Robot Operating System

Lab 2: Programming the “Puppet arm” node

1 Goals

Program a node that can move the left arm of the Baxter robot in a symmetric way with respect to the motion of the right arm (symmetry with respect to the sagittal plane of the robot). The node should be usable in position mode (the joint positions of the master arm are “ copied to ” the slave arm) or in velocity mode (the joint velocities of the master arm are “ copied to ” the slave arm).

2 Deliverables

After validation, the whole package should be zipped and sent by mail (G. Garcia) or through the lab upload form (O. Kermorgant).

3 Information

To create the control loop you must get the current *state* of the right arm and send *commands* to the left arm. To not hesitate to use RViz to compare the frames, some of the joints need to get the negative value of the other arm.

3.1 Tasks

- Identify the topics that the node should subscribe and publish to.
- Create a ROS package (`catkin create pkg <name> --catkin-deps <dependencies>`) with dependencies on `baxter_core_msgs`, `sensor_msgs` and `ecn_common`
- Draw the expected graph of the application.
- Program the node in C++ and/or Python

Once the package is created, feel free to use the provided `lab2_mirror.cpp` file as a template for the node code.

3.2 Avoiding conflicts between controllers

During this lab, all the groups will try to control the left arm and thus it is not advised that you all run your nodes at the same time. In order to avoid this, a tool is provided in the `ecn_common` package that will have your node wait for the availability of Baxter. The code in explained below:

C++: The token manager relies on the class defined in `ecn_common/token_handle.h`. It can be used this way:

```cpp
#include <ecn_common/token_handle.h>
// other includes and function definitions

int main(int argc, char** argv)
{
    // initialize the node
    ros::init(argc, argv, "my_node");

    ecn::TokenHandle token();
    // this class instance will return only when Baxter is available

    // initialize other variables

    // begin main loop
    while(ros::ok())
    {
        // do stuff


        // tell Baxter that you are still working and spin
        token.update();
        loop.sleep();
        ros::spinOnce();
    }
}
```

Python: The token manager relies on the class defined in `ecn_common.token_handle`. It can be used this way:

```python
#! /usr/bin/env python
from ecn_common.token_handle import TokenHandle
# other imports and function definitions

# initialize the node
rospy.init_node('my_node')

token = TokenHandle()
# this class instance will return only when Baxter is available

#initialize other variables

# begin main loop
while not rospy.is_shutdown():
    # do stuff


    # tell Baxter that you are still working and spin
    token.update();
    rospy.sleep(0.1)
```

With this code, two groups can never control Baxter at the same time. When the controlling group ends their node (either from Ctrl-C or because of a crash) the token passes to the group that has been asking it for the longuest time.
Remind the supervisor that they should have a `token_manager` running on some computer of the room.