Assignment 2

DEADLINE: 31th January 2023 at 23:59 (CET)

In this assignment, you have to implement in an assistance robot (namely a Tiago by PAL Robotics) a fetch and delivery behavior for everyday objects.

In the simulated environment there are 7 objects on a pick-up table: A red cube, a green triangle and a blue hexagon, plus 4 gold hexagons (Figure 1 and Figure 2). The gold hexagons are obstacles, and Tiago must not collide with them (or with the table) while grasping the required objects. Once an object is grasped it must be fetched and placed on the cylindrical table of the same color (see Figure 3).

The human user is represented by a ROS node (already implemented for you) which generates the sequence of delivery of the coloured objects.

The pose of the objects on the pick-up table can be obtained by exploiting the Tiago's camera and the AprilTag library thanks to the markers placed on the top of these objects. The AprilTag library identifies the markers through the IDs encoded in the QR codes. Each marker has a specific identifier. Finally, the shape of every object is known (see Figure 1).

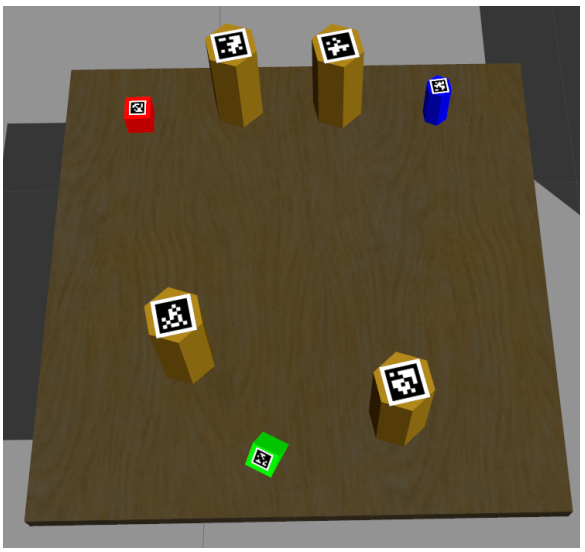


Figure 1

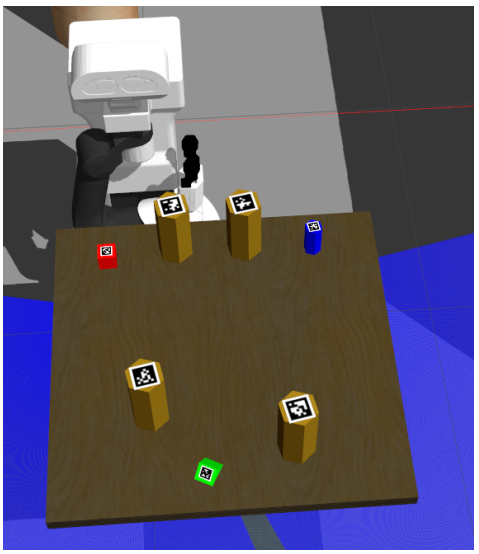


Figure 2

You have to develop a routine, exploiting the MoveIt! library, to pick and place each object (according to the received sequence) from the pick-up table (Figure 1 and 2) to its final destination in accordance with its color (i.e., the red cube on the red cylinder table) (Figure 3).

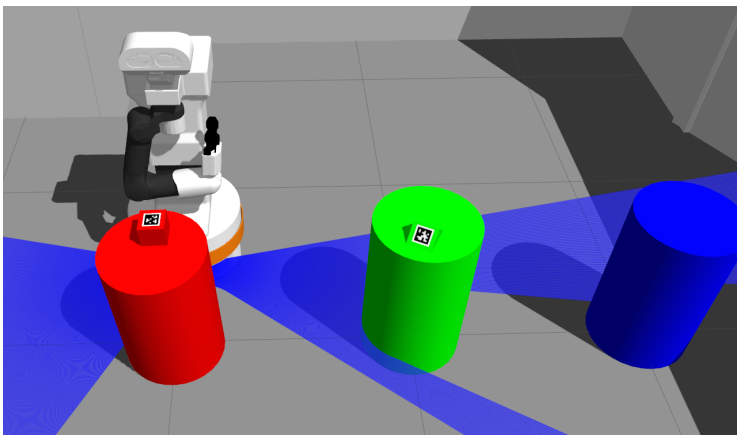


Figure 3

The whole environment is shown in Figure 4.

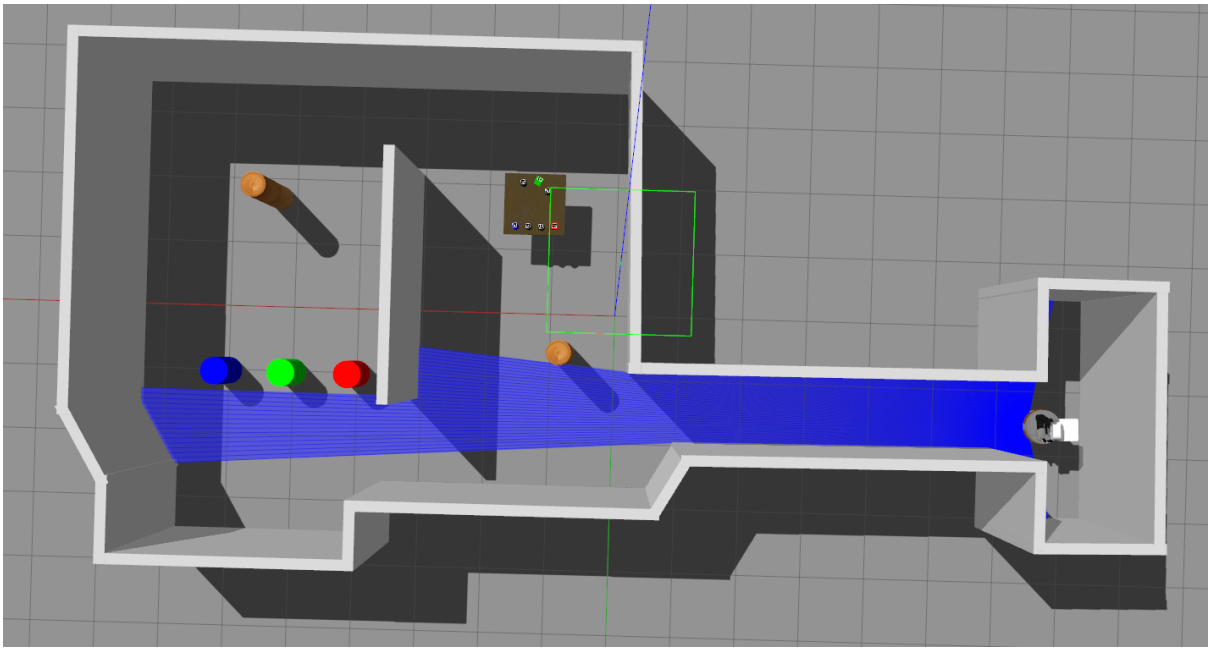


Figure 4

To navigate in the environment and to move from the pick-up table to the delivery tables, you must use the navigation module you developed in the previous assignment.

The assignment is fulfilled if, in the end, the three coloured objects are delivered on top of the corresponding coloured table in the assigned sequence.

SUGGESTION:

We suggest the following pipeline to solve the assignment:

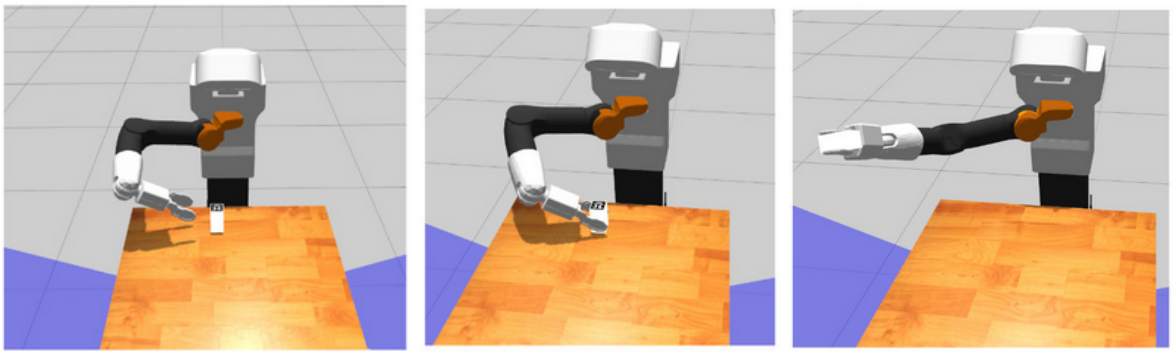
1. Navigate in front of the table with the navigation stack (Figure 2). The position in front of the table is up to you.
2. Use the AprilTag ROS package and Tiago's camera to detect the objects in front of Tiago.
3. Define a collision object for every object you detect and for the table. For the hexagons, you can create a collision object in the shape of a cylinder. The collision

object helps you (i.e. the module MoveIt) to generate a collision-free trajectory. Traditionally, to ensure safety, the collision object is created a little larger than the real object.

4. Assign an initial configuration to the Tiago's arm. With MoveIt you can move only the arm or both torso and arm. (see Tiago Tutorial and MoveIt Tutorial)
5. Make the arm move in a target position. The target position is set at *n* cm above the marker position (align the gripper with the centre of the marker itself) (Figure 5a).
    a. **N.B:** This is true if you make a grasp from the top of the object. But, especially for the hexagon, you can grasp it from the side. In this case, knowing the hexagon position, you can generate a grasping point, i.e. target position, on its side.
6. Through a linear movement complete the grasping. This sub-sequence from 4-6 should help you to implement the correct grasping of the objects (Figure 5b).
7. Remove the target object from the collision objects.
8. Attach virtually the object to the gripper (use arm_7_link) using the appropriate Gazebo plugin: Gazebo_ros_link_attacher (See the section **Instruction to start the homework**).
   (This is a trick to avoid singularities in the physical engine of the GAZEBO simulator which is not good at simulating the friction and so it might happen that the robot grasps the object and the object slips away from its fingers.)
9. Close the gripper.
10. Come back to the previous target position through a linear movement of the arm.
11. Move the arm to an intermediate pose (Figure 5c).
12. Move the arm to a secure pose to avoid collisions before navigate to the destination (i.e. fold the robot's arm close to the robot's body).
13. Navigate to the correct destination (Figure 3). You can decide the final pose of the robot in front of the cylindrical table and use the navigation module to navigate to that point.
14. Place the object on the top of the table.
15. Open the gripper.
16. Detach the object via the Gazebo plugin.

N.B.: We know that with Gazebo there are many problems with the attachment of objects. Thus, you can attach the object only in RVIZ and complete the task. In this case, in Gazebo, you can remove the object manually.

If you have problems with open/close and attach/detach, do not worry, leave these routines in your code, even if they are not performing properly. In the evaluation, we will consider if they are placed in the right place of the pipeline and not if they generate strange behaviors of the objects. Please, specify in the report which problems you encountered anyway.



a b c

Figure 5: Example of the pick routine.

MANDATORY:

Your code must be implemented using the following structure:

- Create a node (A) that sends a request to the human_node using a service. The latter is a node that simulates a human who tells the robot the sequence to pick up the objects. Thus, the human_node returns the object's ID that Tiago needs to pick. We provide the file .srv and the human_node that handles the request and sends the response.
- Use your node (A) to call the Action Server of HW1 and navigate to a position in front of the table.
    - N.B: Since you have only three pickable known objects, you can configure 3 global positions in the front of the table to help you to detect the effective position of the objects on the table (Figure 6).
- Implement two new nodes (B, C) to handle the manipulation of the objects and the detection. The communication with these nodes can be developed using messages, service or Action/Service infrastructure. It is up to you to choose the strategy you think is most appropriate.
    - We suggest you implement object detection in another node (i.e., node B). Using the AprilTag, you know the pose of the objects with respect to the camera reference frame. Then, using the TF you have to transform the pose from the camera frame to the robot frame.
    - With the other node (C), we suggest you implement the MoveIt routine.

N.B: Take advantage of the modularity and scalability of ROS.

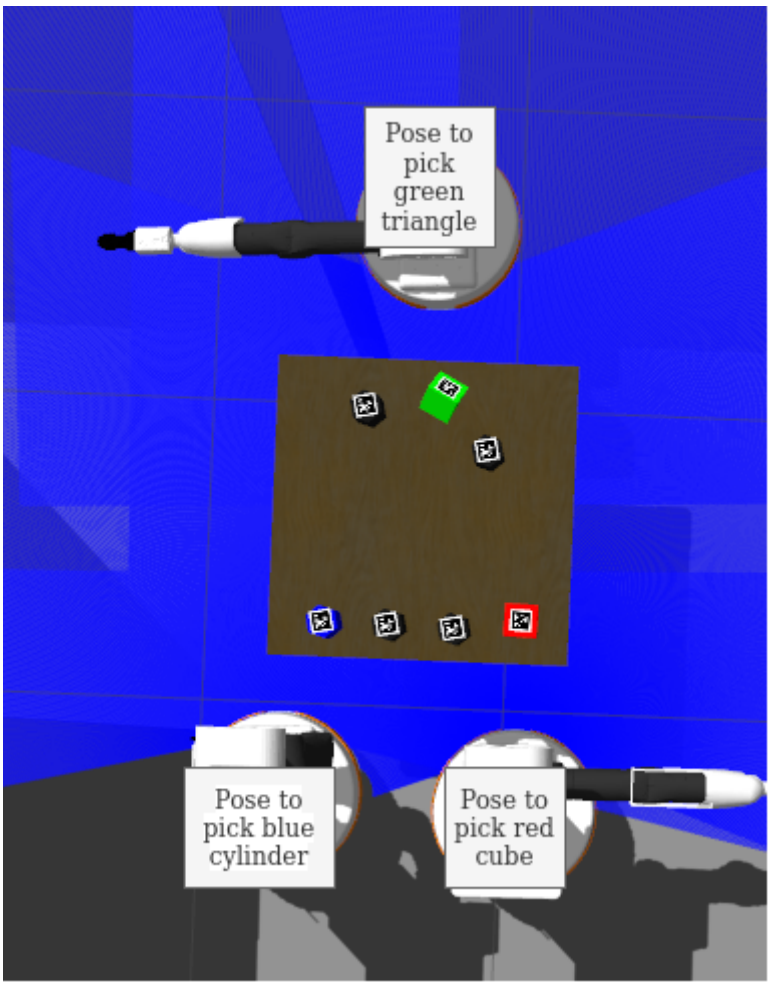


Figure 6

Extra points

Use the laser data to select the pose for the docking routine in front of the cylindrical tables during the object place phase.

In this case, you have to specify in the report that you also implemented this optional solution.

Instruction to start the homework

Follow these instructions to install the environment and start the homework:

For Local ROS Installation

- If you work on a local machine, clone the following repositories in your workspace (src folder) and compile the workspace before going on:

    - Apriltag: https://github.com/AprilRobotics/apriltag.git
    - Apriltag_ros: https://github.com/AprilRobotics/apriltag_ros.git
    - Gazebo_ros_link_attacher: https://github.com/pal-robotics/gazebo_ros_link_attacher.git

For VLAB

- Clone the following repository Gazebo_ros_link_attacher: https://github.com/pal-robotics/gazebo_ros_link_attacher.git

```
$> cd ~/catkin_ws/src
$> git clone https://github.com/pal-robotics/gazebo_ros_link_attacher.git
$> cd .. && catkin build
$> source ~/catkin_ws/devel/setup.bash
```

For everybody

- Go inside the previous workspace

```
$> cd ~/catkin_ws
$> source ~/catkin_ws/devel/setup.bash
```

- Create a new package

```
$> cd src
$> catkin_create_package assigment2_package_name
```

- Start the simulation and MoveIt:

```
$> roslaunch tiago_iaslab_simulation start_simulation.launch world_name:=ias_lab_room_full_tables
```

- AprilTag:

```
$> roslaunch tiago_iaslab_simulation apriltag.launch
```

- Navigation stack:

```
$> roslaunch tiago_iaslab_simulation navigation.launch
```

- Human Service node:

```
$> rosrun tiago_iaslab_simulation human_node
```

NOTE: given the complexity of the pipeline, we advise you to prepare a single launch file that calls all the necessary nodes and other launch files. This will also speed up the development. Have a look at the ROS documentation if you don't know how to do it.

How to submit your solution

To submit your solution, you **must** do the following:

1. Setup your group repository

Use the same repository you set up for the Assignment 1. The repository must contain only the packages you developed for the assignments. Please, **use different packages for Assignment 1 and 2**.

2. Start coding

a. **Push your code (with good comments)** into your group's git repository. Code explainability will be part of the evaluation criteria.
b. We encourage you to commit the code as you write it and not in one block
c. **Add a README.md file** which contains the necessary commands to correctly execute and test your code (e.g. *roslaunch* and *rosrun* commands). Make sure the instructions are working properly. **The first lines of the README.md must contains the following informations**:

```
GROUP XX
Member1's name, email
Member2's name, email
Member3's name, email (if any)
```

You can create a new README.md inside the Assignment 2's package or update the previous README.md, just **clearly distinguish instructions to run Assignment 2 nodes**.
d. We suggest you to test your code step by step in a new clear workspace, cloning the code from your repository in order to be sure that your code is executable by third parties.
e. **Make sure your code is compiling before submitting**. Not compiling codes will be penalized in the final grade.

3. Submission

1. **You are allowed to push the code in the repo until the submission on Moodle**. Later commits will not be considered for the evaluation.
2. To get the maximum grade, you have to submit it within the 31th January 2023 at 23:59 (CET). Later submissions will be penalized.
3. **Prepare a brief PDF report (max. 2-3 pages)** that explains your solution. The report is supposed to explain the high-level ideas, strategies or algorithms you implemented to solve the assignment. **The first lines of the report must be like**:

```
GROUP XX
Member1's name, email
Member2's name, email
Member3's name, email (if any)
LINK TO THE REPOSITORY
LINK TO ADDITIONAL MATERIAL (see point 5)
```

4. **Prepare a short video** (e.g., screen recording) that shows the execution of your software by the robot. This is necessary in case we cannot easily run your code.
5. **Report and video must not be placed in the repository**. We suggest you create a Google Drive folder, upload the video or any additional material and share with us the link to the folder inside the report.
6. We will open a submission to Moodle. You will be able to attach the report to your submission (be sure that it contains all the needed information such as a link to the repo, link to the video, additional info, etc.). **The name of the submission must be as follows: GroupXX_A2**. Please submit only once per group, one member can submit for everybody.

Additional Information

APRILTAG: We provide an example of the AprilTag marker (Figure 7) and the set of IDs and corresponding frame_ids (Table 1).

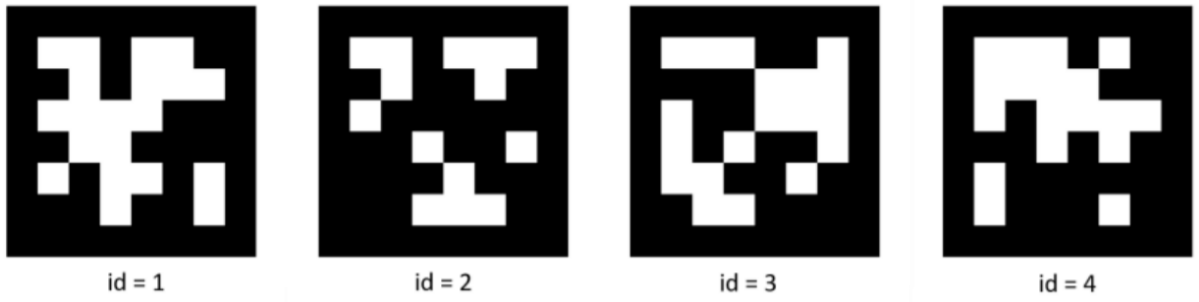


Figure 7

ID	FRAME_ID
1	blue_hexagon
2	green_triangle
3	red_cube
4	gold_obs_0
5	gold_obs_1
6	gold_obs_2
7	gold_obs_3

Table 1

HEAD ACTION: To detect the objects on the table you have to move the head of Tiago. See these two tutorials:

- http://wiki.ros.org/Robots/TIAGo/Tutorials/motions/head_action

- http://wiki.ros.org/Robots/TIAGo/Tutorials/trajectory_controller

Also, rqt_joint_trajectory_controller can be helpful to find the best position of the head. Run the following command:

```
$> rosrun rqt_joint_trajectory_controller
rqt_joint_trajectory_controller
```

GRIPPER OPEN/CLOSE: To open/close the gripper you can use the gripper controller interface. See http://wiki.ros.org/Robots/TIAGo/Tutorials/trajectory_controller

RVIZ: In RVIZ you can test MoveIt using the MotionPlanning (Figure 8)

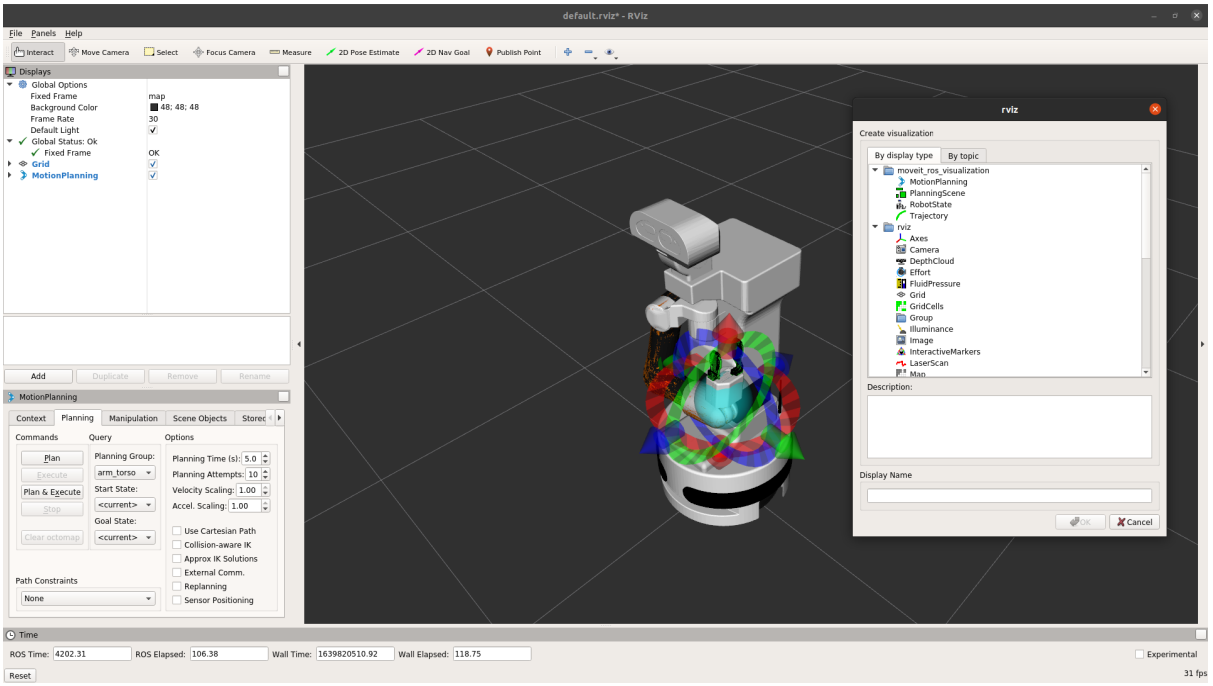


Figure 8

TF: Using rqt you can see the TF Tree (from terminal digit rqt, then in the GUI search plugins-visualization-TF Tree) (Figure 9)

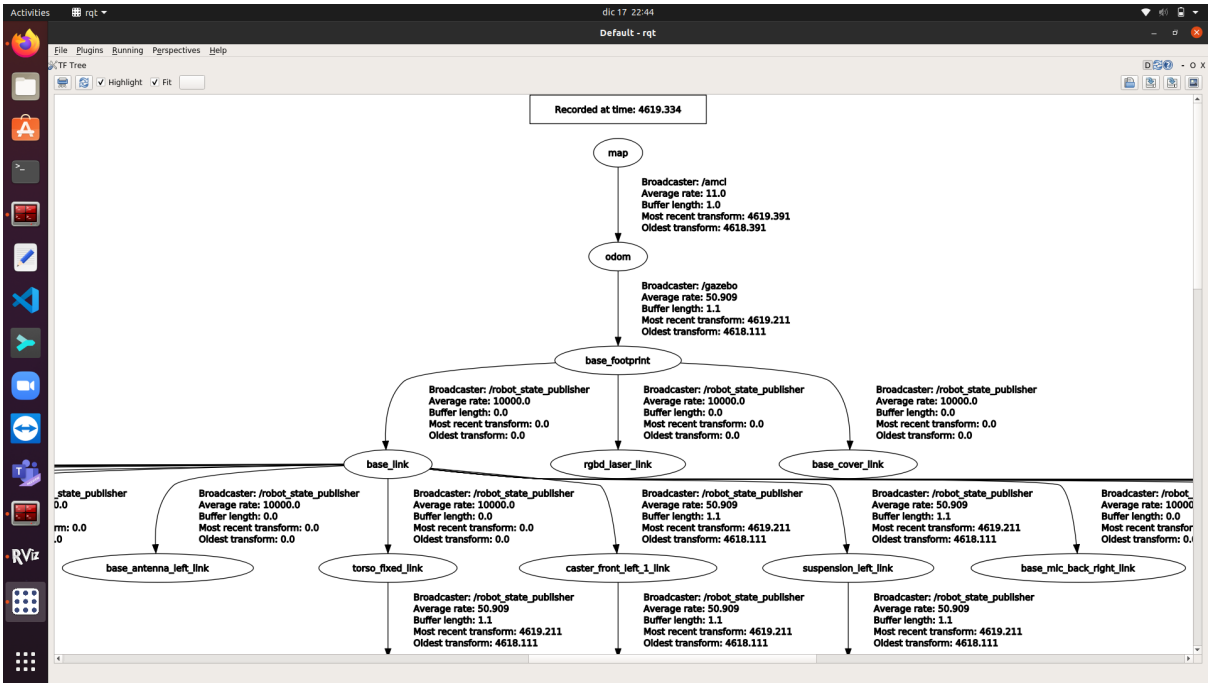


Figure 9