Exercise Session 2

Theory

- ROS package structure
- Integration and programming with VSCode
- ROS C++ client library (roscpp)
- ROS subscribers and publishers
- ROS parameter server
- RViz visualization

Exercise

In this exercise, you will create your first ROS package. The package should be able to subscribe to a laser scan message from the SMB robot and process the incoming data. This node will be the basis for the next exercises. Use VSCode to edit your package (Lecture 2 Slides 9-13).

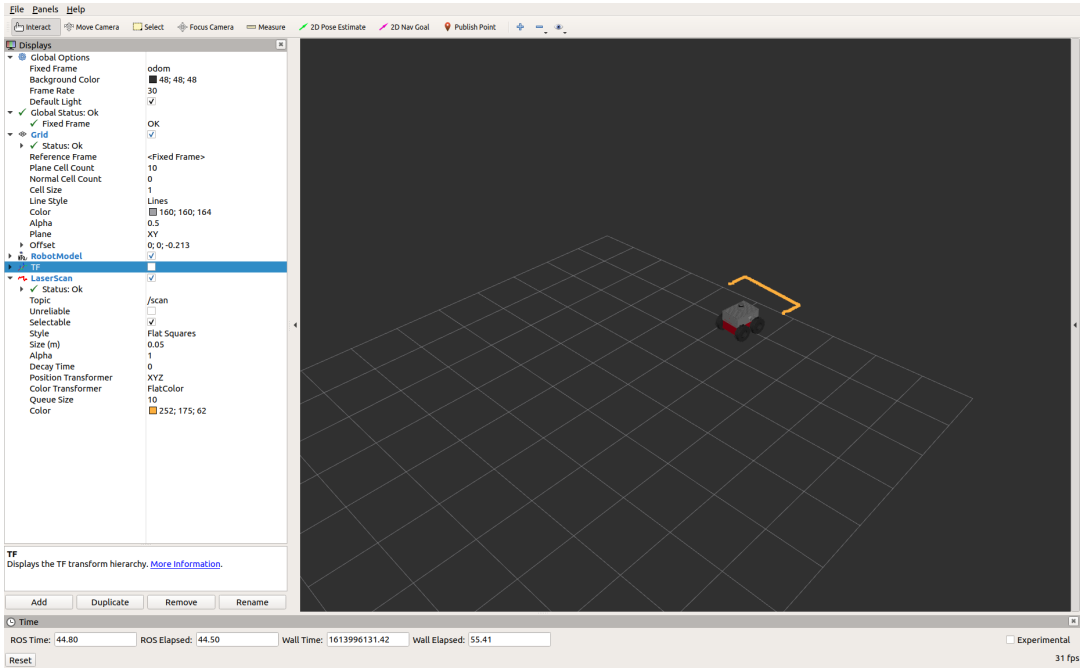
Make sure to look at the ROS template for reference https://github.com/leggedrobotics/ros_best_practices. It will greatly help you with the implementation, as it has a node similar to what you have to do in this exercise!

1. **OPTIONAL** (more difficult): Create the package `smb_highlevel_controller` from scratch. You can use the command `catkin_create_pkg` to create a new package with the dependencies `roscpp` and `sensor_msgs`.

2. **OR** (easy): Download the Zip archive containing prepared files of the package `smb_highlevel_controller` from moodle.

3. Inspect the `CMakelists.txt` and `package.xml` files. (Lecture 2 Slides 5-7)
4. Create a subscriber to the `/scan` topic. (Lecture 2 Slides 19-21)
5. Add a parameter file with the topic name and queue size for the subscriber of the topic `/scan`. (Lecture 2 Slides 22-23)
6. Create a callback method for that subscriber which outputs the smallest distance measurement from the vector ranges in the message of the laser scanner to the terminal. **The callback method has to be implemented as a class member.** Inspect the message type here http://docs.ros.org/en/api/sensor_msgs/html/msg/LaserScan.html
7. Add your launch file from Exercise 1 to this package and modify it to:
    - run the `smb_highlevel_controller` node.
    - load the parameter file.
8. Pass the argument `laser_enabled` from your launch file to the `smb_gazebo.launch` file with the value `true`.

9. Show the laser scan in RViz and add RViz to your launch file. Make sure to set *odom* as the *Fixed Frame* (under *Global Options*) and adapt the size of the laser scan points. You can save your current RViz configuration as the default configuration by pressing ctrl+s. (Lecture 2 Slides 24-26)
10. [OPTIONAL] Check the *pointcloud_to_laserscan* node, and find out what it is doing. Which topic is it publishing on and which is it subscribing on? Visualize the 3D point cloud and the laser scan in Rviz.
11. [OPTIONAL] Create an additional subscriber to the 3D point cloud and print how many points it has.



RViz visualization of a single laser scan. Multiple obstacles are placed around the robot. Note the changed “Fixed Frame” as well as “Size (m)”.

Evaluation

- ❏ Start the launch file and drive around with SMB. There should be changing output from the laser scanner in the terminal. [40%]
- ❏ Check if the node is implemented as the template suggests. [30%]
- ❏ Is a parameter file used? [15%]
- ❏ Is the laser scan visualized in RViz as shown in the image? [15%]

OPTIONAL

- ❏ Correctly explain what the *pointcloud_to_laserscan* node is doing. Is the 3D point cloud changing as the robot moves? [10% bonus]
- ❏ Is the number of points inside the cloud shown in the terminal? Is the callback implemented correctly? [10% bonus]