SMART WAREHOUSE

Santiago Herron Mulet
Robot Programming 2025

Introduction

Context:

- The project focuses on a Smart Warehouse system designed to optimize storage and retrieval processes.
- Implemented using ROS (Robot Operating System) to demonstrate communication and coordination between robotic components.

Objective:

- Develop a system where robots communicate efficiently to perform warehouse tasks.
- Showcase the integration of nodes, topics, services, parameters, and custom messages in a real-world scenario.

Motivation:

- Automating warehouse operations improves efficiency, reduces human error, and optimizes resource use.

The Program

General Description:

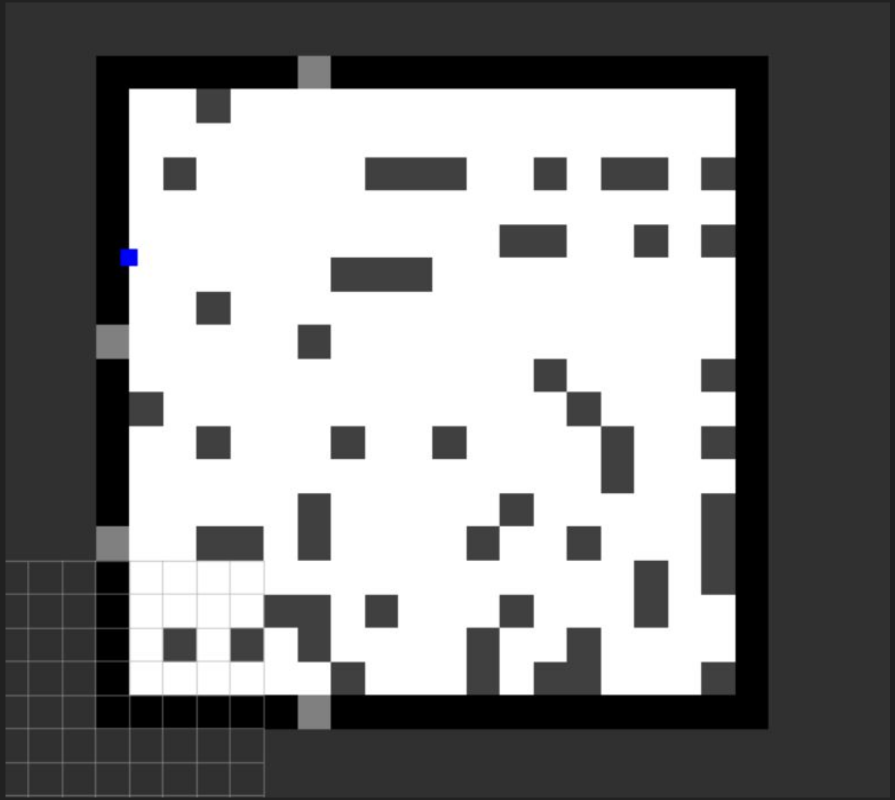
- The program is designed to simulate a warehouse robot navigation system using ROS (Robot Operating System) and Rviz.
- The system includes generating a warehouse map, managing tasks, and controlling the robot's movement.

Main Components:

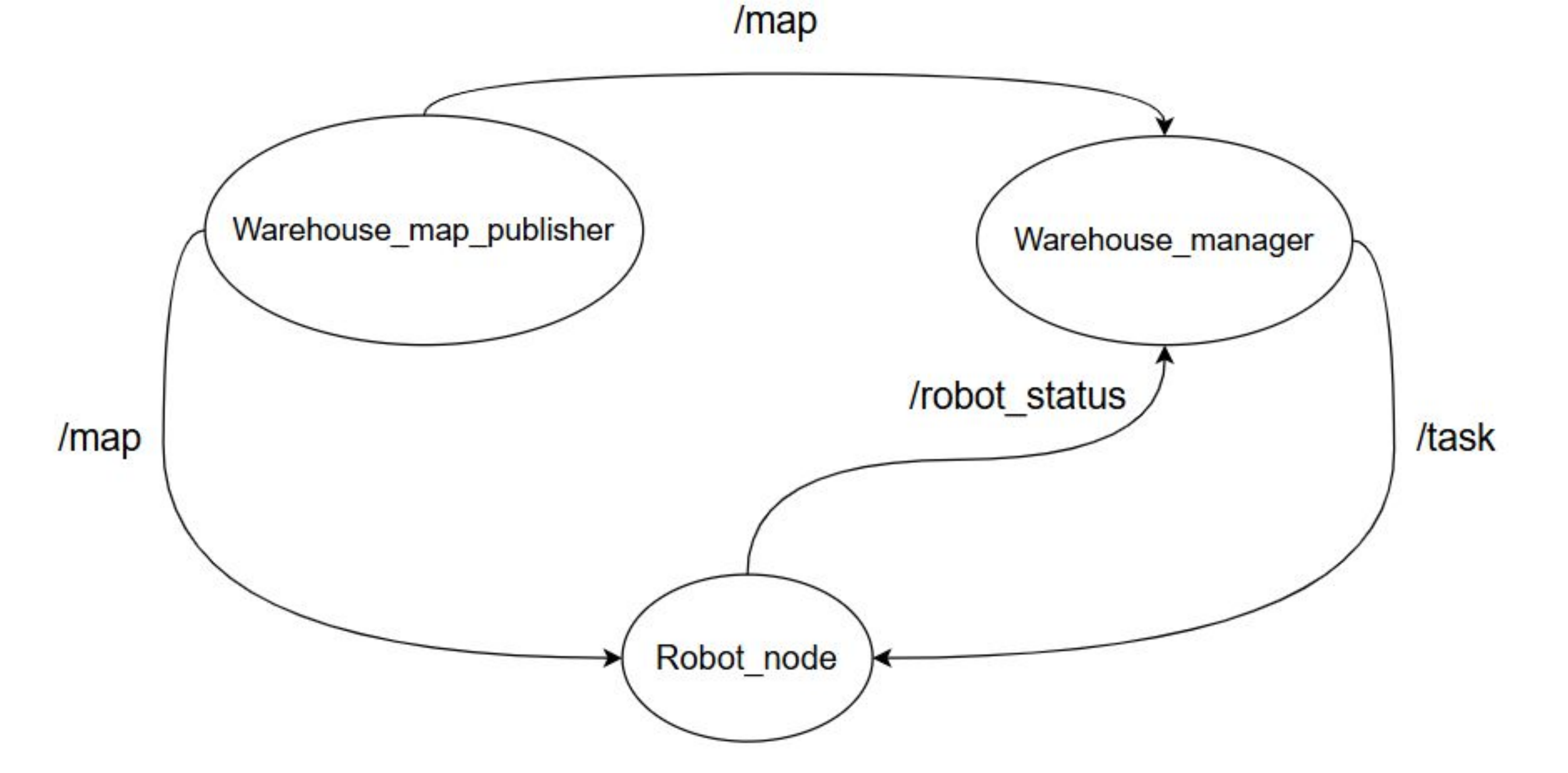
- **Map Generation**: The system generates a map of the warehouse, including charging/loading stations and shelves.
- **Task Management**: The system creates and assigns tasks to the robot, such as loading and unloading items.
- **Robot Control**: The robot navigates the warehouse to complete the assigned tasks and reports its status.

Workflow

- Step 1: The map of the warehouse is generated and published.
- Step 2: Load or Unload tasks are generated and assigned to the robot.
- Step 3: The robot navigates to the specified locations to complete the tasks.
- Step 4: The robot's status is monitored and new tasks are assigned as needed.



Nodes & topics.



Nodes: warehouse_map_publisher

Node Purpose:

- Generate and publish the warehouse map.

Main Functions:

- Initialize the node and read warehouse parameters.
- Create a warehouse map with charging/loading stations and shelves.
- Publish the map on the /map topic.
- Provide a create_map service to generate a new map on demand.

Example of Interaction:

- warehouse_map_publisher publishes the map on the /map topic.
- Other nodes, such as robot_node, subscribe to the /map topic to obtain the map information.

Nodes: warehouse_manager

Node Purpose:

- Manage tasks and assign them to the robot.

Main Functions:

- Initialize the node and subscribe to the warehouse map.
- Generate load/unload tasks and publish them on the /task topic.
- Provide a generate_tasks service to generate new tasks on demand.
- Subscribe to the robot's status on the /robot_status topic and assign new tasks when the robot completes a task.

Example of Interaction:

- warehouse_manager generates a task and publishes it on the /task topic.
- robot_node subscribes to the /task topic and receives the task.

Nodes: robot_node

Node Purpose:

- Control the robot's movement and publish its status.

Main Functions:

- Initialize the node and subscribe to the warehouse map and tasks.
- Navigate to the locations specified in the tasks.
- Publish the robot's status on the /robot_status topic.
- Provide a pause_resume service to pause or resume the robot.

Example of Interaction:

- robot_node subscribes to the /task topic and receives a task.
- robot_node navigates to the specified location and publishes its status on the /robot_status topic.
- warehouse_manager subscribes to the /robot_status topic and assigns new tasks when the robot completes a task.

Topics: /map

Publisher:

- Node: warehouse_map_publisher.
- Message Type: OccupancyGrid.

Subscribers:

- Nodes: robot_node, warehouse_manager
- robot_node: Subscribes to obtain the map information for navigation.
- warehouse_manager: Subscribes to obtain the map information for task generation.

Topics: /task

Publisher:

- Node: warehouse_manager.
- Message Type: Task. (custom).

Subscribers:

- Node: robot_node
- Purpose: To receive tasks and navigate to the specified locations.

Topics: robot_status

Publisher:

- Node: robot_node.
- Message Type: RobotStatus2 (custom)

Subscribers:

- Node: warehouse_manager.
- Purpose: To monitor the robot's status and assign new tasks when the robot completes a task.

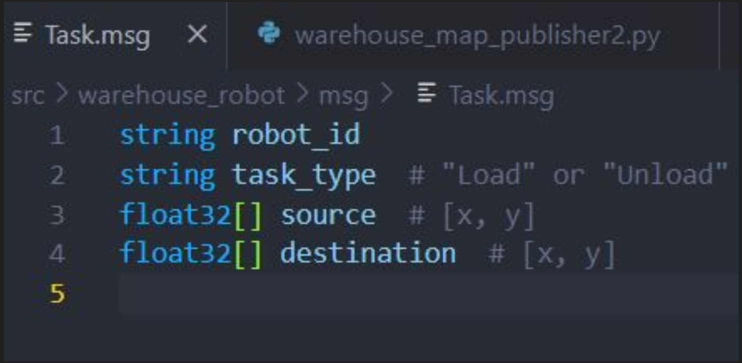
Custom Messages: Task

Fields:

- string robot_id: ID of the robot assigned to the task.
- string task_type: Type of task (e.g., "Load" or "Unload").
- float32[] source: Source location of the task.
- float32[] destination: Destination location of the task.

Example Usage:

- Published by the warehouse_manager node on the /task topic.
- Subscribed by the robot_node to receive tasks and navigate to the specified locations.


Task.msg
warehouse_map_publisher2.py
src > warehouse_robot > msg > Task.msg

```
string robot_id
string task_type  # "Load" or "Unload"
float32[] source  # [x, y]
float32[] destination  # [x, y]

```

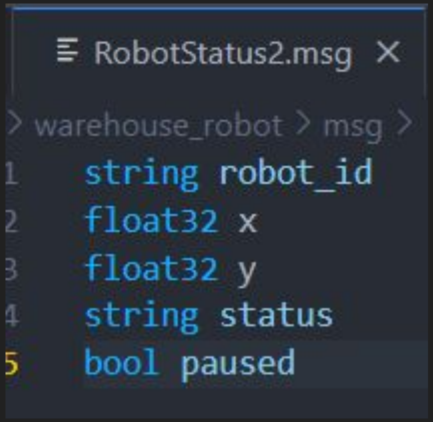
Custom Messages: RobotStatus2

Fields:

- string robot_id: ID of the robot.
- float32 x: X-coordinate of the robot's position.
- float32 y: Y-coordinate of the robot's position.
- string status: Current status of the robot (e.g., "InProgress" or "Completed").
- bool paused: Indicates whether the robot is paused.

Example Usage:

- Published by the robot_node on the /robot_status topic.
- Subscribed by the warehouse_manager to monitor the robot's status and assign new tasks when the robot completes a task.



```
string robot_id
float32 x
float32 y
string status
bool paused
```

Services: PauseResume

Purpose:

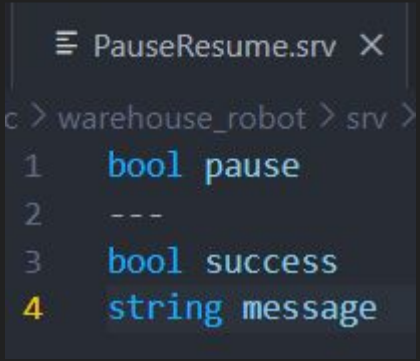
- The PauseResume service is used to pause or resume the robot's operation.

Request Fields:

- bool pause: Indicates whether to pause (true) or resume (false) the robot.

Response Fields:

- bool success: Indicates whether the operation was successful.
- string message: Provides additional information about the operation.



```
bool pause
---
bool success
string message
```

Services: CreateMap

Purpose:

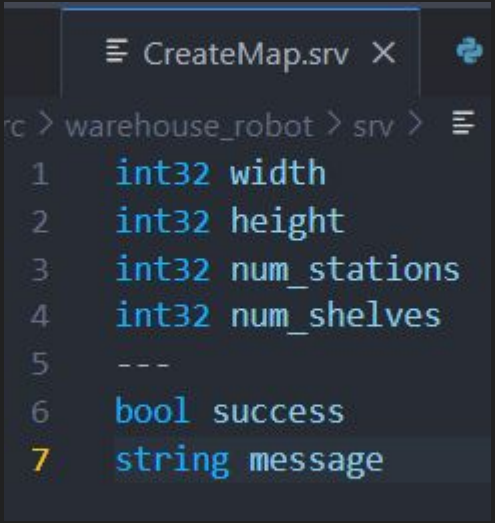
- The CreateMap service is used to generate a new map of the warehouse with specified dimensions, number of stations, and number of shelves.

Request Fields:

- int32 width: Width of the warehouse.
- int32 height: Height of the warehouse.
- int32 num_stations: Number of charging/loading stations.
- int32 num_shelves: Number of shelves.

Response Fields:

- bool success: Indicates whether the map creation was successful.
- string message: Provides additional information about the operation.



```
int32 width
int32 height
int32 num_stations
int32 num_shelves
---
bool success
string message
```

Services: GenerateTasks

Purpose:

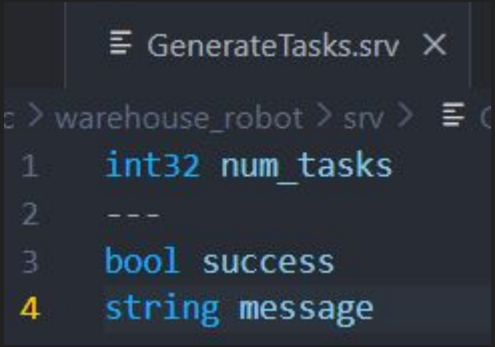
- The GenerateTasks service is used to generate a specified number of tasks for the robot.

Request Fields:

- int32 num_tasks: Number of tasks to generate.

Response Fields:

- bool success: Indicates whether the task generation was successful.
- string message: Provides additional information about the operation.



```
int32 num_tasks
---
bool success
string message
```

Parameters: Warehouse Map Parameters

Purpose:

- These parameters define the dimensions and layout of the warehouse map.

Parameters:

- warehouse_width: Width of the warehouse.
- warehouse_height: Height of the warehouse.
- num_stations: Number of charging/loading stations.
- num_shelves: Number of shelves.

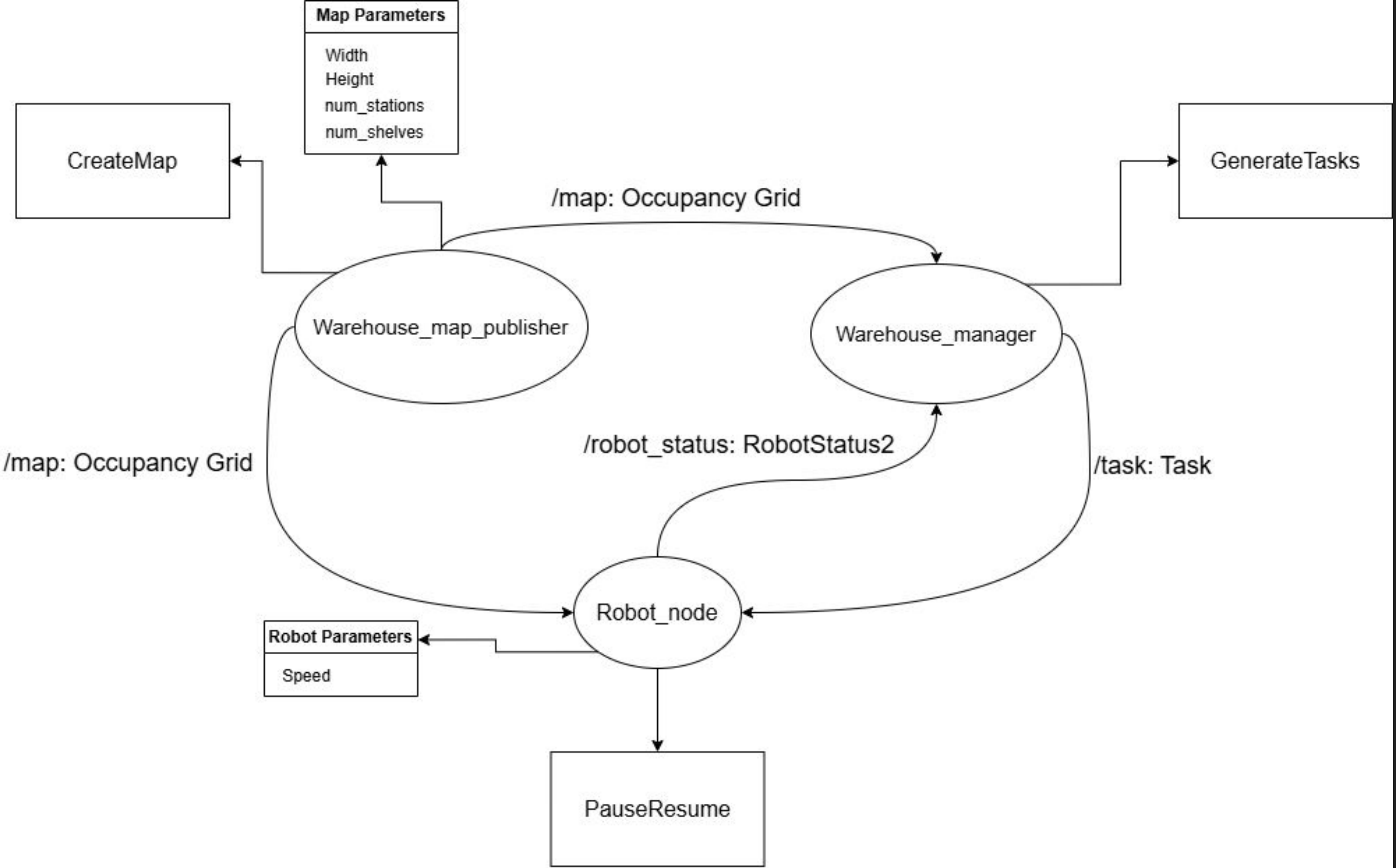
Parameters: Robot Parameters

Purpose:

- These parameters define the operational characteristics of the robot.

Parameters:

- robot_speed: Speed of the robot.


Map Parameters
Width
Height
num_stations
num_shelves
CreateMap
GenerateTasks
/map: Occupancy Grid
Warehouse_map_publisher
Warehouse_manager
/robot_status: RobotStatus2
/map: Occupancy Grid
/task: Task
Robot_node
Robot Parameters
Speed
PauseResume

Project Development Process

Incremental Development:

- Developed step-by-step, adding features incrementally.

Versioning:

- Version 1: Basic node setup and initial map generation.
- Version 2: Task generation and basic robot navigation.
- Version 3: Added walls and detailed map representation.
- Version 4: Task management and robot status updates.
- Version 5: Dynamic map creation and task generation services.
- Version 6 (final): Pause/resume functionality and RViz visualization.

Benefits:

- Early issue detection.
- Continuous integration and testing.
- Iterative improvements.

Implementation Structure

The project is organized into several directories and files, each serving a specific purpose.

```
warehouse_robot/
├── scripts/
│   ├── warehouse_map_publisher6.py
│   ├── warehouse_manager6.py
│   └── robot_node6.py
├── msg/
│   ├── Task.msg
│   └── RobotStatus2.msg
├── srv/
│   ├── CreateMap.srv
│   ├── GenerateTasks.srv
│   └── PauseResume.srv
├── launch/
│   └── warehouse_simulation6.launch
├── urdf/
│   └── robot.urdf
├── package.xml
└── CMakeLists.txt
```

Visualization with Rviz

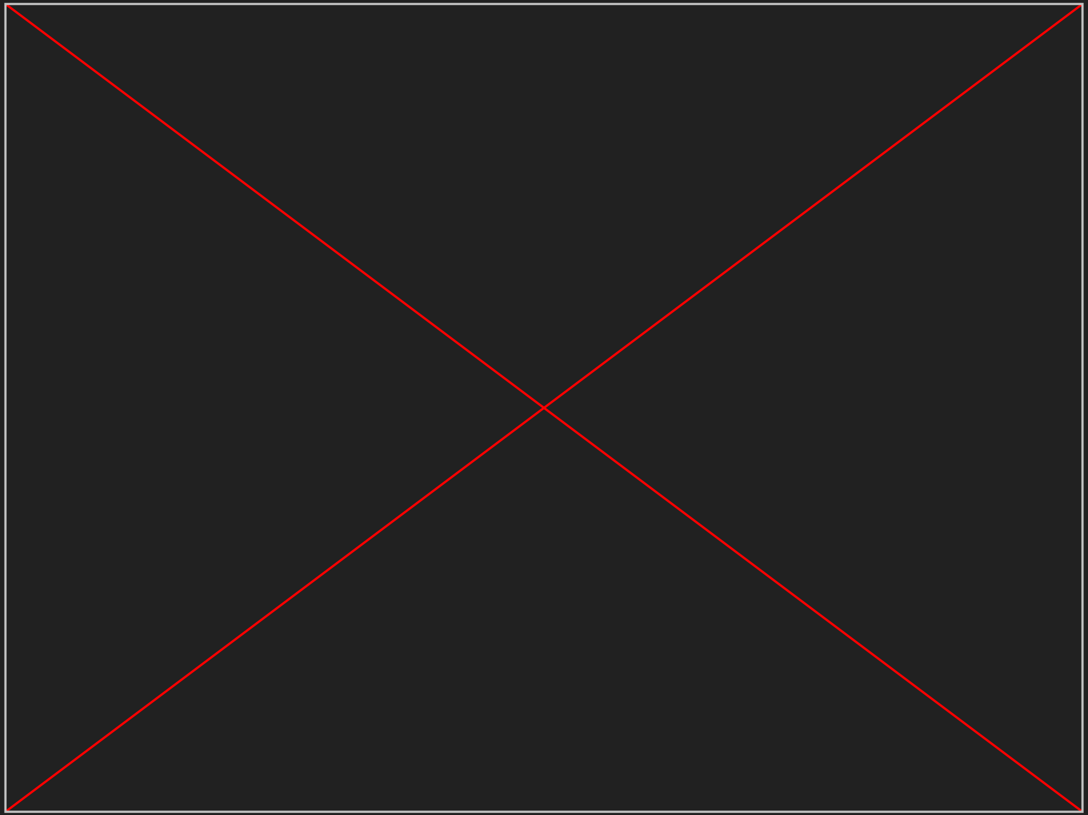
OccupancyGrid:

- Purpose: Represents the warehouse map as a 2D grid.
- Publishing: Published on the /map topic by warehouse_map_publisher node.
- Encoding:
  - Free space: 0
  - Occupied space (walls, shelves): 100, 75
  - Semi-occupied space (stations): 50

Robot Representation:

- URDF File: Defines the robot's physical and visual properties (robot.urdf).
- Position Updates: Published as PoseStamped messages on the /robot_pose topic by robot_node node.

Example of execution



Future Work and Possible Improvements

Visual Enhancements:

- Improve the visual representation in RViz and Gazebo to make the simulation more realistic.
- Add detailed 3D models for the warehouse environment and the robot.

Multi-Robot Coordination:

- Introduce the possibility of having multiple robots operating simultaneously.
- Develop coordination strategies to handle task allocation, collision avoidance, and communication between robots.

Dynamic Task Allocation:

- Implement a dynamic task allocation system that can prioritize tasks based on urgency and robot availability.
- Use machine learning techniques to optimize task scheduling and robot performance.

Conclusion and Learnings

- Managed nodes, topics, services, parameters, and custom messages.
- Implemented map publishing, task management, and robot control.
- Real-time RViz visualization, dynamic map generation, and robust communication.

END