

Exercise 1: Introduction to ROS


Development of Intelligent Systems

2019

The focus of the first exercise is to get familiar with the basics **ROS**. You will learn to write a program within the system, to communicate with other programs and execute it properly. This exercise will introduce several important concepts that are crucial for further exercises so it is recommended that you refresh the topics after the end of the formal laboratory time at home.

1 Setting up

To set up ROS, read the **official guide**. Note that you do not have to do this step on the laboratory computers, however, it is recommended that you have at least one additional computer per group for development and visualization purposes. This year we are using ROS Kinetic installed on top of the Ubuntu 16.04 OS.

 Laboratory laptops have already been set-up. You do not have to create and configure ROS workspace, your team user already has a ROS workspace in `~/ROS` directory.

2 Terminology

ROS is a complex system that introduces a few concepts that are good to know under their established expressions. Read about these concepts on the official webpage:

- **Basic concepts** - file-system organization, process graph elements, name resolving
- **High-level concepts** - coordinate frames, actions and tasks, message ontology

3 Exploring ROS and some useful commands

Let us now start some ROS nodes and explore what is going on in the ROS system. Open a new terminal window and run the command:

```
roscore
```

This command starts the ROS Master. There are other ways to start it, but the ROS Master must be running in order for any nodes to be able to be started. Open a second terminal window and run the command:

```
roslaunch turtlesim turtlesim_node
```

The `roslaunch` command is the simplest way of running ROS nodes. With the previous command we started the `turtlesim_node` which is located in the `turtlesim` package. In a third terminal run the command:

```
roslaunch turtlesim draw_square
```

Now we have started the node `draw_square` from the `turtlesim` package. Now open a forth terminal window and try to find out what is going on in the ROS system with the following commands:

- `rospack`
- `rostopic`
- `rosmmsg`
- `roscat`
- `roscpp`
- `roscpp`
- `rqt_graph`

Note that by typing `-h` or `-help` after the command verb you can get information about the usage of these commands.

Answer the following questions:

- Which nodes are currently active?
- What topics are currently active?
- What is the message type for each topic?
- What topics is each node publishing to?
- What topics is each node subscribed to?
- The used message types belong to which packages?

Additionally try to:

- Get a visualization of all the nodes and topics in the system.
- Get a printout of all the packages installed in the system.
- Get a printout of all the messages installed in the system.
- Print out the messages being published on each topic.
- Publish a message on each topic.

Explore the usage of other commands that are found in the ROS Cheatsheet.

4 Explore a ROS package

Download the zip file for Exercise 1. Unpack it in your workspace and explore the source code of the nodes that are in the package. The following links are useful:

- [Message descriptions](#)
- [Simple publisher and subscriber in C++](#)
- [Simple publisher and subscriber in Python](#)
- [Simple service and client in C++](#)
- [Simple service and client in Python](#)
- [The roslaunch documentation](#)

5 Writing a ROS package

Create a new package using the `catkin_create_pkg` command. Inside the package create a single node that when started with the `roslaunch` command prints out a string (e.g. "Hello from ROS!"). Modify the files so that you can compile the package with the catkin system. Use the following tutorials as a starting point::

- [Creating a package](#)
- [Building a package](#)
- [CMakeLists](#)

6 For homework

- In the new package you created, create a publisher which sends a string and a separate integer field with some message sequence ID. The subscriber should print out these two fields. For this you will need to generate a new message type.
- In the new package you created, write a service node that accepts an array of integers and returns its sum. Demonstrate its correctness by writing a client that generates random sequences of 10 integers and prints out the sequence and the result returned from the service.