Tutorial 2, 22.04.2021 **BILHR 2021**

Goal:
In the previous tutorial, you learned about the basics of ROS. Based on that knowledge, the goal of this tutorial is to learn how to work with the NAO robot. You will send simple motion commands (position control) to the robot and process the received camera images.

Tasks:

1) Make sure you know enough about the following ROS basics: nodes, topics, messages, services, and basic commands (*e.g. catkin build*, *rosrun*, *roslaunch*, *etc.*). Check the ROS online documentation (http://wiki.ros.org) for further information.

2) Create a new package in *src* of your workspace called *tutorial_2*. Download the template file *central_node.py* from the repository and put it in the /*scripts* folder of the package. It implements a node that contains important callback functions needed for the messaging with the nodes of the NAO robot. Understand the callback functions of the template file, make the file executable to run it.

3) Learn how to use the ROS tools *rqt_graph* and *rqt_plot,* which visualize the nodes and signals, respectively.

4) Activate the NAO nodes by using the pre-installed package *nao_bringup*:

$*roslaunch nao_bringup nao_full_py.launch*.
The node for checking the tactile sensors (including bumpers) is located in a different package called *nao_apps*. Activate it by:
$roslaunch nao_apps tactile.launch.
Activate the *central_node* of your package by $*rosrun tutorial_2 central_node*.py You can visualize the communication between the nodes by using *rqt_graph.* Make a screenshot of the *rqt_graph* window showing the active nodes and their wiring.

5) Identify all joint names by looking at the */joint_state* topic

6) Make sure that the stiffness is disabled (via the function inside the node), set it in *central_node.py.* Plot the current positions of the head and both arms by using *rqt_plot.* While *rqt_plot* is plotting the joint positions, move the arms a little, and observe the effects. Make a screenshot of the plots of the 2 DOF shoulder and the 2 DOF elbow joints.

7) Write your own code: You can write your own code within the *central_node.py* file. Nevertheless, you can also write additional nodes. Make sure that any additional file you write is in the */scripts* folder of the package.

Implement the following task: Find a safe home position for the left arm. When touching tactile button (TB) 1, the NAO should move its left arm into that home position. When touching TB 2, the NAO should do a repetitive arm motion with his left arm, starting from the home position. Feel free to create any kind of repetitive arm motion, *e.g.* a waving motion. When touching TB 3, NAO's right arm should move in addition to its left arm, simultaneously mirroring the motion of its left arm.

8) Familiarize yourself with basic image processing using OpenCV of ROS (note that OpenCV is a part of ROS and thus already installed).

Helpful links:

https://wiki.ros.org/cv_bridge/Tutorials/ConvertingBetweenROSImagesAndOpenCVImagesPython
http://docs.opencv.org/doc/tutorials/tutorials.html
https://opencv-python-tutroals.readthedocs.io/en/latest/py_tutorials/py_tutorials.html

9) Consider the raw images delivered by NAO's top camera. Process them to extract red colour blobs. For this purpose, transform the image into HSV colour space. Take the largest colour blob and calculate the position of its center in pixel coordinates. Output this blob position on the terminal or via *rqt_plot*.

Results to submit:

1) Screenshot of task 4).

2) Screenshot of task 6).

3) Source code of task 7) and 9). *Source code !!* No binaries, *etc*.
   Note: **Comment** your code. Also, write **your names** at the beginning of the source code file.

4) Make a demo video showing the results of tasks 7) and 9).

Compress all the required results (1) to 4)) into a .zip or .tar.gz file.
Naming convention:
LastNameOfStudent1_LastNameOfStudent2_LastNameOfStudent3_T*number*.tar.gz

Try to keep the video size compact so that the email is not rejected from the mailserver (below 30mb). You can use Handbrake (https://handbrake.fr/) or ffmpeg to reduce the video filesize. Please keep the length to a maximum of 60 seconds, a minimum framerate of 25 Hz and a minimum resolution of 720p (1280x720).

Submit the file containing your results **before** the deadline to: **bilhr-lecture.ics@ei.tum.de**

Evaluation of your work:
Your work will be evaluated based on your results (required code and other documents).

Deadline: 05.05.2021, 23:59