

# Autonomous Systems & Introduction to Robotics

## ROS practical session

João Quintas & Rute Luz

(Oscar Lima & Carlos Azevedo)

ISR: Institute for Systems and Robotics  
LARSyS: Laboratory for Robotics and Engineering Systems  
IST: Instituto Superior Técnico, Lisboa Portugal

September 24, 2019



# ROS pkg structure<sup>1</sup>

- ROS nodes, a ROS-independent library, a dataset, configuration files, a third-party piece of software, etc
- ROS packages tend to follow a common structure
- For python code it will look like this:

```
carlos@cthinkpad:AutSysPKG $ tree
.
├── CMakeLists.txt
├── common
│   └── src
│       └── AutSysPKG
│           ├── __init__.py
│           └── my_ros_independent_class.py
├── package.xml
├── ros
│   ├── config
│   │   └── config_AutSysPKG.yaml
│   ├── doc
│   │   └── README.md
│   ├── launch
│   │   └── AutSysPKG.launch
│   ├── scripts
│   │   └── AutSysPKG_node
│   ├── src
│   │   └── AutSysPKG_ros
│   │       ├── AutSysPKG_node.py
│   │       └── __init__.py
│   └── test
│       └── AutSysPKG_test.py
└── setup.py
```

- Offers a set of shell commands for using ros with bash (linux terminal)
- Most popular include:
  - ▶ `roscd pkg_name` (cd to `pkg_name` easily)
  - ▶ `roscd pkg_name filename` (quickly edit a file)
  - ▶ `roscat pkg_name filename` (quickly visualize a file in terminal)
  - ▶ `roscat pkg_name executable` (run executable from anywhere without having to give its full path)
- enables tab completion on: `roslaunch`, `rosparam`, `roscat`, `rostopic`, `rosservice`, `rosmmsg`, `rossrv`, `roscat`.

<sup>2</sup><http://wiki.ros.org/rosbash>

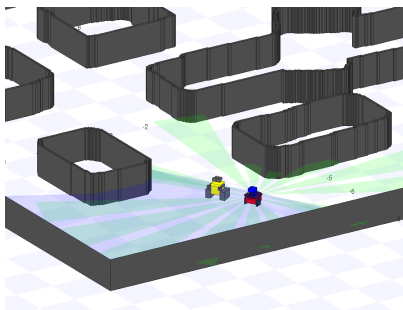
- Part of rosbash suite
- Usage: `roslaunch pkg_name executable_name`
- It will run ONLY executable files
- About files being executable (important!)
  - ▶ make sure your python nodes (i.e. `my_python_node.py`) are executable
  - ▶ check by doing: `ls -l`, if it has an `x` is executable (i.e. `-rwxr-r-`)
  - ▶ alternatively, if your terminal has colors, the file shows green when doing `ls`
  - ▶ roslaunch will also look for your compiled c++ executables (under `devel/lib/pkg_name`)

---

<sup>3</sup><http://wiki.ros.org/roslaunch#roslaunch>

# Stage simulator<sup>4</sup>

- Simulates a population of mobile robots, sensors and objects in a two-dimensional bitmapped environment
- Stage was designed with multi-agent systems in mind, so it provides fairly simple, computationally cheap models of lots of devices rather than attempting to emulate any device with great fidelity.



<sup>4</sup><http://playerstage.sourceforge.net/index.php?src=stage>

- Displays information about ROS topics
- Most useful:
- `rostopic list` (get a list of active topics)
- `rostopic info topic_name` (get topic type, publishers and subscribers)
- `rostopic echo topic_name`
- `rostopic pub topic_name topic_type msg_press_tab!` (publish a topic from console), options:
  - ▶ no args (latched)
  - ▶ `-r float_number` (at a certain rate)
  - ▶ `- -once` (latch for 3 secs, then dies)
- `rostopic hz topic_name` (get the publish frequency rate)

---

<sup>5</sup><http://wiki.ros.org/rostopic>

- A tool for easily launching multiple ROS nodes
- Implemented with XML syntax (`<launch>... </launch>`)
- Allows to load parameters to param server
- A launch file can call other launch files
- Launch a node `<node pkg="..." type="..." name="..." respawn=true ns="..." />`
- Run syntax: `roslaunch pkg_name my_file.launch`

---

<sup>6</sup><http://wiki.ros.org/roslaunch>

## parameter server<sup>7</sup>

- Is a shared, multi-variate dictionary that is accessible via network API
- Nodes can use this server to store or retrieve parameters during runtime
- Is not high performance
- Globally viewable
- Usage from terminal: `rosparam set param_name param_value`,  
`rosparam get param_name`
- Usage from python api: `rospy.set_param(param_name, param_value)`,  
`rospy.get_param("param_name")`
- Suitable for static, non-binary data such as configuration parameters

---

<sup>7</sup><http://wiki.ros.org/Parameter%20Server>



- 3D visualization tool
- Powerful for topic visualization (useful in debugging)
- Sensing state information (laser scans, pointclouds, coordinate frames, cameras)
- Can publish some topics (2D pose estimate, 2D nav goal)
- Is recommended to comply with ROS standard topics to enable topic visualization
- Launch using : `roslaunch rviz rviz` (a roscore must be running)
- Not a simulator

<https://www.youtube.com/watch?v=i--Sd4xH9ZE>

---

<sup>8</sup><http://wiki.ros.org/rviz>

# Thank you!

## Questions? :)

If you have a question please create a Github issue so that we can all benefit from the posted answers under:

[https://github.com/socrob/autonomous\\_systems/issues](https://github.com/socrob/autonomous_systems/issues)