

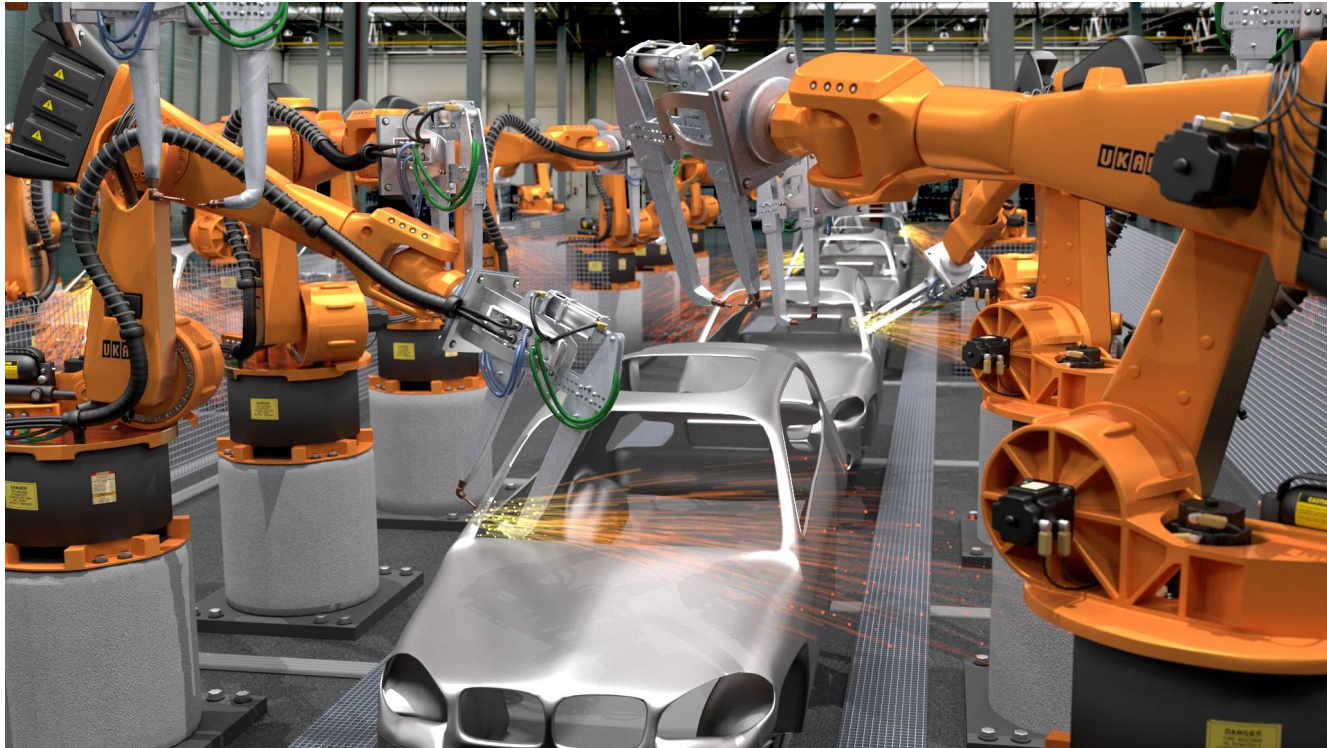
Introduction to Service Robotics Application development with ROS

ludus.russo@gmail.com

<http://www.hotblackrobotics.com/>

<http://www.ludusrusso.cc/>

Robotics Fields - Industrial Robotics



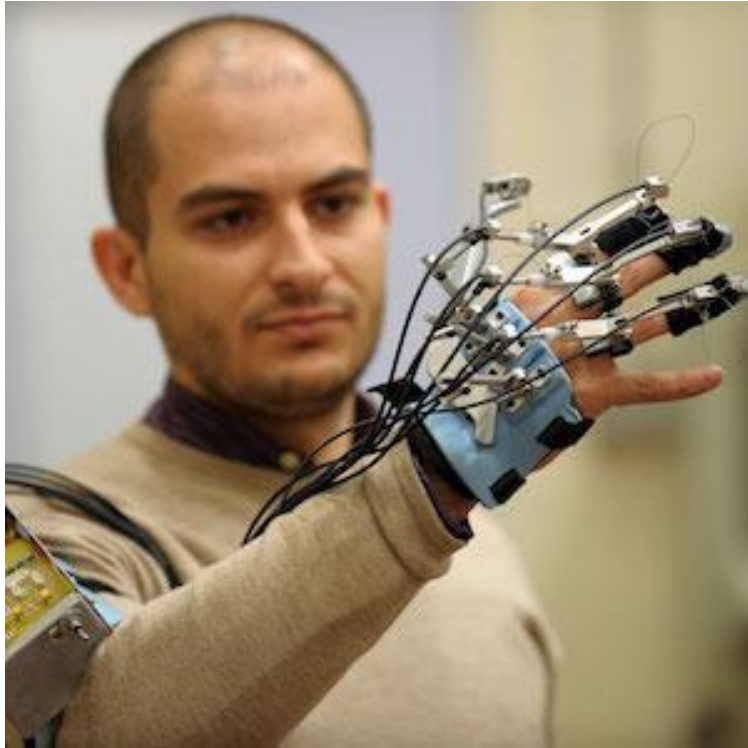
Robotics Fields - Space Exploration



Robotics Fields - Military Robots



Robotics Fields - Wearable Robotics



Robotics Fields - Service Robotics





HOTBLACK ROBOTICS

Cloud Robotics e IoT

Team / HotBlack Robotics was born during the PhD in TIM JOL CRAB (Connected Robotics Application Lab) and Politecnico di Torino



Gabriele Ermacora
CEO

Ph.D. in Robotics
Engineering,
focused on Human
Robot Interaction
and Cloud
Services.



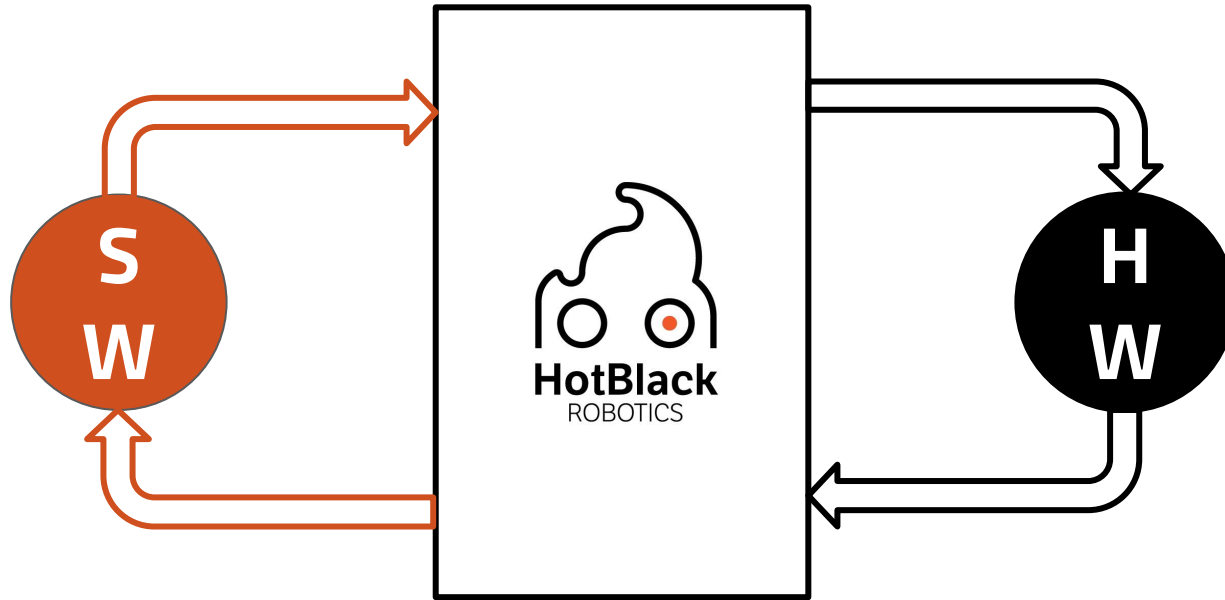
Ludovico O. Russo
CTO

Ph.D. in Robotics
Engineering,
expert in Service
and Cloud Robotics

A group of approximately 15 people, including men and women of various ages, are posed for a group photo in what appears to be a workshop or laboratory. In the foreground, a large, orange industrial robotic arm is visible. The background shows shelves with various items and a whiteboard with some diagrams. The overall lighting is somewhat dim, and the image has a slightly dark, muted color palette.

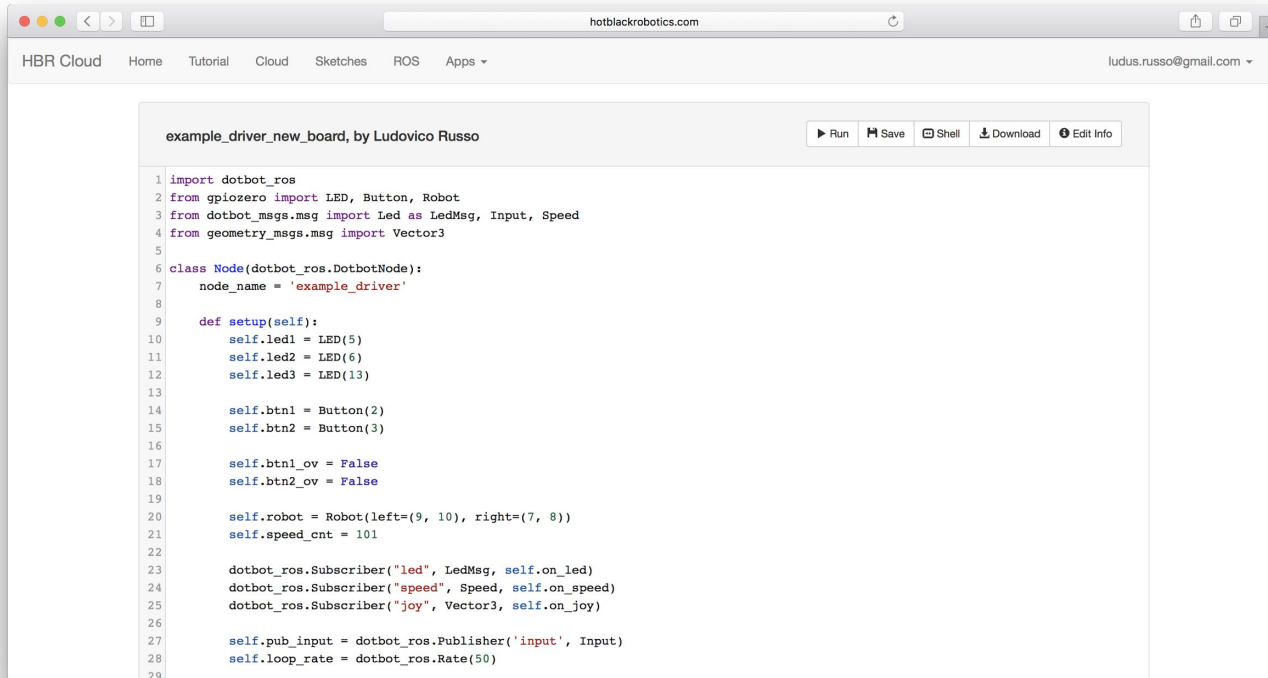
MISSION: Democratize robotics through a cloud platform.

The cloud platform and the community



Sharing of robotics Open Hardware projects and robotic Open Source (Robotic App Store)

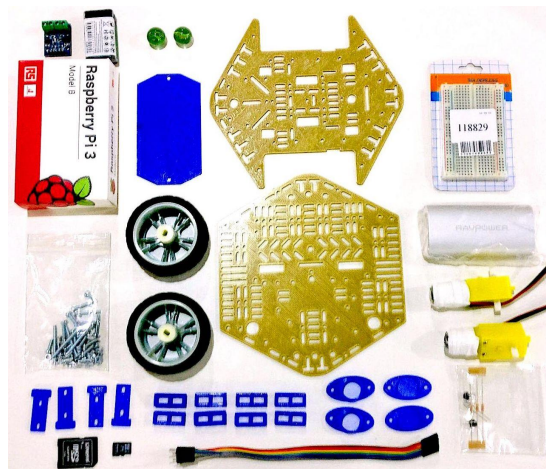
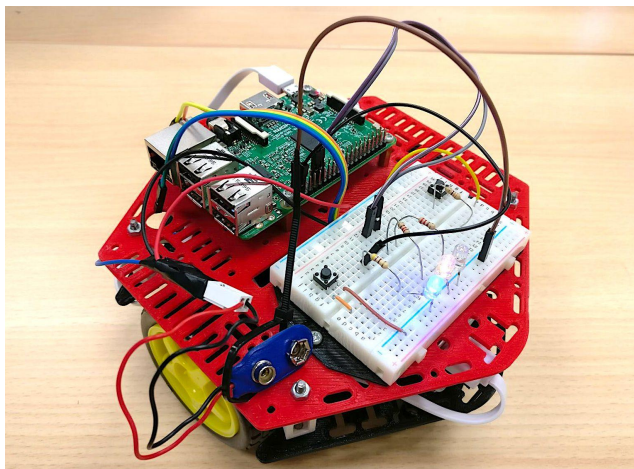
Write a robotic app is easy via cloud



```
example_driver_new_board, by Ludovico Russo
▶ Run ▶ Save ▶ Shell ▶ Download ▶ Edit Info

1 import dotbot_ros
2 from gpiozero import LED, Button, Robot
3 from dotbot_msgs.msg import Led as LedMsg, Input, Speed
4 from geometry_msgs.msg import Vector3
5
6 class Node(dotbot_ros.DotbotNode):
7     node_name = 'example_driver'
8
9     def setup(self):
10        self.led1 = LED(5)
11        self.led2 = LED(6)
12        self.led3 = LED(13)
13
14        self.btn1 = Button(2)
15        self.btn2 = Button(3)
16
17        self.btn1_ov = False
18        self.btn2_ov = False
19
20        self.robot = Robot(left=(9, 10), right=(7, 8))
21        self.speed_cnt = 101
22
23        dotbot_ros.Subscriber("led", LedMsg, self.on_led)
24        dotbot_ros.Subscriber("speed", Speed, self.on_speed)
25        dotbot_ros.Subscriber("joy", Vector3, self.on_joy)
26
27        self.pub_input = dotbot_ros.Publisher('input', Input)
28        self.loop_rate = dotbot_ros.Rate(50)
29
```

Dotbot - the first open hardware project programmable by cloud



Cloud robotics will be the enabler for service robotics



e.g Google Car and cloud robotics

[Google Autonomous car] You see, a driverless vehicle needs to be able to make a decision in a split second. In order to make that decision, it needs to scan and analyze everything in its local environment. While it is certainly possible to create a computer — housed inside the vehicle — that can do all these complex calculations, store all the necessary data and handle the analytics side of it, **that's not what you would call efficient.**

Instead, these vehicles tap into a remote cloud computing network to access boatloads of stored data. This data includes roadmaps, instructions and strategies, visual indicators or identifiers and much more.

Introduction to ROS

What is ROS? Robot Operating System

- **Framework for Robot Application Development**
 - OS-like functionalities (meta-operating system)
 - Hardware Abstraction / Drivers
 - Multiprocess Communication
 - Package Management
- **Collection of Libraries, Tools and Conventions**
 - Stacks
 - Packages
- **Ecosystem and Community**



Why ROS?

- Because creating truly robust, general-purpose robot software is **hard**.
 - Provides plug-and-play solutions to common problems in robotics
- ROS encourages **collaborative Software Development**



ROS Distribution

- Versioned set of ROS packages
 - Like a Linux distributions
 - Allows developers to work on a stable codebase
- Scheduling
 - There is a ROS release **every year in May**.
 - Releases on even numbered years will be a LTS release, supported for five years.
 - Releases on odd numbered years are normal ROS releases, supported for two years.

ROS release history

End-of-life

7 distribution

C Turtle (2010) -> Hydro (2013)



Supported



Indigo Igloo
2014 - 2019



~~Jade Turtle
2015 - 2017~~











































Kinetic Kame
2016 - 2021



Lunar Loggerhead
2017 - 2019

Hardware

 210 Stanley Innovation V3 Segway	 220 Stanley Innovation V3 Segway	 223 Innok Heros	 224 Innok Heros	 Clearpath Robotics Grizzly	 Clearpath Robotics Husky	 Clearpath Robotics Jackal	 Clearpath Robotics Kingfisher
 420 Omni Stanley Innovation V3 Segway	 440LE Stanley Innovation V3 Segway	 440SE Stanley Innovation V3 Segway	 444 Innok Heros	 Clearpath Robotics Ridgeback	 Cogniteam Hamster	 CoroWare Corobot	 Cyton-Gamma
 ABB Robotics (ROS-Industrial)	 Adept MobileRobots Pioneer family (P3DX, P3AT, ...)	 Adept MobileRobots Pioneer LX	 Adept MobileRobots Seekur family (Seekur, Seekur Jr.)	 Dataspeed ADAS Development Vehicle	 Dataspeed Mobility Base	 Denso VS060	 Dr. Robot Jaguar
 Aldebaran Nao	 Allegro Hand SimLab	 AMIGO	 AscTec Quadrotor	 Eddiebot	 Enova Robotics MiniLab	 Erie-Brain	 Erie-Brain 2
 Barrett Hand	 BipedRobin	 Bitcraze Crazyflie	 Blue Robotics BlueROV	 Erie-Copter	 Erie-Copter Ubuntu Core special edition	 Erie-HexaCopter	 Erie-Plane

The ROS Framework

- Lives on an Existing OS
- Supported
 - **Ubuntu (ARM)**
 - **Debian**
- Experimental
 - Mac OS X
 - Others Ubuntu Distro (Gentoo)
- Binded for many languages
- Compiled in
 - **C++**
 - **Python**
- Wrappers
 - JavaScript
 - Java
 - Matlab
 - ...

The ROS Framework - Communication

- **Multi Machines communication**
- **ROS Node**
 - A process in a ROS application
- **Communication Patterns**
 - Publish/Subscribe
 - Service/Client
 - Actions

ROS Organization

- Code
 - Node
 - Library
 - Messages
- Package
 - Collection of Nodes/Libraries/Messages
 - Other related data
- Metapackage
 - Collection of thematically similar Packages.
 - Bundle together code that is developed together and is mutually interdependent.
- Repositories

navigation

electric fuerte groovy hydro indigo jade **kinetic** Documentation Status

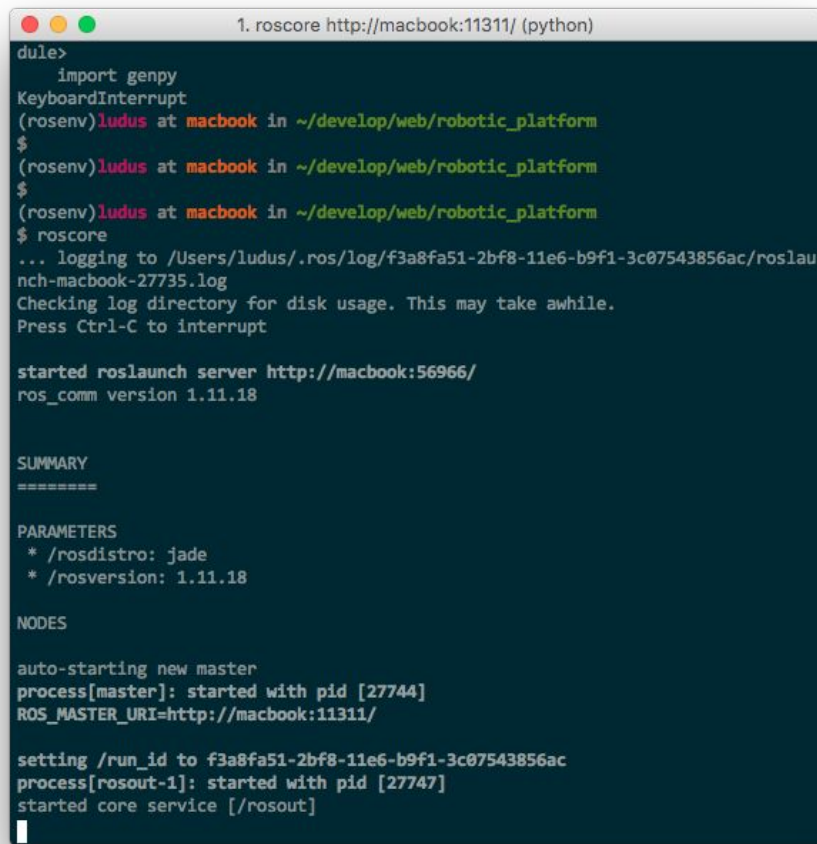
navigation: amcl | base_local_planner | carrot_planner | clear_costmap_recovery | costmap_2d | dwa_local_planner | fake_localization | global_planner | map_server | move_base | move_base_msgs | move_slow_and_clear | nav_core | navfn | robot_pose_ekf | rotate_recovery | voxel_grid

ROS Tools

- Analysis, Debug and Visualization
 - Command Line Tools
 - RVIZ
 - RQT
 - ROS Bag
- Simulation
 - Gazebo

ROS Tools - Command Line Tools

- Main tools to manage ROS
- Allows to visualize and interact with a ROS environment from the command line
 - catkin - Compilation suite
 - roscore
 - rosrn/roslaunch
 - rostopic/rosnode
 - rosmmsg/rossrv
 - rosmmessage/rosservice
 - etc.



```
1. roscore http://macbook:11311/ (python)
dule>
  import genpy
KeyboardInterrupt
(rosenv)ludus at macbook in ~/develop/web/robotic_platform
$
(rosenv)ludus at macbook in ~/develop/web/robotic_platform
$
(rosenv)ludus at macbook in ~/develop/web/robotic_platform
$ roscore
.. logging to /Users/ludus/.ros/log/f3a8fa51-2bf8-11e6-b9f1-3c07543856ac/roslau
nch-macbook-27735.log
Checking log directory for disk usage. This may take awhile.
Press Ctrl-C to interrupt

started roslaunch server http://macbook:56966/
ros_comm version 1.11.18

SUMMARY
=====

PARAMETERS
* /rostdistro: jade
* /rosversion: 1.11.18

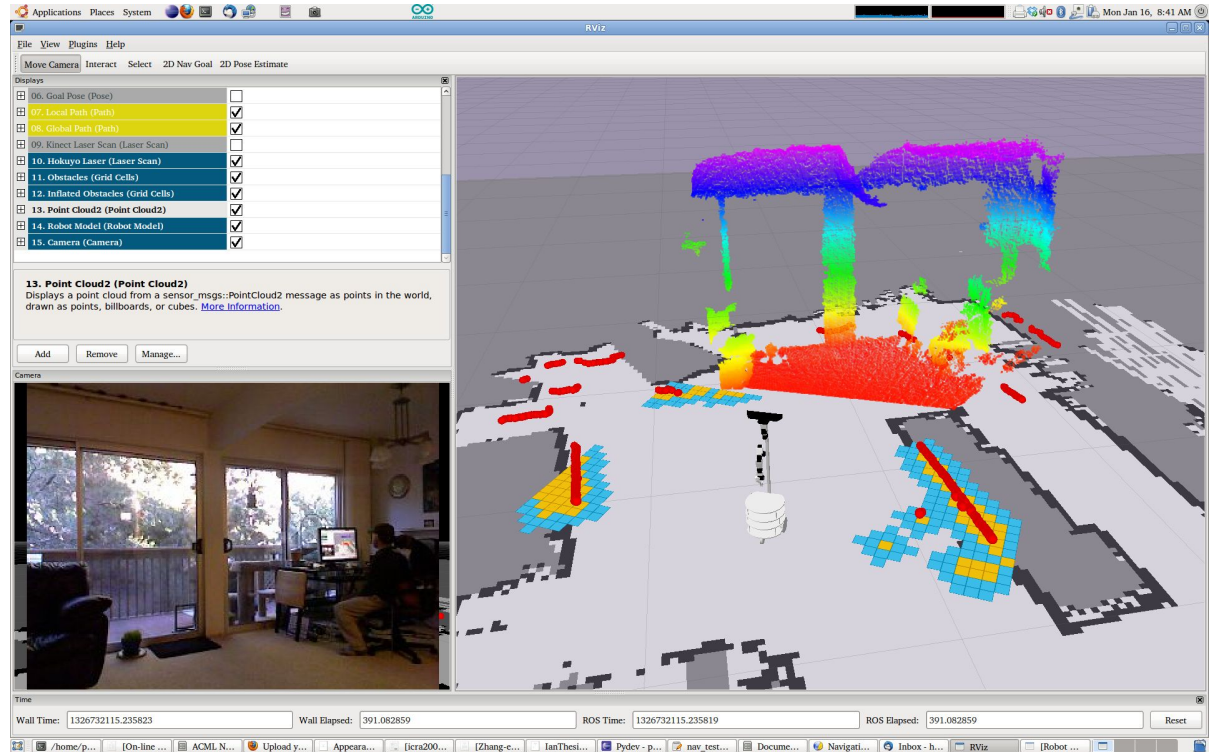
NODES

auto-starting new master
process[master]: started with pid [27744]
ROS_MASTER_URI=http://macbook:11311/

setting /run_id to f3a8fa51-2bf8-11e6-b9f1-3c07543856ac
process[rosout-1]: started with pid [27747]
started core service [/rosout]
```

ROS Tools - RVIZ

3D visualizer for the Robot Operating System (ROS) framework



ROS Tools - RQT

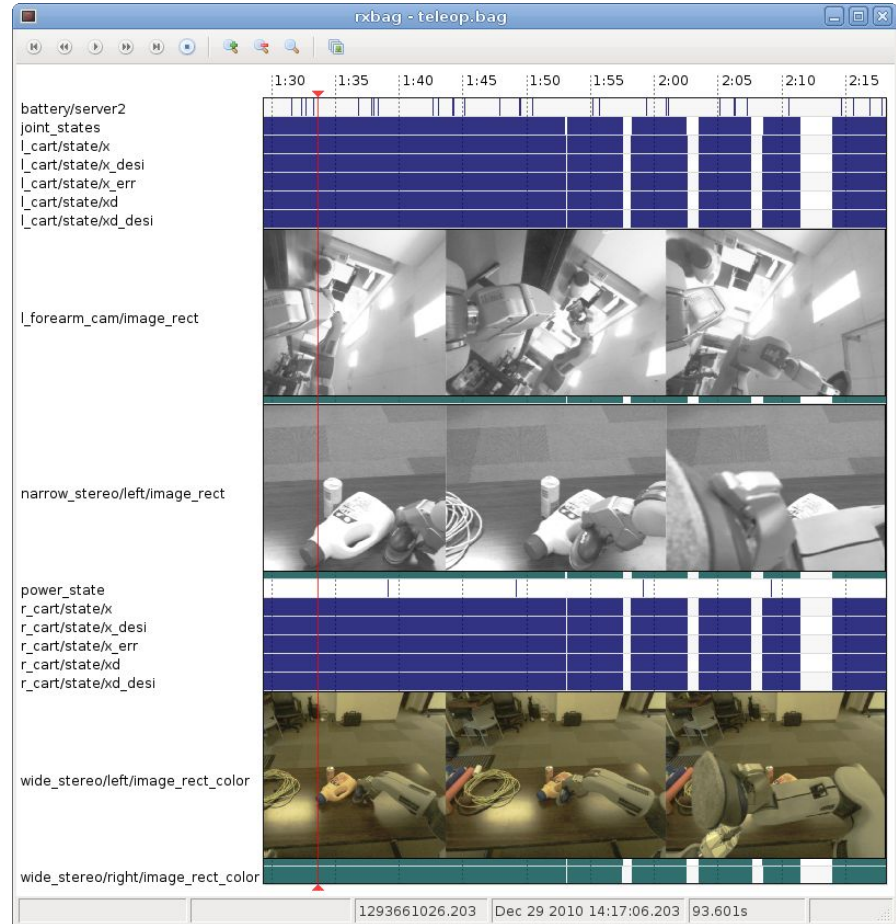
GUI tool for data visualization for the Robot Operating System (ROS) framework

The screenshot displays the RQT (Robot Query Tool) interface. At the top left, a web browser shows the ROS.org website. The main window is divided into several panels:

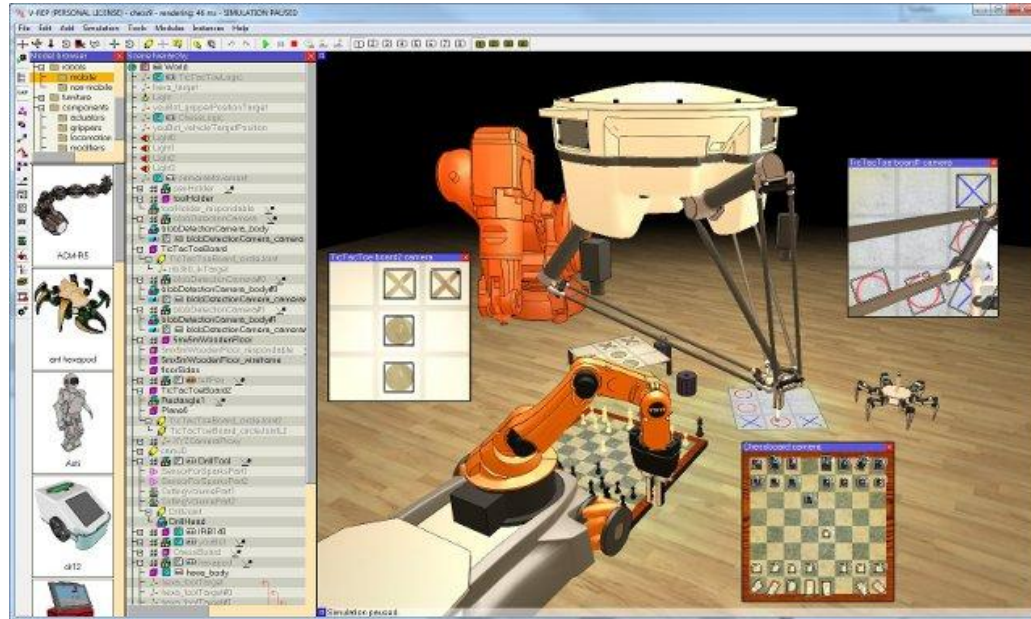
- Topic Publisher:** A table showing active topics. The selected topic is `/cmd_vel3`, which is a `std_msgs/Float32` type published at 5.00 Hz. The expression for the data is `sin(f/20)*10`.
- Robot Steering:** A slider control for the `/cmd_vel` topic, currently set to 1.00 Hz. It includes a 'Stop' button and a 'Refresh' button.
- Loggers:** A panel showing the status of various ROS nodes, including `/rosout`, `/qt_gui_cpp`, and `/rviz_134392`.
- Console:** A message log displaying 9 messages from the `/moveit_setup_assistant` node, including 'Loading Setup Assistant Complete', 'Listening to 'moveit_planning_scene'', 'Starting scene monitor', 'Configuring kinematics solvers', 'Robot semantic model successfully loaded', and 'Setting Param Server with Robot Seman...'. It includes filters for severity and message content.
- Plot:** A graph showing the velocity data for `/cmd_vel3/data` (red line) and `/cmd_vel2/data` (blue line) over time. The x-axis represents time from 0 to 1000, and the y-axis represents velocity from -29 to 29.

ROS Tools - ROSBag

Set of tools for recording from and playback ROS stream information

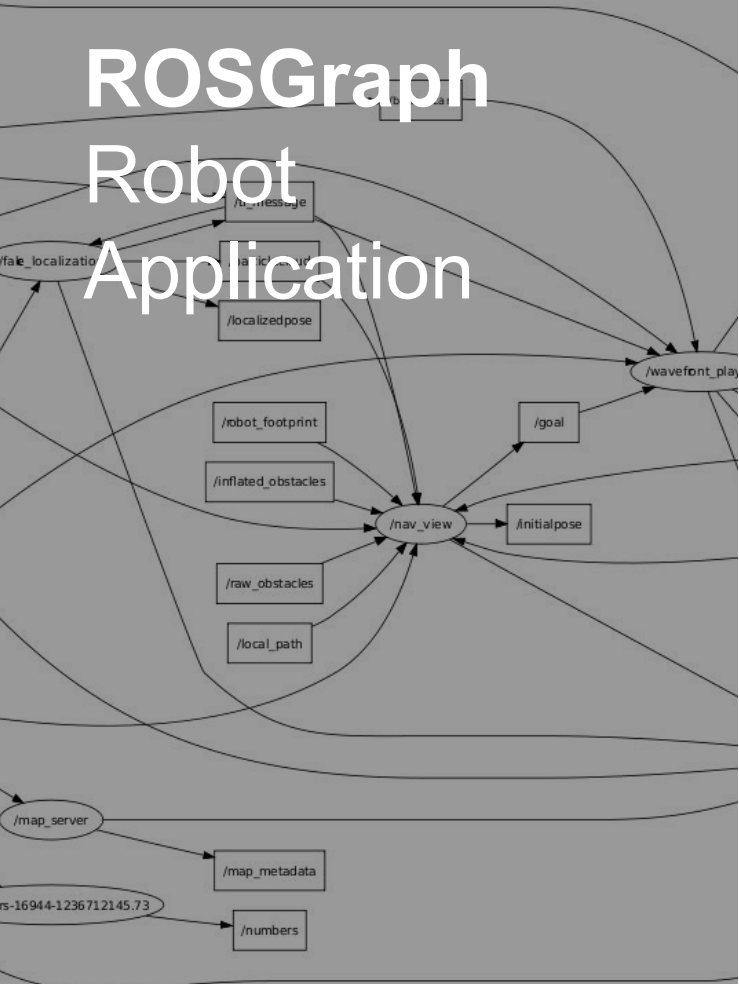


ROS Tools - Simulators



How does ROS work?

ROSGraph Robot Application



An application developed in ROS is a network of processes able to exchange information in a multi-machine communication network:

- The ROS Graph

ROS main components

ROSNODE

- A process in a ROS Application
- Actual Code when running
- A node belongs to a package

ROS Master

- A special node that initialize a ROS application and enables communication

ROSMESSAGE

- Data definition of information that nodes exchange to communicate
- A message belongs to a package

ROSTOPIC

- Main ROS communication
- A channel through which nodes exchange async information

ROS Nodes

A node is a process (a program that is executed) inside a ROS network, that perform a specific task

- A node is identified by an unique string (name) in the network.
- Nodes interact each other by exchanging messages using topics, services and actions
- A node can
 - control hardware (drivers)
 - Perform computations

ROS

Topics

Topics are the main communication channels in a ROS network:

- A topic is identified by a unique string (name) in the network
- On each topic, data can be sent in a specific format (Message Type)
- A node that sends data to a topic is called **publisher**
- A node that receives data from a topic is called **subscriber**

ROS Messages and Message Types

Topics exchange data encapsulated in messages.

- **Message Types** describes the format of a message.
- Each topic has a specific type.
- Can be simple (**bool, int, float, string**)
- Can be complex struct
 - geometry_msgs/Point

File: `geometry_msgs/Point.msg`

Raw Message Definition

```
# This contains the position of a point in free space
float64 x
float64 y
float64 z
```

ROS

Services

Services are communication channels in a ROS network:

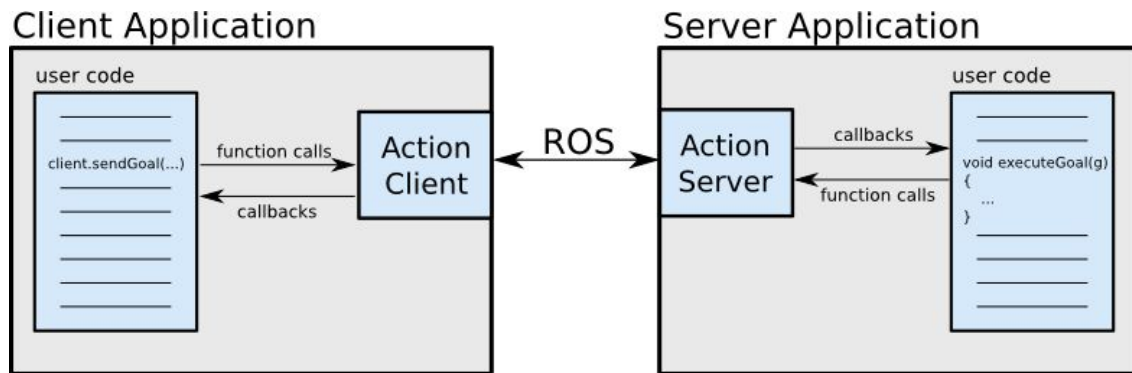
- A node that send a message on a service is called as **Client**
- A node that receive a message on a service is called as **Server**
- They works like topics, but a service call is blocking
 - Client waits a response from a server

ROS

Actions

Actions are a complex communication system designed for long executing tasks:

- Works like services but with a non blocking system:
 - feedback topic
 - Result topic



Starting a Node Master

- **roscore** is the command to start a ROS master
- The ROS master must be started before any other node
- Usage

```
$ roscore
```

Bring up ros nodes - rosrun

- **rosrun** is the command to run a specific node
- The ROS master must be started before any other node
- Usage: `rosrun <package_name> <node_name> <OPTIONS>`

```
$ rosrun turtlesim turtlesim_node
```

```
$ rosrun turtlesim turtle_teleop_key
```

Bring up ros nodes - roslaunch

- rosrund requires a shell per node to be execute
 - In large project, rosrund is too complex to be used
- Roslaunch is a solution to launch several nodes at the same time
- Usage: roslaunch <package_name> <launchfile.launch>

```
<launch>
  <param name="somestring1" value="bar" />
  <!-- force to string instead of integer -->
  <param name="somestring2" value="10" type="str" />

  <param name="someinteger1" value="1" type="int" />
  <param name="someinteger2" value="2" />

  <param name="somefloat1" value="3.14159" type="double" />
  <param name="somefloat2" value="3.0" />

  <!-- you can set parameters in child namespaces -->
  <param name="wg/childparam" value="a child namespace parameter" />

  <!-- upload the contents of a file to the server -->
  <param name="configfile" textfile="$(find roslaunch)/example.xml" />
  <!-- upload the contents of a file as base64 binary to the server -->
  <param name="binaryfile" binfile="$(find roslaunch)/example.xml" />

</launch>
```


Topics

- A **Topic** is a communication channel in ROS
- Each node can **listen** or **publish** on a topic
- Nodes send **messages** over a topic
- **Rostopic** is a ros command to interact using the terminal with topics
 - rostopic list
 - rostopic echo <topic_name>
 - rostopic pub <topic_name> <message_type> <message>

```
$ rostopic echo /turtle1/cmd_vel
```

```
$ rosrn pub ....
```

Message Type

- A **Message Type** represents a data type sent over a topic
- Each topics transfer specific message type information
- **rosmmsg** allows to get informations about a message type in ROS

```
$ rosmmsg show geometry_msgs/Twist
```

```
geometry_msgs/Vector3 linear
```

```
float64 x
```

```
float64 y
```

```
float64 z
```

```
geometry_msgs/Vector3 angular
```

```
float64 x
```

```
float64 y
```

```
float64 z
```

Message Type

ROS defines standard messages to handle common type of communication

- **std_msgs**
 - Int8 - Int16 ..
 - Float8 - Float16
 - String
 - Empty
 - Time
 - Bool
 -
- Arrays
 - Int8MultiArray
 -
- **geometry_msgs**
 - Accel
 - Pose
 - Quaternion
 - Twist
 - ...
- Different version of the same message that can be **stamped** and **WithCovariance**
 - PoseStamped
 - PoseWithCovarianceStamped

ROS Service/Client

- Synchronous communication between nodes
- A **Service** is a couple of messages
 - Request (a message)
 - Response (a message)
- The Client sends a **Request** and waits for a **Response** from the **Server**
- `rosservice` is the command line tool to hand Services

```
$ rosservice list  
$ rosservice call /kill turtle1  
$ rosservice call /reset
```

ROS Service Type

- Like message Type, a Service Type are the data type sent over a Service/Client communication
- **rossrv** allows to get informations about a service type in ROS

```
$ rossrv show turtlesim/Spawn  
float32 x  
float32 y  
float32 theta  
string name  
---  
string name
```

ROS Param Server

- Store and manipulate data accessible by all nodes
- ROS Parameter Server
- rosparam is the command line tool to access to the rosparam server

```
$ rosparam list
/background_b
/background_g
/background_r
/rosdistro
/roslaunch/uris/host_imacdiludovico_home
net_telecomitalia_it__50532
/rosversion
/run_id

$rosparam set background_g 160
```

Example



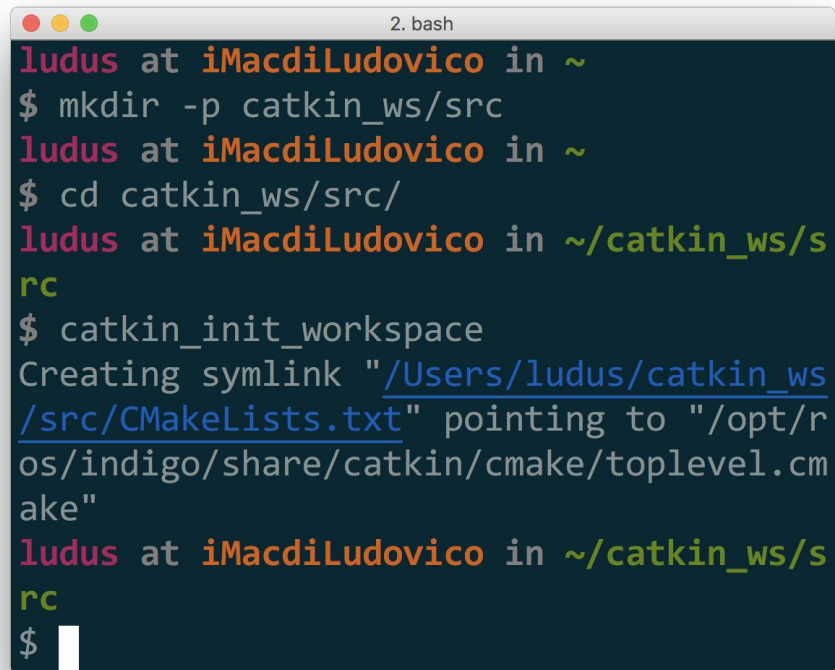
```
1. roscore http://iMacdiLudovico.homenet.telecomitalia.it:11311/ (turtle_teleop_ke)
$
ludus at iMacdiLudovico in ~
$
ludus at iMacdiLudovico in ~
$
ludus at iMacdiLudovico in ~
$
ludus at iMacdiLudovico in ~
$ rosrn turtlesim
draw_square      mimic          turtle_teleop_key  turtlesim_node
ludus at iMacdiLudovico in ~
$ rosrn turtlesim turtlesim_node
^Z
[2]+  Stopped                  rosrn turtlesim turtlesim_node
ludus at iMacdiLudovico in ~
$ bg
[2]+ rosrn turtlesim turtlesim_node &
ludus at iMacdiLudovico in ~
$
ludus at iMacdiLudovico in ~
$
ludus at iMacdiLudovico in ~
$ ros[ INFO] [1465244667.273363000]: Starting turtlesim with node name /turtlesim
m
[ INFO] [1465244667.283853000]: Spawning turtle [turtle1] at x=[5,544445], y=[5,544445], theta=[0,000000]
run turtlesim turtle_teleop_key
Reading from keyboard
-----
Use arrow keys to move the turtle.
```

ROS Developing - Catkin

- **Catkin** is the build system for ROS
- Catkin allows to
 - Organize packages
 - Build packages
- Simplify the development of reusable code
- Developed over Standard Build tools
 - CMake
 - Python-build
- Allows developer to
 - Create workspace
 - Build Nodes
 - Generate Messages and Services types

Catkin Workspace

- A Catkin workspace is a workspace project where code and executables are placed
 - catkin_ws/
 - src/ (all packages code)
 - devel/
 - install/
- To generate a catkin workspace, you need
 - Create the folder catkin_ws/src/
 - Run the command catkin_init_workspace within catkin_ws/src



```
2. bash
ludus at iMacdiLudovico in ~
$ mkdir -p catkin_ws/src
ludus at iMacdiLudovico in ~
$ cd catkin_ws/src/
ludus at iMacdiLudovico in ~/catkin_ws/s
rc
$ catkin_init_workspace
Creating symlink "/Users/ludus/catkin_ws
/src/CMakeLists.txt" pointing to "/opt/r
os/indigo/share/catkin/cmake/toplevel.cm
ake"
ludus at iMacdiLudovico in ~/catkin_ws/s
rc
$
```

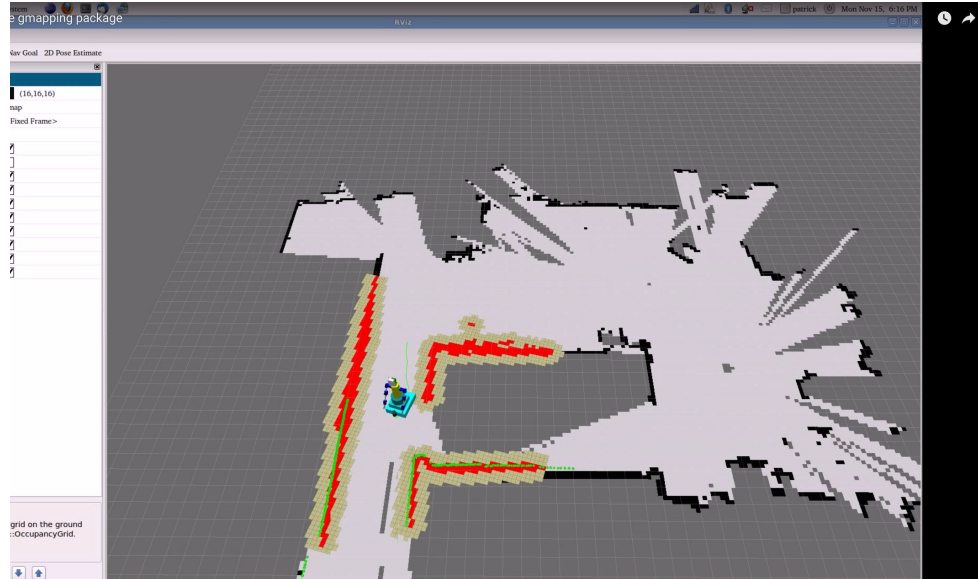
Catkin package

- A catkin package is placed within the src folder
- package/
 - package.xml
 - CMakeLists.txt
 - src/ (c++ Code)
 - msg/ (messages definition)
 - srv/ (services definition)
 - scripts/ (Python code)
- catkin_create_pkg <package_name> <deps>
- package.xml
 - Meta data information about the package
 - Name, description, version etc...
- CMakeLists.txt
 - CMake file containing information on how to build the package

ROS Useful Packages

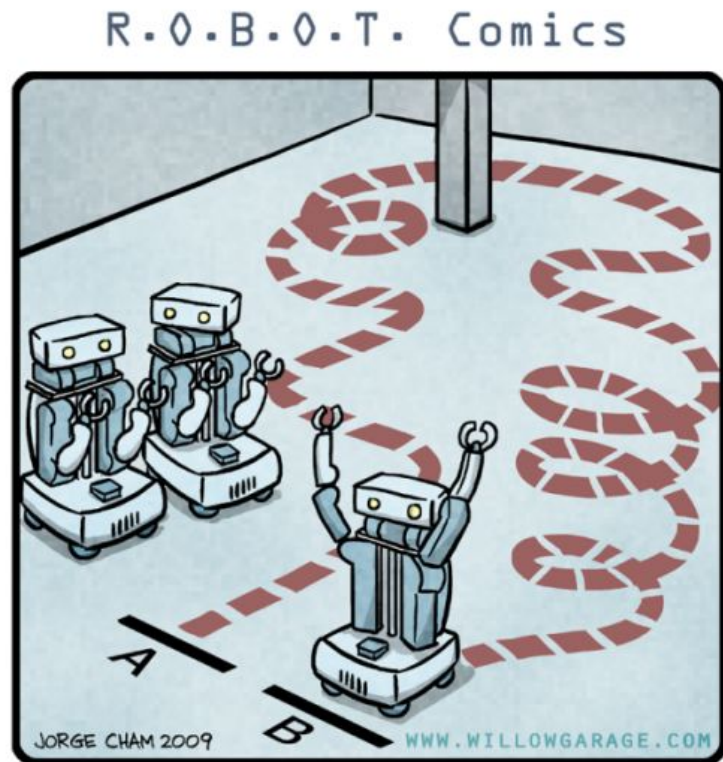
gmapping

- A state of the art node for 2D mapping
- <http://wiki.ros.org/gmapping>
- <https://www.youtube.com/watch?v=khSrWtB0Xik>



Navigation metapackage

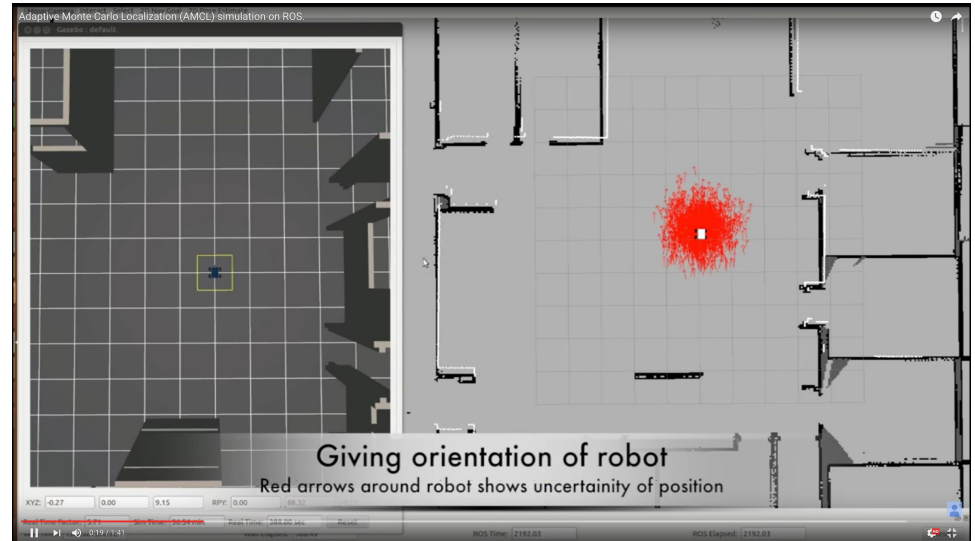
- A 2D navigation stack
- Implement and solves common problems in Robot Navigation
- <http://wiki.ros.org/navigation>



"HIS PATH-PLANNING MAY BE SUB-OPTIMAL, BUT IT'S GOT FLAIR."

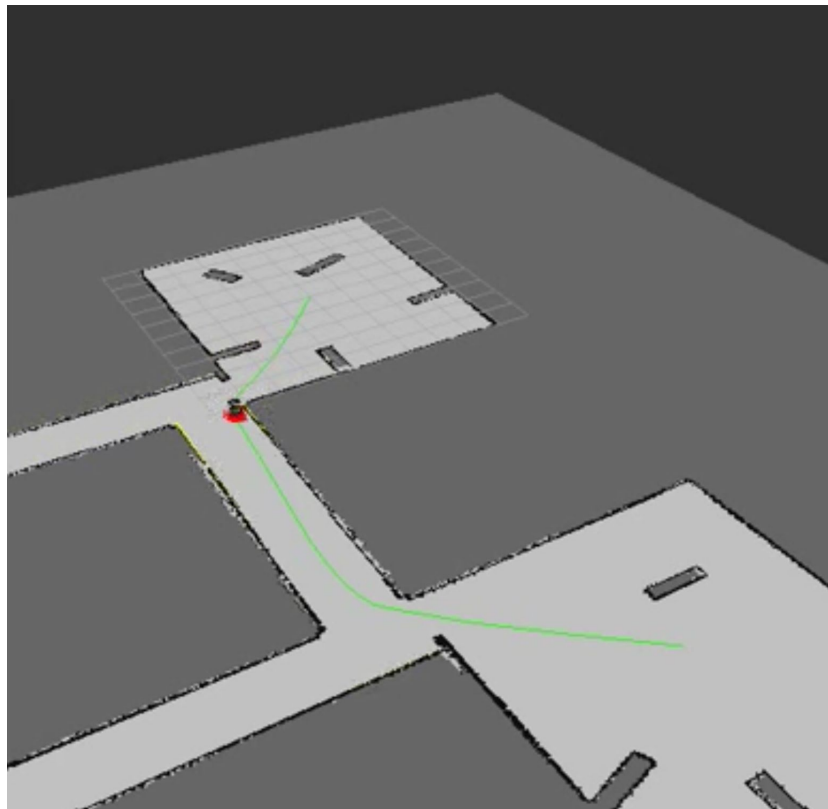
AMCL

- Amcl implements a robot localization algorithm based on particle filters
- <https://www.youtube.com/watch?v=xaeUc1-r9rY>



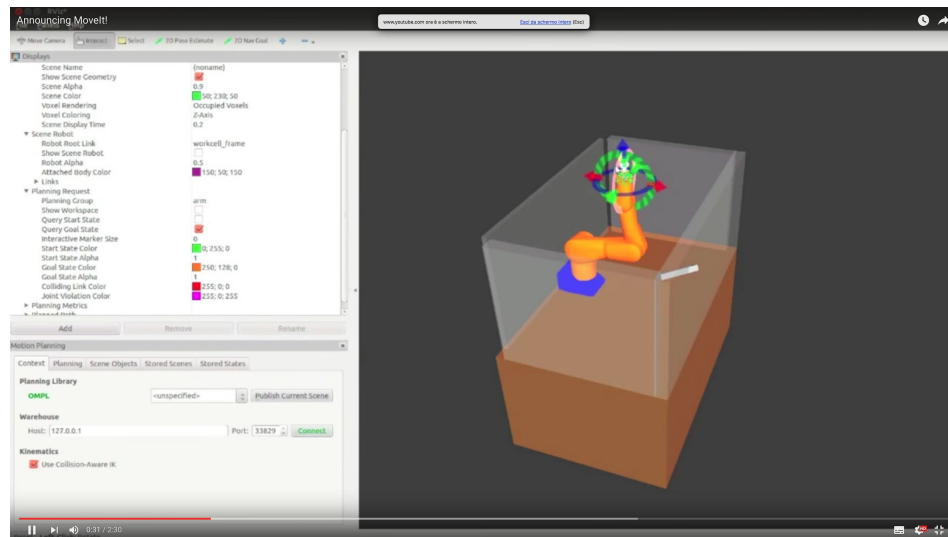
move_base

- Path planning and following algorithm
- http://wiki.ros.org/move_base
- <https://www.youtube.com/watch?v=QFXTCL-i72A>



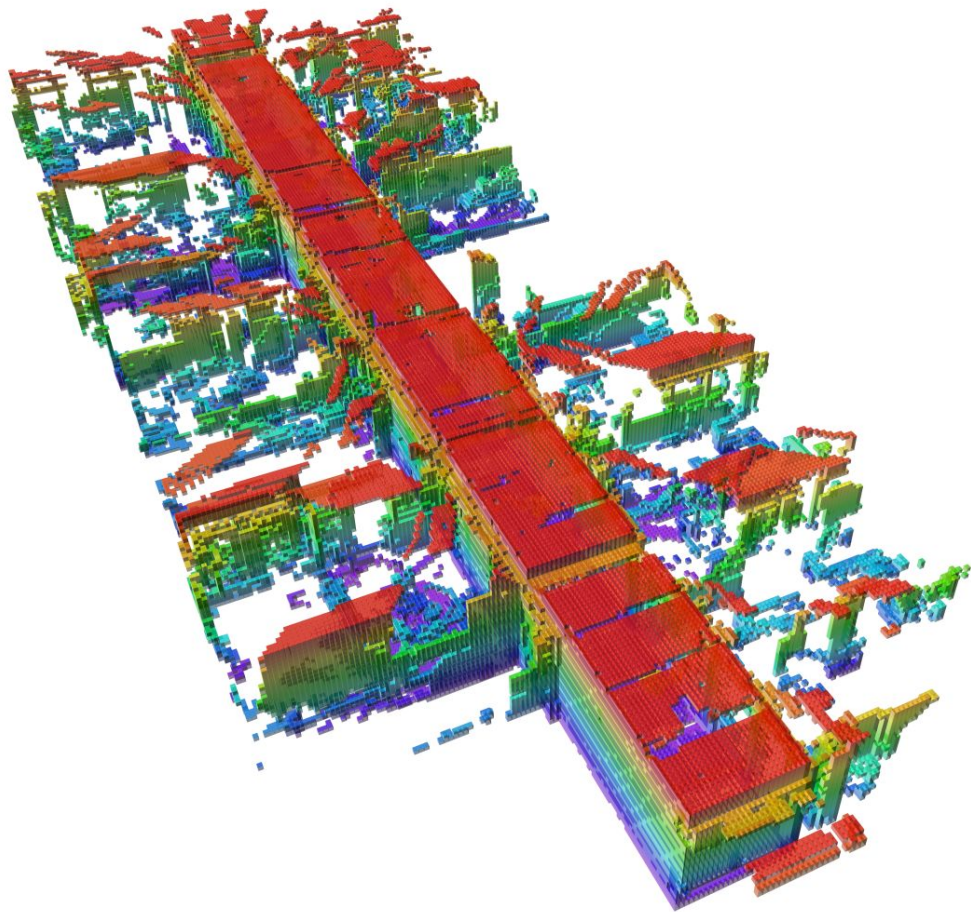
MoveIt - Mobile Manipulation

- Set of tools for mobile manipulation
 - Simulation
 - Motion planning
 - Control
 - 3D Perception
- <http://moveit.ros.org/>
- <https://www.youtube.com/watch?v=vAeEEoxVhAo>



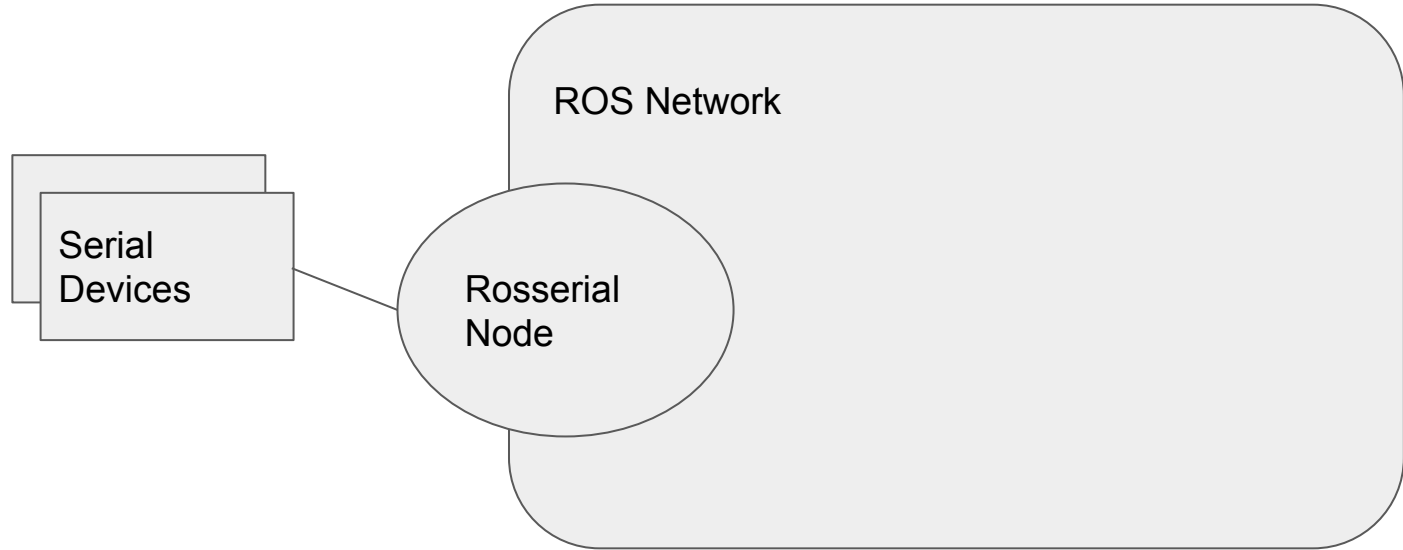
Octomap

- 3D Mapping Framework
- Equivalent package to gmapping
- <https://octomap.github.io/>
- <https://www.youtube.com/watch?v=JwbZegYgxzc>



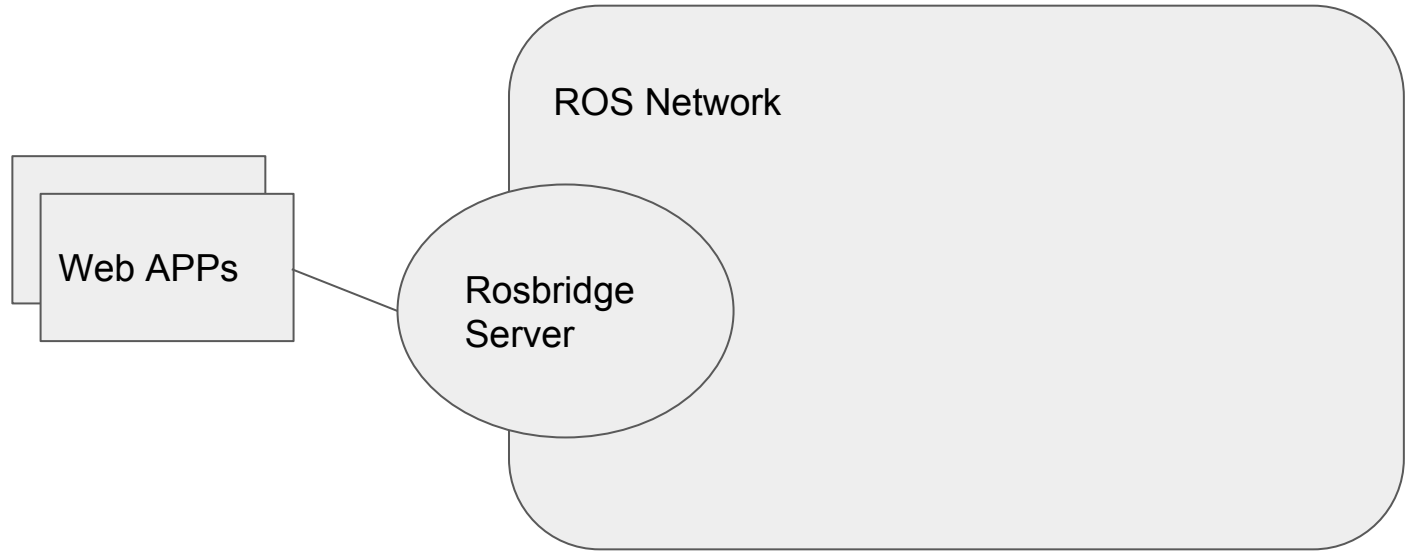
ROSSerial <http://wiki.ros.org/rosserial>

- A protocol for wrapping ROS serialized messages and multiplexing multiple topics and services over a character device such as a serial port or network socket



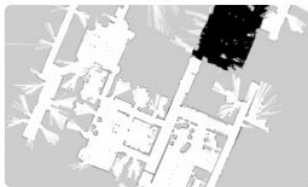
ROS Bridge http://wiki.ros.org/rosbridge_suite

- Rosbridge provides a JSON API to ROS functionality for non-ROS programs.



Robot Web Tools <http://robotwebtools.org/>

```
98 var feedback = {
99   ros : this.ros,
100   name : this.serverName +
101   message : this.actionName +
102   };
103 };
104 var statusListener = new ROSLIB.Topic({
105   ros : this.ros,
106   name : this.serverName + '/status',
107   messageType : 'actionlib_msgs/GoalStatus'
108 });
109 var resultListener = new ROSLIB.Topic({
110   ros : this.ros,
111   name : this.serverName + '/result',
112   messageType : 'actionlib_msgs/GoalStatus'
```



ROSLIBJS

THE STANDARD ROS JAVASCRIPT LIBRARY

CDN: ([min](#)) | ([full](#))

Doc: ([ISDoc](#))

Source: ([GitHub](#))

Wiki: ([ROS Wiki](#)) | ([Tutorials](#))

ROS2DJS

2D VISUALIZATION LIBRARY FOR USE WITH THE ROS

JAVASCRIPT LIBRARIES

CDN: ([min](#)) | ([full](#))

Doc: ([ISDoc](#))

Source: ([GitHub](#))

Wiki: ([ROS Wiki](#)) | ([Tutorials](#))

ROS3DJS

3D VISUALIZATION LIBRARY FOR USE WITH THE ROS

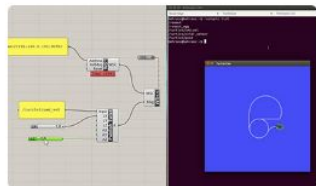
JAVASCRIPT LIBRARIES

CDN: ([min](#)) | ([full](#))

Doc: ([ISDoc](#))

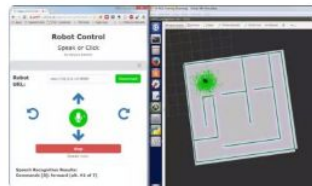
Source: ([GitHub](#))

Wiki: ([ROS Wiki](#)) | ([Tutorials](#))



ROS.GH

A SET OF GRASSHOPPER COMPONENTS FOR



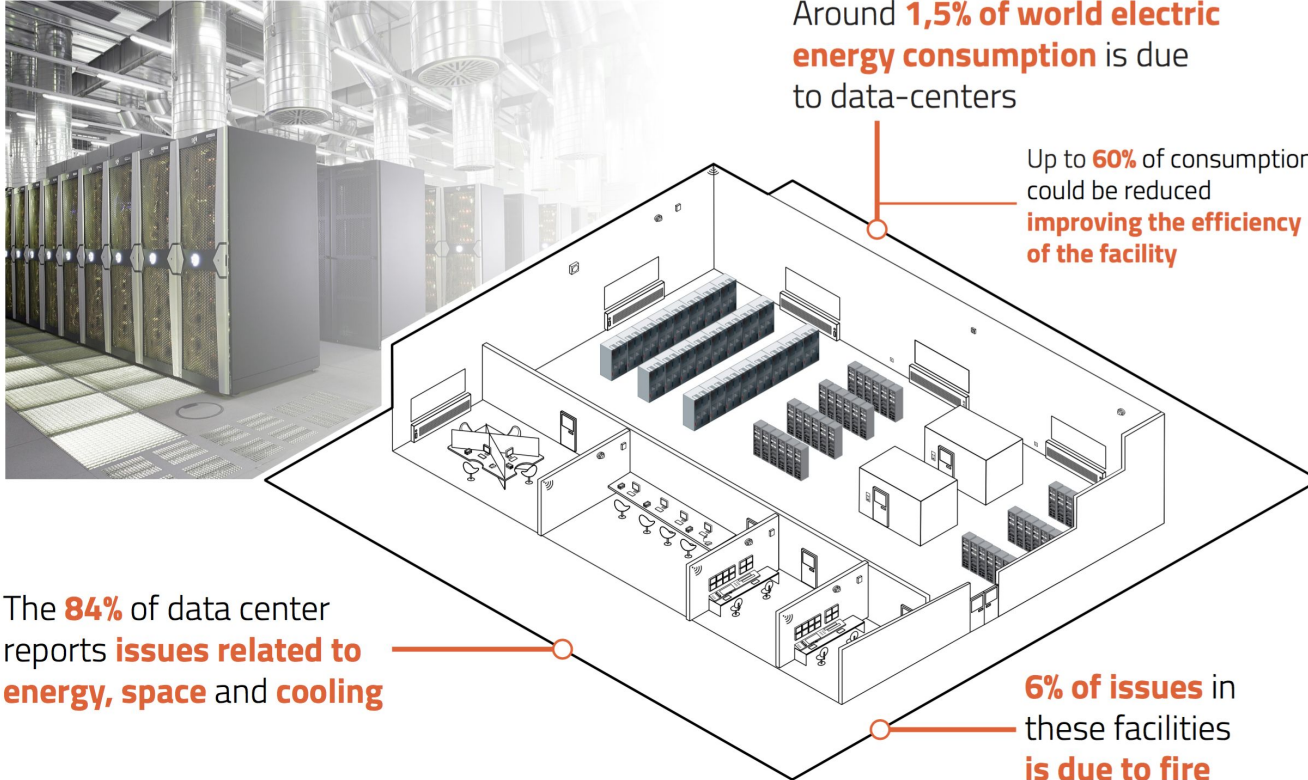
SPEECH COMMANDS

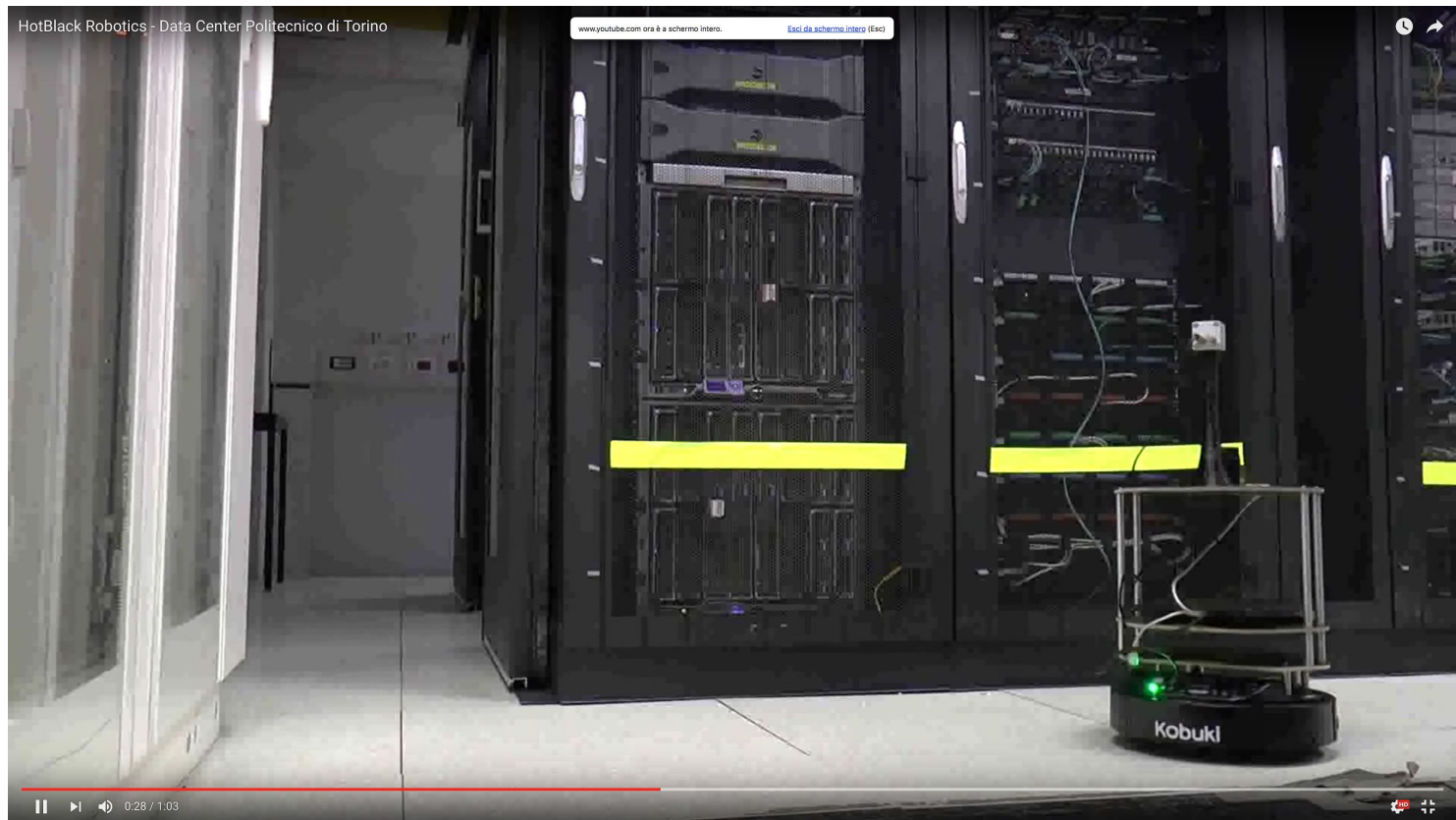
SPEECH CONTROL OF A ROS ROBOT

What you can do with ROS?

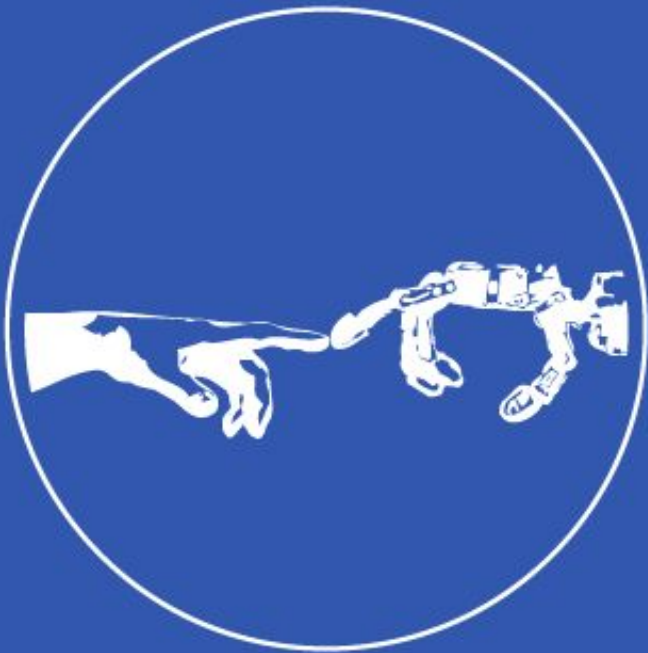
Robot@CED

Data-Center





<https://www.youtube.com/watch?v=HIUB0oHuXrc>



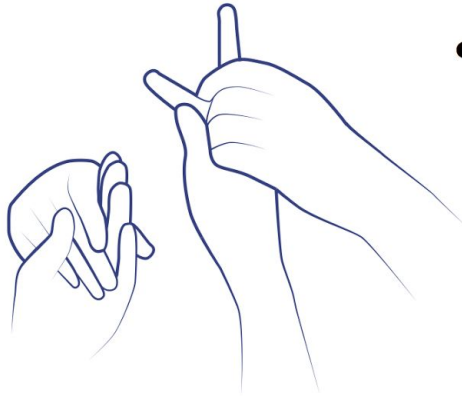
PARLOMA

A Telecommunication system
for deafblind people.

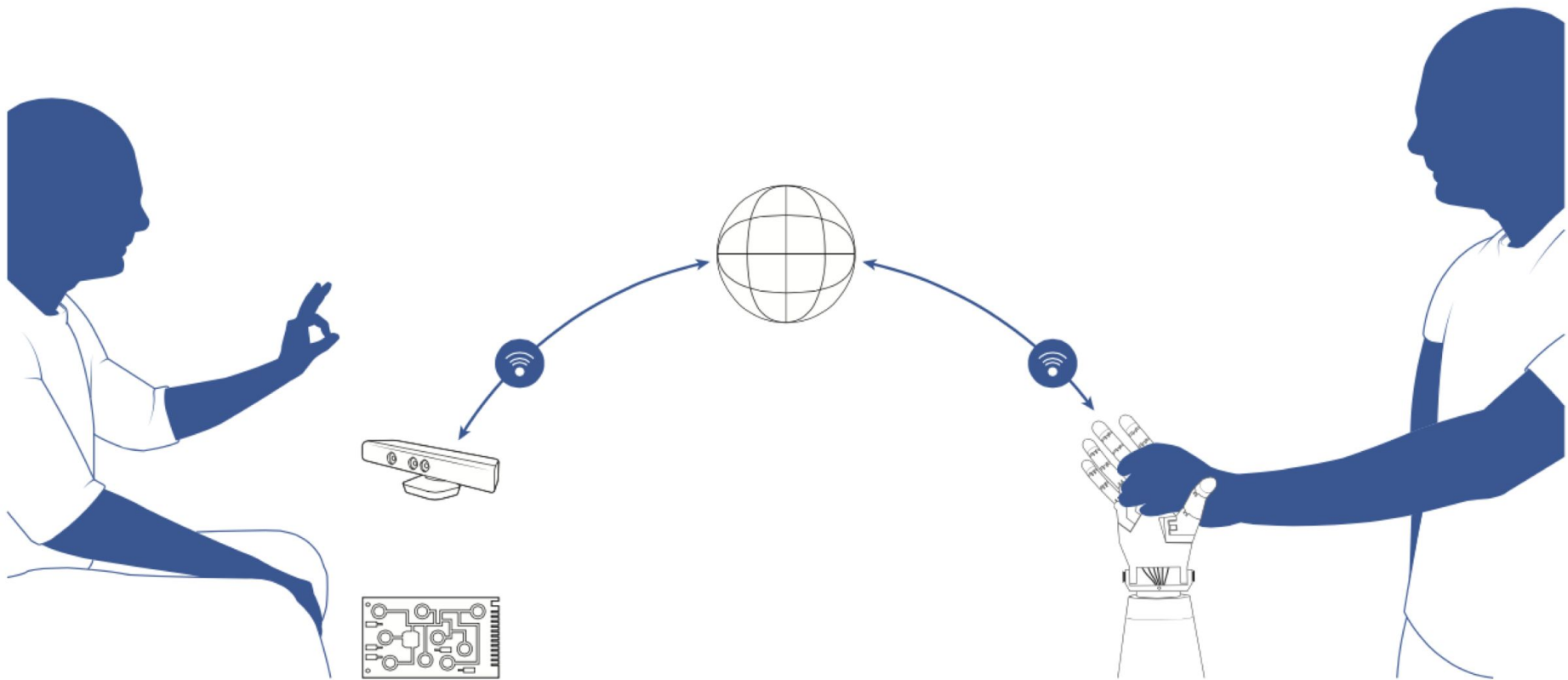


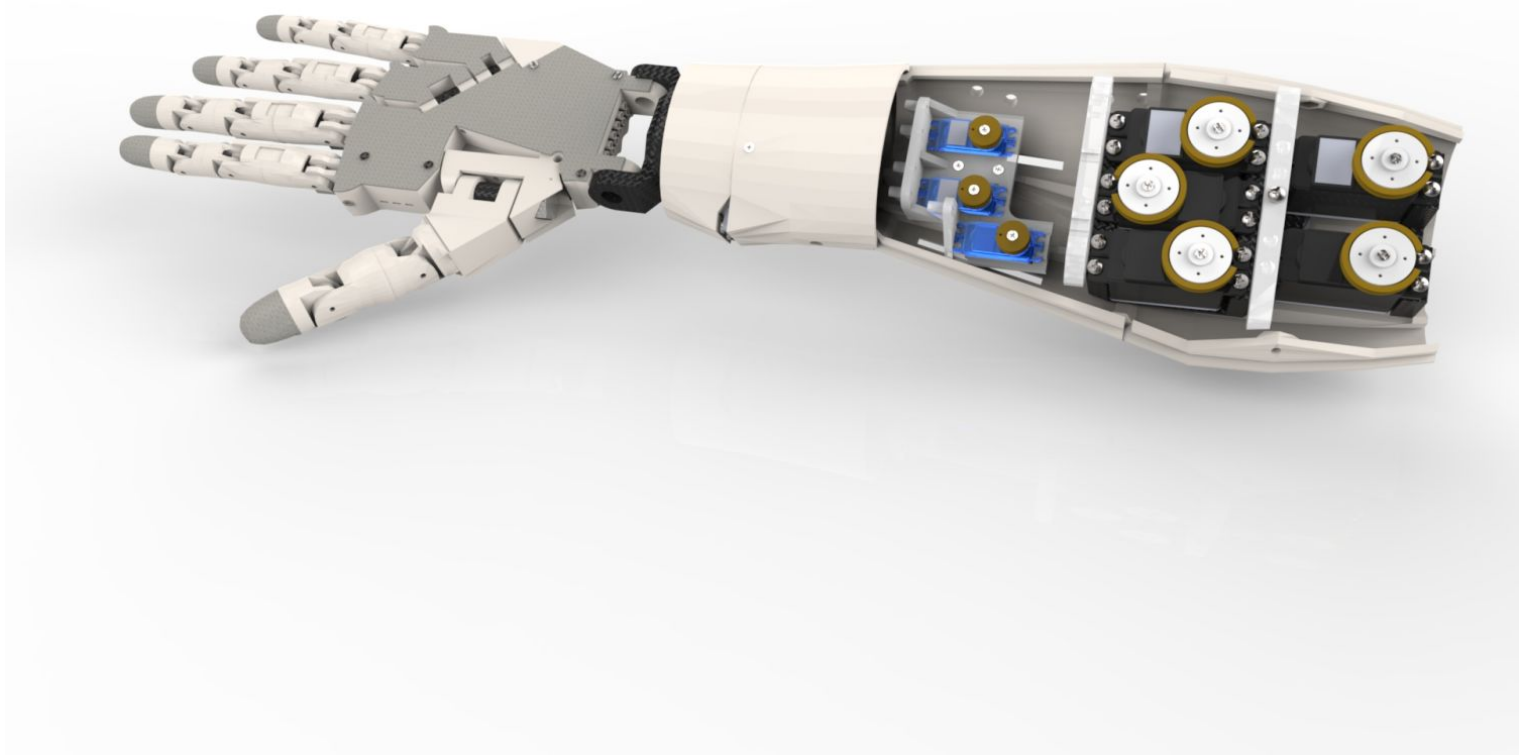
- **Deafblind people have lost both sight and hearing senses**
 - They cannot see nor hear
- **Deafblind people **communicate with tact****
- This is an huge limitation in communication
- They communicate only if the are in the same place

- Deafblind people **communicate mainly with tact**
- Communications skills are strongly related with the **impairment evolution**
 - In some cases, they use communications solutions based on spoken language (such as braille or malossi)



- Most deafblindness people are affected by **Husher's Syndrome**
 - They born deaf and then they lost sight
 - They learn Sign Language (SL) and then they evolve their skills in tactile SL

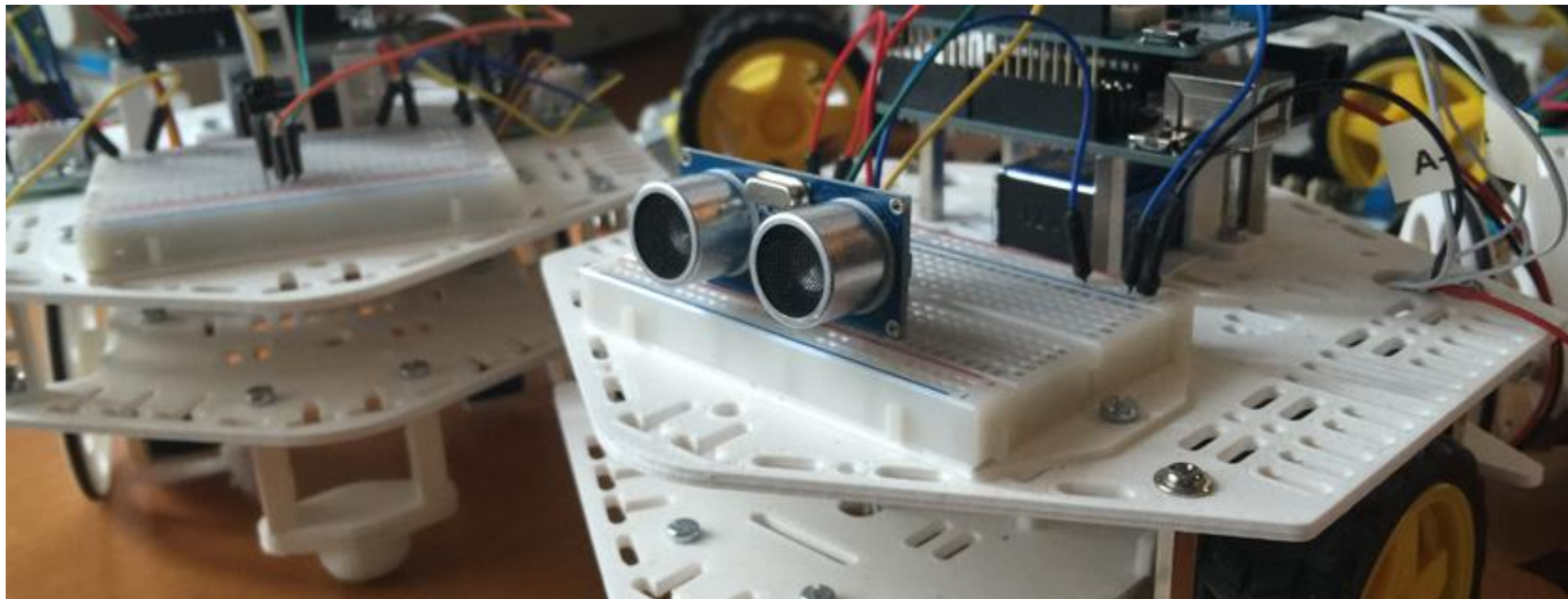




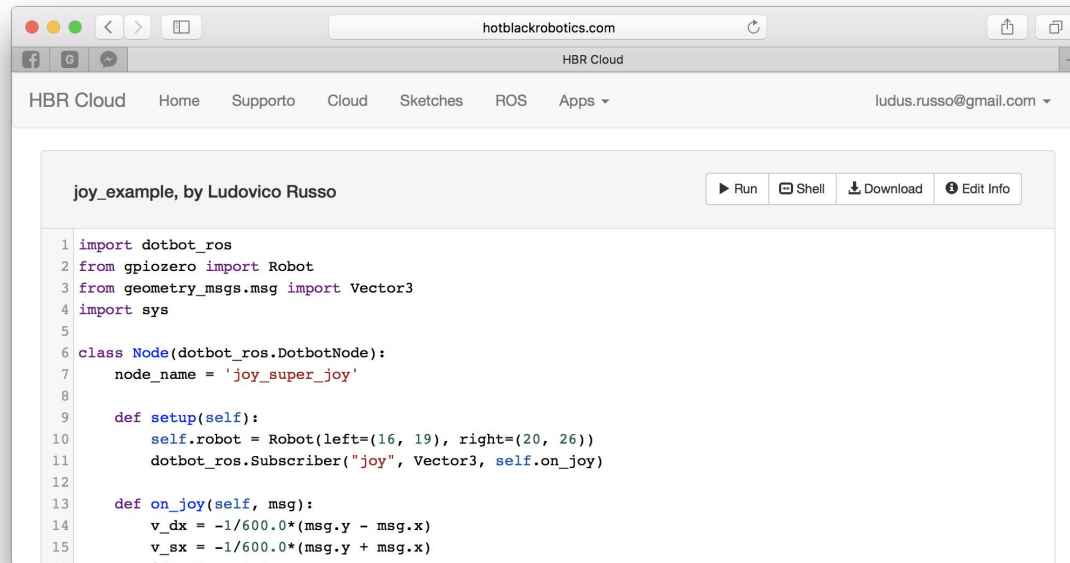


<https://www.youtube.com/watch?v=EJ5-uBt7rHs>

DotBot



Examples Using the HotBlack Robotics Cloud Platform



The screenshot shows a web browser window at hotblackrobotics.com. The page title is "HBR Cloud" and the user is logged in as ludus.russo@gmail.com. The navigation menu includes Home, Supporto, Cloud, Sketches, ROS, and Apps. The main content area displays a code example titled "joy_example, by Ludovico Russo". The code is as follows:

```
1 import dotbot_ros
2 from gpiozero import Robot
3 from geometry_msgs.msg import Vector3
4 import sys
5
6 class Node(dotbot_ros.DotbotNode):
7     node_name = 'joy_super_joy'
8
9     def setup(self):
10         self.robot = Robot(left=(16, 19), right=(20, 26))
11         dotbot_ros.Subscriber("joy", Vector3, self.on_joy)
12
13     def on_joy(self, msg):
14         v_dx = -1/600.0*(msg.y - msg.x)
15         v_sx = -1/600.0*(msg.y + msg.x)
```


ROKERS

www.rokers.io



Thank you!