GUARDIAN MOBILE ROBOT

System Architecture Manual

Version 4.0

Robotnik Automation, S.L.L.

15/03/2012

RBTNK-DOC-120315A

INDEX

1. Software Architecture

This manual describes the GUARDIAN software architecture.

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

The next section gives a brief description of the ROS open source architecture, and the following sections describe the different robot software components.

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.

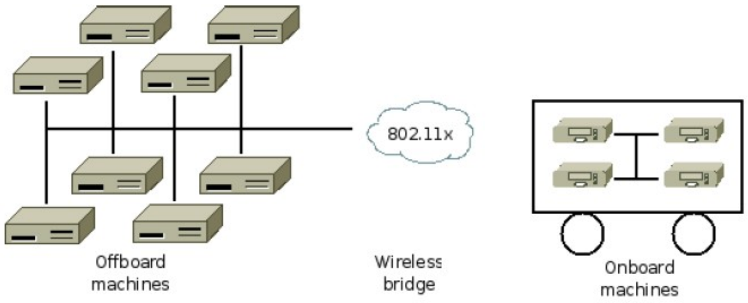


Figure 1 - ROS typical network configuration

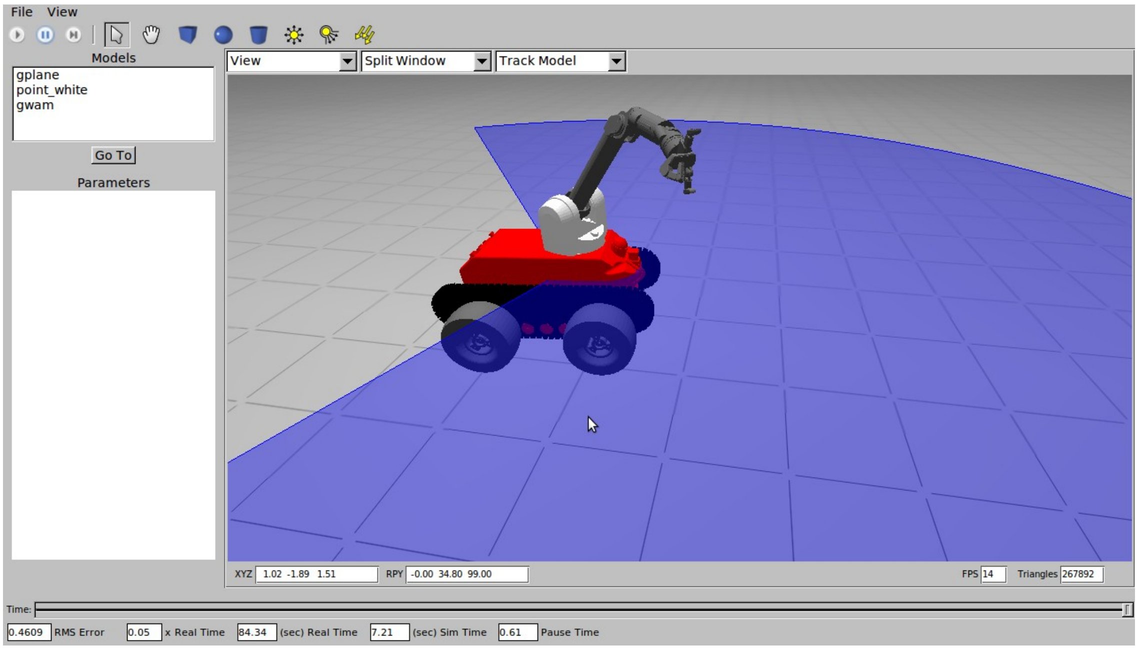
ROS offers a framework to solve common research and development needs of robot systems:

- 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*.



gplane
point_white
gwam

XYZ 1.02 -1.89 1.51 RPY -0.00 34.80 99.00 FPS 14 Triangles 267892

0.4609 RMS Error 0.05 x Real Time 84.34 (sec) Real Time 7.21 (sec) Sim Time 0.61 Pause Time

Figure 2 – ROS Gazebo Guardian simulation

3. GUARDIAN Robot Architecture in ROS

GUARDIAN ROS architecture is the result of the cooperative operation of several nodes.

3.1 GUARDIAN Stacks

The Guardian software is provided in two ROS stacks, one for simulation and another for the real robot:

guardian_sim : simulation stack

This stack provides all the necessary packages for the robot simulation (including teleoperation and AMCL/SLAM navigation).

The stack includes the following packages:

- **guardian_2dnav**
- **guardian_controller**
- **guardian_description**
- **guardian_joystick_teleop**
- **guardian_odometry**
- **guardian_tf**

Official repository: ***https://gwam-ros-pkg.googlecode.com/svn/trunk/guardian***

guardian_robot : robot control stack

This stack provides all the necessary packages for the real robot control and navigation. The stack includes some adapted sensor drivers and it has the following packages:

- guardian_node
- guardian_pad
- guardian_navigation
- guardian_static_tf
- sphereptz
- usb_cam
- microstrain_3dmgx2_imu
- modbus_io
- guardian_complete
- openni_kinnect
- hokuyo_node

The stack makes use of other ROS stacks not mentioned like audio_common, laser_drivers, etc.

3.2 GUARDIAN Simulation stack

The **guardian_sim** simulation stack is composed of the following packages:

- **guardian_2dnav**

This package contains all the configuration files needed to execute the AMCL and SLAM navigation algorithm in simulation.

- **guardian_controller**

It’s the robot’s Gazebo plugin controller. It implements the control of the differential kinematics of the Guardian robot.

- **guardian_description**

It contains the urdf, meshes, and other elements needed in the description are contained here.

- **guardian_joystick_teleop**

This package allows controlling the robot using a joystick or gamepad, sending the messages received through the joystick input, correctly adapted, to the "guardian_controller/cmd_vel" topic. It also allows controlling the pan-tilt.

- **guardian_odometry**

It computes the odometry of the robot using the joint movements and publishes these values to the topic “/odom”.

- **guardian_tf**

It publishes the TFs (Transform configurations) of the robot.

- pantilt_2632hd_sim

It‘s the pantilt’s Gazebo plugin controller. It simulates the motors for the Pan-Tilt motion of the device.

GUARDIAN Robot Stack

The **guardian_robot** real robot control stack is composed of the following packages:

- **guardian_node**

This node controls the communication with the motors and publishes the robot's odometry.

- **guardian_pad**

This node has the same functionality as the simulation one. Apart from the simulation functionality, it allows controlling the sphere cam pan-tilt motion and the lights of the robot.

- **guardian_static_tf**

Simple package that publishes a homogeneous transform needed by the navigation stack.

- **sphereptz**

It controls the pan-tilt motion of the Logitech Sphere cam.

- **usb_cam**

It manages the camera of the Logitech Sphere cam.

- **microstrain_3dmgx2_imu**

Package based on the ROS package ***http://ros.org/wiki/microstrain_3dmgx2_imu*, ** developed by Jeremy Leibs and Blaise Gassend with minor modification***.*** It configures and reads the inertial measurement and publishes the values.

- **modbus_io**

Package intended to manage the digital and analog input/output board included in to the robot. The communication with this device is through the modbus protocol.

- **guardian_complete**

It contains several launch files in order to launch all the components of the robot at the same time.

- **hokuyo_node**

It manages the communication with Hokuyo laser sensor.

- **openni_kinnect**

It manages the communication with the Kinnect device.

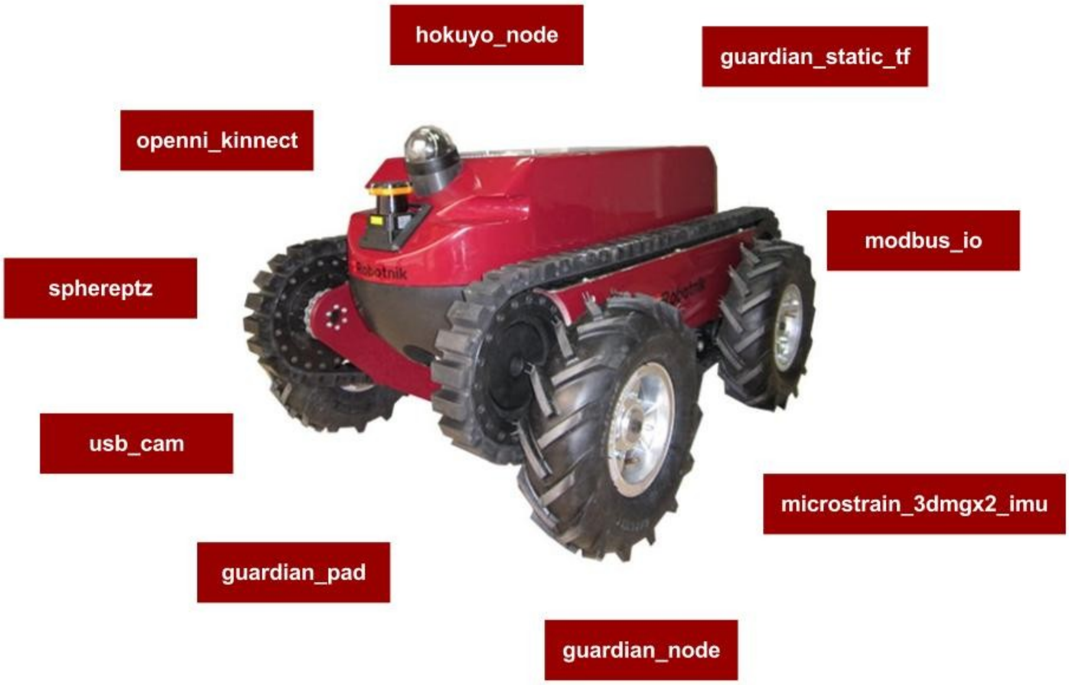


Figure 3 - GUARDIAN's ROS nodes