MIDWA Lab 3
The « puppet hand» node
Using services and transformations

Goal and purpose

The goal of this lab is to program a node that can move the left arm of the Baxter robot in such a way that the left hand remains at a given distance from the right hand, with their `z` axes opposed.

The purpose is to teach you how to use ROS services and transformation listeners/broadcasters.

Deliverables

After validation, send a zip file of the full folder, including the PDF document to this address: **`salvador_8fb7@sendtodropbox.com`**

Create a single zip file with all deliverables inside. Naming convention will be **`Name1_Name2_packagename_code.zip`** where code is a random string of four characters to make sure your submission cannot be overwritten, e.g. « Smith_Baker_xyrt.launch ».

No need for subject or message body: the document goes straight to a dedicated folder of my Dropbox account.

How to reach the goal

One way to obtain the desired behavior of the « puppet hand » is to attach an additional frame to the right hand of the Baxter robot at the position and orientation where we wish to move the reference frame of the left hand.

In order to set the goal pose of the left arm the « **tf broadcaster** » broadcasts the transformation between the new frame and the base frame of the Baxter robot.

The controller node is a « **tf listener** »: it monitors the position and orientation of the target with respect to the base frame of the robot and calls a **service** (Baxter inverse kinematics service) to calculate the joint positions which allows reaching the goal when sending a « joint_command » topic for the left hand. To use a ROS service, the client node sends a **service request message** and waits for the corresponding **service response message**. Both messages are strictly typed data structures. You must check before starting to use the service that it is available. Moreover, you have to take into accound that maybe the service call is unsuccessful because there is no solution for that pose.

Informations

You will need to read about **tf listeners** and **tf_broadcasters** :

- http://wiki.ros.org/tf/Tutorials/Writing%20a%20tf%20listener%20%28C%2B%2B%29
- http://wiki.ros.org/tf/Tutorials/Adding%20a%20frame%20%28C%2B%2B%29

The service you will need to use is **Baxter's Inverse Kinematics Service**. Beware: ROS defines a more general IK service. Most information you will find online using a search engine is likely to be about the general ROS IK service, not Baxter's IK service.

Baxter's IK service messages are defined at:

- https://github.com/RethinkRobotics/baxter_common/blob/master/baxter_core_msgs/srv/SolvePositionIK.srv

You will need to write a service client, which is described at :

- http://wiki.ros.org/ROS/Tutorials/WritingServiceClient%28c%2B%2B%29

Some classes/methods you'll neet to have a look at :

- `tf::Quaternion`, and method `setRPY`
- `tf::Vector3`, and method `setValue`
- `tf::Transform`, and methods `setRotation`, `setOrigin`
- `tf::TransformBroadcaster`, and method `sendTransform`
- `ros::ServiceClient`, and methods `exists`, `call`, `isValid`

Tasks

- Define the nodes and topics that your application will use. Draw the node/topic graph of the application. Write down the main design choices you make to comply with the goal. Do that in a short PDF documentation file (you can use LibreOffice Writer for that) that you will put in the node folder. The node/topic graph should be in this PDF document.
- Program the node(s).
- Write a launch file to show that your application works and have the teacher validate your results.

How to manage the tests

- A single node (from one group) at a time will be controlling the robot.
- We will use a **token** to manage access to the robot. The team who have the token are allowed to run their node. Others wait until they get the token. The token is the T-rex figurine (handle with care: I'm attached to it!)
- **Any** failure of the test causes the token to pass to the next group.
- Tests have no reason to be long for this lab.
- Tests can only be run **using launch files**.

At the end of the lab session: backup your work, just in case. Contrary to PCs of the ECN platform, there is no automatic backup.