Smart Cleaning Robot

A ROS-Based Autonomous Cleaning Robot with Indoor Mapping, Coverage Path Planning, Voice Control, and Expandable Functionality

A COSI 119 Autonomous Robotics Final Project

Team Members: Pang Liu, Zhenxu Chen

Presentation: Pang Liu (Nov 23, 2024)

IRL Demo

Gazebo Simulation Demo

Modules and Responsibility Distribution

Panel Module (Pang Liu)
Self-designed GUI for program entrance
Integrated all modules in this module
Use buttons or voice to control program

Mapping Module (Zhenxu Chen)
Based on Explore_Lite package
Customized for the program
Explore fast and save the map

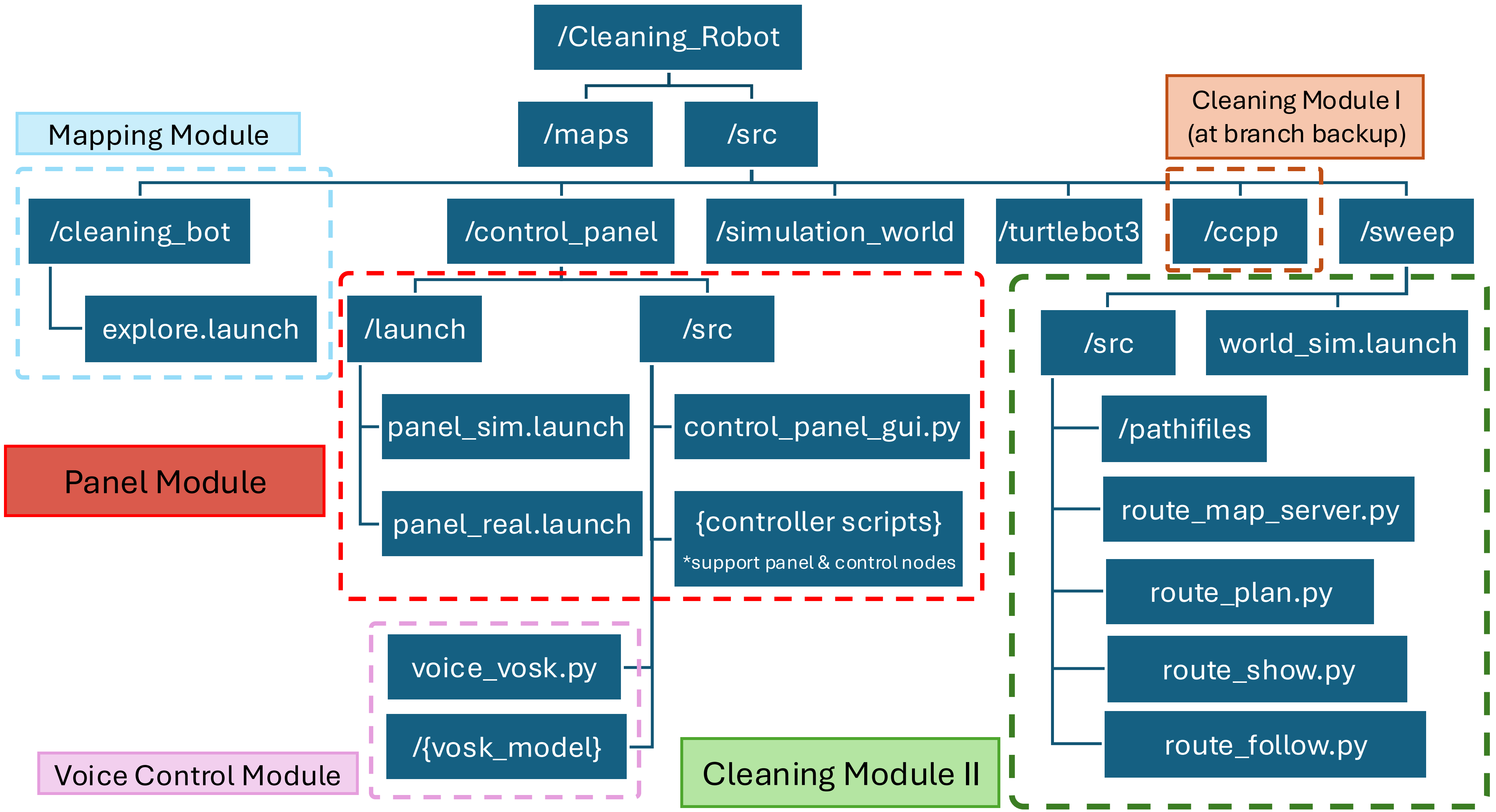
Voice Control Module (Pang Liu)
Based on Vosk model
Recognize voices
Real-time monitoring in the background

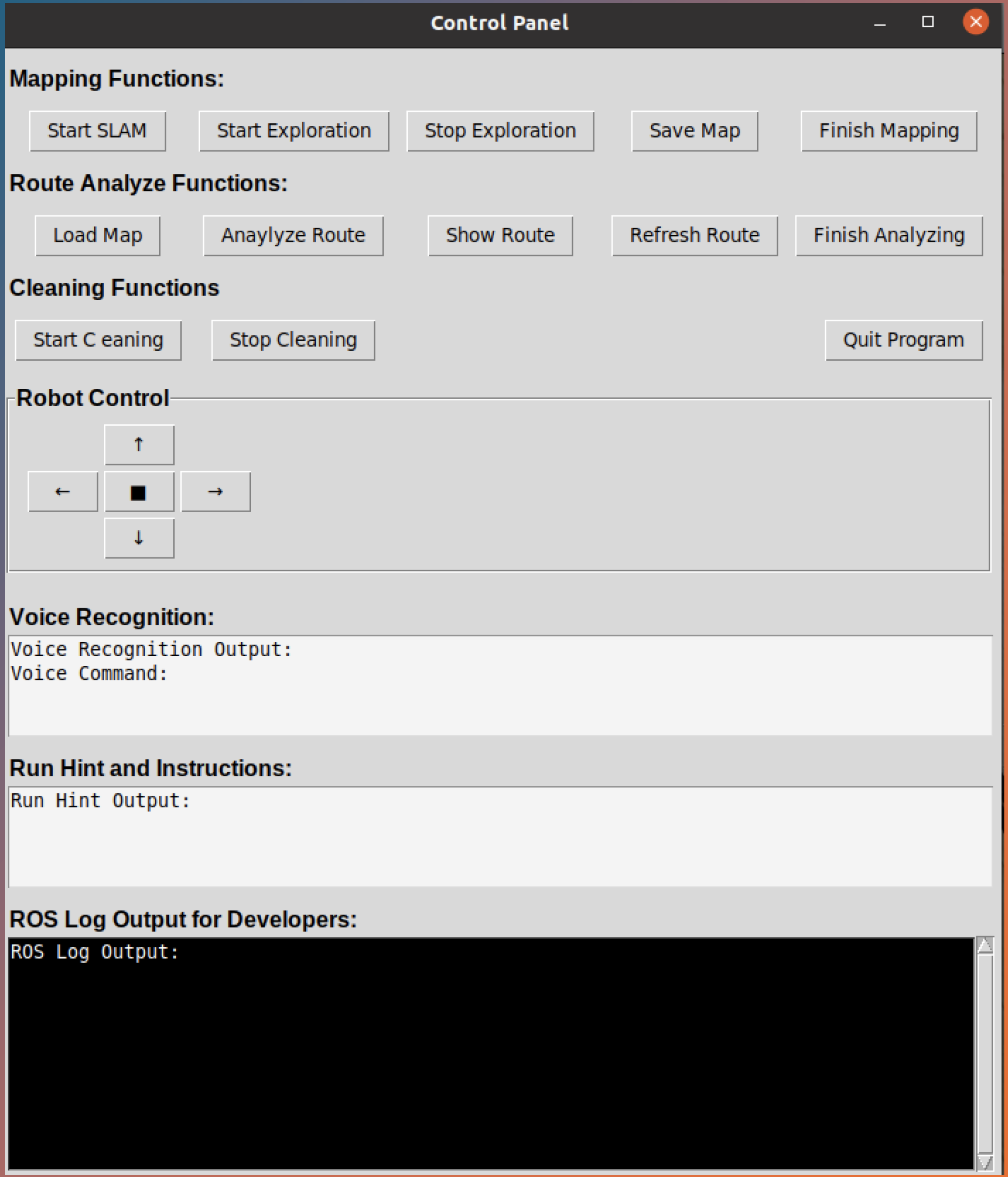
Two Different Cleaning Modules

Cleaning Module I (Zhenxu Chen)
Based on CCPP package and move_base
Customized for the program

Cleaning Module II (Pang Liu)
Sub Module I: Route Analysis
Fully Self-designed Route Analysis program
Load saved map and analyze cleaning route
Can mark with three kind of values: obstacles, uncleaned point, and cleaned point
Sub Module II: Route Follow
Fully Self-designed Route Follow program
DO NOT NEED move_base!

Main Project Files Directory Structures





```
Voice Recognition Output:
Voice Command:
```

```
Run Hint Output:
```

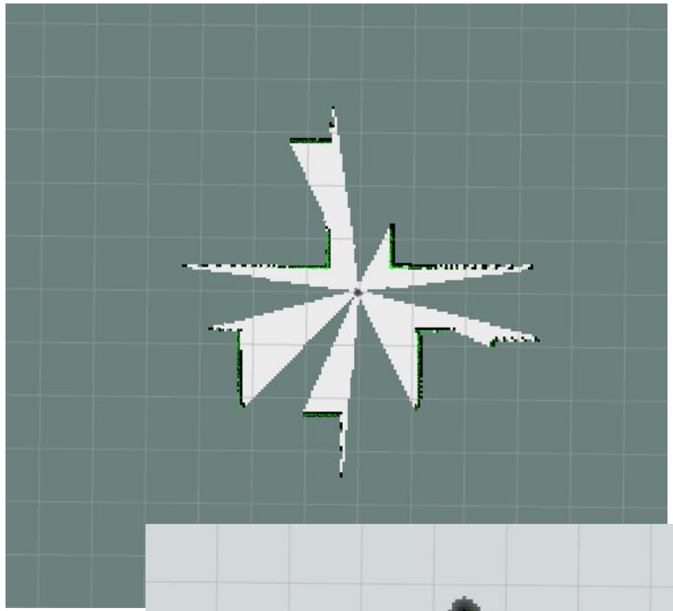
```
ROS Log Output:
```

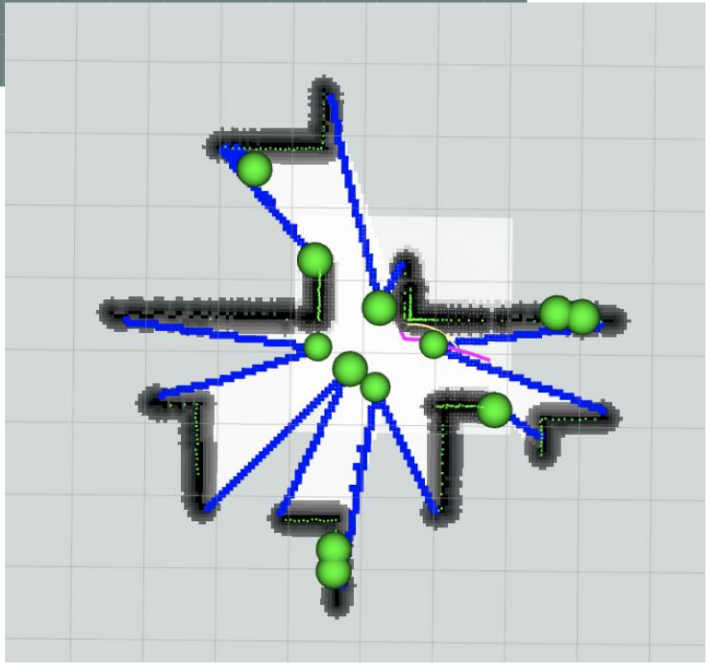
Panel Module

- All-in-on self designed GUI Panel to control all tasks and allow voice control.
- The recognized voice will show on the panel.
- Hint and instructions for people who don't understand codes.
- Can start SLAM, explore and build the map.
- Can save the multiple maps.
- Can load the newest map and analyze route and can show the points and route on Rviz.
- Can start cleaning and stop cleaning.
- Can quit program by clicking quit program.
- Allow robot control, don't worry hit the wall.
- Developer Friendly:
  - Can optimize route algorithm easily, just modify the algorithm in source file and click Refresh Route button.
  - The ROS Log Output helps developers find bugs.

Mapping Module

- **Step 1**: Click **[Start SLAM]** to open the SLAM gmapping and RViz.
- It will run the **turtlebot3_slam.launch** to start SLAM, RViz, and sensor supporting.





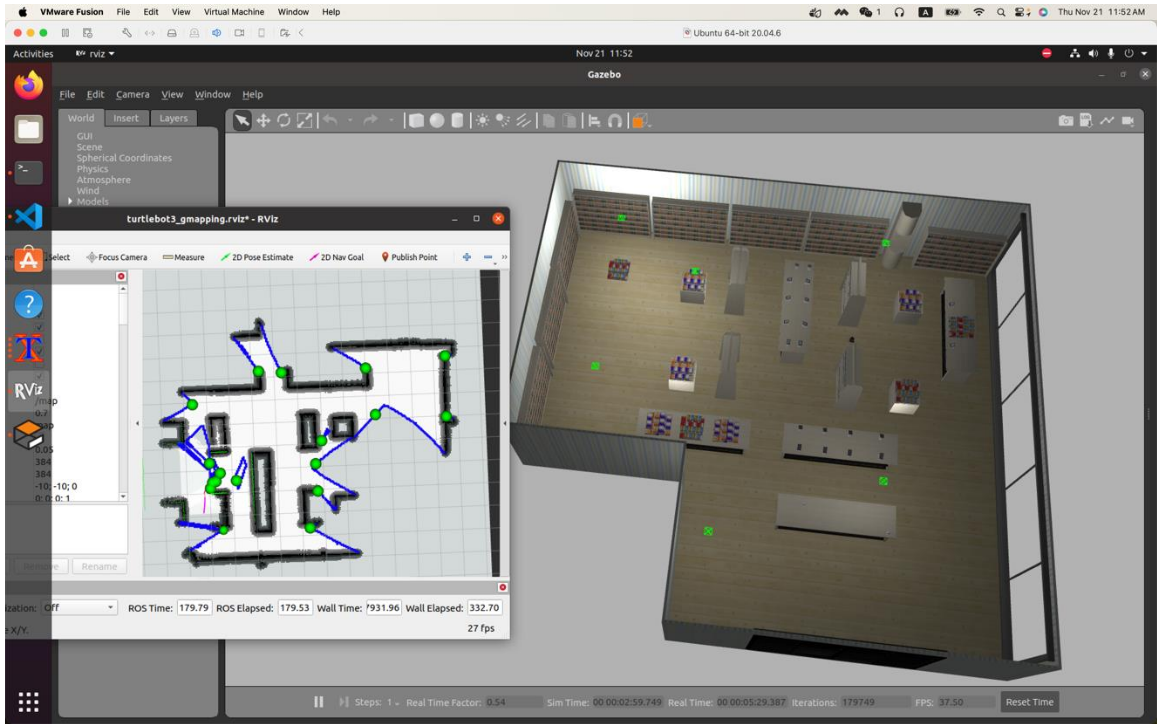
- **Step 2**: Click **[Start Exploration]** to start frontier exploration.
- It essentially runs the **explore.launch**, which is the launch file of the explore_lite package, used to start the robot's autonomous exploration function. Its main purpose is to allow the robot to explore unknown areas through the frontier-based exploration method and generate a complete map.

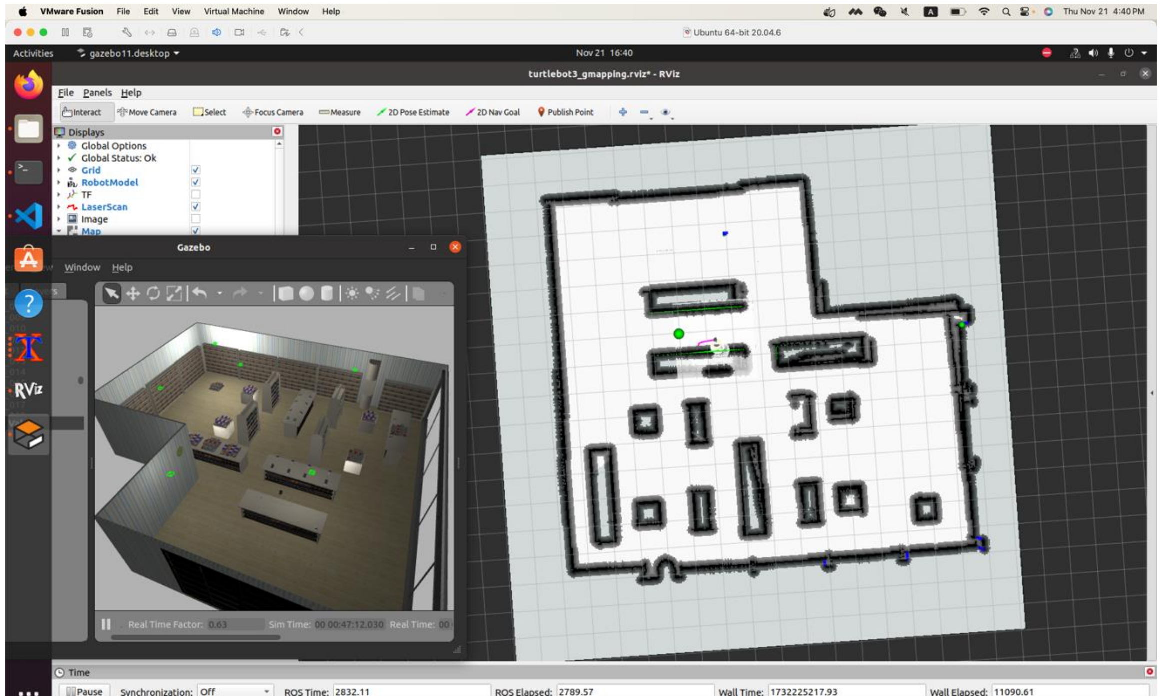
Mapping Module

The robot will continue exploring the map, using frontier exploration logic.

The user can see the map and click [Save Map] to save the map.

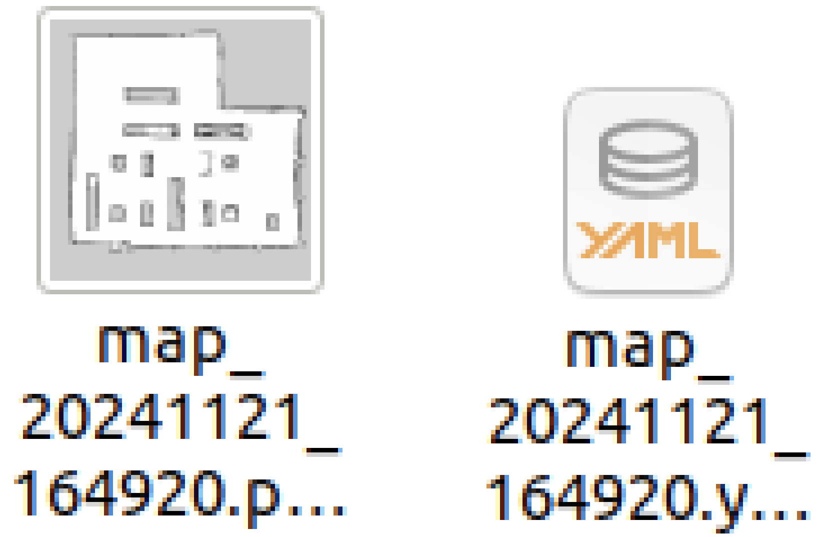
The robot will continue exploring for one hour, and automatically save the map and stop exploration, if user doesn't stop manually.



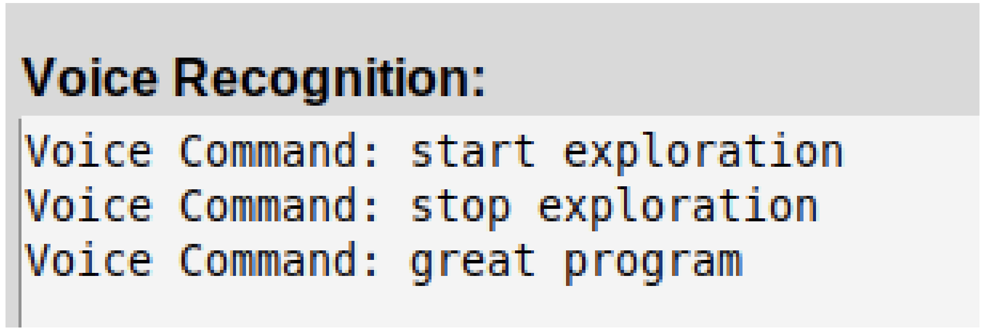


Mapping Module

- The user can manually stop exploration and save the map, or wait for the robot explore for one hour, automatically stop exploration and save the map.
- The map will save in two files:
- PGM and YAML files are two common file formats in ROS and robot-related applications, which are used to store map data and map metadata, respectively.



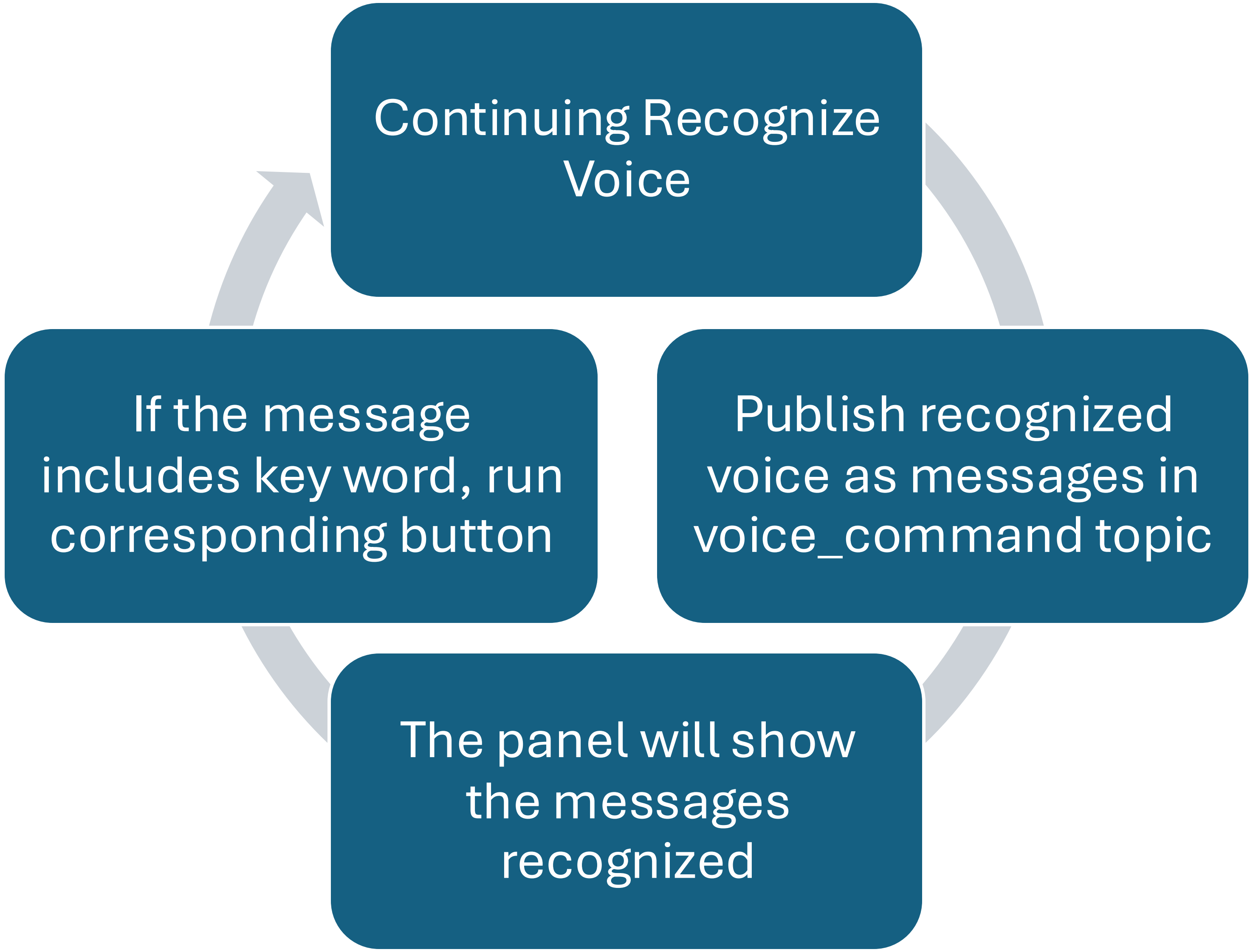
Voice Control Module



Voice Recognition:

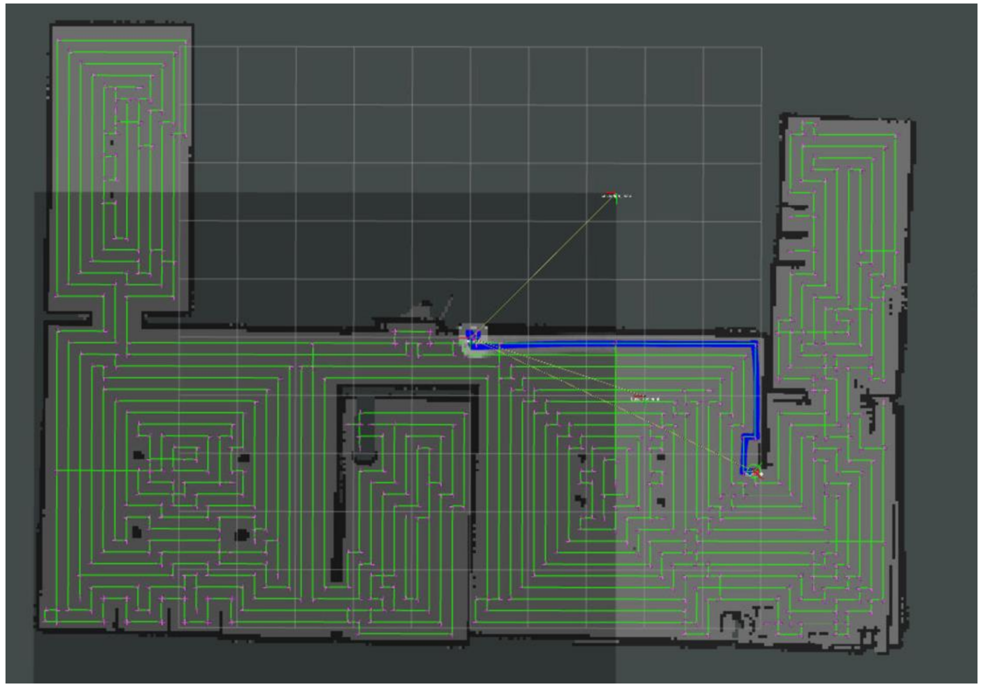
```
Voice Command: start exploration
Voice Command: stop exploration
Voice Command: great program
```

Vosk processes audio input in real-time by extracting features from the sound and using a pre-trained acoustic and language model to convert speech into text.
The recognized text is then published in voice_command topic, and continuing listened by the panel node, enabling the script to parse and act on specific voice commands.


Continuing Recognize Voice
Publish recognized voice as messages in voice_command topic
The panel will show the messages recognized
If the message includes key word, run corresponding button

Cleaning Module I

- The Cleaning Module I use CCPP package (https://wiki.ros.org/full_coverage_path_planner) to achieve the cleaning function.
- The CCPP package will use saved map to plan a full coverage route and allow the robot following the route.
- Video: https://drive.google.com/file/d/1F1Hh0JKD9KMvRVsC_EX5ZwptzUVWLEi8/view?usp=drive_link



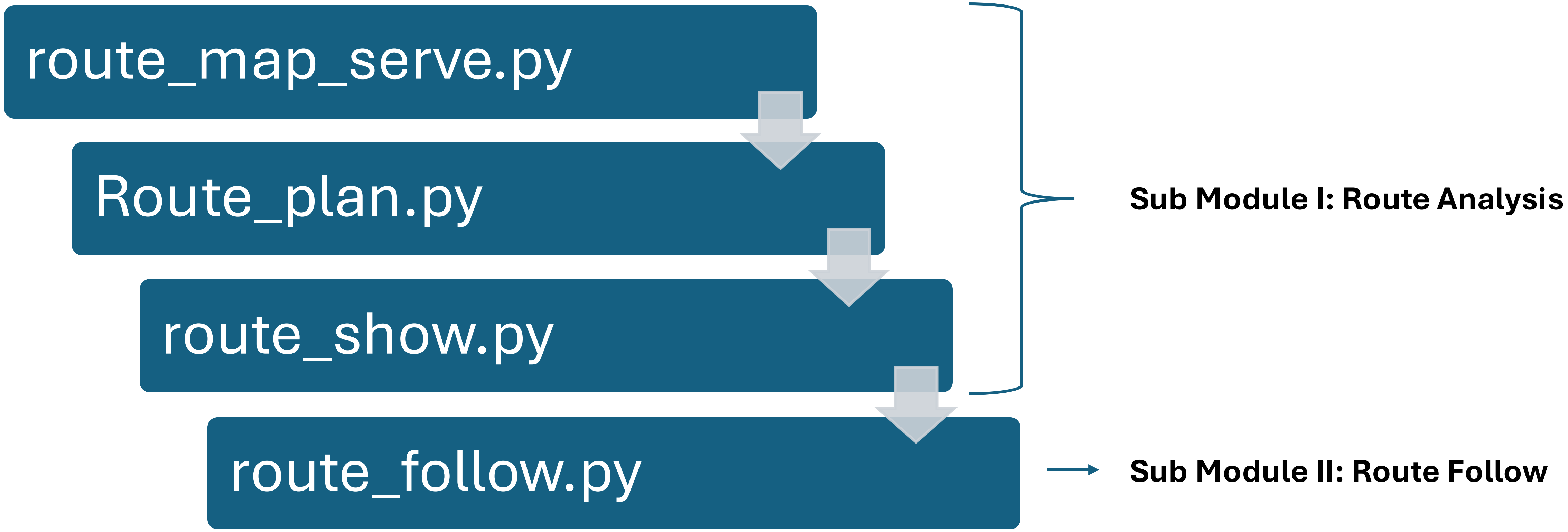
Cleaning Module II: Why Self Design?

- Innovative requirements for core functions:
  - **The cleaning function is the core function of the cleaning robot**, which is directly related to the actual value and user experience of the product. In order to make the product commercially competitive and innovative, the cleaning function needs to be independently developed, rather than simply relying on existing third-party packages.
- Customizability and scalability:
  - Although the use of ready-made packages can quickly implement functions, **it will limit the flexibility in subsequent development and cannot expand functions and optimize accuracy according to specific needs.** Therefore, the development team decided to design the cleaning function from scratch to better control the development process and technical implementation.

Cleaning Module II: Goal

- Improve technological innovation capabilities:
  - The self-developed cleaning function helps **accumulate core algorithms knowledge** and **lays a technical foundation for future product optimization** and innovation. It might seem stupid now, but with the continuous development, it will become more and more reliable and efficient.
- Assist education and scientific research (Further Plan)
  - The successfully developed robot cleaning function can be used as part of research and laboratory equipment to promote more scientific research cooperation and application scenario exploration.
  - The project will be continuing update and will be made into a tutorial for new learners to know how to make a cleaning robot step by step.
  - The frontier exploration module will also be self designed after this final term is finished completely.
  - **It will be reorganized to an open-source project** on smart cleaning robot, and welcome everyone come to do it together!

Cleaning Module II: Processing Sequence



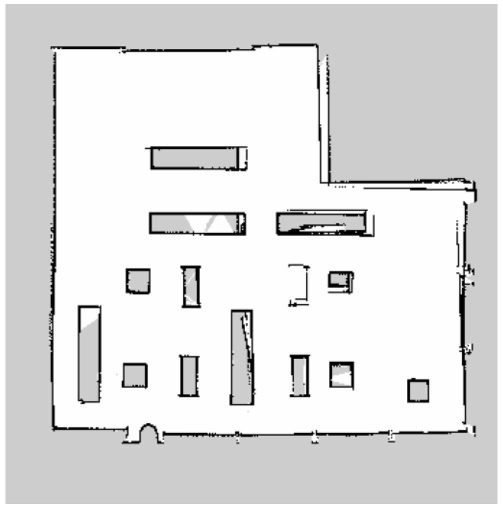
Cleaning Module II => Sub Module I Route Plan Logic (1) – (5)

- Detailed introduction of route_plan.py (core script):
  - (1) Get the latest map (map data of OccupancyGrid message type) through /map topic.
  - (2) Convert OccupancyGrid data to a grid map represented by a NumPy array.
  - (3) Perform obstacle expansion on the map (taking into account the safety distance of the robot)
  - (4) **Generate a three-value map**: -1, 0, and 1 are used to represent obstacles, unvisited areas, and visited areas respectively
  - (5) Generate path points in the map through a fixed sampling interval. Each path point includes world coordinates and grid coordinates

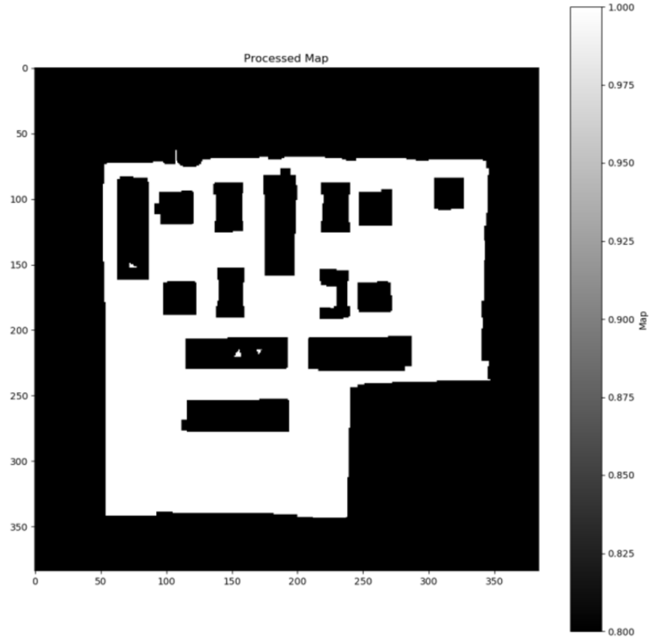
Cleaning Module II => Sub Module I
Route Plan Logic (6) – (7)

- (6) **Use a greedy algorithm to find valid connections between path points and check whether there are obstacles between two points**
- (7) After the connection is completed, use matplotlib to draw the path points and connected line segments and save them as image。
  - The logic of finding valid connections here is:
  - Each path point can only be connected to the path points connected to it.
  - The definition of connection is up, down, left, and right
  - Isolated path points are not considered in the connection

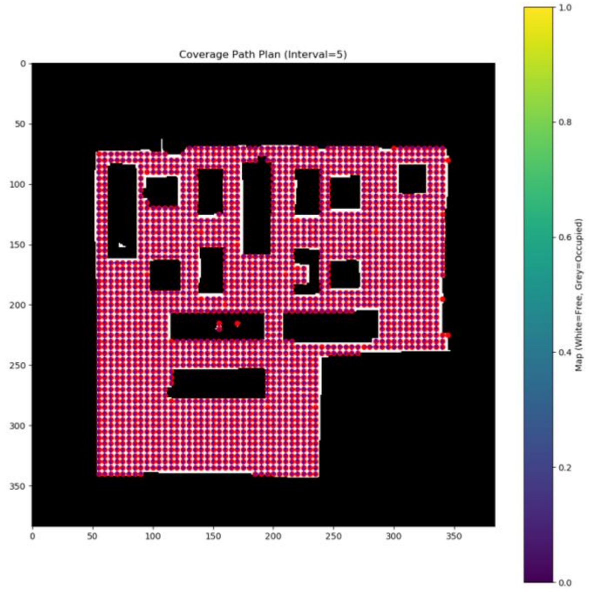
Cleaning Module II => Sub Module I Route Plan Logic



Original map file

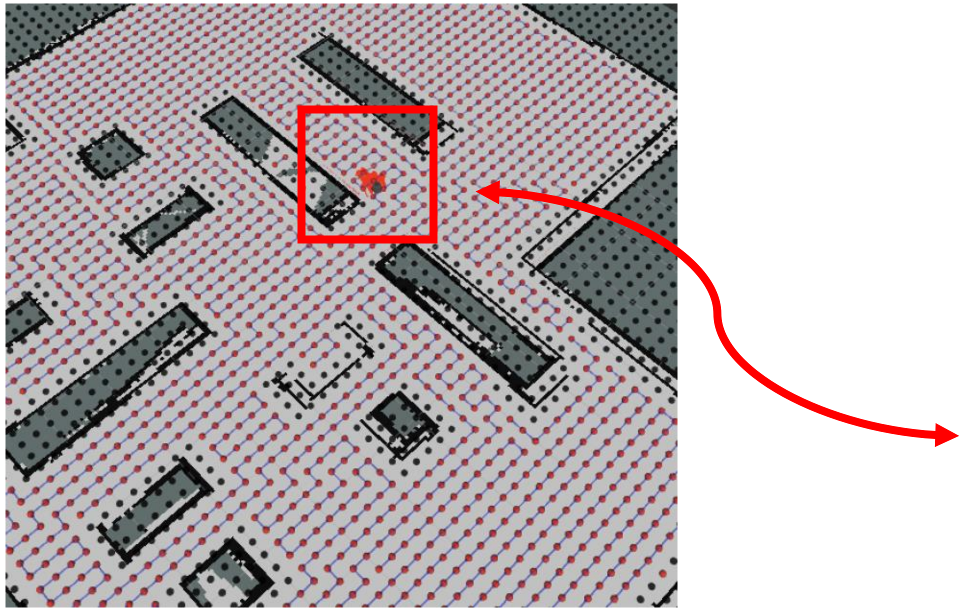


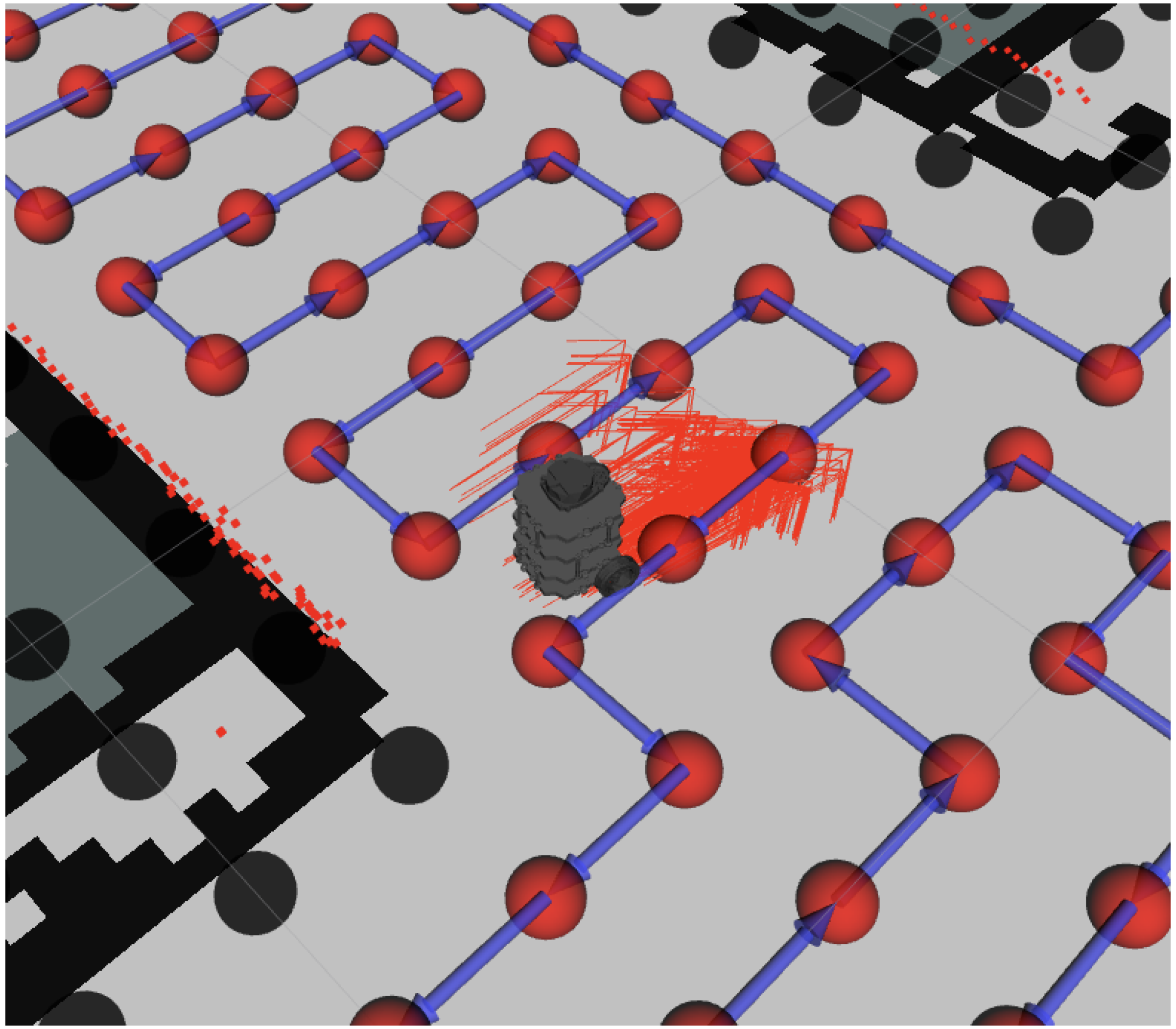
Expand the map to include obstacles (taking into account the robot's safety distance)


Coverage Path Plan (Interval=5)
Map (White=Free, Grey=Occupied)

Map with waypoints and directional arrows

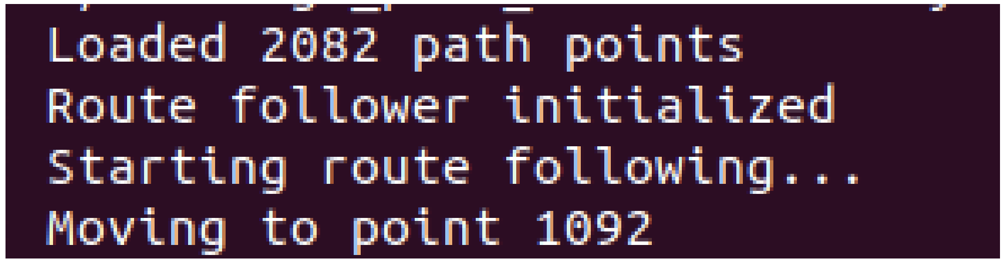
Cleaning Module II => Sub Module I Planned Route showed in RViz





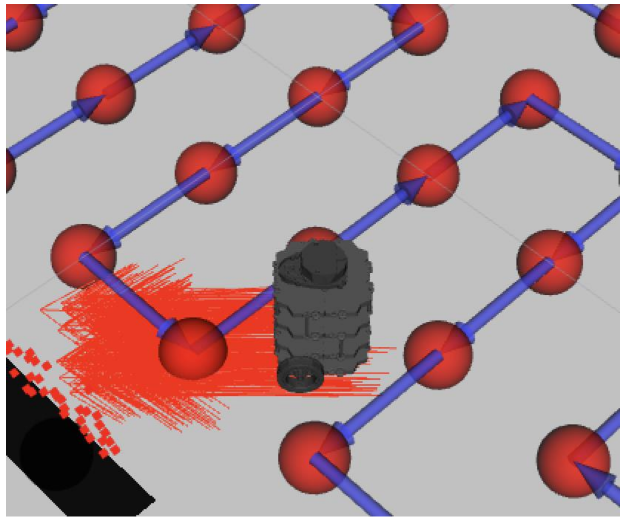
You can see red point (uncleaned point)
And the path with arrow now!!

Cleaning Module II => Sub Module II Route Following (Still Debugging, Not Launched Yet)



```
Loaded 2082 path points
Route follower initialized
Starting route following...
Moving to point 1092
```

The robot will follow the points and planned route direction, following the red points one by one.
Once it reached a red point, that point will turn to green point, meaning the point that has been cleaned.



Supported Modules

- /simulation_world
  - Provide different simulation maps for different circumstances
  - Allow testing in simulations efficiency, allowing agile development
  - Use world:={map} to open corresponding map
    - Eg. (bash$) roslaunch control_panel panel_sim.launch world:=house

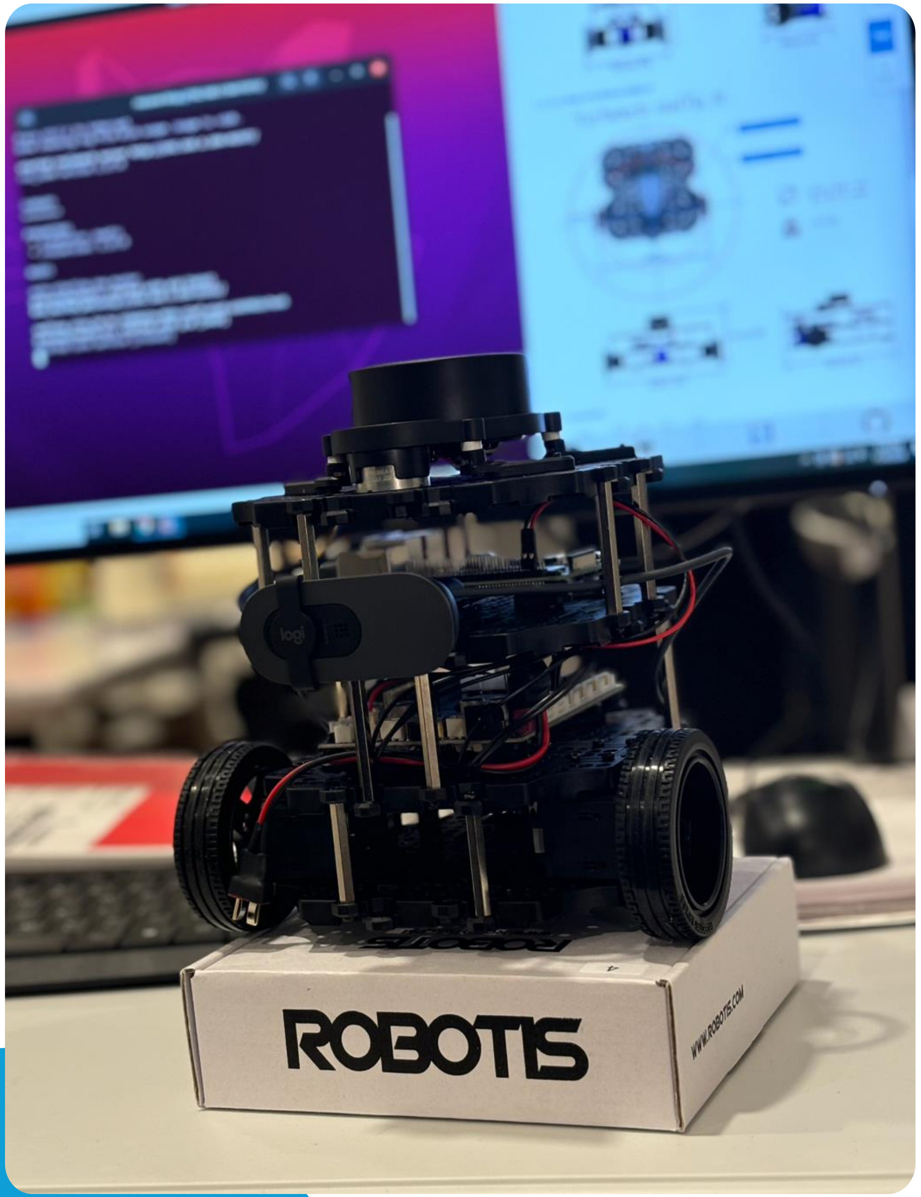
bookstore (default)

house

warehouse

Supported Modules

- /turtlebot3
  - Provide turtlebot3 supported gmapping SLAM, used in Mapping Module
  - Provide the required support for TurtleBot3 hardware and Gazebo simulation models includes LiDAR and IMU (Inertial Measurement Unit) sensors.



Turtlebot3 Built by ourselves