SSY236

Assigment 03: Perception, Reasoning & Control

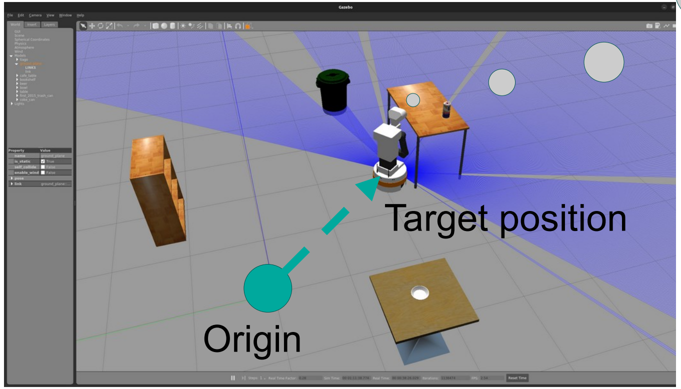
Karinne Ramirez-Amaro
Emmanuel Dean

Assignment 03

In this assignment you can accumulate a maximum of 12 points

Goal: The robot should save and retrieve its knowledge base of all the seen objects from the Gazebo environment. Then, the robot should navigate to a desired target object provided by the user

Saving the position of the table



Assignment 03

1) Open Docker

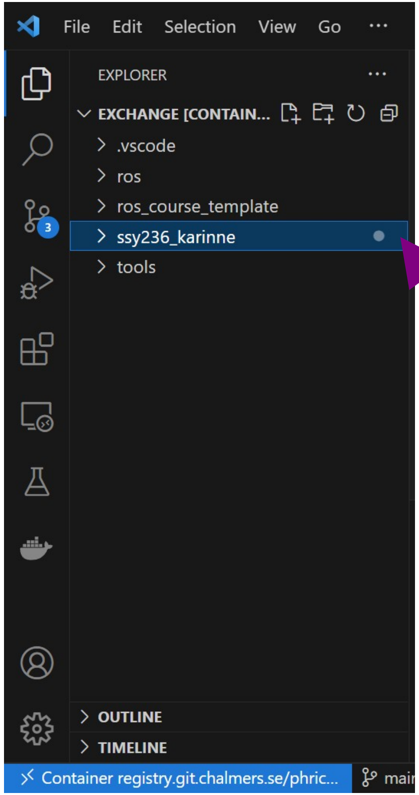
2) Run the container

3) Open and configure Xlaunch

4) Open vscode

   - Attach the vscode to the container

Assignment 03



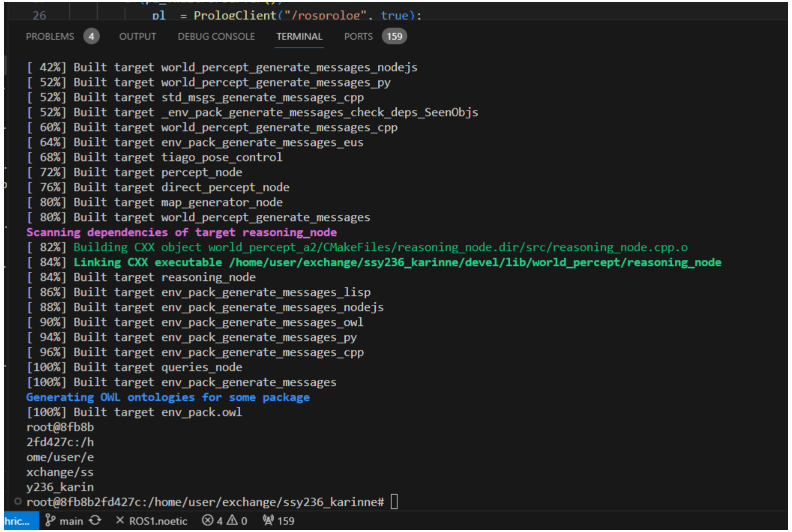
1) Download the folder “assignment_3.zip” from Canvas → Study week 4 → Assignment

2) Unzip the file. Inside this folder you will find a folder “world_percept_assig3”

3) In your vscode, move to your exchange folder in the “EXPLORER” menu. Then, copy the folder “world_percept_assig3” inside your src folder, for example:

- ssy236_karinne
  - src
    - world_percept_assig3

Assignment 02



```
pl  = PrologClient("/rosprolog", true);
```

```
[ 42%] Built target world_percept_generate_messages_nodejs
[ 52%] Built target world_percept_generate_messages_py
[ 52%] Built target std_msgs_generate_messages_cpp
[ 52%] Built target _env_pack_generate_messages_check_deps_SeenObjs
[ 60%] Built target world_percept_generate_messages_cpp
[ 64%] Built target env_pack_generate_messages_eus
[ 68%] Built target tiago_pose_control
[ 72%] Built target percept_node
[ 76%] Built target direct_percept_node
[ 80%] Built target map_generator_node
[ 80%] Built target world_percept_generate_messages
Scanning dependencies of target reasoning_node
[ 82%] Building CXX object world_percept_a2/CMakeFiles/reasoning_node.dir/src/reasoning_node.cpp.o
[ 84%] Linking CXX executable /home/user/exchange/ssy236_karinne/devel/lib/world_percept/reasoning_node
[ 84%] Built target reasoning_node
[ 86%] Built target env_pack_generate_messages_lisp
[ 88%] Built target env_pack_generate_messages_nodejs
[ 90%] Built target env_pack_generate_messages_owl
[ 94%] Built target env_pack_generate_messages_py
[ 96%] Built target env_pack_generate_messages_cpp
[100%] Built target queries_node
[100%] Built target env_pack_generate_messages
Generating OWL ontologies for some package
[100%] Built target env_pack.owl
root@8fb8b
2fd427c:/h
ome/user/e
xchange/ss
y236_karin
root@8fb8b2fd427c:/home/user/exchange/ssy236_karinne# 
```

1) Make sure that you source the right workspace:

2) source /knowrob_ws/devel/setup.bash

3) Make sure you have included the new package “world_percept_assig3”

4) Remember to CATKIN_IGNORE your previous packages

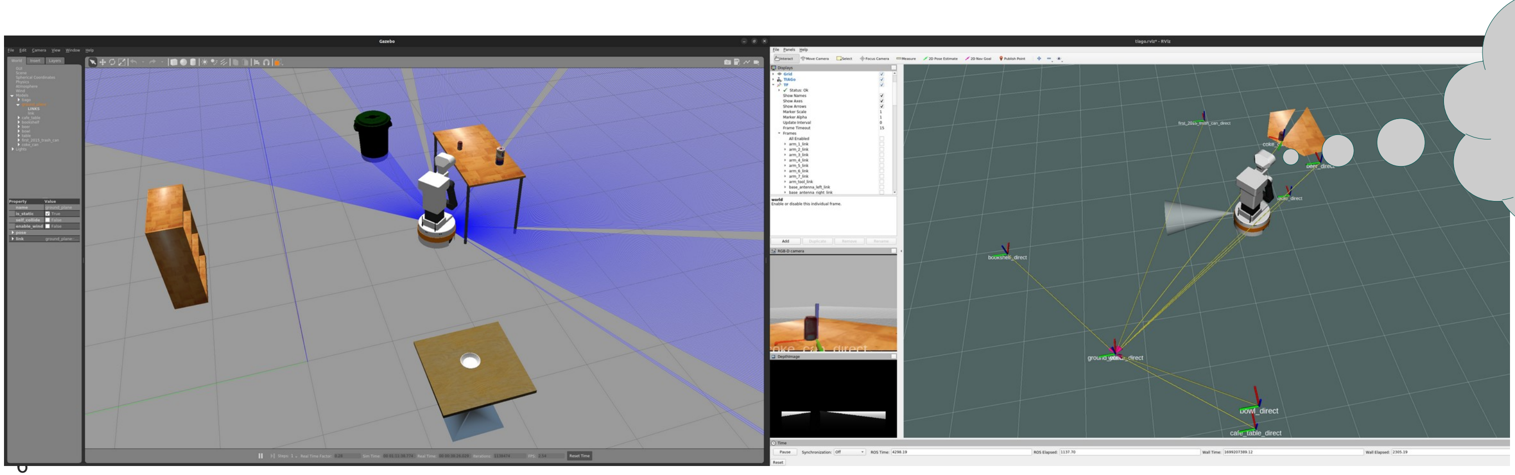
5) Then, simply compile:

6) catkin_make

Assignment 03 – Task 1

Recap from Assignment 01 & 02. The robot explores the environment and discovers the name and positions of new objects in the scene! Then, the robot is able to store into its knowledge the information of the seen objects.

However, every time we stop the program, the robot forgets everything!



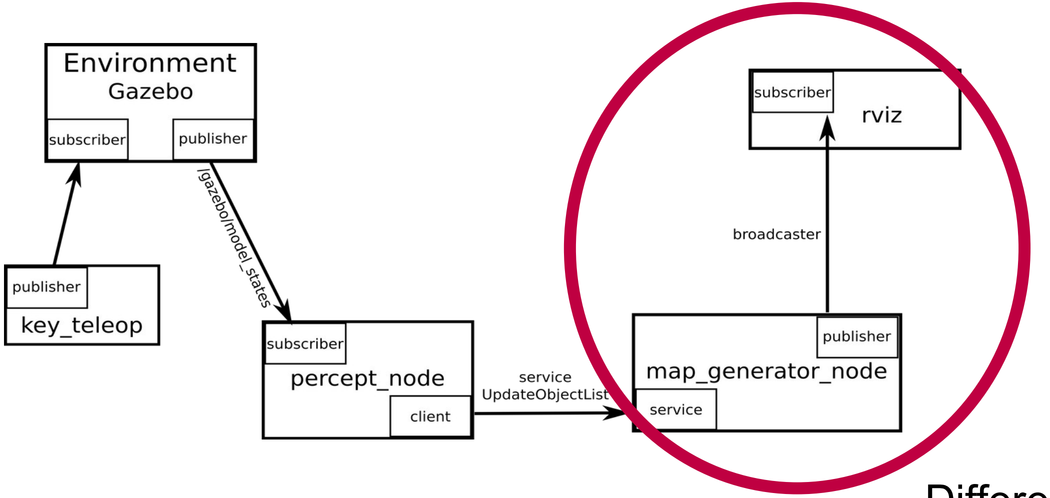
Saving that there is a table in the environment

Assignment 03 – Task 1

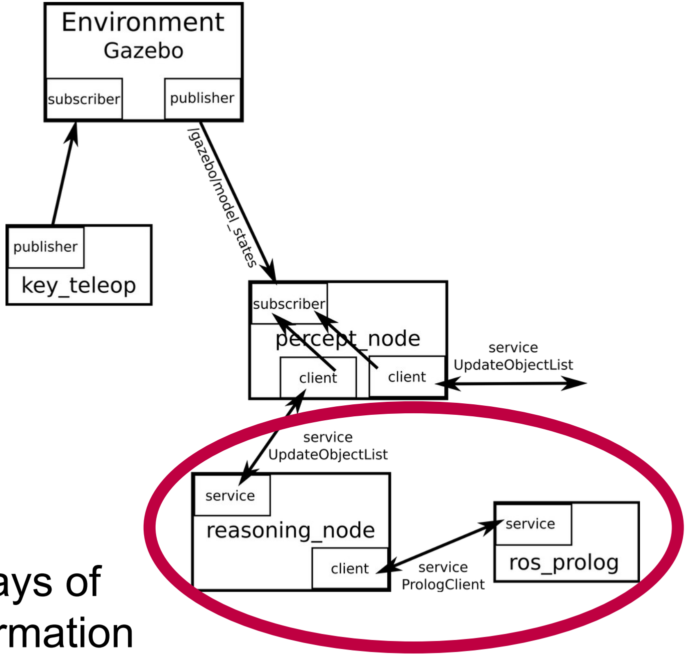
Note: For this assignment you need to source:

```
source /knowrob_sw/devel/setup.bash
```

Assignment 01

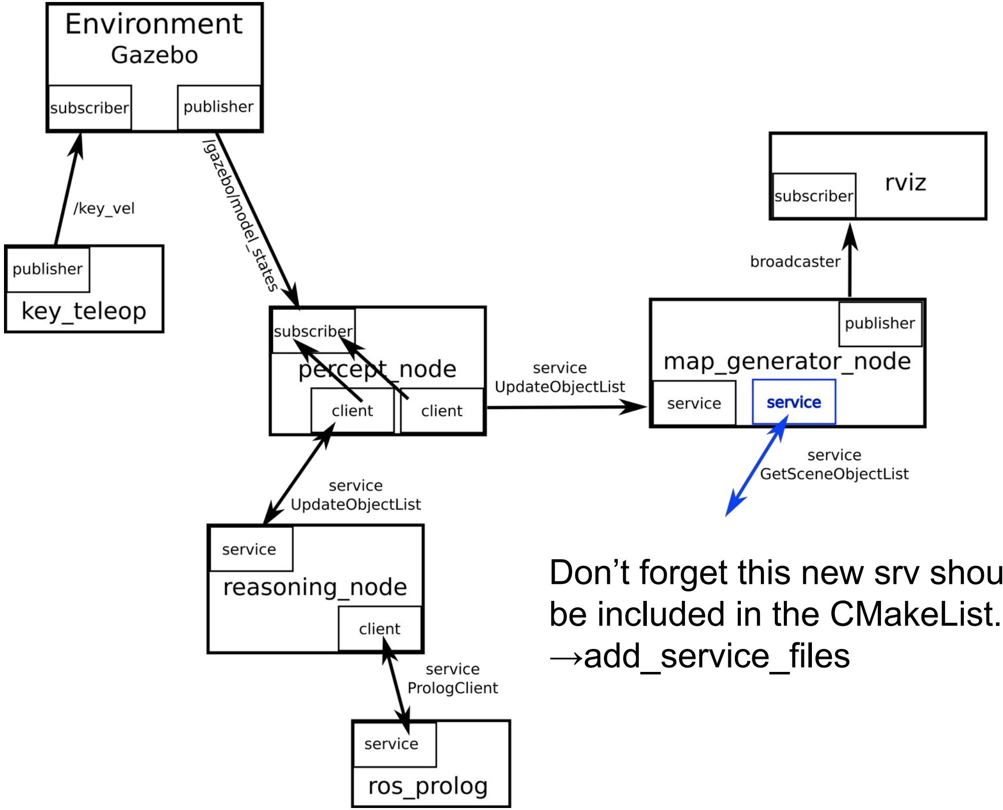


Assignment 02



Different ways of storing information

Assignment 03 – Task 1



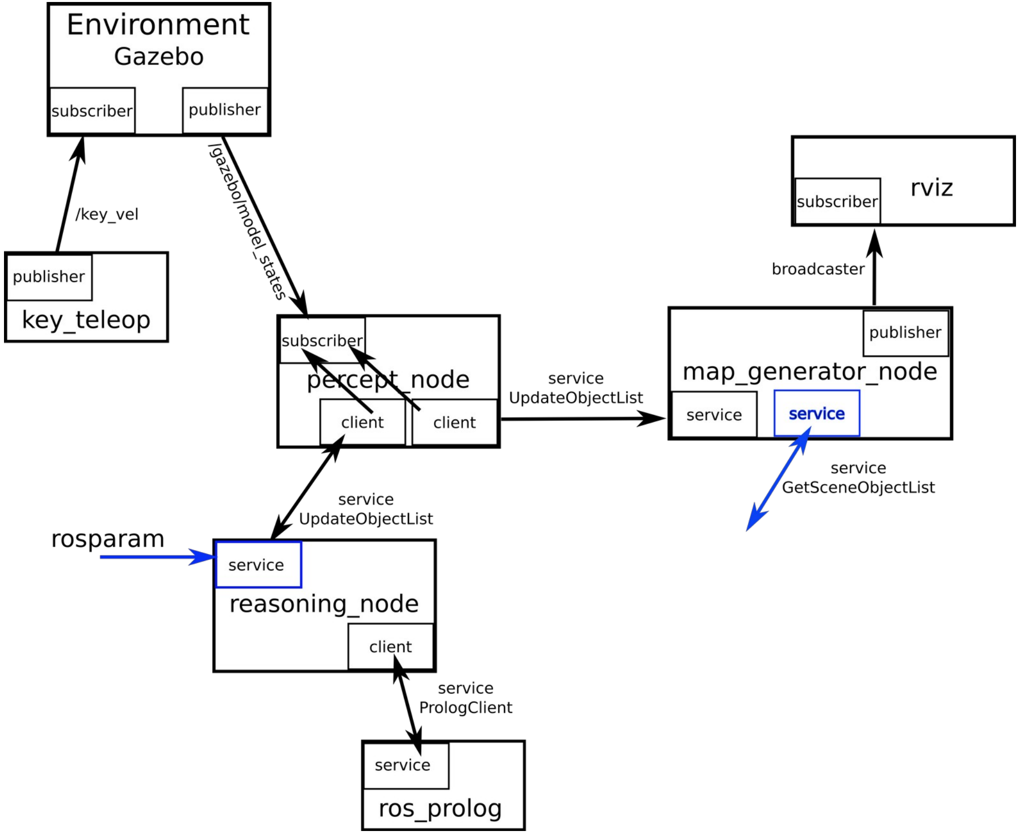
Don’t forget this new srv should be included in the CMakeList.txt →add_service_files

A03.T01. The goal of this task is to modify the new service inside the node “map_generator_node” (1 pt)

- First we, create a new srv file called “GetSceneObjectList.srv”
- We define the service variables inside the class MapGenerator as “private variables”
- We define the new service name and advertise the new service in the constructor
- Modify the callback function “srv_get_scene_obj_callback” → look for the #TODO A03.T01

http://wiki.ros.org/ROS/Tutorials/CreatingMsgAndSrv

Assignment 03 – Task 2



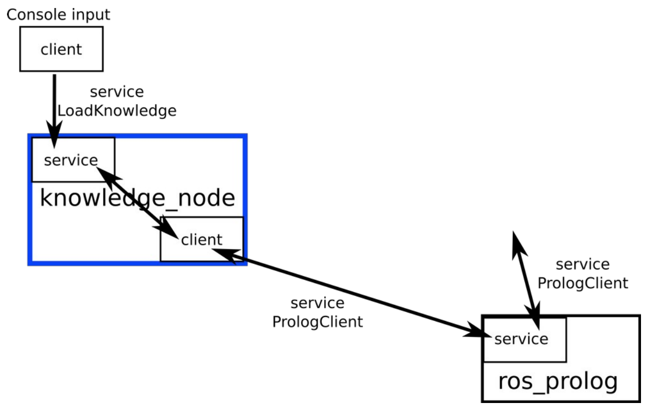
A03.T02. The goal of this task is to use “rosparam” inside the node “reasoning_node” (1 pt)

- First we, create a new yaml file inside the folder “config” called “loadKnowledge.yaml”
- Look into the TODOS inside the main function (0.6 pts)
- Create a new function “setOutQueriesFiles” to create and open a new file. This new function needs to be “public” (0.4 pts)

http://wiki.ros.org/rosparam

http://wiki.ros.org/roscpp/Overview/Parameter%20Server

Assignment 03 – Task 3



Don’t forget this new srv should be included in the CMakeList.txt →add_service_files

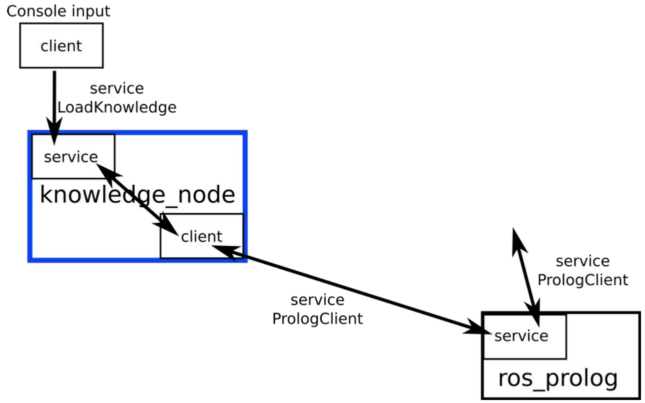
A03.T03. The goal of this task is to create a new node called “knowledge_node” (2 pt)

1) First, create a new srv file called “loadKnowledge.srv” (0.25 pts)
   - Request is an integer called “start”
   - Response is a boolean called “confirm”
2) Create a new c++ file called “knowledge_node.cpp”
   - The main function (0.5 pts):
     - should open the same yaml file as the one in the node “reasoning_node”. This main function is also expecting one value as input argument (similar to the reasoning_node)
     - Should call a function “void setQueryFile (std::string fileName_Q)” to open a file if it exists, if not just print “File not found and exit the function” This function does not have a returning argument.

http://wiki.ros.org/rosparam

http://wiki.ros.org/roscpp/Overview/Parameter%20Server

Assignment 03 – Task 3



3) Define the variables needed to advertise the service, e.g. define the name of the service “srv_load_knowledge_name_” and the variable used when advertising the service. Then, advertise the service (0.25 pts)

4) Now create a PrologClient to the node /rosprolog. Similarly to the one we did inside the node “reasoning_node.cpp”

5) Create the callback function (0.25 pts) similar to:

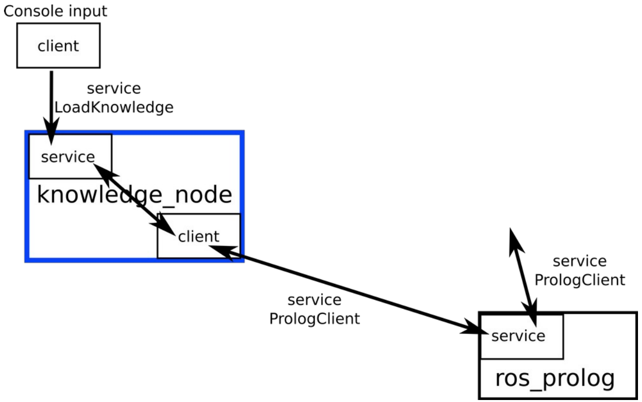
```
    bool callback_load_knowledge(world_percept::LoadKnowledge::Request
&req,world_percept::LoadKnowledge::Response &res)
  { }
```

When this service is call you need to load the prolog queries inside the open file. This means that inside this callback function you should call a function call “loadQueries” when the service is called

http://wiki.ros.org/rosparam

http://wiki.ros.org/roscpp/Overview/Parameter%20Server

Assignment 03 – Task 3



6) Create the function void loadQueries(){} (0.75 pts).

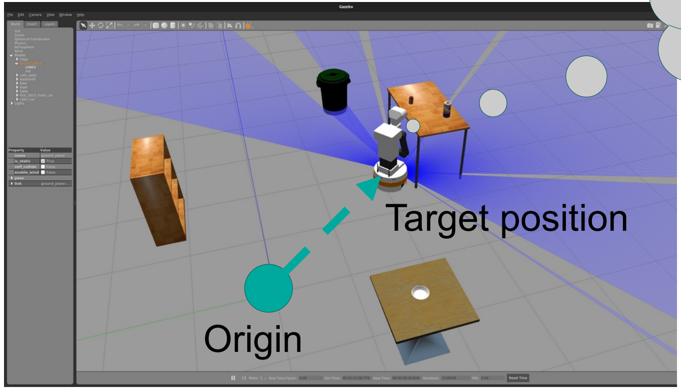
- This function should read each line of the loaded file until the end of the file.
- When a new line is found the line is send to the clientProlog to assert this line (query) inside the knowledge base (Hint: Take a look inside the “reasoning_node”, we use a similar solution there.)

Hint: You can test this node from the console as follows:
Terminal 01: rosrun world_percept knowledge_node my/path/to/prolog/file
Terminal 02: rosservice call /load_knowledge "start: 1"

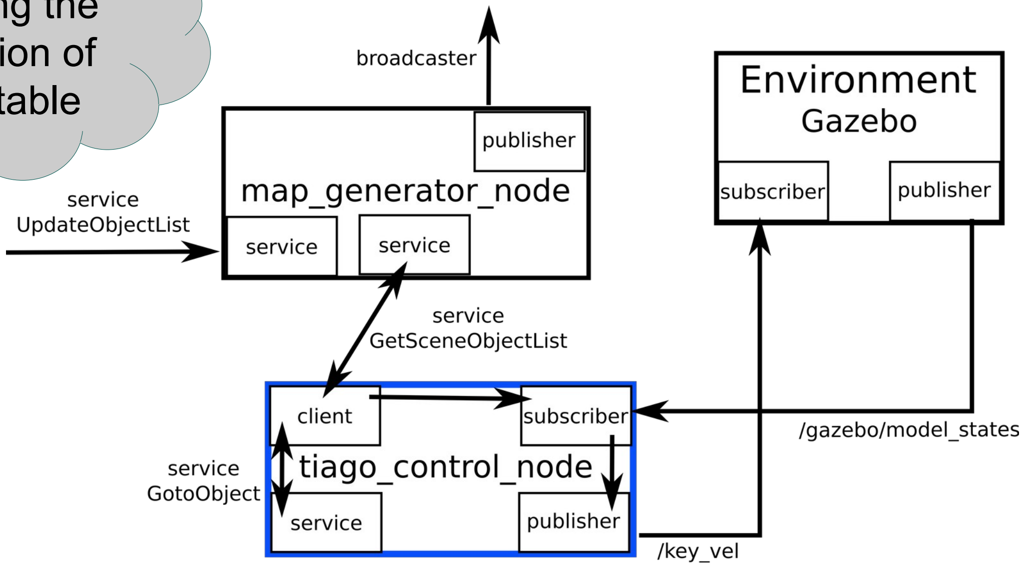
http://wiki.ros.org/rosparam http://wiki.ros.org/roscpp/Overview/Parameter%20Server

Assignment 03 – Task 4

A03.T04. Now, we want the TIAGO robot to use the information it has collected from the environment to move from its origin to a target position

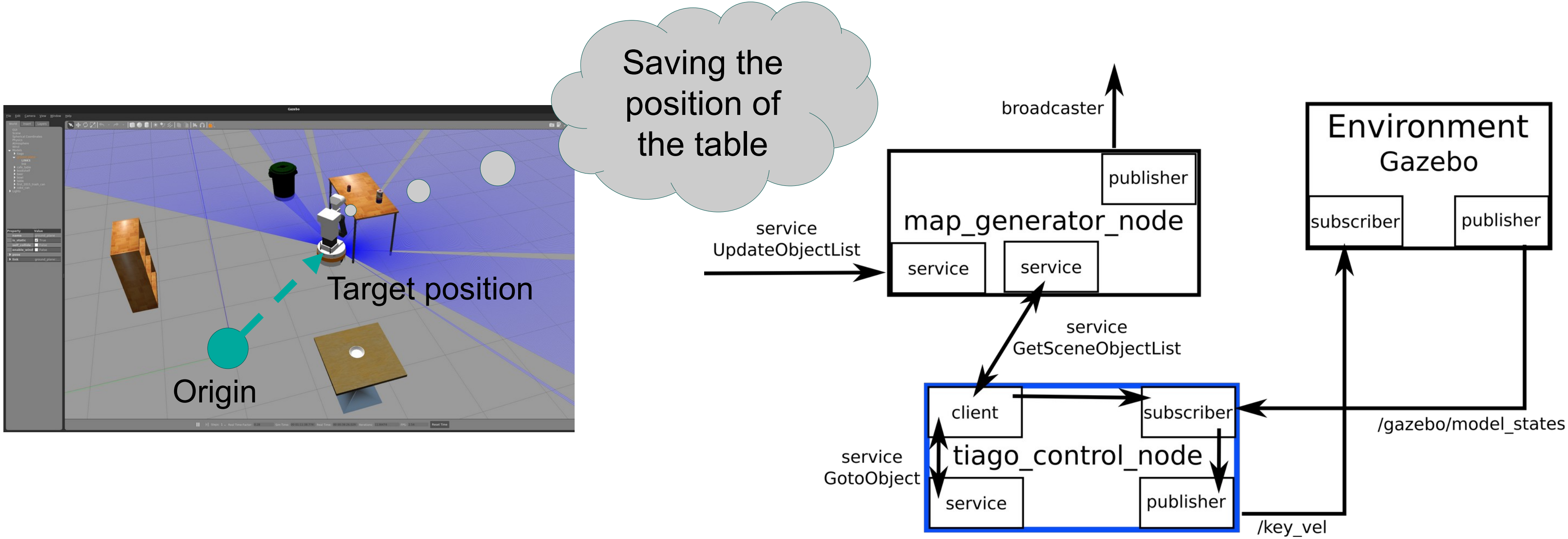


Saving the position of the table

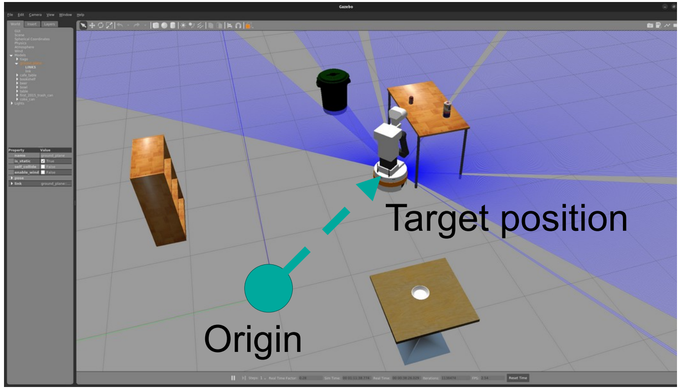


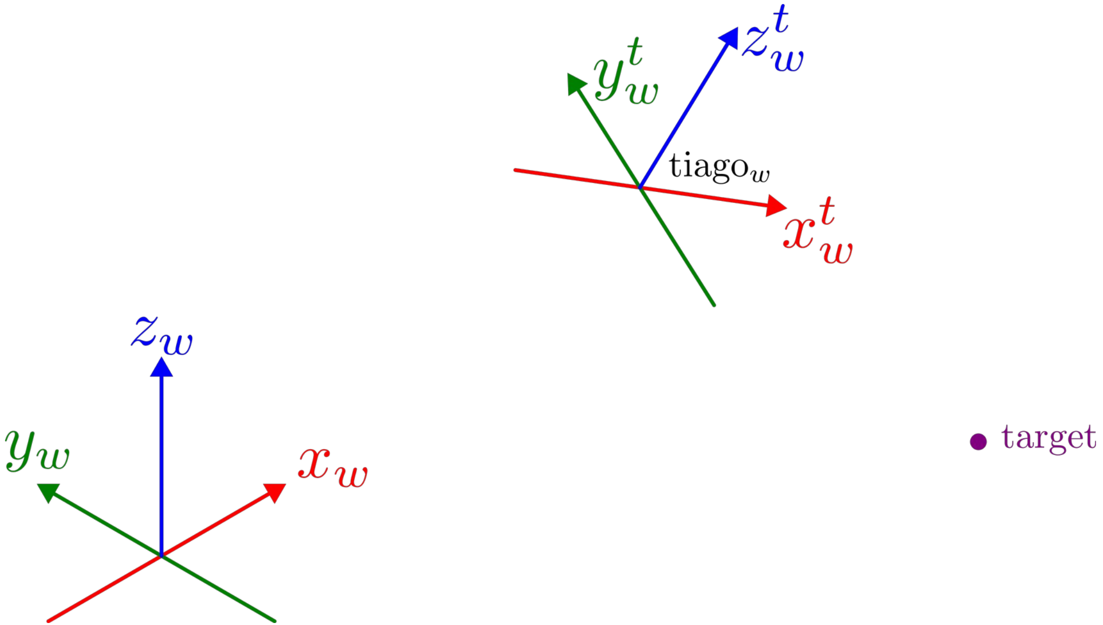
Assignment 03 – Task 4

A03.T04. Now, we want the TIAGO robot to use the information it has collected from the environment to move from its origin to a target position

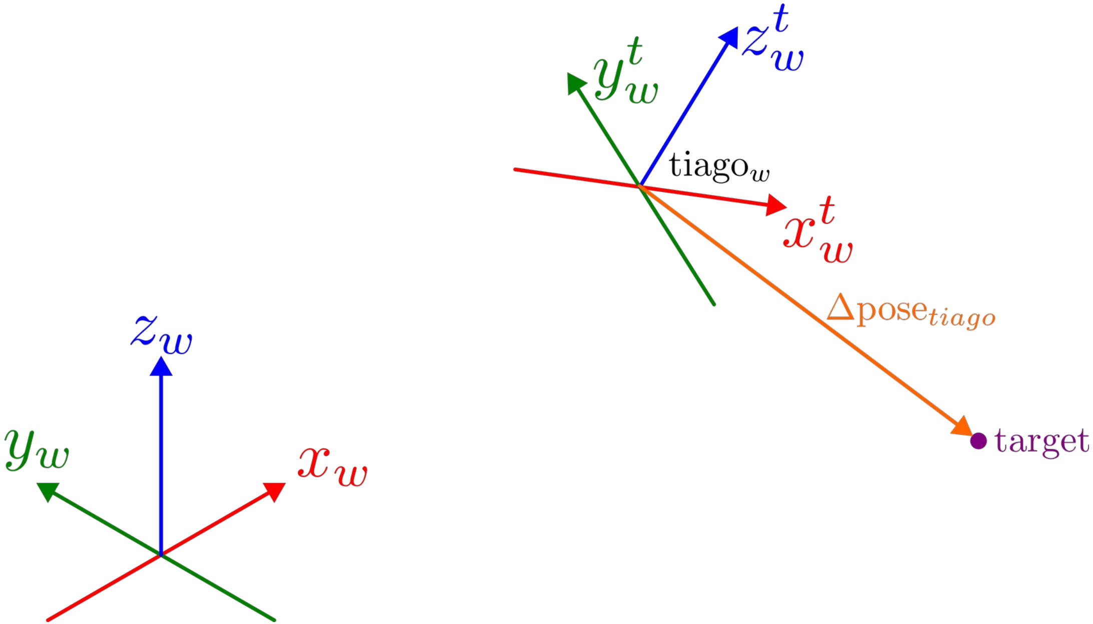


Assignment 03 – Task 4

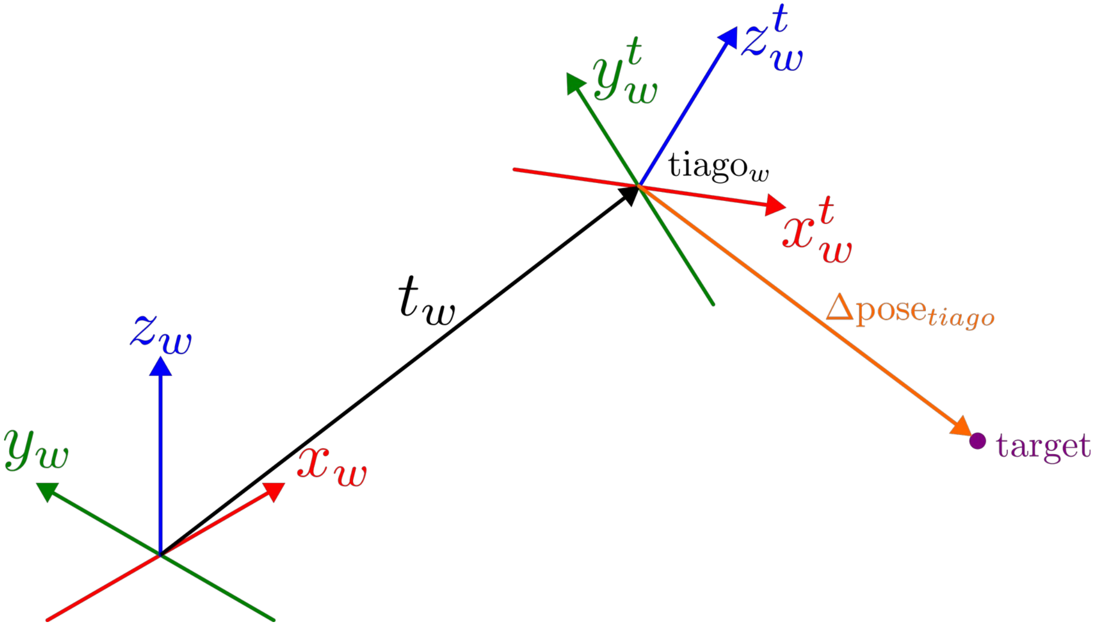




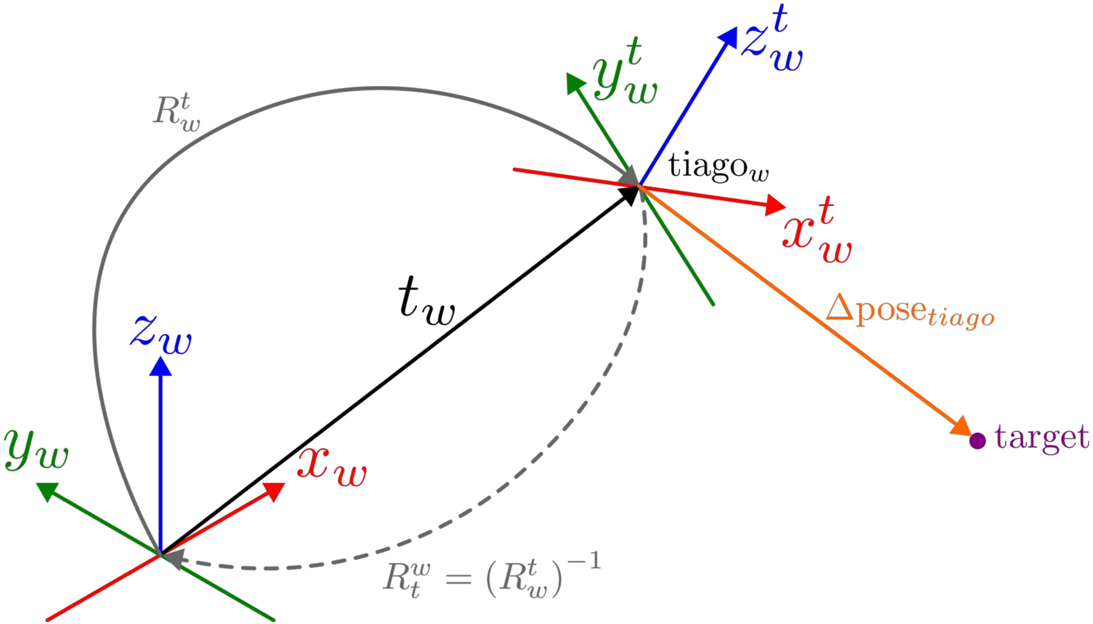
Assignment 03 – Task 4



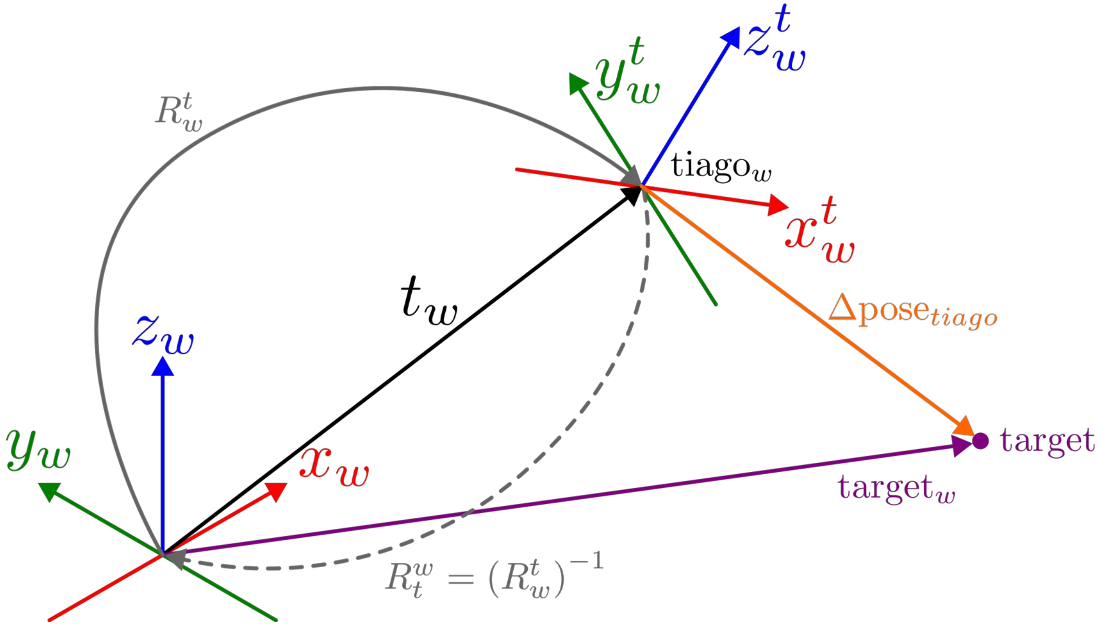
Assignment 03 – Task 4

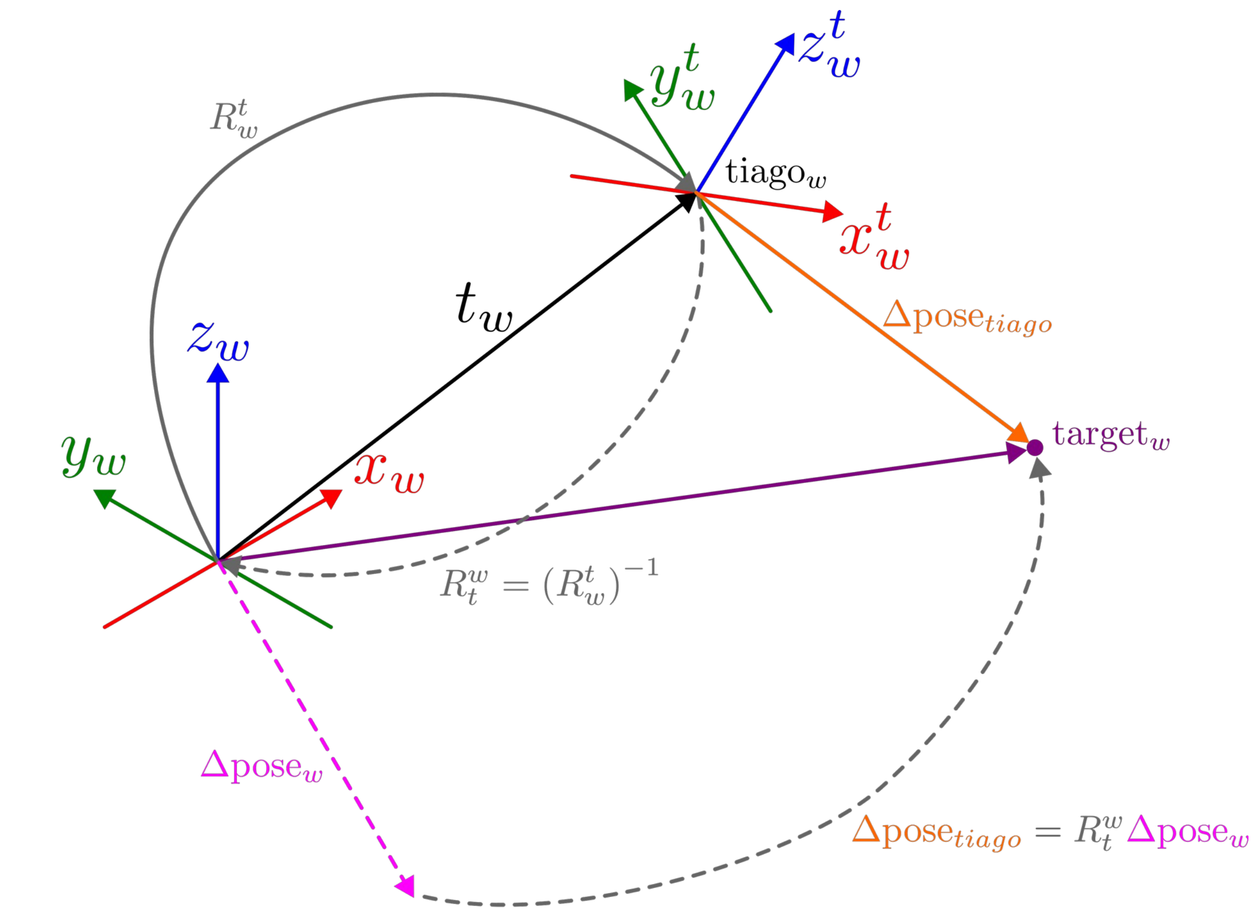


Assignment 03 – Task 4



Assignment 03 – Task 4



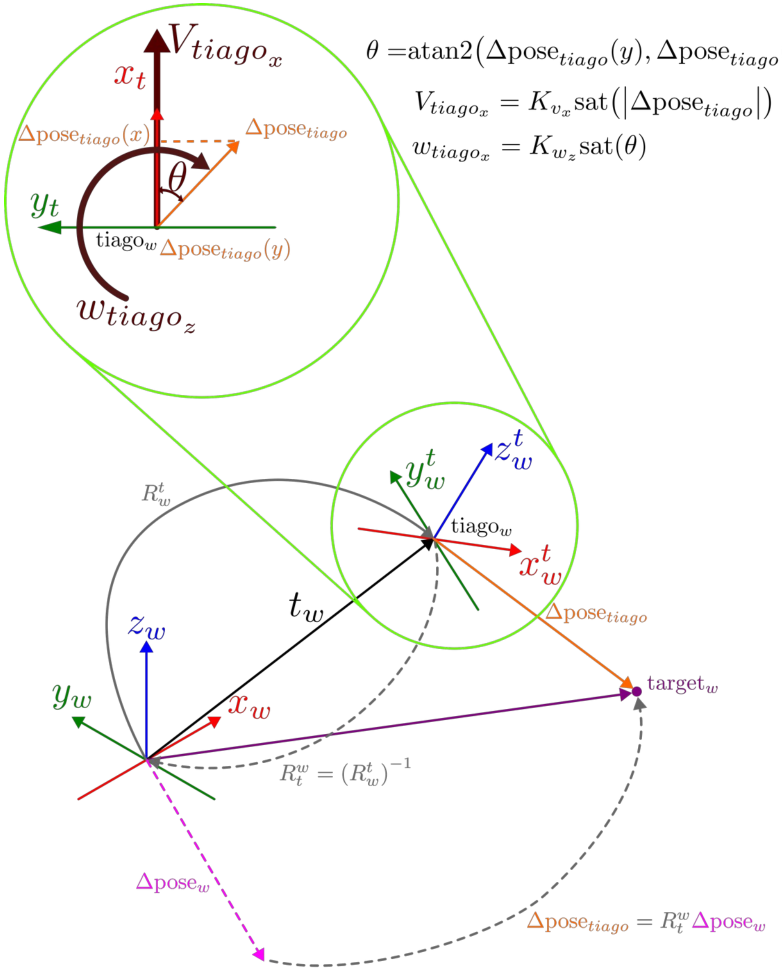


$$\Delta\text{pose}_w = \text{target}_w - \text{tiago}_w$$

$$\theta = \text{atan2}(\Delta\text{pose}_{tiago}(y), \Delta\text{pose}_{tiago}(x))$$

$$V_{tiago_x} = K_{v_x}\text{sat}(|\Delta\text{pose}_{tiago}|)$$

$$w_{tiago_x} = K_{w_z}\text{sat}(\theta)$$



$$\Delta\text{pose}_w = \text{target}_w - \text{tiago}_w$$

$$\theta = \mathrm{atan2}\big(\Delta\mathrm{pose}_{tiago}(y), \Delta\mathrm{pose}_{tiago}(x)\big)$$

$$V_{tiago_x} = K_{v_x}\mathrm{sat}\big(|\Delta\mathrm{pose}_{tiago}|\big)$$

$$w_{tiago_x} = K_{w_z}\mathrm{sat}(\theta)$$

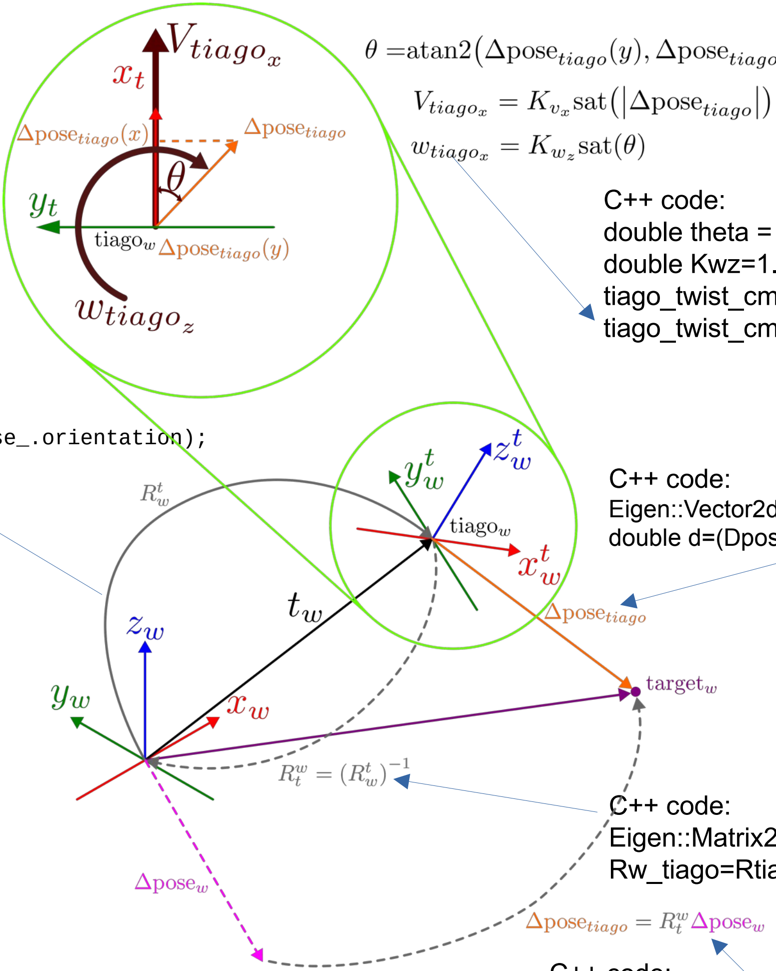
C++ code:

```
double theta = std::atan2(Dpose_tiago(1),Dpose_tiago(0));
double Kwz=1.1, Kvx=0.1;
tiago_twist_cmd.linear.x = Kvx*d;
tiago_twist_cmd.angular.z = Kwz*theta;
```

```
//Get 2D Rotation matrix from quaternion
Eigen::Matrix2d Rtiago_w=q2Rot2D(tiago_pose_.orientation);
```

C++ code:

```
Eigen::Vector2d Dpose_tiago=Rw_tiago*Dpose_w;
double d=(Dpose_tiago.norm() <= 1.3) ? 0.0: Dpose_tiago.norm();
```



C++ code:

```
Eigen::Matrix2d
Rw_tiago=Rtiago_w.inverse();
```

C++ code:

```
Dpose_w=target_w-tiago_w;
```

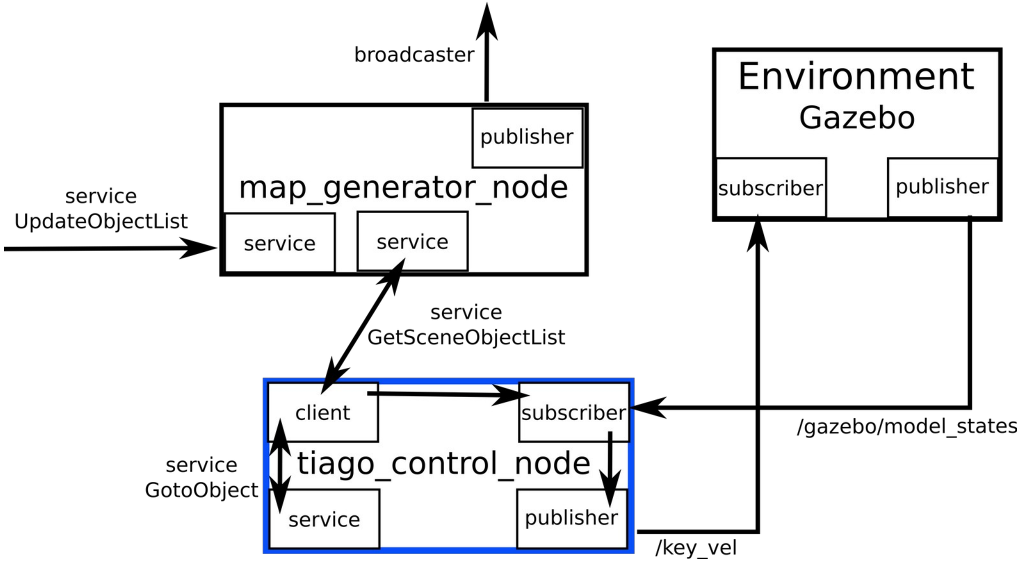
$$\Delta\mathrm{pose}_w = \mathrm{target}_w - \mathrm{tiago}_w$$

C++ code:

```
Eigen::Vector2d Dpose_tiago=Rw_tiago*Dpose_w;
```

Assignment 03 – Task 4

A03.T04. The goal of this task is to create a more complex node called “tiago_control_node” (8 pts)



- This node will have:
  - One subscriber to the topic “/gazebo/model_states (2 pts)
  - One publisher that will send the twist information of the robot to the topic /key_vel (2 pts)
  - Once client that will be inside the callback function of the subscriber. This client will also connect to a service inside the node (2 pts) “map_generator_node”
  - Once service that will calculate the linear and angular velocity of the robot to move towards the target object (2pts)

Assignment 03 – Task 4

Hints for the callback function of the subscriber:

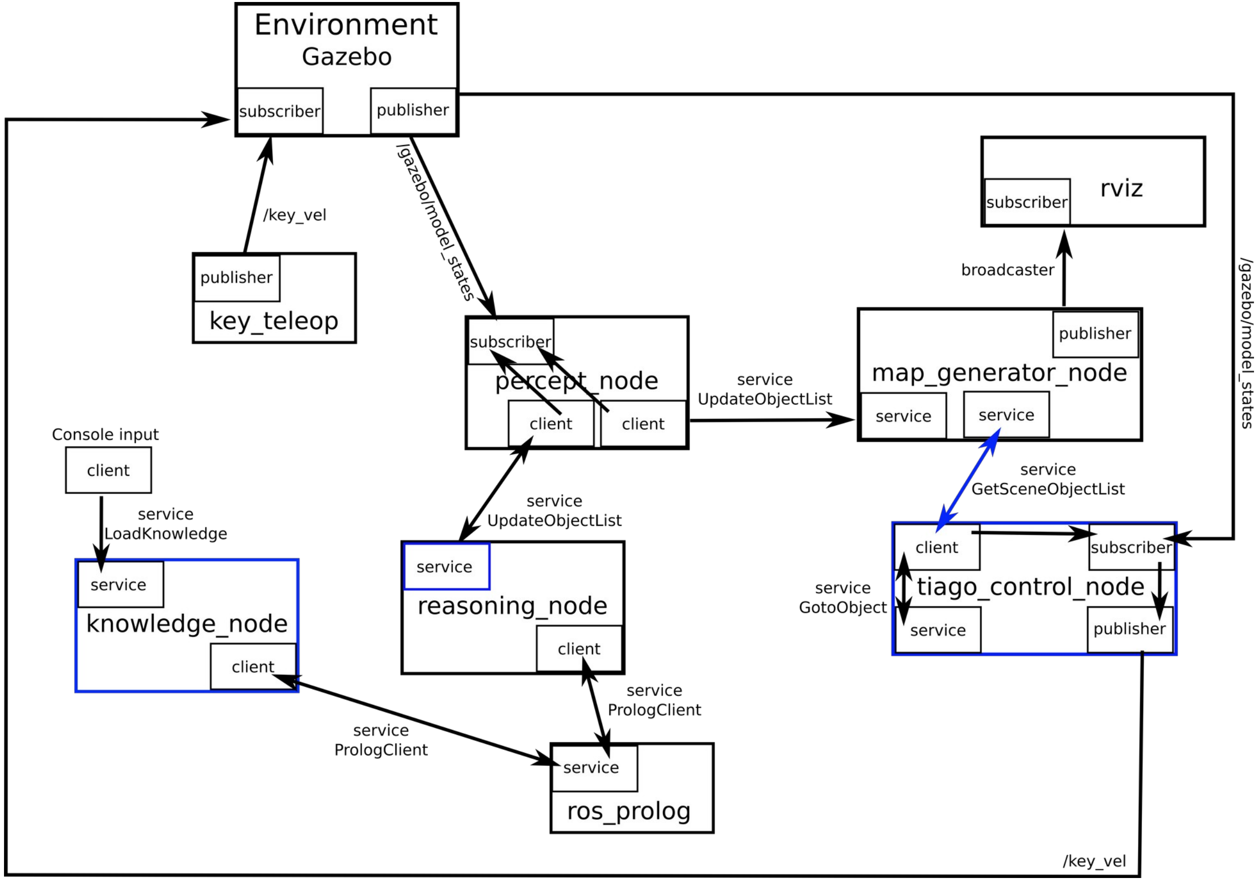
```
// Use the following variable to send the information to the publisher
geometry_msgs::Twist tiago_twist_cmd;

// Get the 2D Rotation matrix from quaternion
Eigen::Matrix2d Rtiago_w=q2Rot2D(tiago_pose_.orientation);
```

You will need the following function to compute the transformations:

```
Eigen::Matrix2d q2Rot2D(const geometry_msgs::Quaternion &quaternion)
    {
        Eigen::Quaterniond eigenQuaternion(quaternion.w, quaternion.x, quaternion.y, quaternion.z);
        Eigen::Matrix2d rotationMatrix = eigenQuaternion.toRotationMatrix().block(0,0,2,2);
        return rotationMatrix;
    }
```

Assignment 03



Assignment 03

Deadline for Assignment 03: **Nov 29 at 11:59 pm**

- **Individual assignment**
- Please upload original material before the deadline
- Upload your whole package in a zip file
  - Within this zip file you should include a README file that explains how to run your code
  - Please name the file “world_percept_assig3_CID_A01.zip”

CHALMERS

UNIVERSITY OF TECHNOLOGY