# Robot Operating System

Lab 3: the “puppet arm” node

## 1 Goals

In this lab, again the left arm will be controlled depending on the position of the right arm. This time, a constant 3D transform will be imposed between the left and right grippers.

### 1.1 On the real robot

If you are in P-Robotics room, you can do the lab on the real robot. You have to run the bridge in a ROS 1 terminal, then you can work on ROS 2.

```
source ~/ros/baxter.sh # so that your ROSMASTER is on Baxter
rosrun baxter_simple_sim ros2_bridge.py
```

You can move Baxter’s right arm by grabbing the wrist.

### 1.2 In simulation (including virtual machine users)

The simulation should be started in a ROS 1 console with:

```
roslaunch baxter_simple_sim simulation.launch lab:=puppet rviz:=false
```

You can then use only ROS 2 terminals.

## 2 ROS concepts

### 2.1 Publishing to a topic

In order to control the left arm, you need to publish a command message on the suitable topic (as in the lab 2)

### 2.2 Using TF

In ROS, the `/tf` topic conveys many 3D transforms between frames, forming a tree. A TF listener can be instantiated in a node in order to retrieve any transform between two frames.

For `/tf` to get all the transforms for Baxter, a `robot_state_publisher` must be run. It is a standard node that:

- loads the description of a robot (URDF file)
- subscribes to the joint states topic
- publishes all induced 3D transforms with the direct geometric model

A custom launch file already exists to do this and can be run with:

```
ros2 launch baxter_description baxter_state_publisher_launch.py
```

## 2.3 Running RViz2

A configuration file for RViz2 is available in `lab3_puppet/launch`. Go in this directory and run:

```
rviz2 -d ./config.rviz
```

This configuration runs RViz2 already configured for Baxter. Note that all topics from the real robot in ROS 1 are not bridged: no image, sonar or range sensors.

## 2.4 Adding a new frame

In our case, the desired pose of the left gripper will be defined as a new frame, relative to the right gripper and called `'left_gripper_desired'`.
The node `static_transform_publisher` from the `tf2_ros` package is designed to publish this kind of fixed transform between frames:

```
ros2 run tf2_ros static_transform_publisher x y z yaw pitch roll frame_id child_frame_id
```

where:

- `frame_id` is the reference frame (here `right_gripper`)
- `child_frame_id` is the target frame (here `left_gripper_desired`)
- `x y z yaw pitch roll` is the 3D transform (here `0 0 0.1 0 3.14 0`)

## 2.5 Putting all together

A single launch file should be written in order to regroup the previous commands:

- run the node `static_transform_publisher` with the correct arguments
- include the launch file `robot_state_publisher_launch.py` from the `baxter_description` package
- run RViz with the argument `-d (path to config.rviz from the lab3_puppet package)`

With the simulation and your launch file running, check that you can indeed retrieve the current 3D transform between the frames `base` and `left_gripper_desired`:

```
ros2 run tf2_ros tf2_echo base left_gripper_desired
```

# 3 Tasks

- Identify the topics that the node should publish to: names, message type
- Identify the services that the node need in order to convert 3D transform to joint positions: names, service type
- Check the online documentation "ROS 2 C++ services" to get the overall syntax. This work requires at least a publisher, a timer, tf, and service client
- Program the node in C++ (and then in Python3 if you feel like it)

The package is already created for this lab (`lab3_puppet`), you just have to update the C++ file and compile it.
Feel free to keep this package as a template for future packages / nodes that you will create.