Table of Contents

1. ROS Introduction 3

1.1. An introduction 3

1.2. Nodes 4

1.3. Topics 4

1.3.1. Subscribing 4

1.3.2. Publishing 5

2. ROS navigation 6

2.1. Navigation stack 6

2.2. Path calculation 7

2.3. Autonomous driving 7

Welcome

Project Willy

- Willy
- Publicity
- Sponsors

Startup Willy

- Driving Willy
- Remote
- Willy Web

Configuration

- GIT Setup
- Ubuntu
- Remote
- Wiki

ROS

- Introduction to ROS
- Navigation

Technical

- Development Guide
- Findings
- Hardware
- Known Bugs
- Parameters
- Software

Web interface

- Development Guide
- SAD
- RosNodeJs
- Interaction

Research

- Hardware
- Peripherals
- Sensors
- Social interaction

- Software
- Web interface

Design

- Background
- Design Guide
- Technical
- Realisation

Status and Advice

- Status
- Todo & Advice

Archive

- (2016/2) Initial design
- (2017/1) Base & Functionalities
- (2017/2) Research

Welcome

Project Willy

- Willy
- Publicity
- Sponsors

Startup Willy

- Driving Willy
- Remote
- Willy Web

Configuration

- GIT Setup
- Ubuntu
- Remote
- Wiki

ROS

- Introduction to ROS
- Navigation

Technical

- Development Guide
- Findings

- Hardware
- Known Bugs
- Parameters
- Software

Web interface

- Development Guide
- SAD
- RosNodeJs
- Interaction

Research

- Hardware
- Peripherals
- Sensors
- Social interaction
- Software
- Web interface

Design

- Background
- Design Guide
- Technical
- Realisation

Status and Advice

- Status
- Todo & Advice

Archive

- (2016/2) Initial design
- (2017/1) Base & Functionalities
- (2017/2) Research

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.

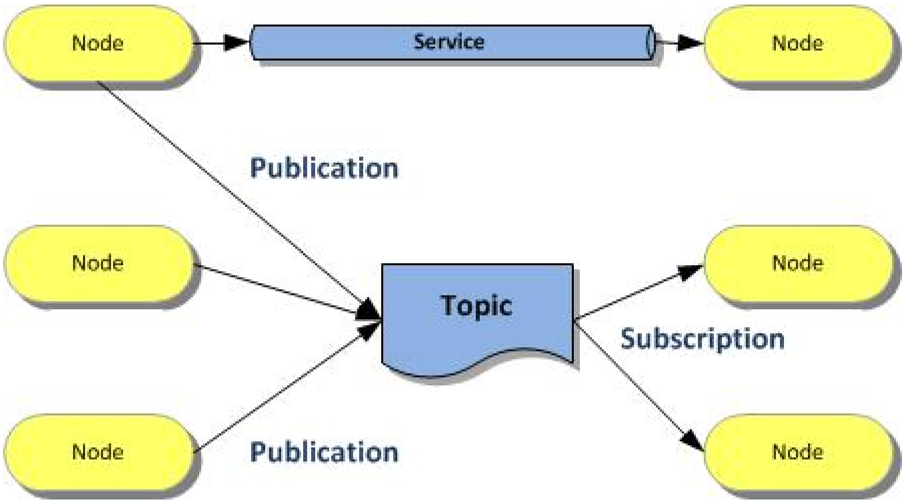
1.2. Nodes

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

- Willy
- Publicity
- Sponsors

Startup Willy

- Driving Willy
- Remote
- Willy Web

Configuration

- GIT Setup
- Ubuntu
- Remote
- Wiki

ROS

- Introduction to ROS
- Navigation

Technical

- Development Guide
- Findings
- Hardware
- Known Bugs
- Parameters
- Software

Web interface

- Development Guide
- SAD
- RosNodeJs
- Interaction

Research

- Hardware

- Peripherals
- Sensors
- Social interaction
- Software
- Web interface

Design

- Background
- Design Guide
- Technical
- Realisation

Status and Advice

- Status
- Todo & Advice

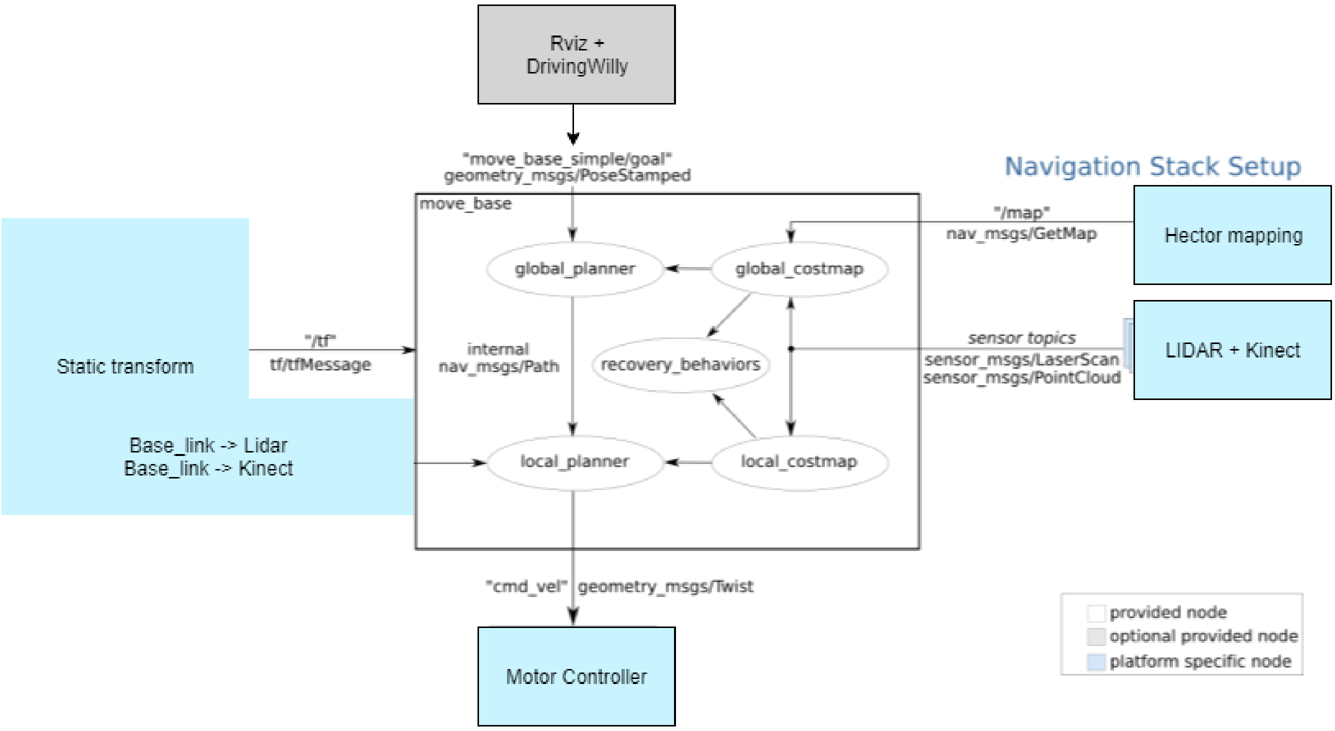
Archive

- (2016/2) Initial design
- (2017/1) Base & Functionalities
- (2017/2) Research

2. ROS navigation

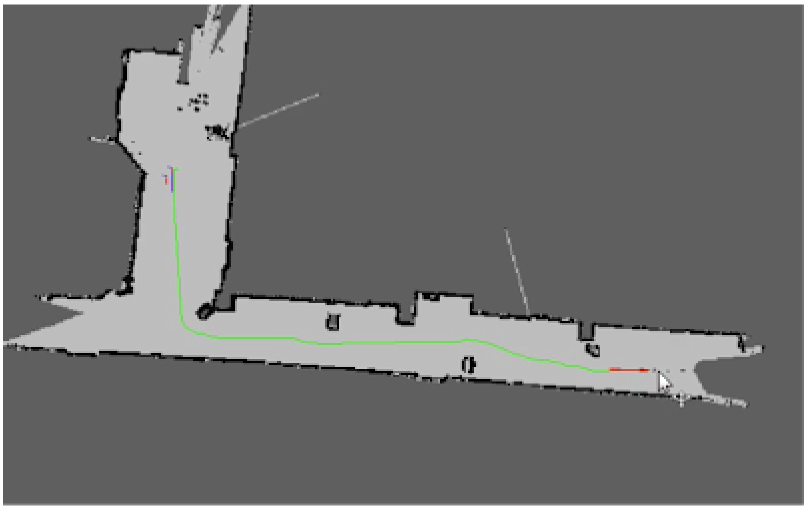
2.1. Navigation stack

Willy uses the ROS navigation stack to navigate indoor. The ROS navigation stack is a complex collection of multiple ROS packages. These packages read the sensor data we provided from our sensors to create a map of the environment. When the user provides a goal from rviz, the global planner will provide a path to that goal and calculate the costs in that map. costs can be compared with obstacles. The bigger the obstacle, the higher the cost.



2.2. Path calculation

In the image above you can see the overview of the ROS navigation stack in our project. We use the move_base from ROS. We send goals from rviz or C++ to the move_base. The move_base converts the goal into a path and calculates a route. This proces is shown below.



2.3. Autonomous driving

When the path and costs are calculated, the move_base will send geometry::Twist messages to the mobile base. In our case this is the motorcontroller with the cmd_vel topic. When the motorcontroller receives a geometry::Twist message, the motor will go driving on the given X, Y and Z velocity from the Twist message.

The ROS navigation stack has a lot of parameters. These parameters change the way the path is calculated and the maximum speed etc. For more information about these parameters see:

https://artofrobotics.github.io/WillyWiki/Willy/Parameters.html