Robot Operating System (for lab assistants)

How to setup Baxter

1 General commands on Baxter and ROS 1/2

The computers use the `ros_management_tools` to handle the ROS version and connection on another ROSMASTER (ROS 1).
Here are the main commands:

```
ros1ws # use ROS 1
ros2ws # use ROS 2
ros_baxter # ROS 1 uses Baxter's ROSMASTER
ros_turtle 1 # ROS 2 will connect to turtlebot1's ROS_DOMAIN_ID
ros_reset # ROS 1 uses local ROSMASTER, ROS 2 uses local domain ID only
```

All these commands only have to be typed once. They will be forwarded to any new terminal, and to QtCreator is it is open with the corresponding icon (”for ROS 1” / ”for ROS 2”).

2 Use in simulation

In the simulation, only ROS 2 is needed. Hence the first step is:

```
ros2ws && ros_reset
```

The simulation can be run with:

```
ros2 launch baxter_simple_sim sim_launch.py
```

3 Use on the real robot

The real robot first needs to be run once. Then, each student must run their own bridge between ROS 1 and ROS 2.

3.1 Running the robot

Baxter is a pure ROS 1 robot so running it is done on ROS 1:

```
# power on!
ros1ws && ros_baxter # use ROS 1 on Baxter's ROSMASTER
rosrun baxter_tools enable_robot.py -e # enable
rosrun baxter_tools tuck_arms.py -u # untuck the arms
```

3.2 Allowing (or not) several publishers on the same topic

On the ROS 1 side, if several people publish on the joint command topics they will all move the robot.
This can be controller on the ROS 2 side as `baxter_bridge` checks for a special parameter. In practice, **only the first lab** should allow all students to control the arm at the same time. To ensure this, type:

```
ros1ws && ros_baxter # so that your ROSMASTER is Baxter
rosparam set allow_multiple true
```

4 Powering off

```
ros1ws && ros_baxter # use ROS 1 on Baxter's ROSMASTER
rosrun baxter_tools tuck_arms.py -t # tuck the arms
rosrun baxter_tools enable_robot.py -d # disable
# power off by holding the power button for 2-3 s
```