

CPE 521 - Final Project

The goal of this project is to utilize ROS to map an environment using SLAM given a dataset with the following data:

- Laser Scan Data
 - `front.csv`
 - `rear.csv`
- Odometry Data
 - `odom.csv`
- IMU Data
 - `imu.csv`

Dataset

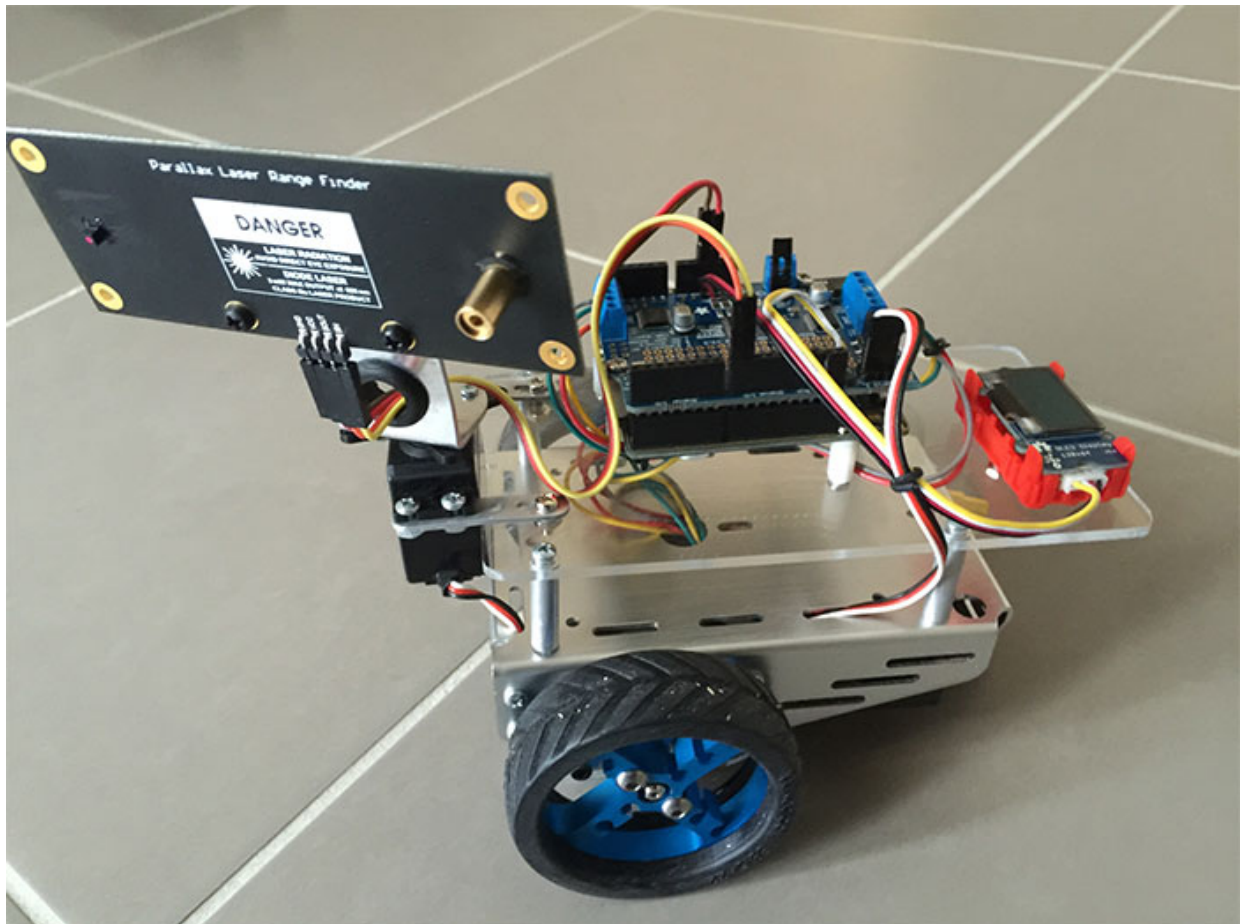
Given Data

From the files given, we will only be using the odometry and laser scan data.

Laser Scan Data

2 Laser Range Finders were installed on the robot. One in the front and one in the rear.

An example of a hardware setup that might get us this data is shown below:



The data is stored in a CSV file with the following format:

- **Column 1:** timestamp (in seconds)
- **Column 4-185:** distance readings (in meters) from the front laser range finder

Odometry Data

Odometry data could be obtained from the robot's wheel encoders. The data is stored in a CSV file with the following format:

- **Column 1:** timestamp (in seconds)
- **Column 5:** x position (in meters)
- **Column 6:** y position (in meters)
- **Column 7:** orientation (in radians)

Conversion

A 'bag' file is a ROS file format that stores ROS message data. We will be converting the given CSV files into a bag file to use with ROS.

The `csv_to_bag.py` script is ran and the bag file `output.bag` is created.

SLAM

Methodology

The robot will be using the `gmapping` package to perform SLAM.

The `gmapping` package is a ROS wrapper OpenSlam gmapping algorithm. It provides laser-based SLAM, as a ROS node. The SLAM approach is based on the Rao-Blackwellized particle filter utilizing range data from a laser range finder. The ROS node subscribes to the tf system and builds a map of the environment with a 2D occupancy grid using the laser and pose data.

Implementation

It's installed using the following command:

```
sudo apt-get install ros-melodic-gmapping
```

And can be ran using the following command:

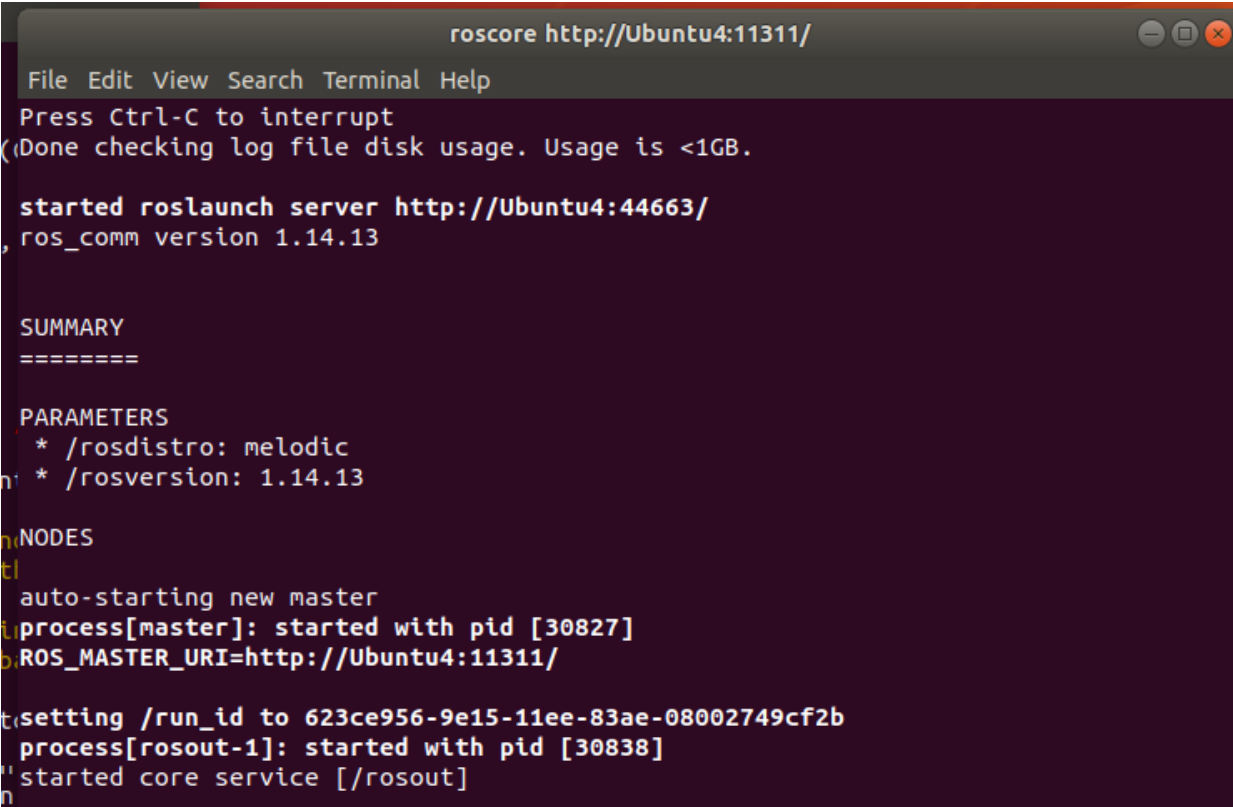
```
roslaunch gmapping slam_gmapping scan:=base_scan
```

Implementation

Starting the ROS Core

```
roscore
```

This will start the ROS Master that will act as the central hub for all ROS nodes.



```
roscore http://Ubuntu4:11311/
File Edit View Search Terminal Help
Press Ctrl-C to interrupt
(Done checking log file disk usage. Usage is <1GB.

started roslaunch server http://Ubuntu4:44663/
, ros_comm version 1.14.13

SUMMARY
=====

PARAMETERS
* /rostdistro: melodic
* /rosversion: 1.14.13

NODES
auto-starting new master
process[master]: started with pid [30827]
ROS_MASTER_URI=http://Ubuntu4:11311/

setting /run_id to 623ce956-9e15-11ee-83ae-08002749cf2b
process[rosout-1]: started with pid [30838]
started core service [/rosout]
```

Note that we have to set the simulation to use simulated time so that the time stamp on the messages will be the same as the time stamp on the messages in the bag file.

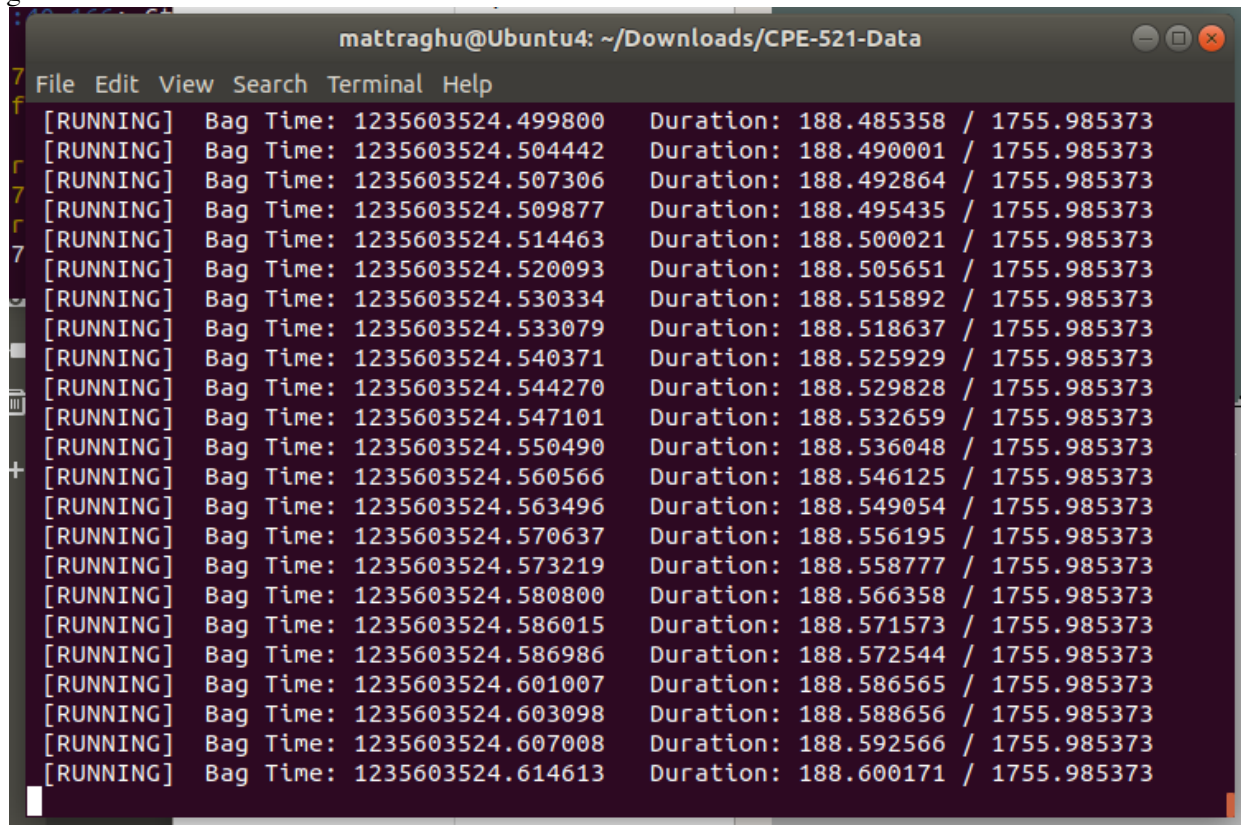
This is done by running the following command:

```
roscparam set use_sim_time true
```

Playing the Bag File

```
rosbag play output.bag
```

This will play the bag file and publish the messages to the ROS Master. Essentially, it will act as a live stream of the data as if it were coming from the robot in real time.

A terminal window titled 'mattraghu@Ubuntu4: ~/Downloads/CPE-521-Data' displays a list of 20 bag files. Each line shows the status '[RUNNING]', the file name 'Bag', the time, and the duration. The times range from 1235603524.499800 to 1235603524.614613, and the durations range from 188.485358 to 188.600171. The window has a menu bar with 'File', 'Edit', 'View', 'Search', 'Terminal', and 'Help'.

File	Time	Duration
[RUNNING] Bag	1235603524.499800	188.485358 / 1755.985373
[RUNNING] Bag	1235603524.504442	188.490001 / 1755.985373
[RUNNING] Bag	1235603524.507306	188.492864 / 1755.985373
[RUNNING] Bag	1235603524.509877	188.495435 / 1755.985373
[RUNNING] Bag	1235603524.514463	188.500021 / 1755.985373
[RUNNING] Bag	1235603524.520093	188.505651 / 1755.985373
[RUNNING] Bag	1235603524.530334	188.515892 / 1755.985373
[RUNNING] Bag	1235603524.533079	188.518637 / 1755.985373
[RUNNING] Bag	1235603524.540371	188.525929 / 1755.985373
[RUNNING] Bag	1235603524.544270	188.529828 / 1755.985373
[RUNNING] Bag	1235603524.547101	188.532659 / 1755.985373
[RUNNING] Bag	1235603524.550490	188.536048 / 1755.985373
[RUNNING] Bag	1235603524.560566	188.546125 / 1755.985373
[RUNNING] Bag	1235603524.563496	188.549054 / 1755.985373
[RUNNING] Bag	1235603524.570637	188.556195 / 1755.985373
[RUNNING] Bag	1235603524.573219	188.558777 / 1755.985373
[RUNNING] Bag	1235603524.580800	188.566358 / 1755.985373
[RUNNING] Bag	1235603524.586015	188.571573 / 1755.985373
[RUNNING] Bag	1235603524.586986	188.572544 / 1755.985373
[RUNNING] Bag	1235603524.601007	188.586565 / 1755.985373
[RUNNING] Bag	1235603524.603098	188.588656 / 1755.985373
[RUNNING] Bag	1235603524.607008	188.592566 / 1755.985373
[RUNNING] Bag	1235603524.614613	188.600171 / 1755.985373

Running the SLAM Node

```
roslaunch gmapping slam_gmapping scan:=front
```

This will run the SLAM node and subscribe to the `front` topic that is being published by the bag file.

It will then publish the map to the `map` topic.

```
mattraghu@Ubuntu4: ~/Downloads/CPE-521-Data
File Edit View Search Terminal Help
update ld=1.00787 ad=0.0101805
Laser Pose= 39.0909 52.045 1.687
m_count 35
Average Scan Matching Score=121.049
neff= 16.3751
Registering Scans:Done
update frame 6788
update ld=1.00578 ad=0.007
Laser Pose= 38.9725 53.0257 1.694
m_count 36
Average Scan Matching Score=121.415
neff= 16.8931
Registering Scans:Done
update frame 6905
update ld=1.00003 ad=0.00671468
Laser Pose= 38.8462 54.0173 1.70071
m_count 37
Average Scan Matching Score=115.385
neff= 16.8066
Registering Scans:Done
update frame 7024
update ld=1.00814 ad=0.0240801
Laser Pose= 38.7156 55.0166 1.68667
m_count 38
progress = float(progress)
```

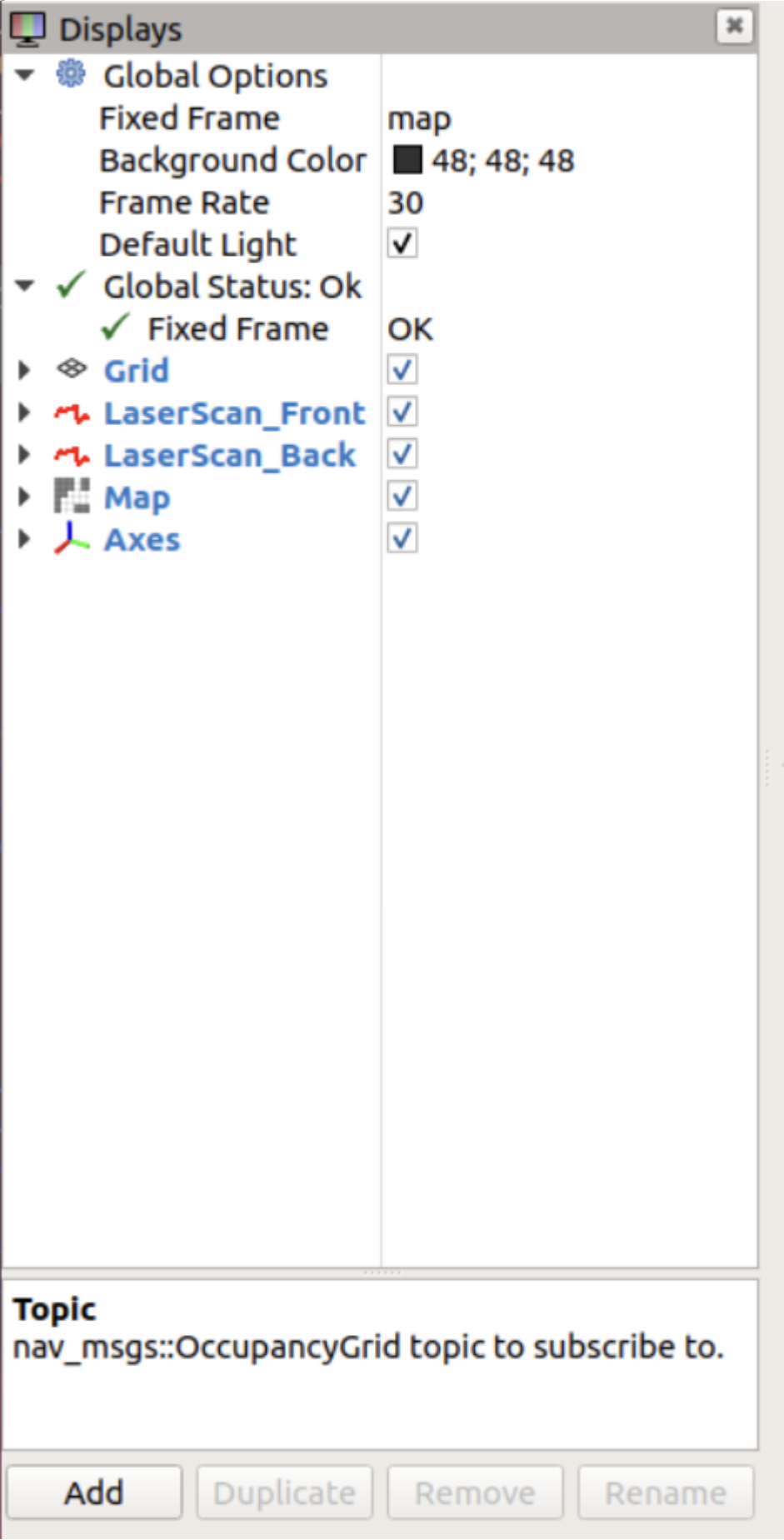
GUI Visualization

```
rviz
```

This will open the RViz GUI.

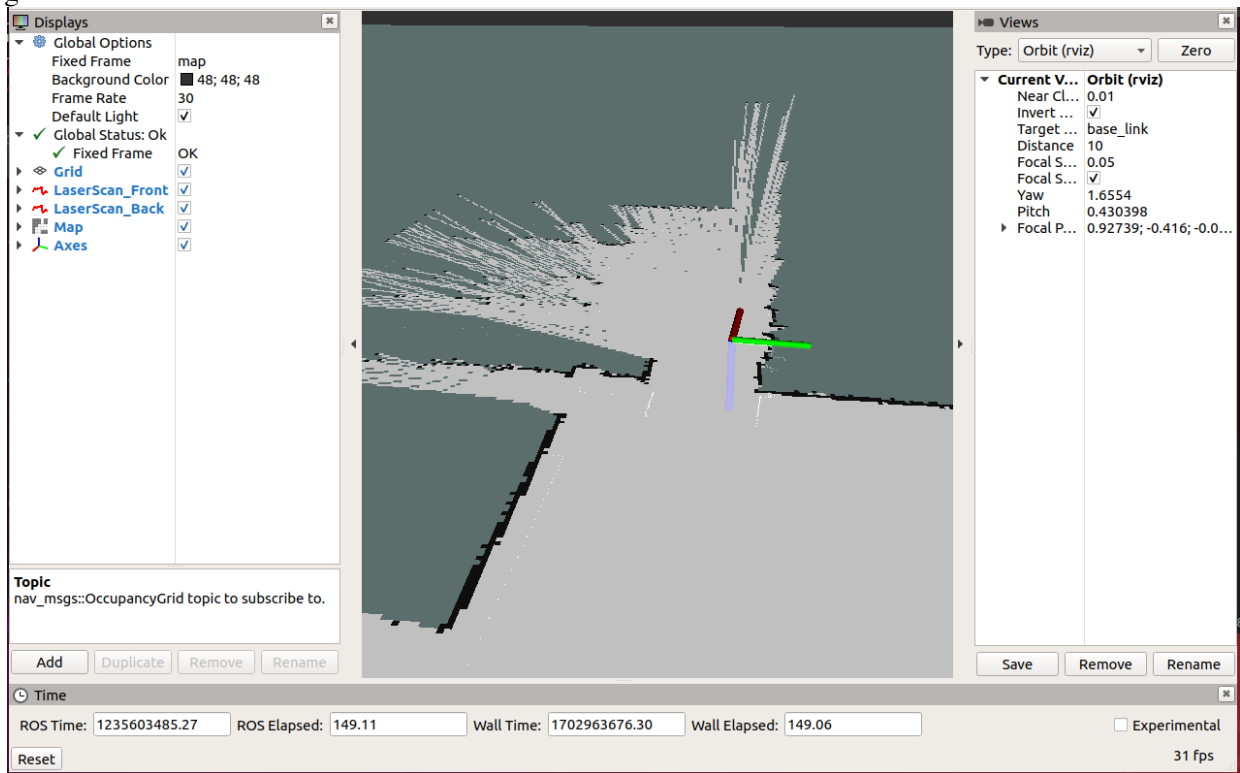
The following displays are added and configured:

- **Map** - Displays the map. Configured to subscribe to the `map` topic.
- **LaserScan_Front** - Displays the laser scan data from the front laser range finder. Configured to subscribe to the `front` topic.
- **LaserScan_Rear** - Displays the laser scan data from the rear laser range finder. Configured to subscribe to the `rear` topic.
- **Axes** - Displays the robot's current position and orientation. Configured to subscribe to the `odom` topic.
- **Grid** - Displays a grid on the map.



Results

The following is the map that was generated:



In simulation time, the robot was able to map the entire environment. From my tests, it seems that there is a slight delay between when the laser data is published and when the map is updated.

Discussion

During this lab, I learned a lot about ROS and how it works overall. From the way that the nodes communicate with each other to the way that the messages are published and subscribed to. I also learned about the `gmapping` package and how it can be used to perform SLAM.

In fact, for one of my projects that I'm working on, I'm going to have to use ROS to implement a SLAM algorithm on an autonomous drone. While I'm not going to be using openslam's `gmapping` package, I will be using ROS to implement the SLAM algorithm. This lab helped me get a better understanding of how ROS works and how I can use it to implement SLAM.