Introduction

Yanai Avila and Abraham Garcia

CPE 476 Mobile Robotics

Assignment 8 – Final Report

Author Note

[Include any grant/funding information and a complete correspondence address.]

Table of Contents

Introduction

We accomplished all the tasks asked for in this report. By creating a URDF file, we were able to create a model of our robot to view in Rviz and Gazebo. By running teleop keyboard and joy commands, we were able to move the rover through simulation in Rviz and Gazebo. We were also able to move the actual rover with these commands after connecting the ROS controller. Finally, we were able to implement mapping, localization and navigation once we hooked up the LIDAR and Jeston Nano.

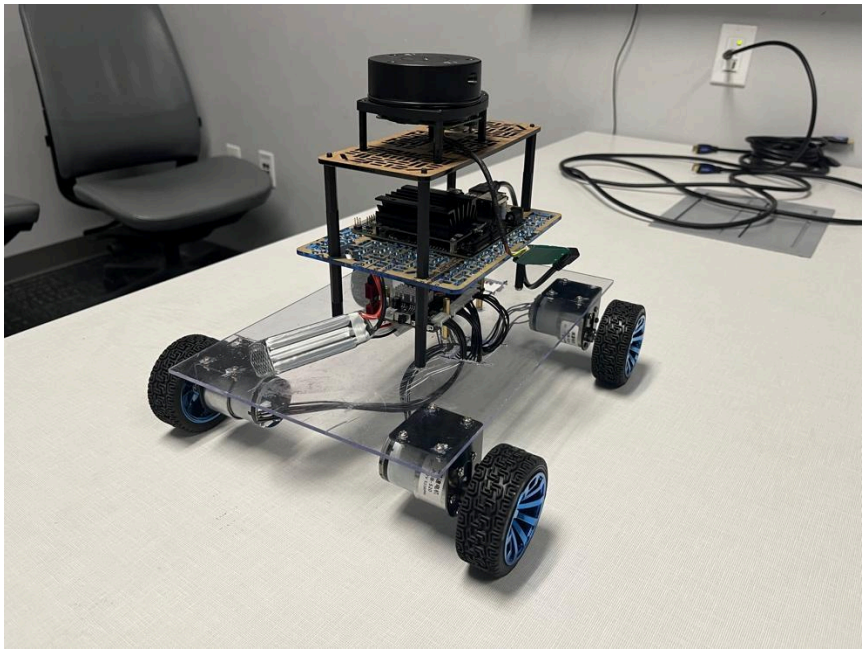
Rover

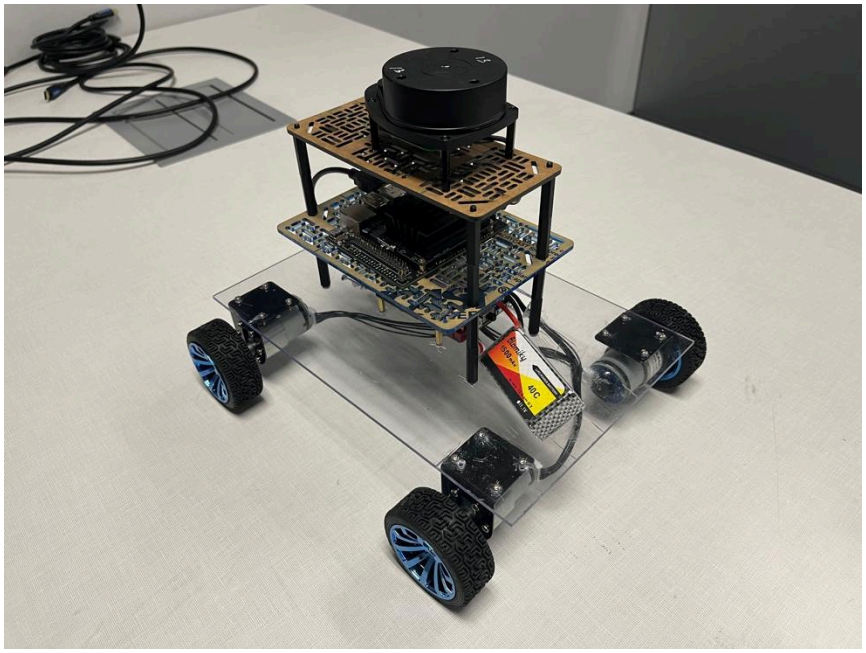
Hardware components of the rover:

- Chassis + 2 layers
- Encoder geared motor + motor bracket + coupling + rubber tire (x4)
- ROS controller
- RPLIDAR A1
- Jestson NANO 4GB with Micro SD Card
- 11.1V T-plug Battery with Battery charger
- Cables (HDMI, USB, Miro USB, JN power)

The Rover

Pictures:



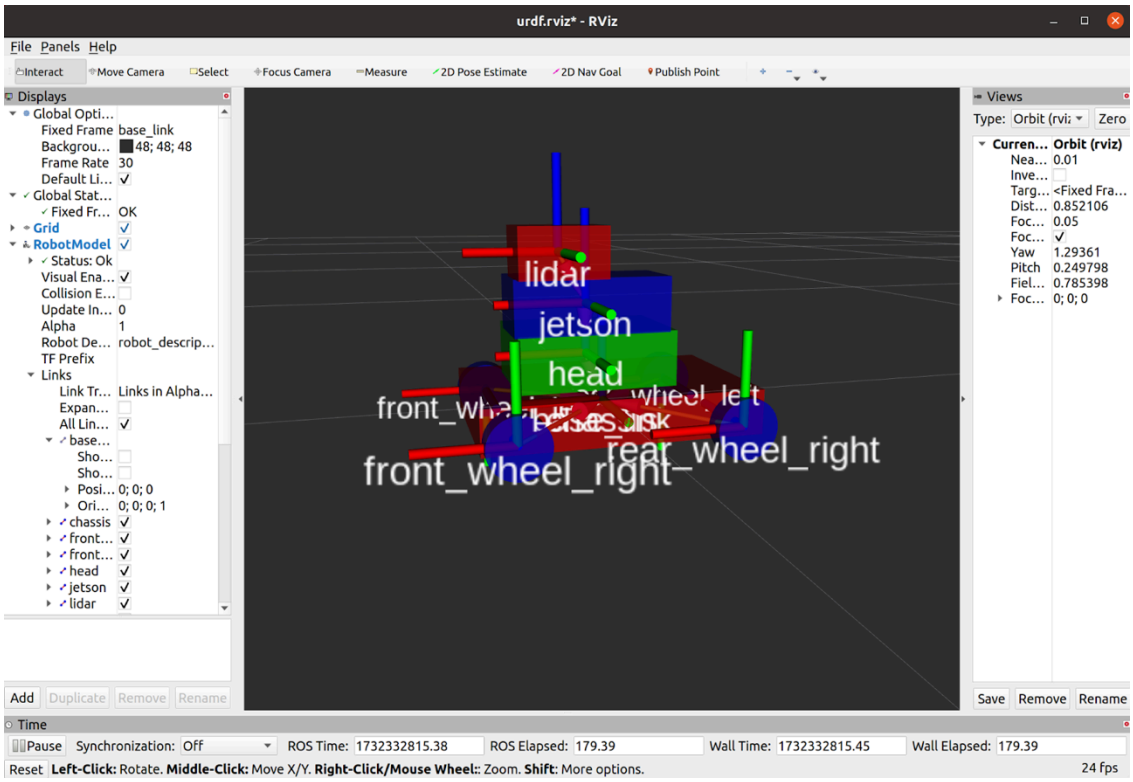


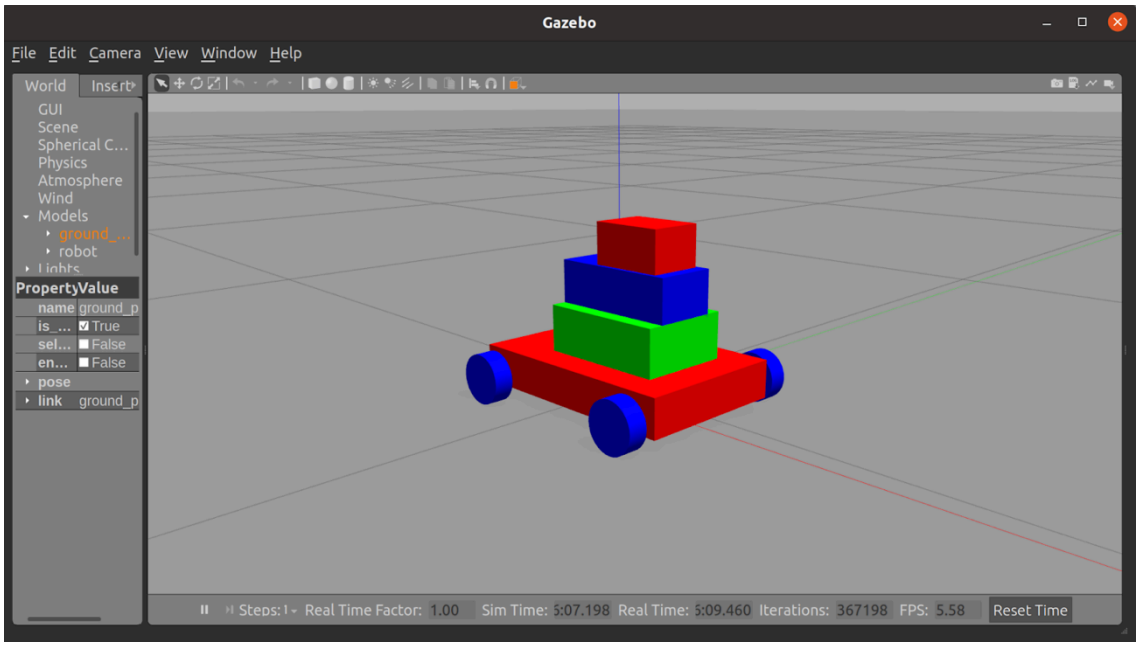
Description: Our rover has 3 layers. The first layer contains the ROS controller, the second layer contains the Jetson Nano, and the third layer contains the LiDAR.

Rover URDF

1. Rviz and Gazebo showing your "Robotmodel" similar to your assembled rover

    a) Corresponding URDF file with modifications: URDF and XACRO





Rover Simulation

1. Description: Using the modified URDF files and creating a model of our assemble rover, we moved the rover through simulation in Rviz and Gazebo by running teleop keyboard and joystick commands. We used the host machine's keyboard and a USB joystick.
2. Commands executed:
    a) roslaunch ros_mobile_robot drive_robot.launch
    b) roslaunch ros_mobile_robot mobile_robot.launch
    c) rosrun teleop_twist_keyboard teleop_twist_keyboard.py cmd_vel:=/robot_diff_drive_controller/cmd_vel
    d) rqt_robot_steering rqt_robot_steering
3. Video of your simulated rover moving in Rviz/gazebo when executing the
    a) teleop_keyboard command: key_gazebo
    b) joy_teleop command: joy_gazebo

Rover Testing

1. Description: Using teleop keyboard and joystick commands, we moved the actual rover using our host machine's keyboard and a USB joystick.
2. Commands executed:
    a) roslaunch yahboomcar_bringup yahboomcar.launch
    b) rosrun teleop_twist_keyboard teleop_twist_keyboard.py
    c) roslaunch yahboomcar_ctrl yahboom_joy.launch
3. Video of your actual when executing the
    a) teleop_keyboard command: keyboard
    b) joy_teleop command: joystick

LIDAR Testing

1. Testing and working of the LIDAR: We were able to successfully test the working of our LiDAR. This can be seen in the following sections for mapping, localization, and navigation. We then save the map using this command: rosrun map_server map_saver -f ~/yahboomcar_ws/src/yahboomcar_nav/maps/my_map Lidar

Mapping

1. Description: In this section the lidar camera is scanning the room locating obstacles and free space to maneuver. The goal is to fully scan a room with the robot to get a complete map of the room. Once the room is fully scanned we save the map to then use for localization.
2. Commands executed:
    a) roscore

b) roslaunch yahboomcar_nav laser_bringup.launch

c) roslaunch yahboomcar_nav yahboomcar_map.launch use_rviz:=false map_type:=gmapping

d) roslaunch yahboomcar_nav view_map. launch

e) rosrun teleop_twist_keyboard teleop_twist_keyboard.py

f) rosrun map_server map_saver -f ~/yahboomcar_ws/src/yahboomcar_nav/maps/my_map

3. Video of your actual rover and simulated rover mapping (gmapping) in Rviz/gazebo when executing the teleop_keyboard or joy_teleop commands:

https://youtu.be/pltYc0STbqU Mapping_Video2

Localization

1. Description: Once the map is saved we upload it to rviz in navigation viewing mode. We then adjust the robots directional view and position in the map using the 2D pose estimation tool to line up the current reading of the obstacles (red lines) with the map's obstacles in the room (black lines).
2. Commands executed:

a) roscore

b) roslaunch yahboomcar_nav laser_bringup.launch

c) roslaunch yahboomcar_nav yahboomcar_navigation.launch use_rviz:=false map:=my_map

d) roslaunch yahboomcar_nav view_navigate.launch

3. Video of your actual rover and simulated rover localization in Rviz/gazebo. Move the rover or use 2D Pose Estimation: https://youtu.be/3Si69RB9X2k Local_Video2

Navigation

1. Description: Once we have lined up the obstacles that are the same to match the pose of the robot and obstacles to its real place in the room, we used the 2D Pose goal tool to move the robot in the desired pose. The robot will use the map and current obstacles in mind to navigate itself to the desired position on the map.
2. Commands executed:
    a) roscore
    b) roslaunch yahboomcar_nav laser_bringup.launch
    c) roslaunch yahboomcar_nav yahboomcar_navigation.launch use_rviz:=false map:=my_map
    d) roslaunch yahboomcar_nav view_navigate.launch
3. Video of your actual rover and simulated rover navigation in Rviz/gazebo. Move the rover using 2D Nav Goal: https://youtu.be/l4VggpO8kI0 Nav_Video2

Collusion Avoidance

1. Description: The robot when encountering a collision detects obstacles and stops. It then calculates a new route to the goal pose then turns and drives around the obstacle.
2. Commands executed:
    1. roscore
    2. roslaunch yahboomcar_nav laser_bringup.launch

3. roslaunch yahboomcar_nav yahboomcar_navigation.launch use_rviz:=false map:=my_map
4. roslaunch yahboomcar_nav view_navigate.launch

3. Video of your actual rover and simulated rover navigation and avoidance in Rviz/gazebo.

Collision_Avoidance

References

DA7

DA6

See **12.12 Navigation and avoiding @**

http://www.yahboom.net/study/ROSMASTER-X1.