The objective of this course is to introduce students in the use of ROS as a powerful robotics tool. Specifically, a familiarization with the middleware concept and the software structure of a robot. There will be a special emphasis on sensing and control of robots using ROS, both in simulation and in real environments.

Learning Outcomes:
Learn how to setup a Linux O.S. environment to work with ROS.
Understand the ROS communications architecture.
Use ROS in the different process layers, from sensing to control or actuation.
Implement simple ROS projects with both simulation and real robots.

Mandatory contents:
Install and setup ROS in a native O.S. Linux (Ubuntu).
Know and understand the internal procedures of ROS and its modules functionalities (master, nodes, and so on).
Identify and use the ROS tools and formats related to the internal communication between nodes (topics, actions, services, ...).
Use ROS visualization and debugging tools.
Design and program C++ algorithms using ROS as a middleware.
Use debugging tools to verify the compilation and the algorithm functionalities. Configure and use a simulation environment with the designed algorithms. Managing acquisition, analysis and display of data obtained from different sensors using ROS. Manage and send control commands to a robot using ROS, both using simulation and real settings.

<table>
<thead>
<tr>
<th>Study load</th>
<th>Total learning time: 112h 30m</th>
<th>Hours large group:</th>
<th>27h</th>
<th>24.00%</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Hours small group:</td>
<td>13h 30m</td>
<td>12.00%</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Self study:</td>
<td>72h</td>
<td>64.00%</td>
<td></td>
</tr>
</tbody>
</table>
# Content

## 1. ROS Basic concepts

**Learning time:** 5h  
Practical classes: 3h  
Self study: 2h

**Description:**  
1.1. Introduction  
1.2. ROS core components  
1.3. Applications  
1.4. Install instructions  
1.5. ROS command-line tools

## 2. Development Tools

**Learning time:** 5h  
Practical classes: 3h  
Self study: 2h

**Description:**  
2.1. Programming  
2.2. Building executables  
2.3. The ROS build system  
2.4. Good practices  
2.5. Version control using GIT

## 3. Communications using topics

**Learning time:** 5h  
Practical classes: 3h  
Self study: 2h

**Description:**  
3.1. An example: The package agitr_chapter3  
3.2. A publisher program  
3.3. A subscriber program  
3.4. Standard and common messages
### 4. The launch utility

**Description:**
- 4.1. Using launch files
- 4.2. Understanding launch files
- 4.3. Graph resource names
- 4.4. Managing names in launch files
- 4.5. ROS parameters

**Learning time:** 5h  
Practical classes: 3h  
Self study: 2h

### 5. Communications using services

**Description:**
- 5.1. Services
- 5.2. The package agitr chapter8
- 5.3. A client program
- 5.4. A server program
- 5.5. Standard services
- 5.6. Defining non-standard services

**Learning time:** 5h  
Practical classes: 3h  
Self study: 2h

### 6. Tools

**Description:**
- 6.1. The tf tool
- 6.2. Robot Modeling and visualization tools
- 6.3. The rosbag Tool
- 6.4. The rqt tool

**Learning time:** 5h  
Practical classes: 3h  
Self study: 2h
# 7. Communications using actions

**Description:**
- 7.1. Working with ROS actionlib
- 7.2. Building and running a simple example
- 7.3. The ROS action server
- 7.4. The ROS action client
- 7.5. The pan-tilt example

**Learning time:** 5h
- Practical classes: 3h
- Self study: 2h

---

# 8. Simulation - basic issues

**Description:**
- 8.1. Gazebo basics
- 8.2. Integration to ROS
- 8.3. Configuring launch files
- 8.4. ROS-aware Gazebo plugins
- 8.5. Tunning URDF models

**Learning time:** 5h
- Practical classes: 3h
- Self study: 2h

---

# 9. Simulation - sensors

**Description:**
- 9.1. Available ROS plugins
- 9.2. The camera ROS plugin
- 9.3. The depth camera ROS plugin
- 9.4. ROS plugins for some other sensors

**Learning time:** 5h
- Practical classes: 3h
- Self study: 2h
# 10. Robot control

<table>
<thead>
<tr>
<th>Description:</th>
</tr>
</thead>
<tbody>
<tr>
<td>10.1. ros_control overview</td>
</tr>
<tr>
<td>10.2. Controllers</td>
</tr>
<tr>
<td>10.3. Hardware Abstraction Layer</td>
</tr>
<tr>
<td>10.4. Using ros_control in Gazebo</td>
</tr>
</tbody>
</table>

**Learning time:** 5h  
- Practical classes: 3h  
- Self study: 2h

---

# Case study

<table>
<thead>
<tr>
<th>Description:</th>
</tr>
</thead>
<tbody>
<tr>
<td>Definition of the solution</td>
</tr>
<tr>
<td>Sensing module</td>
</tr>
<tr>
<td>Planning module</td>
</tr>
<tr>
<td>Action module</td>
</tr>
</tbody>
</table>

**Learning time:** 15h  
- Practical classes: 9h  
- Self study: 6h

---

## Qualification system

The acquired competences and capabilities will be assessed on the basis of three qualification grades: exercises (20%), deliverable (20%) and final project (60%).  
Re-evaluation: new final project (60%).
Bibliography

Basic:

Others resources:
- ROS wiki page: http://wiki.ros.org/
- ROS tutorials: http://wiki.ros.org/ROS/Tutorials/
- Gazebo tutorials: http://gazebosim.org/tutorials/
- Catkin tutorials: http://jbohren.com/tutorials/
- ROS cheatsheet: https://github.com/ros/cheatsheet/releases/download/0.0.1/ROScheatsheet_catkin.pdf /

Hyperlink
- Introduction to ROS: online tutorials
  https://sir.upc.edu/projects/rostutorials/index