

Lecture 1: course overview and introduction to ROS

Jonathan Cacace
jonathan.cacace@unina.it

PRISMA Lab
Department of Electrical Engineering and Information Technology
University of Naples Federico II

www.prisma.unina.it

Robotics Lab: learn how to program advanced robotic systems (mobile and industrial).

- Formal teacher: Prof. Vincenzo Lippiello
- Lecturer: dr. Jonathan Cacace
 - Postdoctoral researcher @ PRISMA Lab: building 3A, room S.07
 - <http://wpage.unina.it/jonathan.cacace/>
 - jonathan.cacace@unina.it, [gmail.com](mailto:jonathan.cacace@gmail.com)
 - **Teaching assistantship:** Monday 14.30/16.30 (on appointment!)

Robotics Lab: first time for us!

Objectives:

- Improve your skills in programming and usage of Linux OS
- Learn to design a whole robotic application interfacing sensors, decision and control
- Learn to use off-the-shelf software to speed up the development process

This topics discussed during the course can be classified as follow:

Prerequisite:

- Basic usage of Linux Operating System
- Basic usage of version control tools (git)
- Recall of c++ programming and compilation tools (cmake)
- Robot Operating System (ROS)



Robot Programming Tools:



- Robotic simulation
- ROS Packages:
 - Robot modelling
 - Kinematic
 - Robot control
 - Localization
 - Navigation
 - ...
- Matlab for robot programming
- Image elaboration
- Sensors


Robotic application:



- Discussion about robotic applications programmed with ROS

Motivation:


Robotics Software Engineer (m/w/d)
 iks Engineering GmbH
 Ulm, DE
 Diventa uno dei primi candidati
 5 giorni fa


Robotics Support Engineer - Europe
 Locus Robotics
 Köln, DE
 Diventa uno dei primi candidati
 2 giorni fa


Robotics Engineer
 micropsi industries
 Berlin, DE
 1 mese fa


Konstrukteur Robotics
 Brunel
 Lübeck, DE
 Diventa uno dei primi candidati

Must Have Qualifications

- Knowledge of robotic manipulator design
- Experience in working with robot kinematics and solving inverse kinematics
- Knowledge of robot control algorithms and real world experience
- Willingness to own code and ideas
- Python, C or C++ skills
- Excellent written and interpersonal communication skills

Nice to have


- Previous experience in the robotics integration business
- Engineering skills, particularly CAD and rapid prototyping
- ROS experience


Perks / Benefits


- A chance to contribute to and be responsible for a piece of exciting AI technology
- Work towards applications while keeping in touch with academic research
- A friendly work atmosphere with smart people who value kindness, humility, curiosity, rational decision making, and rigor.
- Competitive compensation.


Please upload your resume and a short motivation letter mentioning your earliest possible start date. We are looking forward to your application.

Motivation:


Robot QA Engineer (m/f/x)
 Magazino GmbH
 München, Bayern, Deutschland
 2 settimane fa


Multi-robot Coordination Engineer (f/m/x)
 Magazino GmbH
 München, DE
 Diventa uno dei primi candidati
 1 settimana fa


Senior Robot Navigation Engineer (f/m/x)
 Magazino
 München, DE
 Diventa uno dei primi candidati
 3 mesi fa


Senior Robot Navigation Engineer (f/m/x)
 Magazino GmbH
 München, DE
 Diventa uno dei primi candidati
 5 ore fa

...emerging scene, representing the state-of-the-art in robotics research, a fantastic atmosphere, a bundle of benefits and a fantastic team. Are you interested in how this looks like? Come and see!

Your tasks

- Work closely with our roboticists to design, implement, execute and maintain QA procedures for all components in the system
- Analyze, troubleshoot and report complex and fascinating robotic problems before they reach our customer's facilities
- Improve our software quality framework, including software unit tests, integration tests, functional hardware-in-the-loop tests, and robot performance tests
- Help to drive our software deployment procedures from the developers to the robots, which includes maintaining our buildfarm, building software packages, integrating with automated testing procedures, and managing releases

Your profile

- Bachelor or Master in robotics, IT, mechatronics, or similar
- Knowledge of ROS (Robot Operating System), Linux, Python and/or C++
- Experience with robotic and/or embedded software development
- Basic understanding of mechanics, electrical systems, networks, IT and Linux


Nice to have:


- First experience with software testing and hardware-in-the-loop testing
- Experience with Jenkins or similar build automation tools, software deployment concepts, Debian package building, and APT repository maintenance
- Fluent English required, basic German skills helpful


Motivation:


Lausanne Area, Switzerland

2 settimane fa [in](#) Candidatura semplice

 **Robotics Faculty**
DPS Coimbatore
Coimbatore, IN
🕒 Diventa uno dei primi candidati
1 mese fa

 **Robot QA Engineer (m/f/x)**
Magazino GmbH
München, Bayern, Deutschland
2 settimane fa

 **Mechatronics and Robotics- Assistant Professor**
Hindustan University
Bangalore, IN
🕒 Diventa uno dei primi candidati
1 mese fa

 **Robotics Engineer**
Tesla
📍 Fremont, CA, US
1 mese fa

Requirements

- BS in mechatronics engineering or equivalent experience.
- 3+ years extensive experience Robotic programming and applications.
- Excellent robotic high level and back ground programming skills on Fanuc, KUKA, ABB or equivalent brands.
- Hands on experience commissioning and programming industrial robots
- Robotic mechanical and electrical trouble shooting skills.
- Experience with Robot simulation packages as well as off-line programming using Robcad, Delmia 3DX or Process Simulate (preferred).
- Virtual commissioning experience of a robotics system is a plus.
- Knowledge on programming languages such as Python, Java, C, C++, HTML, ROS
- Experience on robotics applications such as Spot Weld, Stud Weld, Adhesive, Self Piercing Rivet, Vision, Arc Weld, Roll hemming.
- Experience on Automotive industries is a plus.
- Good overall people management skills.
- Proficient in CAD (CATIA is a plus)
- Basic PLC programming knowledge is a plus.

Livello di anzianità

Esperienza minima

Tipo di impiego

A tempo pieno

Settore


Marketing e pubblicità,
Settore automobilistico, Internet

Funzioni lavorative

Ingegneria, Informatica


Visualizza meno ~

Motivation:




Software Engineer
Talents in Motion
Genova, Liguria, Italia


1 settimana fa · [in](#) Candidatura semplice




Product Engineer Robotics
Omron Industrial Automation Europe
Milano, IT

 1 ex studente lavora qui


6 giorni fa




Control System Specialist
FCA - Fiat Chrysler Automobiles
Orbassano, Piedmont, Italy

 19 ex dipendenti lavorano qui

1 settimana fa · [in](#) Candidatura semplice



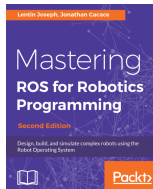
Machine Learning Engineer
Camlin Group
Parma, Emilia-Romagna, Italy

 Diventa uno dei primi candidati

- Education: Automation Engineering M.Sc.;
- Knowledge of Control Systems design, Estimation, Optimization and Vehicle Control Systems as ESC (Electronic Stability Control), EPS (Electronic Power Steering);
- Knowledge of Vehicle Dynamics and Ride Comfort;
- Technical skill in:
 - Control Design, Development and Analysis by using Matlab/Simulink environment (e.g. Control Systems Toolbox, Model Predictive Control Toolbox, ...);
 - Physical modeling, identification and simulation with Matlab/Simulink (e.g. System Identification Toolbox, Neural Networks Toolbox);
 - Rapid Prototyping ECU programming (e.g. dSpace MicroAutobox, ...);
 - Vehicle Dynamics Commercial Models usage (i.e. IPG CarMaker / dSPACE ASM);
 - CAN Data Analysis by using Vector SW (e.g. CANalyzer, CANape, CANdb++...) and HW (CANcaseXL, CANlog3, GL2000, ...);
 - **Robotics Development environment (i.e. ROS);**
 - C, C++ basic programming languages and code generation process knowledge;
- Good knowledge of English language: it is required the ability to handle in English language both the release of technical documents and meetings with suppliers and FCA worldwide organization;
- To be passionate of:
 - Control theory and related applications;
 - Rapid prototyping technologies and related applications.

Material:

- **Main:** lecture notes. Each lesson is associated with a set of slide, pdf lecture notes and eventually materials (source code, etc...).
- Text book: L. Joseph, J. Cacace, 'Mastering ROS for Robotics Programming 2nd edition'. Get it 'here'.



- Google group: rl2020 (on invitation)

Evaluation:

- Simulation scenes to program industrial and mobile (ground and aerial) robots
- Each candidate chooses a simulation scene to develop a project
- The project can be made in working groups composed by 2/3 persons
- *Project*: ROS package to accomplish a given task
- *Exam*: discussion of the project plus questions to prove student' knowledge

Each lesson is associated with:

- A set of slides
- Lecture notes
- Sample code (if needed)

Step-by-step exercises will help you to become familiar with robotics programming.

Any questions or doubts about the organization of the course?

ROS

Robot Operating System (ROS) represents a flexible framework, providing various tools and libraries to write robotic software. It offers several powerful features to help developers in such tasks as message passing, distributing computing, code reusing, and implementation of state-of-the-art algorithms for robotic applications.



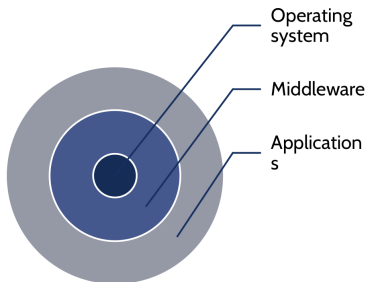
Originally developed in 2007 at the Willow Garage and Stanford Artificial Intelligence Laboratory under GPL license

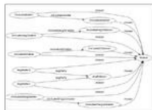
- Since 2013 managed by OSRF
- Today used by many robots, universities and companies
- Standard for robots programming

Motivation:

The most common problem of robotics at the time was that they spent too much time to re-implement the software infrastructure required to build complex robotics algorithms like: drivers to the sensors and actuators, and communications between different programs inside the same robot and too little time dedicated to build intelligent robotics programs that were based on that infrastructure

The correct definition for ROS is a robotic middleware: a software that connects different software components or applications.





Plumbing

- Process management
- Inter-process communication
- Device drivers

+



Tools

- Simulation
- Visualization
- Graphical user interface
- Data logging

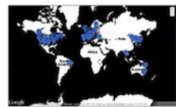
+



Capabilities

- Control
- Planning
- Perception
- Mapping
- Manipulation

+



Ecosystem

- Package organization
- Software distribution
- Documentation
- Tutorials

ros.org

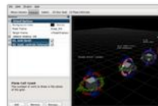
ROS allows communication between processes (nodes). In particular, it provides publish-subscribe messaging infrastructure designed to support the quick and easy construction of distributed (local and remote) computing systems. For example, consider that your application uses data from a camera. You can use the ROS node deployed by the vendor of your camera to implement your application.



Plumbing

- Process management
- Inter-process communication
- Device drivers

ROS provides an extensive set of tools to configure, manage, debug, visualize data, log and test your robotic application.



Tools

- Simulation
- Visualization
- Graphical user interface
- Data logging

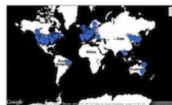
ROS provides a broad collection of libraries that implement useful robot functionalities, like manipulation, control, and perception. In addition, ROS can be connected to other external software like OpenCv, PCL, and so on, thanks to proper wrappers.



Capabilities

- Control
- Planning
- Perception
- Mapping
- Manipulation

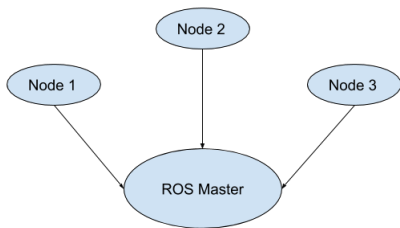
ROS is supported and improved by a large community, with a strong focus on integration and documentation. On ROS webpage: ros.org you can find basic and advanced tutorial to learn how to program in ROS, while the Q&A website (answers.ros.org) allow you to directly ask you solution for your own problems (and contains thousand of question already answered).



ros.org

Ecosystem

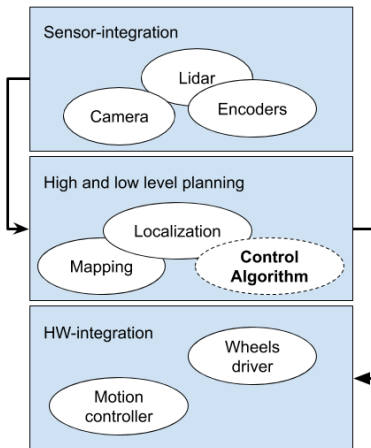
- Package organization
- Software distribution
- Documentation
- Tutorials



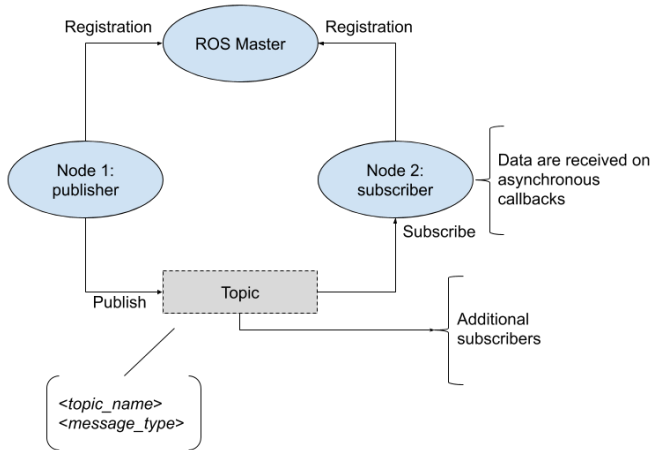
- Peer to peer.
- Distributed.
- Multi-language.
- Light-weight.
- Free and open-source.

- Nodes are the processes that perform computation (the executable). Each ROS node is written using ROS client libraries implementing different ROS functionalities, such as the communication methods between nodes, which is particularly useful when different nodes of our robot must exchange information between them. Using the ROS communication methods, they can communicate with each other and exchange data. One of the aims of ROS nodes is to build simple processes rather than a large process with all the functionality (modularity).

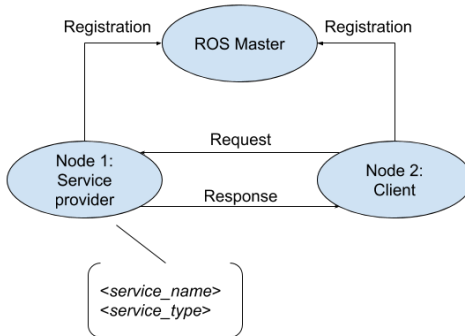
- Master: The ROS Master is a particular ROS node that provide the name registration and lookup to the rest of the nodes. Nodes will not be able to find each other, exchange messages, or invoke services without a ROS Master. In a distributed system, we should run the master on one computer, and other remote nodes can find each other by communicating with this master.



Publish-subscribe:



Services:



ROS communication relies on a set of standard and custom data structures called ROS messages. The types of data are described using a simplified message description language called ROS messages. These datatype descriptions can be used to generate source code for the appropriate message type in different target languages. The message definition consists in a typical data structure composed by two main types: fields and constants.

`geometry_msgs::PoseStamped` is used to share the pose of an object:

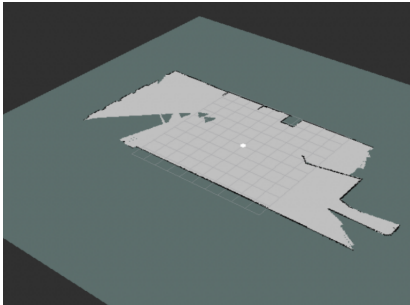
```
1  std_msgs/Header header
2      uint32 seq
3      time stamp
4      string frame_id
5  geometry_msgs/Pose pose
6      geometry_msgs/Point position
7          float64 x
8          float64 y
9          float64 z
10     geometry_msgs/Quaternion orientation
11         float64 x
12         float64 y
13         float64 z
14         float64 w
```

Service: request - response:

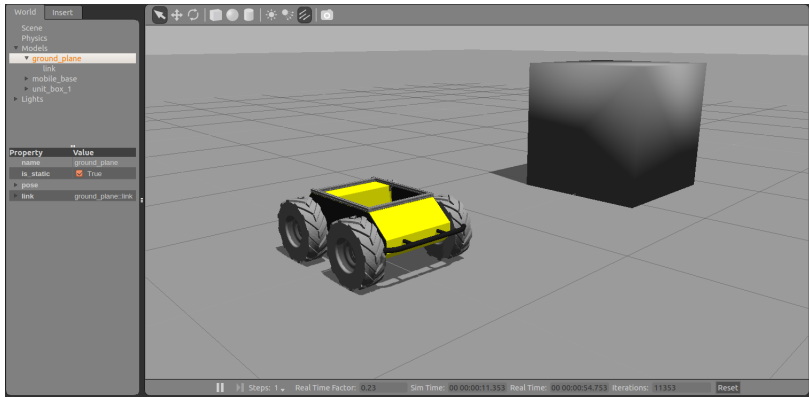
```
1 #Request message type
2 string str
3 ---
4 #Response message type
5 string str
```

The first section is the message type of the request that is separated by `---` and in the next section is the message type of the response. In these examples, both Request and Response are strings.

RViz: ROS Visualization



Gazebo ROS



ROS issue

- No QoS in communication
- Security: the ROS master will respond to requests from any device on the network

ROS cannot be used in **industry** as is.

To programming using ROS you must setup a developer's PC:

- Ubuntu 18.04: <https://ubuntu.com/tutorials/tutorial-install-ubuntu-desktop#1-overview>
- ROS Melodic:
<http://wiki.ros.org/melodic/Installation/Ubuntu>

Dual boot vs Virtual Machine: A simplest way to setup the developer's PC is installing Linux on a Virtual Machine. This is not suggested due to the high hardware resources needed by the simulation scenes used in the evaluation. Moreover, you can use it in the first part of the course to test initial robotic programming examples.

During we will start to discuss the technologies used to program robots:

- C++
- C++ compilation using Cmake
- Git (version control)
- Linux command line tools