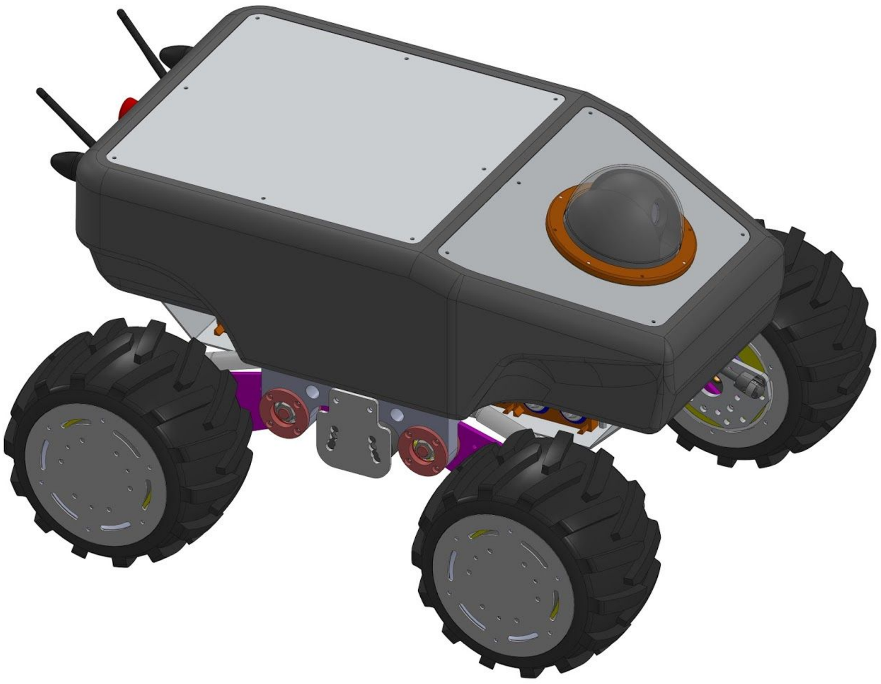
SUMMIT XL
MOBILE PLATFORM



SYSTEM SOFTWARE ARCHITECTURE MANUAL

RBTNK-DOC-160609A

Robotnik Automation SLL, Spain

Contents

1. Software Architecture

This manual describes the SUMMIT XL software architecture.

The SUMMIT XL 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 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.

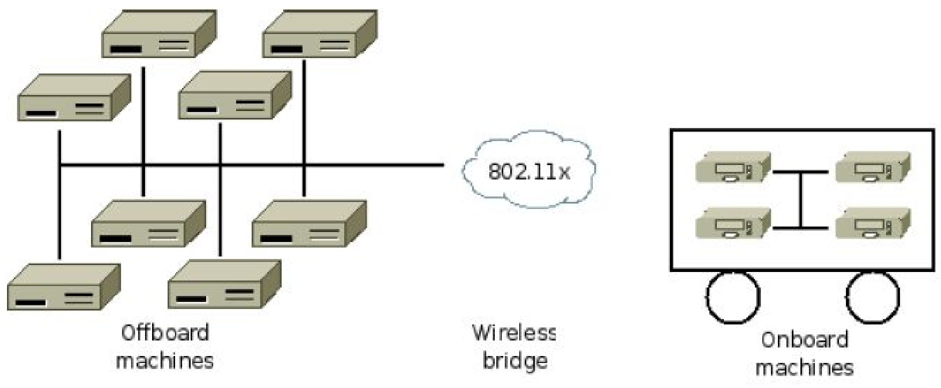


Figure 1 - ROS typical network configuration

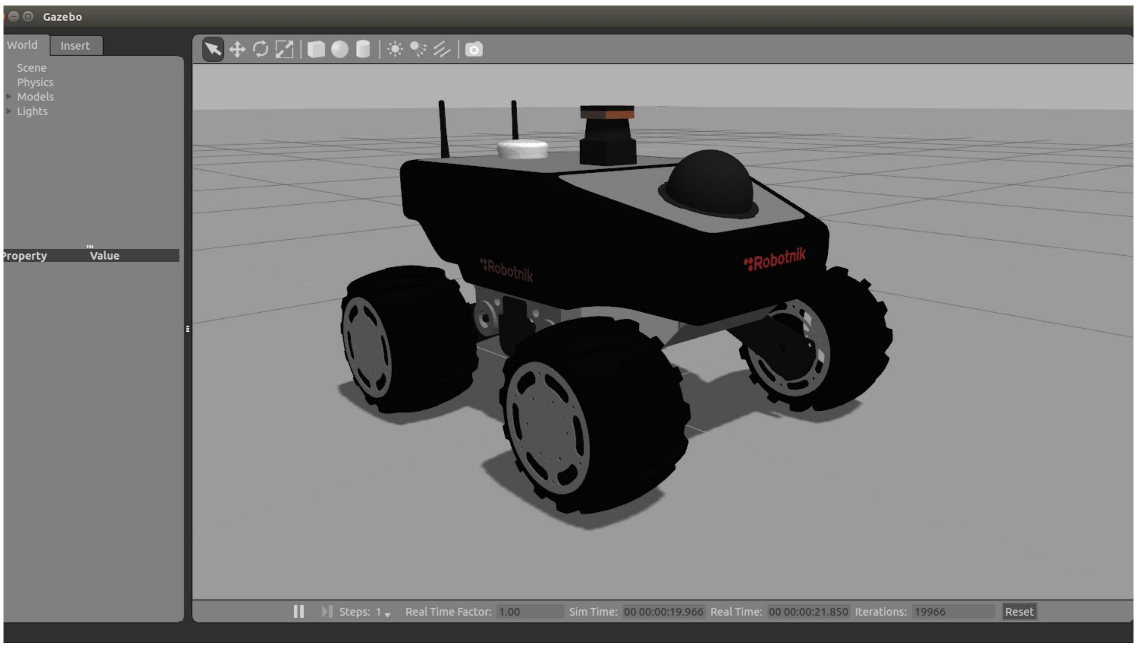
ROS offers a framework to solve common research and development needs of robot systems:

- 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), 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.



Steps: 1 Real Time Factor: 1.00 Sim Time: 00 00:00:19.966 Real Time: 00 00:00:21.850 Iterations: 19966

Figure 2 – ROS gazebo robot simulation

3. SUMMIT XL Robot Architecture in ROS

SUMMIT XL - 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 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.

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

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

- **robotnik_trajectory_suite**

A trajectory control metapackage. This package is needed if your Summit XL robot mounts a robot arm. It coordinates the trajectories generated by MoveIt! and sends them to the specific actuators.

https://github.com/RobotnikAutomation/robotnik_trajectory_suite

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

- **hokuyo_node**

A package for the Hokuyo laser range finder.

- **openni2_camera**

Driver for the orbecc frontal 3d camera.

4. Network Configuration

Install the matching ros version in the remote machine (follow instructions in www.ros.org).

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:
Wifi SSIDs: Summit XL (A..Z)
Wifi Password: R0b0tn1K (R and K capital letters)

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

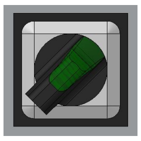
Robot Computer:
User/Password: summit / R0b0tn1K

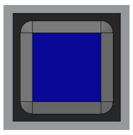
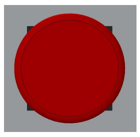
And the password for the AXIS camera is also R0b0tn1K

5. Robot startup

5.1 Start-up sequence

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

	**ON/OFF switch**

	**PC button**
	**Emergency button**

The general ON/OFF switch (green) must be activated for giving energy to all the elements of the system. Press the 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 is possible to connect to the system in a remote way or connect to the robot manually.

To move the robot, the EMERGENCY button 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 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 must be illuminated)
3. Wait until the computer is started.
4. Power on the Gamepad pressing the Start button.
5. 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.

6. Pressing the Dead man button you should be able to move the robot and the Axis camera (optional).
7. 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.

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

By default, the configuration of the buttons is the following:

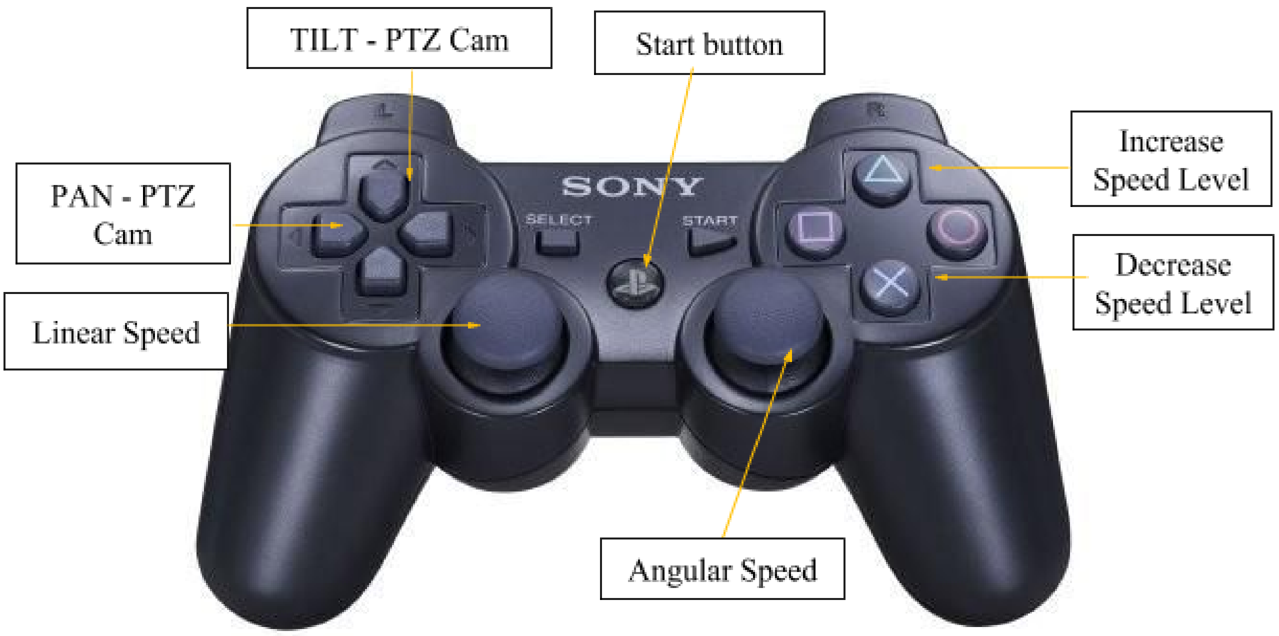


Figure 2 – Dualshock top view

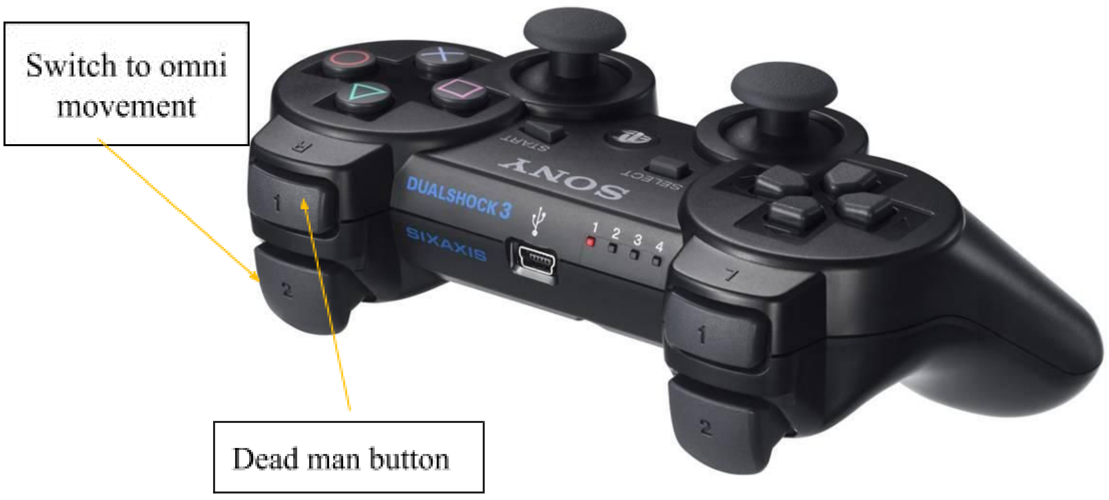


Figure 3 – Dualshock front view

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

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

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

```
ps3.yaml
num_of_buttons: 15
axis_linear: 1
axis_angular: 2
scale_linear: 1.0
scale_angular: 2.0
button_speed_up: 12
button_speed_down: 14
button_dead_man: 11
button_output_1: 15
button_output_2: 13
output_1: 0
output_2: 2
button_ptz_tilt_up: 4
button_ptz_tilt_down: 6
button_ptz_pan_right: 5
button_ptz_pan_left: 7
```

5.3 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:

-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:

```
rosrun ps3joy sixpair
rosrun ps3joy ps3joy.py
```

5.4 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 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 ROS_MASTER

Add the following line to your /etc/hosts file

```
192.168.0.200   summit
```

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

```
>export ROS_MASTER_URI=http://summit:11311

>rostopic list
```

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

```
>ssh summit@summit
>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

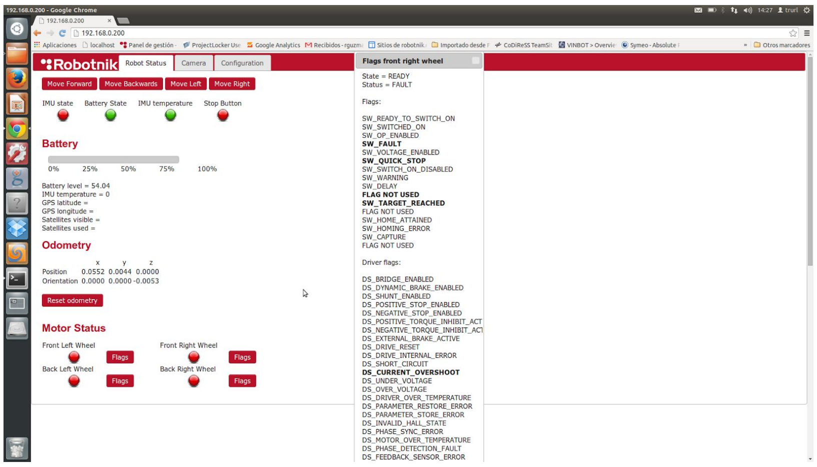
Instructions for the robot simulation can be found in the README.md file of the summit_xl_sim repository.

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 summit_xl_web of summit_xl_robot).

Just write the robot address in the browser: http://summit

The status tab gives information about errors, robot odometry, imu status, e-stop button status, battery level, as many other variables. In addition it allows you to see the status of the servoamplifiers and their internal flags.



Battery level = 54.04
IMU temperature = 0
GPS latitude =
GPS longitude =
Satellites visible =
Satellites used =

```
               x       y       z
Position       0.0552  0.0044  0.0000
Orientation    0.0000  0.0000 -0.0053
```

State = READY
Status = FAULT

Flags:

SW_READY_TO_SWITCH_ON
SW_SWITCHED_ON
SW_OP_ENABLED
SW_FAULT
SW_VOLTAGE_ENABLED
SW_QUICK_STOP
SW_SWITCH_ON_DISABLED
SW_WARNING
SW_DELAY
FLAG NOT USED
SW_TARGET_REACHED
FLAG NOT USED
SW_HOME_ATTAINED
SW_HOMING_ERROR
SW_CAPTURE
FLAG NOT USED

Driver flags:

DS_BRIDGE_ENABLED
DS_DYNAMIC_BRAKE_ENABLED
DS_SHUNT_ENABLED
DS_POSITIVE_STOP_ENABLED
DS_NEGATIVE_STOP_ENABLED
DS_POSITIVE_TORQUE_INHIBIT_ACT
DS_NEGATIVE_TORQUE_INHIBIT_ACT
DS_EXTERNAL_BRAKE_ACTIVE
DS_DRIVE_RESET
DS_DRIVE_INTERNAL_ERROR
DS_SHORT_CIRCUIT
DS_CURRENT_OVERSHOOT
DS_UNDER_VOLTAGE
DS_OVER_VOLTAGE
DS_DRIVER_OVER_TEMPERATURE
DS_PARAMETER_RESTORE_ERROR
DS_PARAMETER_STORE_ERROR
DS_INVALID_HALL_STATE
DS_PHASE_SYNC_ERROR
DS_MOTOR_OVER_TEMPERATURE
DS_PHASE_DETECTION_FAULT
DS_FEEDBACK_SENSOR_ERROR

Figure 4 – Robot web server status page

The web interface permits also to control the ptz camera or even the robot motion.

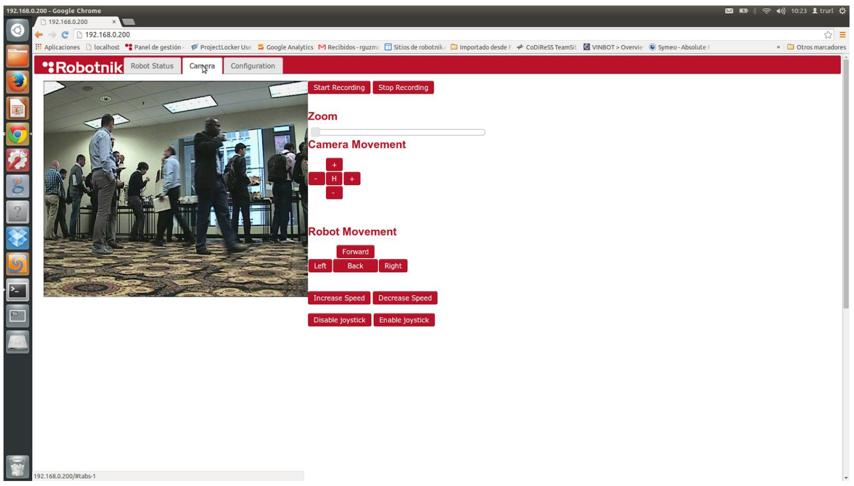


Figure 4 – Robot web server control page

7. Scripts and Start Configuration (in the rover)

For the Start-up and the execution of the robot control programs, the Summit XL is configured with some scripts. This usually does not need to be changed by the user, but it is explained to simplify customization.

7.1 /etc/init/ttyX.conf

/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 summit user in tty2 terminal:

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

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

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

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 *ttyX.* For the user *summit*, you need to edit the file /home/summit/.bashrc.

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

```
#### BOOT ####
   echo "ROBOTNIK SUMMIT XL"
   Terminal=`tty`
   case $Terminal in
     "/dev/tty2") roscore;;
     "/dev/tty3") sleep 20;
     roslaunch summit_xl_bringup summit_xl_complete.launch;;
   esac
```

For the user script *ttyX.* For the user *root*, the file /home/root/.bashrc has been modified to include:

```
#### BOOT ####
   echo "ROBOTNIK SUMMIT XL"
   Terminal=`tty`
   case $Terminal in
     "/dev/tty1") sleep 25;
          rosrun ps3joy sixpair;
          rosrun ps3joy ps3joy_node.py;;
   esac
```

Some robots use alternative joystick drivers as *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.