SPACE NAVIGATOR

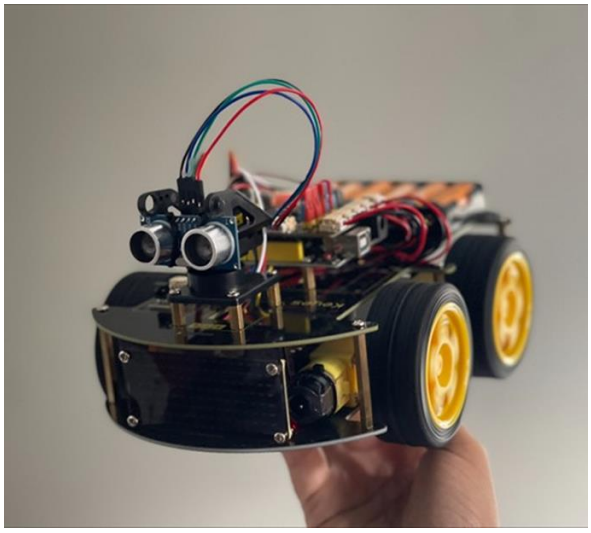
Dr. Shayok Mukhopadhyay

Dept. of ECECS University of New Haven

smukhopadhyay@unh.newhaven.edu

Abstract

The purpose of this project was to converge the theory and technical expertise gained from this course into a project which displays our understanding of autonomous vehicles. With this in mind, our goals consisted of hardware interfacing, control, and navigation. In summary, our project was somewhat successful albeit not perfect. We succeeded in hardware interfacing: with the Arduino handling low-level control of the ultrasonic sensor, servo 'neck', and motor speed with direction; reporting the distance sensed through USB. In addition, we successfully implemented PID controllers so the robot can maintain a constant distance from an object or use an IMU to change its orientation. However, largely due to time constraints, we failed to achieve autonomous navigation or IMU displacement control. So here we present our experience, detail our goals, methodology, and how we were on track to achieve autonomous navigation in time.



ELEC 4101-02 F23

Report Finalized: 12–14–2023

Group: Francisco Estévez, Priyanka, Akash

"In pursuit of our abstraction objectives, our team collaboratively delineated responsibilities, and I undertook the commitment to focus on the navigation component of the project through the implementation of Simultaneous Localization and Mapping (SLAM). Diligently, I have endeavoured to fulfil this responsibility, striving to contribute meaningfully to the overall success of our project."

I initially submitted a term paper reflecting only half of my progress. Following the mid-term project submission, I extended my efforts, particularly in the area of SLAM. While I have made significant strides in understanding and implementing map creation, time constraints have regrettably hindered me from fully achieving the project's objectives."

KEYWORDS ACHIEVED SO FAR:

-knowledge gained about ROS2 and Navigation 2

-What is the Navigation2 stack, and why do we need it?

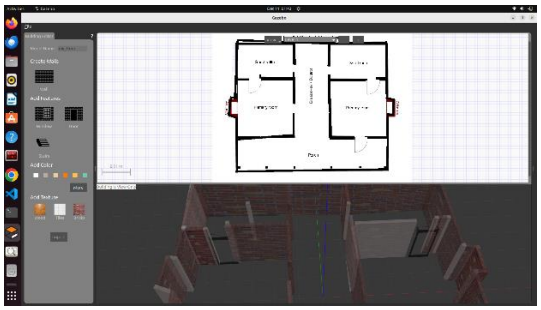
-Setup and Installation for ROS2 and Nav2

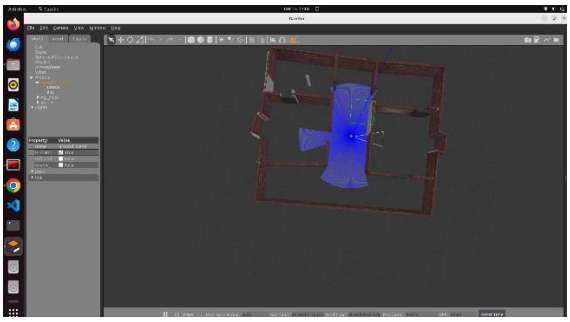
-Generating a map with SLAM

-Understanding the Nav2 Stack

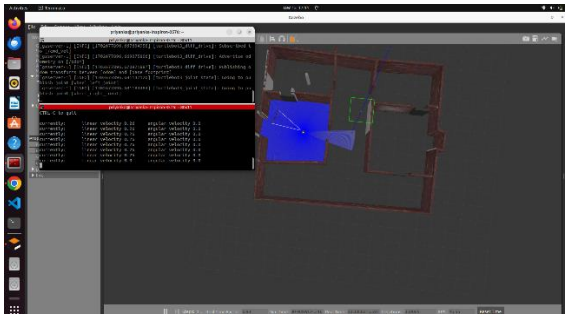
-Building my own house world for Navigation in Gazebo

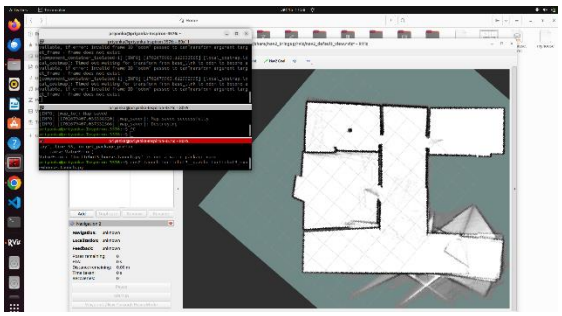
-Agapting a Custom Robot for Nav2



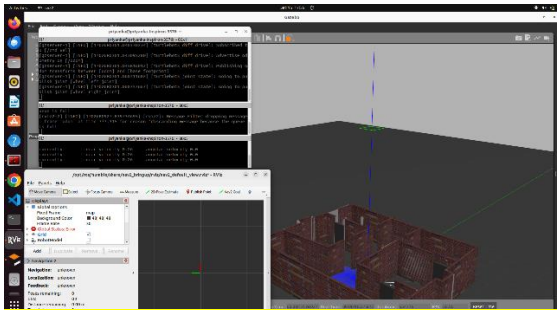


File Name: My_floor





Folder Name:Map-File: mynewhousemap.yaml



Trans Form:

We could need to know where a robot is relative another robot where a laser scan is relative to the map origin or to another part of the robot.

TFs neede for Nav2

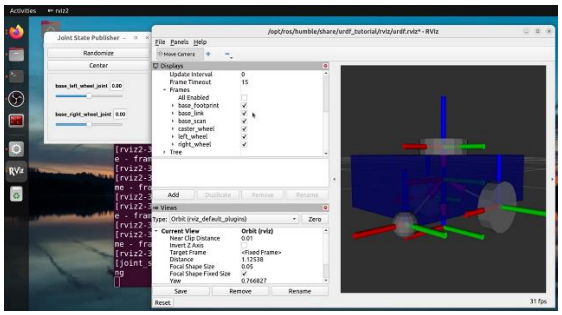
1 map→odom

2 odom>base_link

3base_link>base_scan

Focusing on transforms. Then following with optometry another sensor.

Controller that takes a common velocity and translates it into real comments for the motors of the wheels so that the robot can move.



Have different frames,f or example the base link,the base footprint base etc.

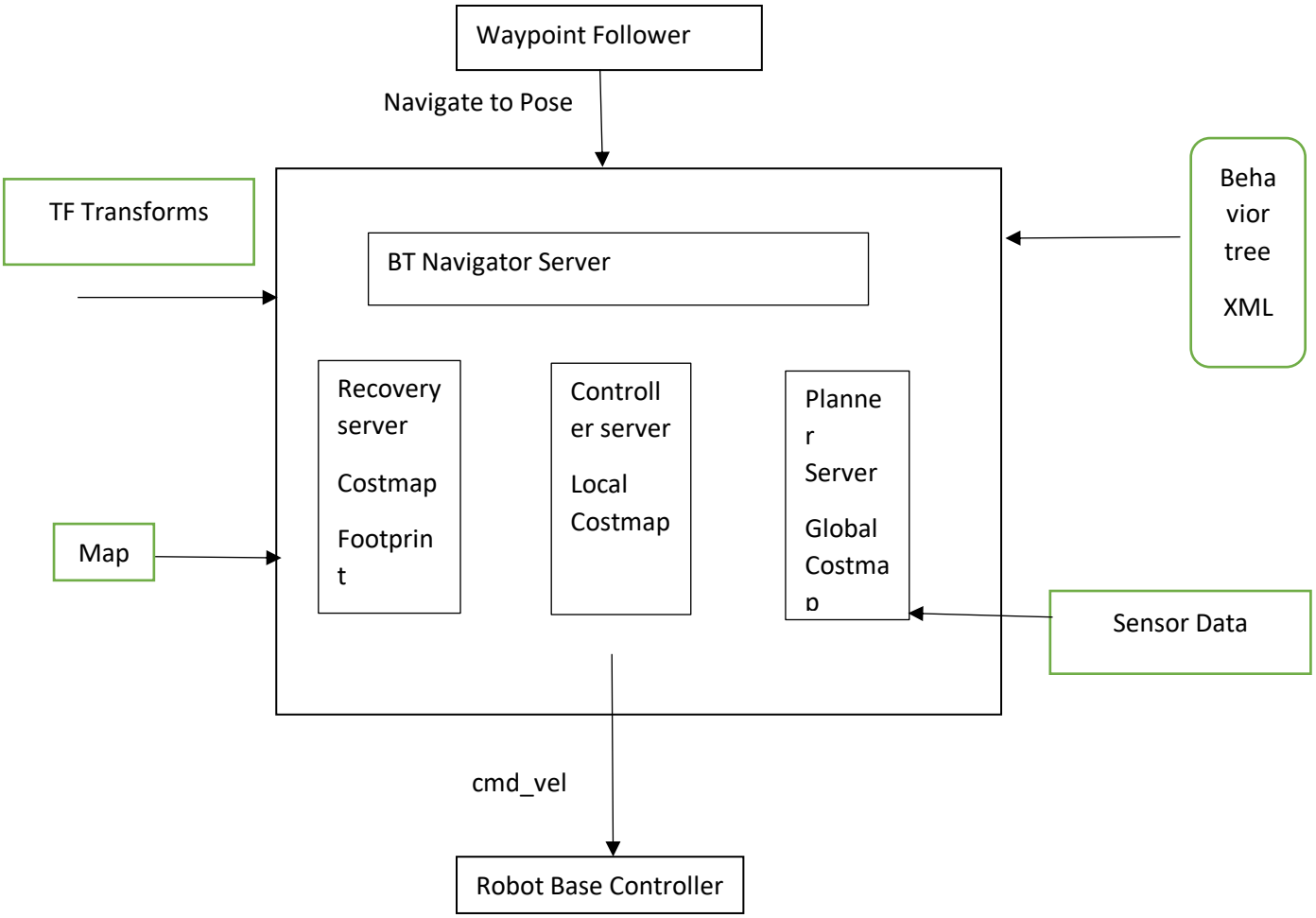
$ ros2 launch urdf_tutorial display.launch.py model:=my_robot.urdf	Command to launch urdf file

Inputs/Outputs for Navigation2

-Odometry

-Sensors

Controller



SLAM and Navigation Steps for Any Robot (ROS2 Nav2)

Those commands are the general commands to run for SLAM and Navigation when using any robot that is configured for the Nav2 stack.

Steps - SLAM

Need to install the slam_toolbox package:

$ sudo apt install ros-humble-slam-toolbox

1. Start the robot
   This will be specific to robot.
   Example with simulation:
   $ ros2 launch turtlebot3_gazebo turtlebot3_world.launch.py
2. Start a Navigation launch file
   $ ros2 launch nav2_bringup navigation_launch.py use_sim_time:=True
3. Start SLAM with slam_toolbox
   $ ros2 launch slam_toolbox online_async_launch.py use_sim_time:=True if using Gazebo
4. Start Rviz
   $ ros2 run rviz2 rviz2
   Need to configure RViz
5. Generate and save map

Make the robot move in the environment.
Example with simulation:
$ ros2 run turtlebot3_teleop teleop_keyboard Save the map: $ ros2 run nav2_map_server map_saver_cli -f ~/my_map

Steps - Navigation

1. Start the robot
   This will be specific to own robot.
   Example with simulation:
   $ ros2 launch turtlebot3_gazebo turtlebot3_world.launch.py

2. Start the main Navigation2 launch file $ ros2 launch nav2_bringup bringup_launch.py map:=path/to/map.yaml sim_time:=True

3. Start RViz $ ros2 run rviz2 rviz2 (you will need to configure RViz)

4. Send navigation commands Use the “2D Pose Estimate” button to set the initial pose, and the “Nav2 Goal” button to send navigation goals.

Note: instead of using RViz to send commands, you can directly interact with the Nav2 interfaces in our own code, for example using the Simple Commander API

Run Navigation with custom Robot(Using SLAM-toolbox)

Using a package that is named Slam Toolbox

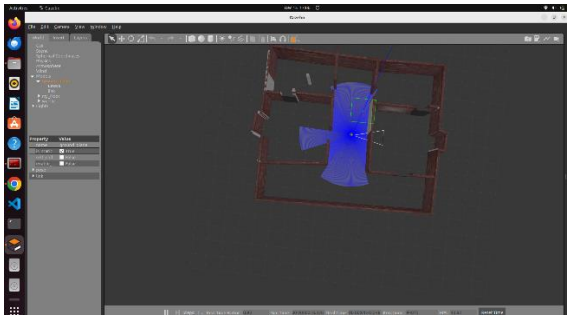
```
$sudo apt install ros-humble-slam-toolbox
```

-A laser scan needs to be published

```
$ros2 launch turtlebot3_gazebo turtlebot3_mynewhouse.world.launch.py
```

```
$ Ros2 launch nav2_bringup navigation_launch.py

$ ros2 launch turtlebot3_navigation2 navigation2.launch.py use_sim_time:=True map:=/mynewhouse.yaml/
```



Future Goals

--Interact programmatically with the Navigation stack

Attaching Reference Folder Name as – Final_Priyanka.zip