Table of Contents

- Sonar
- Lidar
- Localization and navigation
- Motor controller
- Joystick
- Social interaction
- Speech
- Speech recognition

Skylab

- Setup Skylab
- Python scripts
- Webserver
- Functions of the webserver
- Skylab servers
- ROS installation on Ubuntu VMs in Skylab
- DNS,DHCP, pfSense & Ubuntu

Radeffect App

- Radeffect App

Lessons learned

- Todo & Advice
- Lessons Learned

Archive

- Previous Groups
- Research Archive

Welcome

Project Willy

- History of Willy
- Project Willy
- Publicity
- Sponsors

Getting started

- Development Guide
- Driving Willy
- Documentation

Build of Willy

- Design history
- Requirements
- Design reference
- Physical build
- Hardware

Robotic Operating System

- Introduction to ROS
- ROS Tutorials
- Multi master

Architecture

- Software Architecture
- Hardware Architecture
- Skylab Architecture
- ROS topic design

Hardware nodes

- sensor node
- si node
- power node
- WillyWRT

Components

- ROS master
- New ROS master on Lubuntu
- Brain
- Sonar
- Lidar
- Localization and navigation
- Motor controller
- Joystick
- Social interaction
- Speech
- Speech recognition

Skylab

- Setup Skylab

1. ROS Introduction

1.1. An introduction

As a requirement from the product owner, **ROS** is used as framework on Willy. ROS, the Robot Operating System, is a flexible software **framework** for use in robots. It consists of a collection of libraries, tools and conventions that provide basic infrastructure to communicate between different parts of the robot.

In the case of Willy, ROS is especially handy because Willy is made with a modular design. All modules can be removed without disrupting the other functionalities of Willy. For example, when the web interface is removed, Willy is still able to drive, but with another module as for example the keyboard controller. Or the removal of the motor driver makes Willy still able to interact with public.

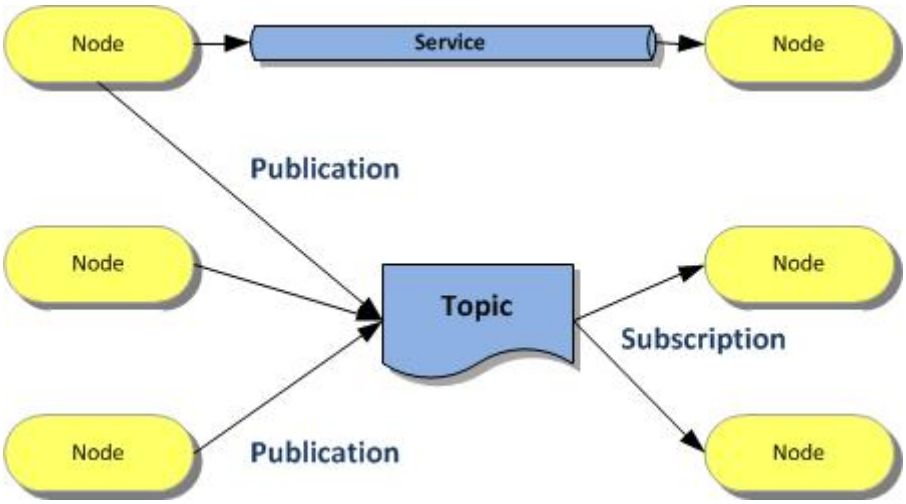
1.2. Nodes

A **node** in ROS can be seen as a module. It is an executable that communicates through ROS to other nodes to send and receive data. A node can be for example a c++ application, or a piece of Python code, or even an Arduino connected with USB running code. A piece of information a node receives or sends is called a **message**.

More information can be found at http://wiki.ros.org/ROS/Tutorials/UnderstandingNodes

1.3. Topics

A topic is a bus over which nodes can exchange data messages. A topic always has a name, so all topics can be identified.



More information can be found at http://wiki.ros.org/Topics

To interact with a topic, two methods are used, subscribing and publishing.

1.3.1. Subscribing

Subscribing is getting data from a topic. Everytime the data in a topic is updated, a message will be passed to all subscribing nodes. This way a node can use this information.

1.3.2. Publishing

Publishing is sending data to a topic. When a node has new information, a message will automatically be sent to the linked topic, so this data is updated.

Welcome

Project Willy

- History of Willy
- Project Willy
- Publicity
- Sponsors

Getting started

- Development Guide
- Driving Willy
- Documentation

Build of Willy

- Design history

- Requirements
- Design reference
- Physical build
- Hardware

Robotic Operating System

- Introduction to ROS
- ROS Tutorials
- Multi master

Architecture

- Software Architecture
- Hardware Architecture
- Skylab Architecture
- ROS topic design

Hardware nodes

- sensor node
- si node
- power node
- WillyWRT

Components

- ROS master
- New ROS master on Lubuntu
- Brain
- Sonar
- Lidar
- Localization and navigation
- Motor controller
- Joystick
- Social interaction
- Speech
- Speech recognition

Skylab

- Setup Skylab
- Python scripts
- Webserver

- Functions of the webserver
- Skylab servers
- ROS installation on Ubuntu VMs in Skylab
- DNS,DHCP, pfSense & Ubuntu

Radeffect App

- Radeffect App

Lessons learned

- Todo & Advice
- Lessons Learned

Archive

- Previous Groups
- Research Archive

2. ROS Tutorials

We found out the hard way, multiple times, that ROS can be quite difficult to comprehend. The principle seems rather easy, have a working network, communicate on TCP-IP and DNS name, and start coding. However, there are a variaty of pitfalls where you can go wrong.

We have used multiple ROS tutorials to understand more of the technical aspects of ROS as well as the (im)possibilities it holds. Due to these tutorials we kept Willy as it was, but also gained knowledge about ROS. Therefore we strongly suggest to read and try some of the tutorials ROS provides on their webpage.

http://wiki.ros.org/ROS/Tutorials

Welcome

Project Willy

- History of Willy
- Project Willy
- Publicity
- Sponsors

Getting started

- Development Guide
- Driving Willy
- Documentation

Build of Willy

- Design history

- Requirements
- Design reference
- Physical build
- Hardware

Robotic Operating System

- Introduction to ROS
- ROS Tutorials
- Multi master

Architecture

- Software Architecture
- Hardware Architecture
- Skylab Architecture
- ROS topic design

Hardware nodes

- sensor node
- si node
- power node
- WillyWRT

Components

- ROS master
- New ROS master on Lubuntu
- Brain
- Sonar
- Lidar
- Localization and navigation
- Motor controller
- Joystick
- Social interaction
- Speech
- Speech recognition

Skylab

- Setup Skylab
- Python scripts
- Webserver

- Functions of the webserver
- Skylab servers
- ROS installation on Ubuntu VMs in Skylab
- DNS,DHCP, pfSense & Ubuntu

Radeffect App

- Radeffect App

Lessons learned

- Todo & Advice
- Lessons Learned

Archive

- Previous Groups
- Research Archive

3. Multi master setup on Willy

The Willy environment is built on two locations (Willy and Skylab) which are coupled with an Wifi connection. By using an multi master setup there are two advantages:

- Through the setup it can be decided which topics will be transferred between the two environments, thereby potentially saving bandwidth (configured on master_sync node)
- Skylab will have a dedicated master even when Willy is not operational

To have the sync module always up-and-running, it seems that this module will be running on the master in Skylab. It has to be tested if the master on Willy will be automatically be recognized when it is started or some actions has to be taken. Seems to be a task of the master_discovery node.

It has to be tested in the Willy environment if this setup works. Actions are:

- ☑ Make snapshot of test master in Skylab
- ☑ Install multimaster on the (now) test master in Skylab
- ☑ Install test master with multimaster on a RPi
- ☑ Confirm that it works
- ☑ Evaluate the stop and restart of master on RPI and document possible restart actions

Go/no go decision

Go:

- ☐ Install multimaster on production RPi on Willy
- ☐ Depreciate test master on RPi
- ☐ Document and make instructions

No go:

- [x] Revert snapshot on master in Skylab
- [x] Make decision on further action

Decision is that the nodes in Skylab will be using the ROS master of Willy.

3.1. Installation and using multimaster

Install multimaster on every master Kinetic node with:

- sudo apt-get install ros-kinetic-multimaster-fkie
- Check DNS, localhost and host name
- Enable multicast feature on both hosts and for all ROS nodes:
  - edit the /etc/sysctl.conf file and add the following line:
    - net.ipv4.icmp_echo_ignore_broadcasts=0

Start multi master:

- first master:
  - export ROS_MASTER_URI=http://brainnode:11311
  - roscore >/dev/null 2>&1 &
  - rosrun master_discovery_fkie master_discovery >/dev/null 2>&1 &
  - rosrun master_sync_fkie master_sync >/dev/null 2>&1 &
- second master:
  - export ROS_MASTER_URI=http://u1634:11312
  - roscore --port 11312 >/dev/null 2>&1 &
  - rosrun master_discovery_fkie master_discovery >/dev/null 2>&1 &
  - rosrun master_sync_fkie master_sync >/dev/null 2>&1 &

3.2. For info:

- start node manager with: node_manager
- rosservice call /master_discovery/list_masters

Take note: there must exist a binding between the IP address and the host name for all the computers in all the networks in order for the ROS framework to work properly.

References:

http://wiki.ros.org/multimaster_fkie/Tutorials/Setup%20a%20ROS%20master%20synchronization