Rosbridge web interface

Martin Blaha, Michal Kreč, Petr Marek,
Tomáš Nouza, Tadeáš Lejsek

blahama7@fel.cvut.cz, krecmich@fel.cvut.cz, marekpe9@fel.cvut.cz,
nouzato1@fel.cvut.cz, lejsetad@fel.cvut.cz

May 28, 2013

Available at
https://cw.felk.cvut.cz/doku.php/misc/projects/nifti/sw/start

**Supervisor: Ing. Michal Reinštein, Ph.D.,
Ing. Marek Polčák**

Center for Machine Perception, Department of Cybernetics
Faculty of Electrical Engineering, Czech Technical University
Technická 2, 166 27 Prague 6, Czech Republic
fax +420 2 2435 7385, phone +420 2 2435 7637, www: http://cmp.felk.cvut.cz

Rosbridge web interface

Martin Blaha, Michal Kreč, Petr Marek,
Tomáš Nouza, Tadeáš Lejsek

May 28, 2013

Abstract

A universal web GUI interface for mobile robots running ROS (Robot Opearating System) was developed as a semester project during the course A3M99PTO (in Czech called "Práce v týmu a její organizace"). The system allows to operate a mobile robot remotely using only a web browser. It does not matter, if an operator is next to the robot or on the opposite side of the planet. The system was successfully deployed on two different robotic platforms.

ROS.org
rosbridge.org

Contents

1 Introduction

The aim of the project was to design and create a universal web GUI interface that can be used for teleoperating mobile robots with the system ROS (Robot Operating System) [1]. The goal was to create a lightweight interface that can run independently on the platform and can be easily transferred between different robotic platforms. Two robots were available for evaluation and testing. Both were running the ROS version *Fuerte* with operating system *Ubuntu 12.04* [2].

1.1 NIFTi robot

The NIFTi robot is being developed in the NIFTi project [3]. It involves 6 different European universities, two fire departments and a robot manufacturer. The robot should be suited for outdoor conditions. One of its purposes is to take part in Urban Search and Rescue (USAR) missions, where it should map the situation and help locate victims of a catastrophe. The robot is shown in the Fig. 1.

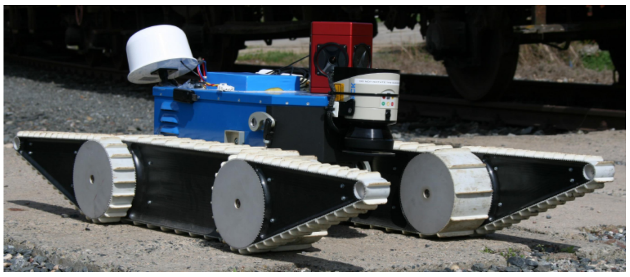


Figure 1: The NIFTi robot.

1.2 Quanti robot

The Quanti robot belongs to the company Quanti [4] whose boss is Ing. Marek Polčák. The robot is intended for indoor use only and it takes part in different robot competitions. Our web GUI will be used during the testing and on the tournaments in the category called freeride. The robot is shown in the Fig. 2.

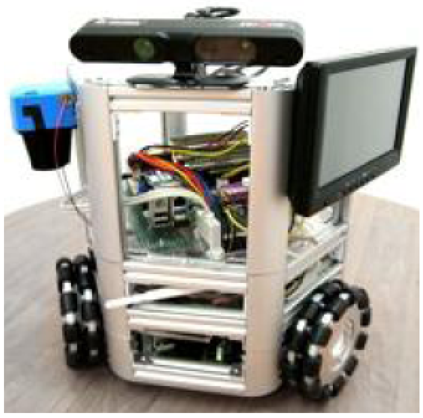


Figure 2: The Quanti robot.

2 Software requirements

Although the system is designed to be platform independent for the user side, it has strict requirements for the robot side. It is not surprising that the robot must have installed ROS. Since the only officially supported OS for the ROS is currently the *Ubuntu*, the other required software has a *Linux* as a preferable platform. Even though *Windows* platform is labeled as experimentally supported on the ROS web site, no one from our team succeed in installing ROS on it.

For correct usage of our interface, the following must be installed on the robot:

- `Rosbridge` suite [5]
- `mjpeg_server` [6]
- `OpenSSH` server [7]
- `Shell In A Box` [8]
- `ros_pub_tf_echo` node (part of our system)

2.1 Rosbridge

The most important part of our software is based on the *rosbridge* version 2.0. It is an applications layer network protocol specification for the ROS emphasising the client/server model. Specifically, *rosbridge* allows clients to publish and subscribe topic messages and invoke services in the server's runtime environment. *Rosbridge* transports JSON-formatted messages over TCP sockets and web-sockets.

The figure 3 shows how the *rosbridge* is integrated with other layers of the system. The current implementation of the *rosbridge* is distributed as a ROS package but it is only a protocol specification and it is probable that there will be more implementations for other middlewares in the future.

In order to communicate with the *rosbridge* JSON messages, there is a javascript library `Ros.js` which provides methods for subscribing and publishing topics and call services. Because it is not much, there is also the package *rosapi* distributed with the *rosbridge* suite which enables commonly used services such as: get topics, get services, get nodes, get parameter, set parameter, etc. Unfortunately, there is currently no service for remotely starting a node or a launch file like `rosrun` or `roslaunch`.

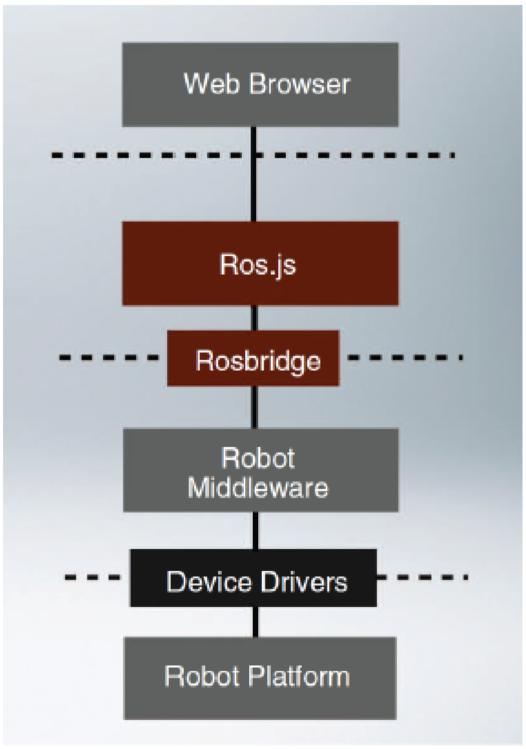


Figure 3: Rosbridge diagram

2.2 Mjpeg server

The *mjpeg_server* is a streaming server that subscribes to requested image topics in ROS and publishes those topics as MJPEG streams via HTTP.

While *rosbridge* is capable of streaming video, as it is just another message type from ROS, the web browser is optimized to efficiently download images in binary format. Thus for increased performance benefits this additional communication channel is used. In order to deal with a users specific requirements the video streams can be provided at a desired quality and size to accommodate different connection speeds and interface designs.

2.3 OpenSSH server

The mobile robot has often no monitor and keyboard connected, when operating. If an operator wants to make some more difficult operation (e.g. copy some file, restart some application, etc.), he or she can connect to the robot remotely using SSH protocol and do what is needed. Commonly used *Linux* tool is the *OpenSSH*, which is distributed as a package with most of distribution.

2.4 Shell In A Box

To enable the user to command a robot using SSH, the terminal window is a part of our web GUI. *Shell In A Box* is a open-source client/server tool that uses *OpenSSH server* on the server side and a javascript library on the client side. The functionality is the same as has any common terminal application.

2.5 `ros_pub_tf_echo` node

Transformation messages are one of the most important messages in ROS. It provides transformations between every coordinates system on the robot. Our GUI uses them in painting the map, where the position of the laser scanner center is needed. Since the `/tf` messages are published 1000 times per second they must be filtered to prevent network overload. This filtration is made by our `ros_pub_tf_echo` node where only transformation between map and laser frame is transferred. The output filtrated frequency can be set using rosparam `/freq_tf` (default value is 10) also as `/source_tf` (default `/map`) and `/target_tf` (default `/laser`).

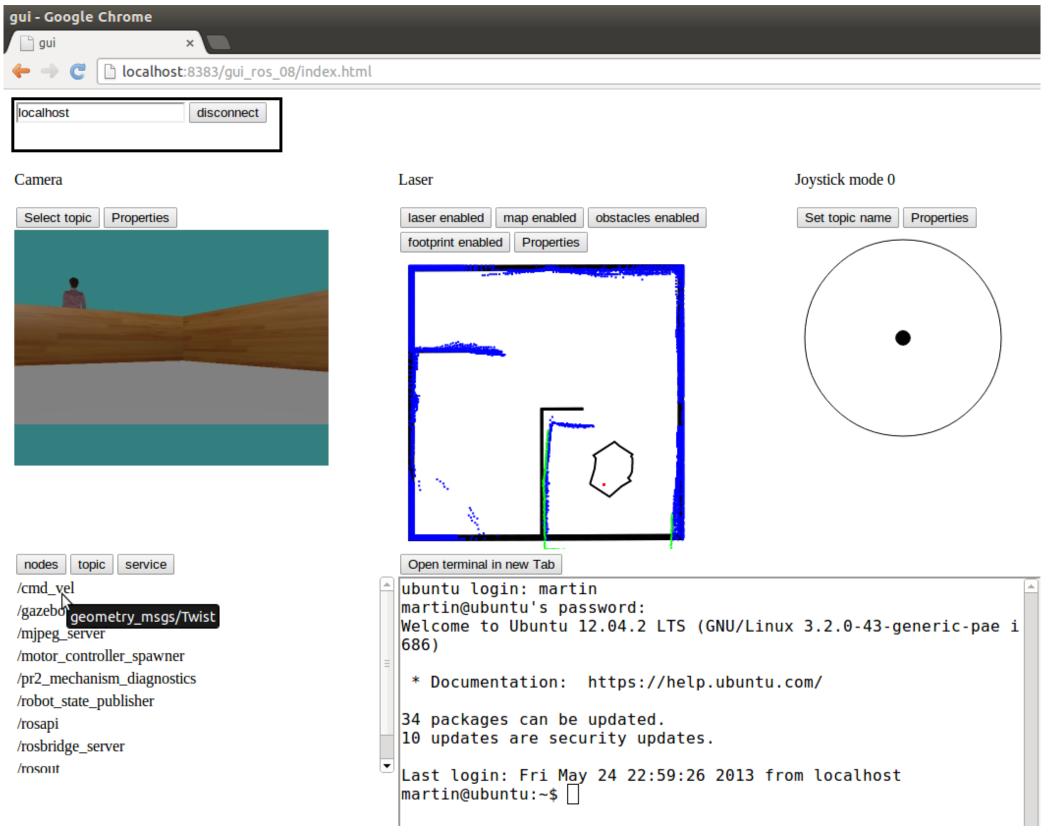
2.6 Webserver requirements

All the above described software must run on the robot. The client only need to connect to a webserver where is our application stored. While only a javascript is used (all the computation is on the client), there are no special needs for the webserver (like PHP, etc.) when using our basin environment. Example can be found here: http://www.rosbridge.felk.cvut.cz/gui.

There is also an option to allow access only for the authorized users. For this feature we made an user management system based on the *WordPress* [9] that has an administration interface for managing the users. This system requires the *MySQL* [10] database installed on the server. Example can be found here: http://www.rosbridge.felk.cvut.cz/wordpress.

3 GUI description

The first step (after optional log-in) is to insert a robot IP address. This can be also an IP address of a machine which uses only a robot *roscore* but has running *rosbridge* and all required software as described above. The GUI has five main parts and is in the Fig. 4.



```
/cmd_vel
/gazebo
/mjpeg_server
/motor_controller_spawner
/pr2_mechanism_diagnostics
/robot_state_publisher
/rosapi
/rosbridge_server
/rosout
```

```
ubuntu login: martin
martin@ubuntu's password:
Welcome to Ubuntu 12.04.2 LTS (GNU/Linux 3.2.0-43-generic-pae i
686)

 * Documentation:  https://help.ubuntu.com/

34 packages can be updated.
10 updates are security updates.

Last login: Fri May 24 22:59:26 2013 from localhost
martin@ubuntu:~$
```

Figure 4: The web GUI with Gazebo simulator

3.1 Camera window

Human gets most of information by his or her eyes thus the camera is most important sensor for an operator. User can choose here any camera topic, that is published by robot and set its image size and quality. Because of the high network load, only one camera image is shown at the webpage.

3.2 Map window

Second most important information is how the robot senses the environment. For this purpose this windows shows the 2D map (message type `nav_msgs/OccupancyGrid` published on the topic `/map`), the laser scan (`sensor_msgs/LaserScan` published on the topic `/scan`), obstacles (`nav_msgs/GridCells` on the topic `/move_base/local_costmap/obstacles`), robot footprint (`geometry_msgs/PolygonStamped` on the topic `/move_base /local_costmap/robot_footprint`) and the laser center position as a red dot (obtained from the topic `/laser_web_tf` produced by the `ros_pub_tf_echo` node). As the robot moves in the environment, the map grows and is automatically downscaled to fit the window.

3.3 Joystick window

For the basic moving with the robot, there is a software emulation of a joystick. It looks like a circle with dot in the middle. The dot is representing the zero position. By mouse clicking and dragging to this circle, the joystick commands are emulated. Before the commanding, the topic for sending `geometry_msgs/Twist` must be selected. Joystick has also two modes. Default mode reads finger position as the speed in the direction of the x-axis and rotation around the z axis. The second mode reads finger position as the velocity in the direction of the x-axis velocity in the y-direction.

3.4 Information window

This window works mainly for debugging. It is often the last part of the GUI that stops working because it has lowest network load. It has three function:

- Show all the nodes currently running on the robot
- Show all the topics currently running on the robot
- Show all the services currently running on the robot

3.5 Terminal window

The most powerful tool of our GUI is the terminal window. It automatically opens a SSH connection to the robot. From here, the user can do almost everything with not only ROS but even with the operating system on the robot computer.

References

[1] "Robot Operating System." http://www.ros.org/wiki/ROS. Accessed: 26/04/2013.

[2] "Ubuntu 12.04 lts." http://www.ubuntu.com/download/desktop. Accessed: 25/05/2013.

[3] "Natural human-robot cooperation in dynamic environments." www.nifti.eu. Accessed: 26/04/2013.

[4] "Quanti: 01 robotics." www.quanti.cz/cz/produkty/tym-01robotics. Accessed: 25/05/2013.

[5] "rosbridge v2.0." http://rosbridge.org/doku.php?id=rosbridge_protocol_v2.0. Accessed: 25/05/2013.

[6] "Mjpeg server." http://www.ros.org/wiki/mjpeg_server. Accessed: 25/05/2013.

[7] "Openssh." http://www.openssh.org/. Accessed: 25/05/2013.

[8] "Shell in a box." http://code.google.com/p/shellinabox/. Accessed: 25/05/2013.

[9] "Wordpress." http://wordpress.org/. Accessed: 25/05/2013.

[10] "Mysql." http://www.mysql.com/. Accessed: 25/05/2013.