Tutorial 1, 25.04.2019 **BILHR SS 2019**

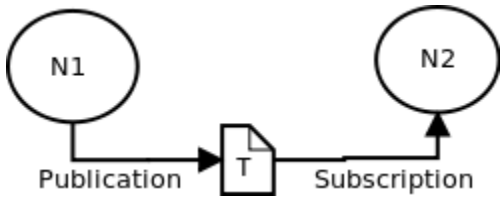
Goal:
The goal of this tutorial is to introduce you to ROS (Robot Operating System). ROS is the software layer between your specific robot programs and the robot (here, the NAO). ROS was developed over many years and it now has a collection of useful libraries and tools for robotic applications. In this tutorial, you learn the following ROS basics:

- Create a workspace and ROS package
- Concept of ROS: nodes and their communication with each other
- Basic message types of the NAO robot

The ICS lab computers run Ubuntu 14.04 LTS (64 bit), along with ROS Indigo. If you want to install ROS on your laptop (optional), we recommend that you install either Ubuntu 14.04 with ROS Indigo, or Ubuntu 16.04 with ROS Kinetic. For documentation on ROS, see http://wiki.ros.org.

1) The main concept of ROS: Communication between nodes.

A node is a process doing some computation. It can receive messages containing input data, process it, and send messages containing the results. Applied to a robot for example, one node can query a specific sensor (*e.g.* tactile sensor), one node can do some higher-level planning or computation, and another node can send and receive motor data (*e.g.* position, velocity, *etc.*). Note that one node does not have to "know" about the existence of another node, in order to communicate with it. For node-to-node communication, it is important that a message topic is defined first. Consider two nodes N1 and N2. N1 sends (`publishes') a message on the topic T. If N2 has `subscribed' to the same topic T, it will receive the message sent by N1. In ROS, you can use pre-defined message structures. Pre-defined messages are data types like int, float, arrays of these basic types, *etc.* Nevertheless, you can also define your own message structures (custom messages). You can define the topic names by yourself.



2) ROS is already installed on your lab computer. Create a workspace by creating a folder in the ~/ros/ directory called */bioinspired_ws/src* and use the command *catkin init* inside. Create a package inside the src folder by using the command *catkin create pkg tutorial_1 –catkin-deps rospy std_msgs*. Study the *CmakeLists.txt and package.xml*-file. Inside the package create a folder called *scripts* and inside create two files called *keyboard_node.py* and central_node.py. At the beginning of each file put the line *#!/usr/bin/env python* to indicate for the interpreter that it is a python script. Make the files executable.
To compile, cd into the workspace and use the command *catkin build.* To run, source the workspace: source *~/ros/bioinspired_ws/devel/setup.bash*. Run the nodes by calling *rosrun tutorial_1 keyboard_node*

Task:
Implement the following simple response to keyboard input: When writing on the keyboard and pressing Enter, the *keyboard_node* should send a message containing the typed sentence. The *central_node* receives that message and displays its content on the terminal window. Visualize

the communication between the nodes by typing *rqt_graph* into the terminal window.
Use the provided ROS Survival Guide and the Tutorials: https://wiki.ros.org/ROS/Tutorials

Tipps:

- use *msg = raw_input('Type a message: ')* to read the keyboard.
- You can automatically source the ws by adding the line to the end of .bashrc in your home folder

There is nothing to hand in this time.