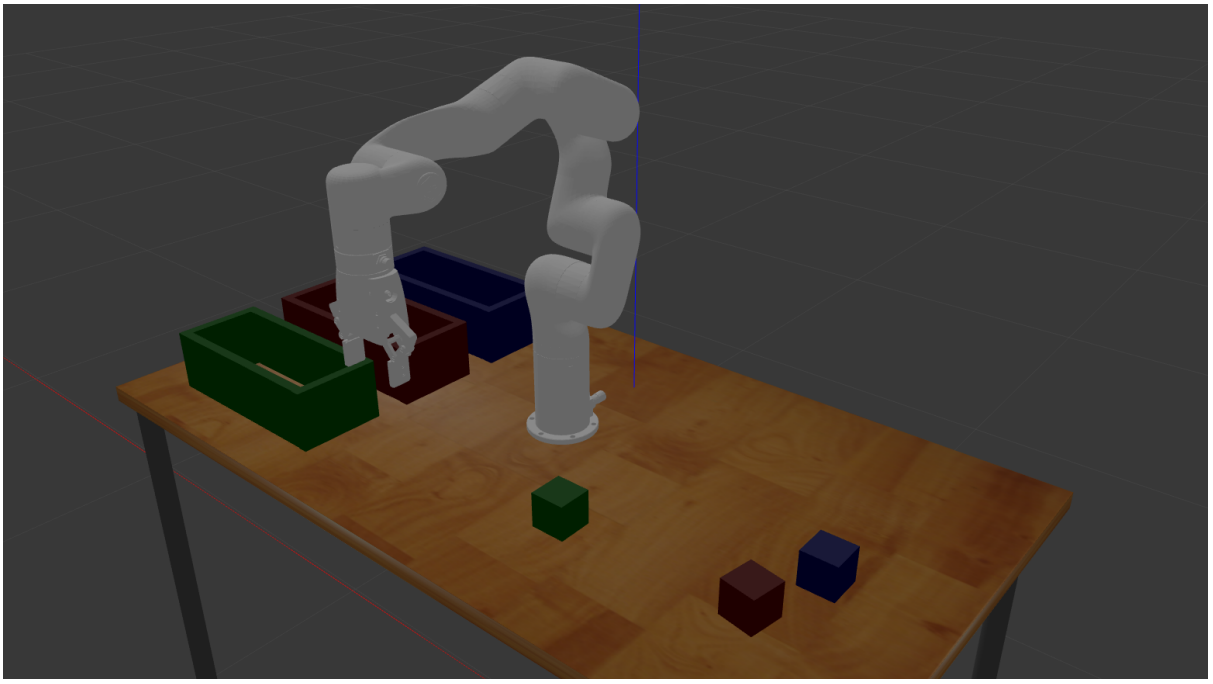
Manchester Robotics Xarm Challenge



TE3001B Robotics Foundation

Luis Humberto Sanchez Vaca A01638029

Owen Jauregui Borbón A01638122

Giancarlo Franco Carrillo A01638108

Thursday March 17, 2022

Challenge

Simulate and control with ROS the solution of a pick and place task using the manipulator arm robot “Xarm 6”. The pick and place task was meant to pick different colored boxes spreaded around the robot’s workspace and place them in the matching containers.

The simulation uses both gazebo and rviz, and the control is made by using the moveit library, along with the tf2 library to do transformations between frames.

Xarm6 Robotic Arm

The *uArm xArm 6 Robotic Arm* is a manipulator robot of 6 degrees of freedom, built for lightweight tasks and for being user friendly, its high precision and the variety of end-effectors allow users to solve different tasks according to the objective. You can easily add the most commonly used end-actuators: a gripper, a vacuum system, and a camera module to help in various tasks.

Some of the most important specifications of the robot are:

Mechanical:

Payload (kg): 5

Reach (mm): 691

Degrees of Freedom: 6

Repeatability (mm): ±0.1

Typical Tool Maximum 5peed{m/s): 1

Weight (kg): 9.5

Performance:

Ambient Temperature Range: 0-50 °C

Power Consumption: Min 8.4 W, Typical 120 W Max 240 W

Input Power Supply: 24 V DC, 10 A

Power Supply: 100-240 V AC, 50-60 Hz

IP Classification: IP54

ISO Class Cleanroom: 5

Robot Mounting: Any

I/O Ports: Digital x 2 485 x 2

Communication:

Programming: Python/C++/ROS Underlying Interface

Communication Protocol: Self-defined

Communication Mode: RS-485/Ethernet TCP-IP

Electrical:

Nominal supply voltage: 24 V DC

Absolute maximum supply voltage: 28 V DC

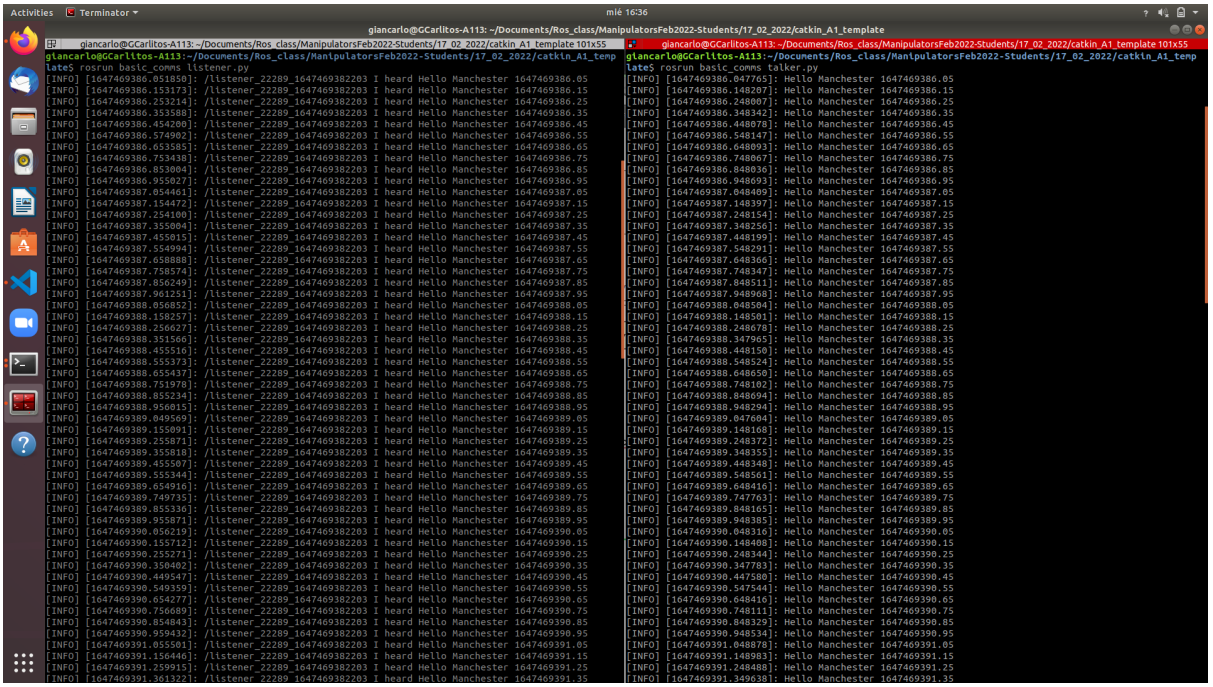
Quiescent power (minimum power consumption): 1.5 W

Peak current: 1.5 A

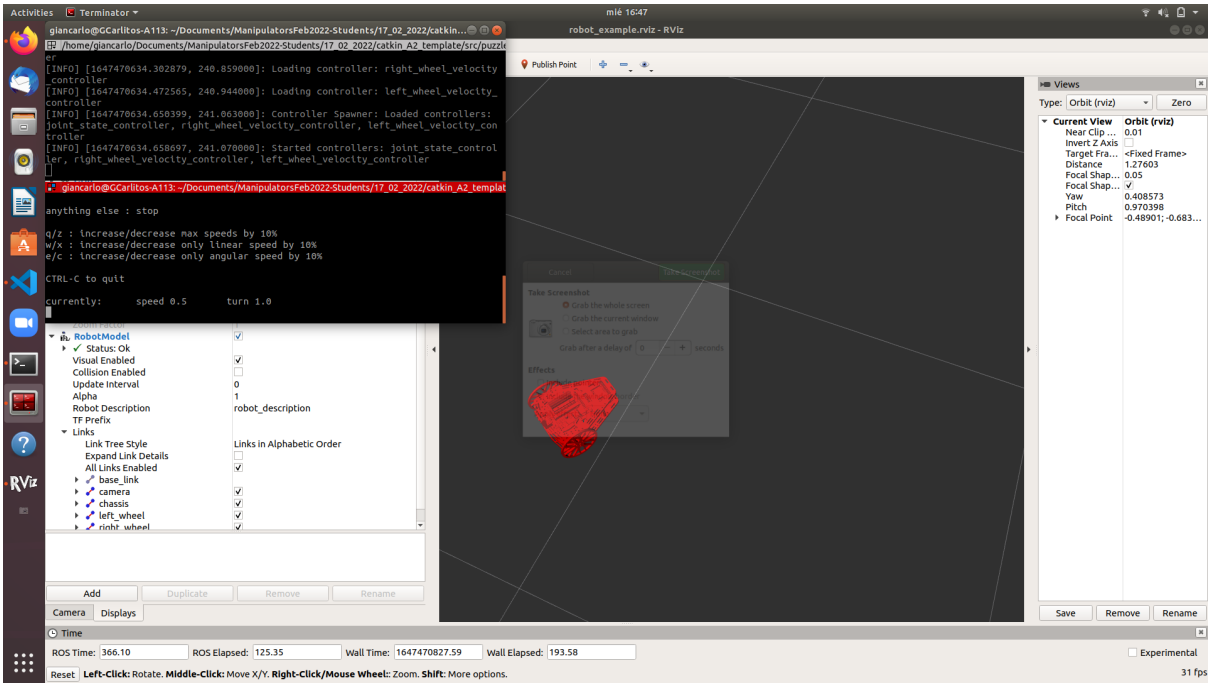
Activities

Before we could do the challenge we needed to learn how to use ROS and its different concepts like publisher, subscriber, nodes, launch files, urdf, xacro, etc. In order to do that we first did the following introductory activities.

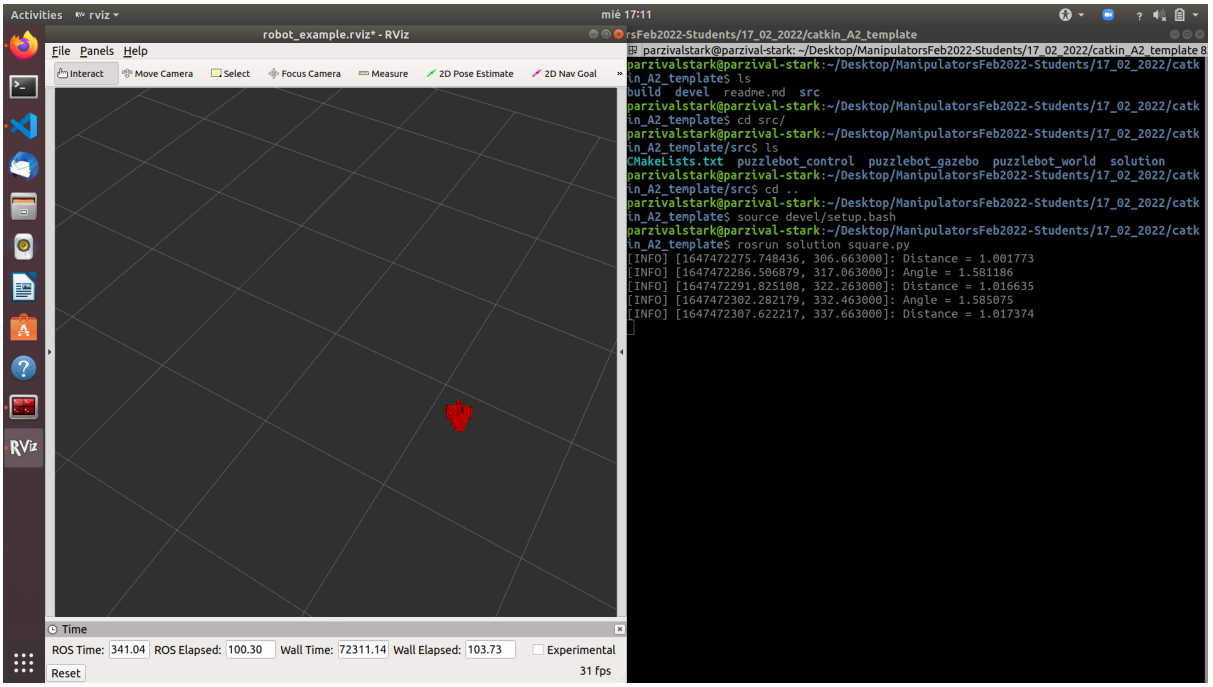
1. Talker and listener



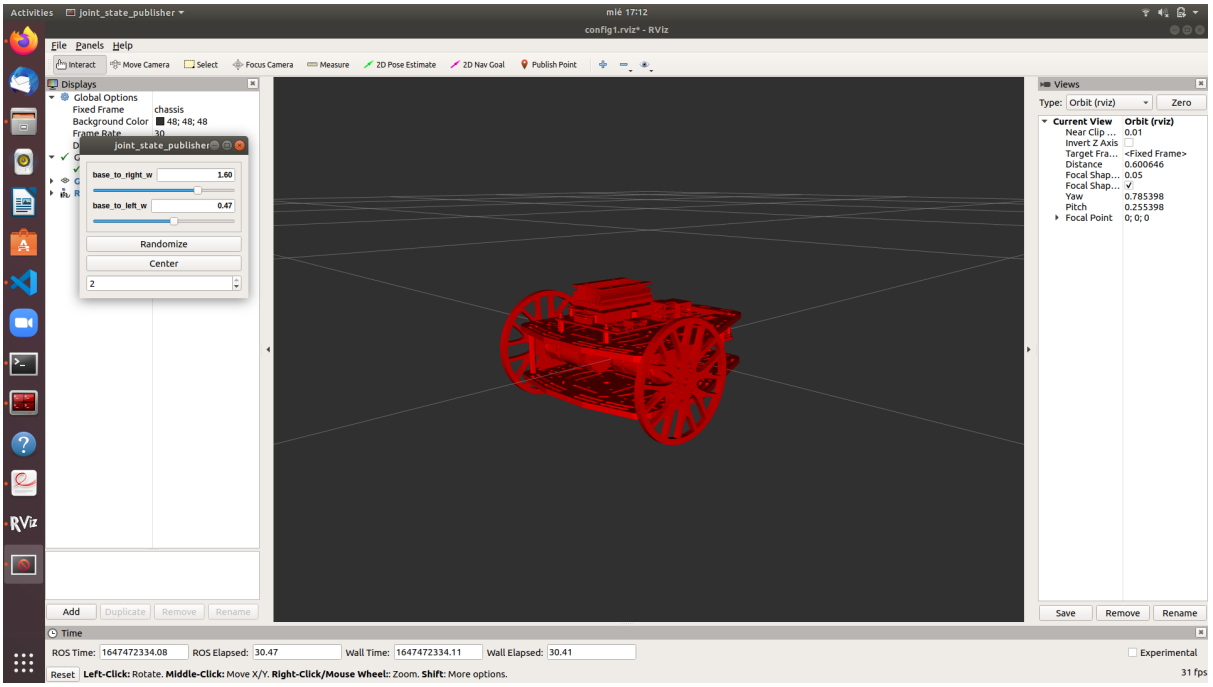
2. Teleoperator using keyboard to control puzzlebot



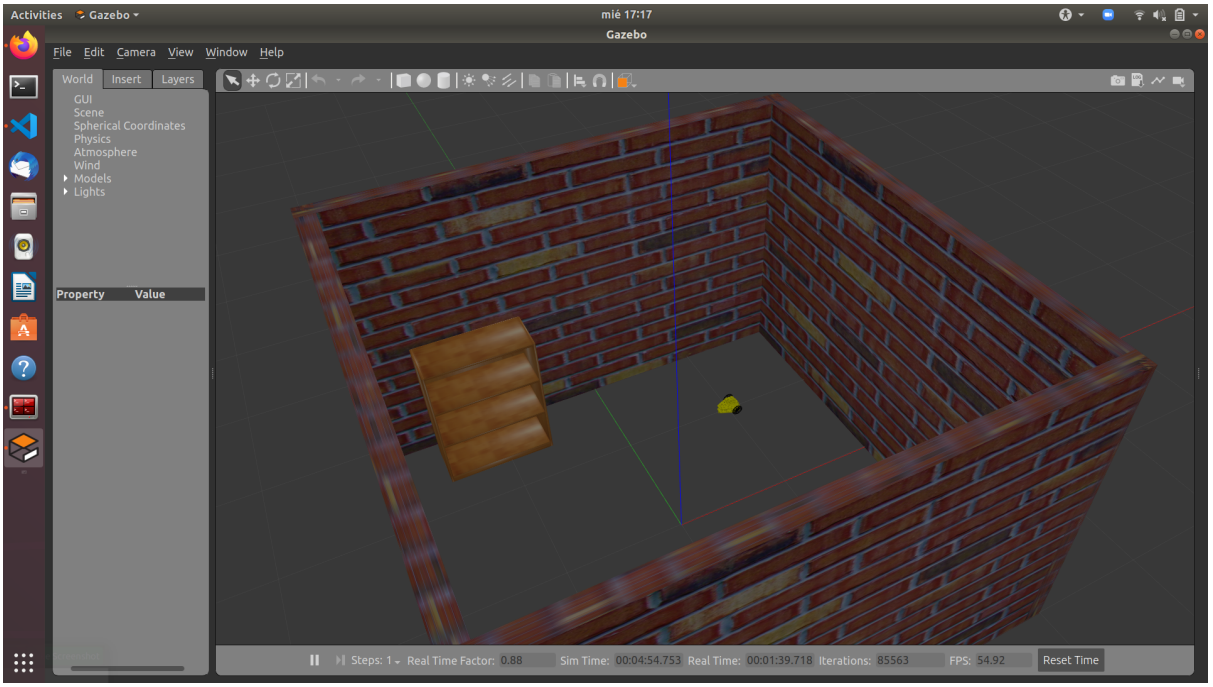
3. Draw square using puzzlebot



4. Place puzzlebot's wheels



5. Puzzlebot in gazebo



Solution

Learn about moveit

MoveIt is an extension tool for RViz that helps users to visualize and debug robots. This plugin allows users to set up virtual environments, create start and goal states for the robot interactively, test various motion planners, and visualize the output.

This tool can be used either by its graphic interface or in code using the moveit_commander Python package, to get familiar with this package first we try the example code using the panda robot, we did our own script to test the different functions that were provided with this package in order to learn which ones were going to be useful in the challenge.

Learn about tf2

Tf2 is a library used for planning the movement of the robot and its frames, keeping the relationship in a tree structure buffered in time, offering the user the possibility to use the transform points, vectors, etc between any two coordinate frames at any desired point in time.

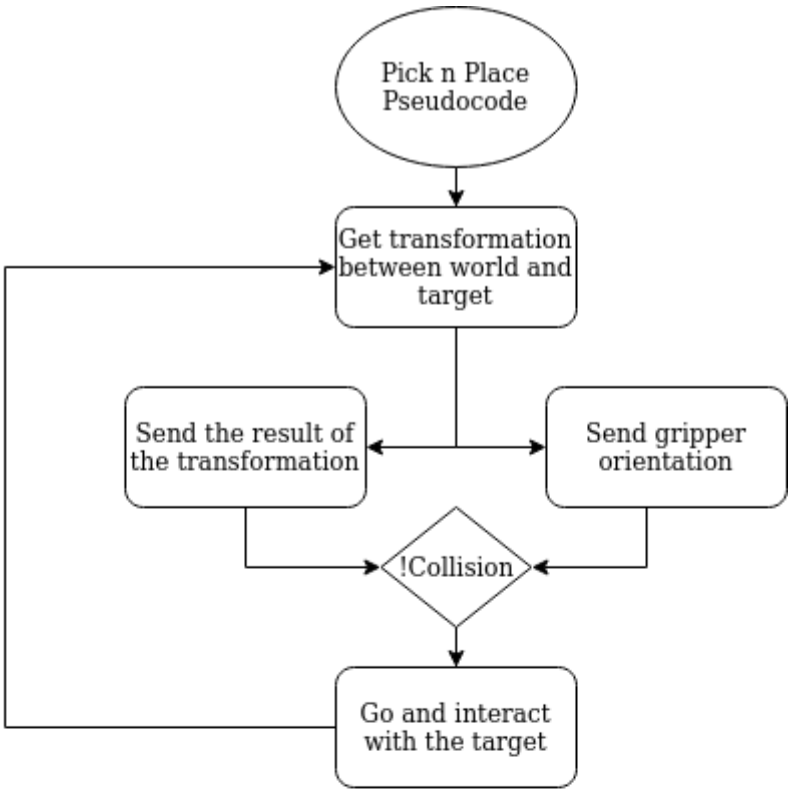
Learn how to use services

In ROS, besides topics we also have services, which function with a server and a client, the client sends requests to the server and it performs different actions accordingly, also it can give an answer.

We knew that there were two services that we needed to interact with to do the pick and place task: GoalPlanner and AttachObject, but first we needed to learn how they work, so we use different commands to know more about them, such as “rosservice info ‘service’”, which gives information like node, type and the arguments we need to provide when calling the service; “rossrv show ‘type’” gives us the type of the arguments and the type of the response; finally “rosservice call ‘service’ ‘args’” can be used to test the kind of responses that this service will give. After getting all this information, we learned enough about the services that we were going to use, to know where and how to use them in the final code.

Write a pseudo code

Now that we have learned about the libraries and the services we were going to use to perform the task, we wrote a pseudocode with the first approach that will be made to perform the pick and place task.



Do tests with the xarm in code

With this approach we decided to implement some of the functions to test them separately in the xarm, we worked in calling the services to see how the response was given, we used to tf2 listener to get the transformation between the frame of the box and the base, then we split the result to get only the part of our interest, this was used with moveit to move the robot to the position of the box, but we had some problems because the planner couldn't find a path, this problem was probably caused by the fact that the transformation result was given with too many decimals, and it was not possible for the planning to find the inverse kinematics. After reviewing the documentation we saw that there was a function to give some tolerance to the planning, and with this the xarm was now able to get to the location of the box.

Write the functions to do the pick and place

Once we tested different commands and saw how the tf2, forward and inverse kinematics for each group had worked, we moved to the solution template in order to make a more proper code. We followed the template as provided and wrote each function, when we finished the basic tasks we debugged them to see how it works and in what order the functions should be called.

Add obstacles in rviz

In order to do motion planning, it is critical to avoid obstacles. For this particular case, the obstacles are the boxes as well as the containers. We can visualize these obstacles in gazebo, but it also needs to be specified in move it for the motion planning to avoid collisions. Once we place these obstacles in moveit, we are able to visualize them in the rviz interface.

It is important to notice that the containers aren't a full obstacle. In fact, these are formed by 4 different obstacles that describe the walls of the container. This way, we ensure they are empty for the motion planning. By identifying the container as an empty box, we can allow the robot to place the boxes inside in a more sophisticated manner.

Do the attachment of the objects

When we already had all the movement and motion planning done, it was crucial to attach and detach the boxes at the right times. Without doing these attachments, the robot doesn't know that there are objects grabbed by it, thus the box could collide against other obstacles. The way to solve this collision problem was to attach the object in rviz using move it so that the planner recognizes the box as part of the robot.

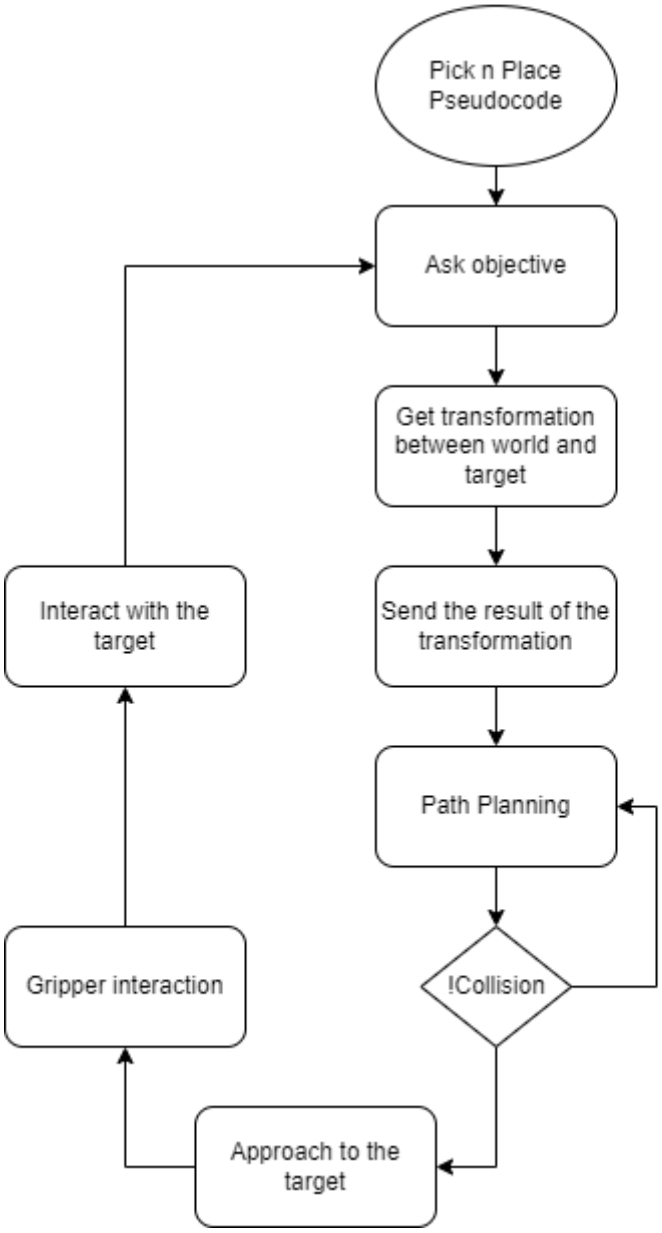
On the other hand, attaching the object in rviz will recognize the object as part of the robot, but isn't going to move the object in the gazebo interface. Therefore, it is important to also make the attachment in gazebo to ensure the proper visualization of the simulation. This way, objects are affected in the motion planning as well as in the gazebo interface.

Close and open gripper

By using forward kinematics and saving the "open/home" position of the gripper we did test and error to simulate the gripper closing itself before attaching to the blocks around the world and open before detaching the block to deposit it into the box, we tried to orientate the gripper with the tf tree but quaternions were out of our knowledge, so that is one of the things we could do for future implementations.

Final solution

Once all the functions provided by the template were successfully debugged and run correctly, codifying a new version of the pseudo code that would result later into the final solution helped us to visualize the correct order in which the code should do the different services and routines.



Additional resources

We decided to use a repository in Github to work in a more efficient way by dividing the different functions, also we wanted to have version control in case the modifications cause the robot to malfunction.

GitHub Repository: https://github.com/ParzivalStark/Challenge_Xarm6

We recorded a video to demonstrate the functionality of the robot performing the pick and place task.

Demo Video: https://youtu.be/vpxlERksXUQ

Conclusions

Luis Humberto Sanchez Vaca:

I think this was a great and really interesting challenge, we needed to learn how to use different tools that are highly used in the field of robotics, like ROS, Gazebo and Rviz, the initial activities were a good start, but to be able to complete the challenge we had to research a lot so we could learn more about this tools and the libraries that we needed to use. I think that even though we didn't directly apply concepts like kinematics or control in the challenge, it was necessary to know about them, and understanding them was key to know how tf2 or moveit work and how they could be applied to solve the pick and place task.

Another important factor was the project management, once all of us had understood the basic concepts that we were applying, we divided the functions of the code to work in a more efficient way, by working in parallel and test them individually we made sure that each part worked correctly before merging it with the rest of the code.

This challenge gave us the opportunity to work in a more real environment, even though we only use simulations, these are the tools that are used in the industry, and simulating is an important step to make sure that the real robot won't have any problems, so I'm sure that the knowledge and skills that we acquired by doing this challenge will be really useful in the future as professionals.

Owen Jauregui Borbón:

Along this challenge, we were required to develop a lot of aptitudes and gain new knowledge. It was imperative to plan ahead the way we were going to work and collaborate as a team. The task distribution was a key factor to complete the challenge and use of project managing technologies was crucial for a great working synergy.

ROS was a great platform to increase our skill and knowledge. As well as ROS, the use of Gazebo and Rviz interfaces was the perfect kind of experience for future simulation, designing and testing projects for our own robots as professionals. At the end we got the opportunity to get more interface options and practice our teamwork skills. This challenge proves the importance of selecting the right tools and technologies, as well as having project management.

Giancarlo Franco Carrillo:

Since the very beginning the pick and place challenge sound very hard having in mind that is the first real robotic application we implement as students, that but also the very first time we use Ros and its extensions such as Gazebo that helped to simulate a more real scenario, or Rviz that was very useful to pre-simulate that scenario and to let us know that the robot was working properly.

Getting deeper into what is going to be our everyday, the fear at the beginning of the semester was something big to talk about, but as the sessions with Manchester Robotics were keeping going and with the help of the professors, that fear disappeared and turned into a very powerful knowledge of how to properly as a future robotic engineer simulate, calculate and codify programs to our future robotic creations.

Finally, selecting the correct team according to our different skills is how we could do proper management on the challenge. Even though this time the challenge was only simulated, I understand that not only this, but many other soft skills such as teamwork, project management, will help me to work properly along with my partners.

References

MoveIt. Concepts. recovered from:

https://moveit.ros.org/documentation/concepts/

ROS. ROS Tutoriales. recovered from:

http://wiki.ros.org/es/ROS/Tutoriales

UFACTORY. uArm xArm 6 Robotic Arm. recovered from:

https://www.robotshop.com/jp/en/uarm-xarm-6-robotic-arm.html