AURO Final Report

- **Overview**

This solution consists of State Machine with Nav2 API, for which we used methodologies like **Simulation**, **Experiment**, and **Functional Testing**. The methodology for developing the solution was influenced by the need for reliability, adaptability, and cost-effective development. Firstly, **Simulation methodology** was the foundation of the development process. With the help of Gazebo simulator integrated with ROS2, the robotic system (sensors, environment, and navigation) was tested virtually. This methodology allowed for iterative development without requiring physical hardware. Moreover, RViz represented visualization of the robot's navigation and task execution. Secondly, **Experimental Testing** was performed to evaluate how robots interact with dynamic elements, such as items and zones in real time, allowing to try different approaches of implementation and rating effectiveness of overall solution. This approach ensures that robots' behaviors, such as obstacle avoidance and picking up items performed as expected. **Functional Testing** checks if navigation and perception work correctly, and every module fulfilled its design goals and blends in with the rest of the system.

- **Architecture**

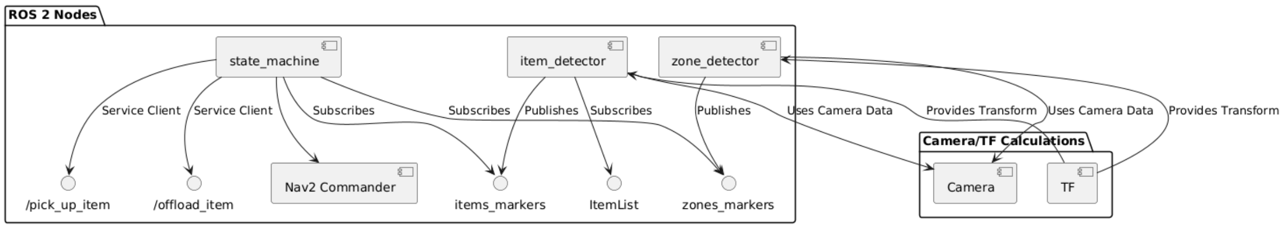


Figure 1

Figure 1 showcases high level architecture diagram, which integrates multiple ROS 2 nodes that work together to achieve autonomous item detection, zone recognition, and navigation. The solution is designed based on the Nav2 Simple Commander API and consists of the primary nodes state_machine, item_detector, and zone_detector, with supporting components like camera, and TF transformations. Each node has a well-defined role in the system's control strategy.

Control Strategy and Role of the Nodes:

1. **Item_detection:** Detects items in the environment to estimate their positions relative to the robot correctly.

- Uses camera data to identify items and calculates their distance and angle using TF transformations.
- Publishes identified items as visualization markers on the items_markers topic.
- Subscribes to the ItemList topic, which is published by the item sensor node, to get raw item data.
- Uses the camera for processing raw images and TF to compute positions in the map frame.

2. **State_Machine**: Acts as the brain of the system, the overall control logic is a state-based approach.

- **Searching State**: The robot navigates the area using Nav2 while looking for items and uses data from items_markers and sets the navigation goal as the item's coordinates.
- **Collecting State**: The robot moves to goal's location and interacts with the /pick_up_item service to pick up the item.
- **Offloading State**: After collecting an item, the robot navigates to the zone using Nav2 using data from zones_markers and interacts with the /offload_item service to offload the item.

3. **Zone Detection:** Identifies zones where items need to be offloaded. Similar approach to Item_Detection node. Publishes zones as visualization markers to the zones_markers topic.

The architecture is modular and follows a clear separation of concerns, ensuring scalability. It Centralizes decision-making and control logic, simplifying the integration of navigation and manipulation tasks. Its reliance on the Nav2 API ensures robust navigation capabilities, such as obstacle detection and path planning. **Item_Detector** handles computationally tasks like item recognition and localization, offloading this work from the state_machine. This design isolates the perception system for easier debugging and optimization.

- **Control**

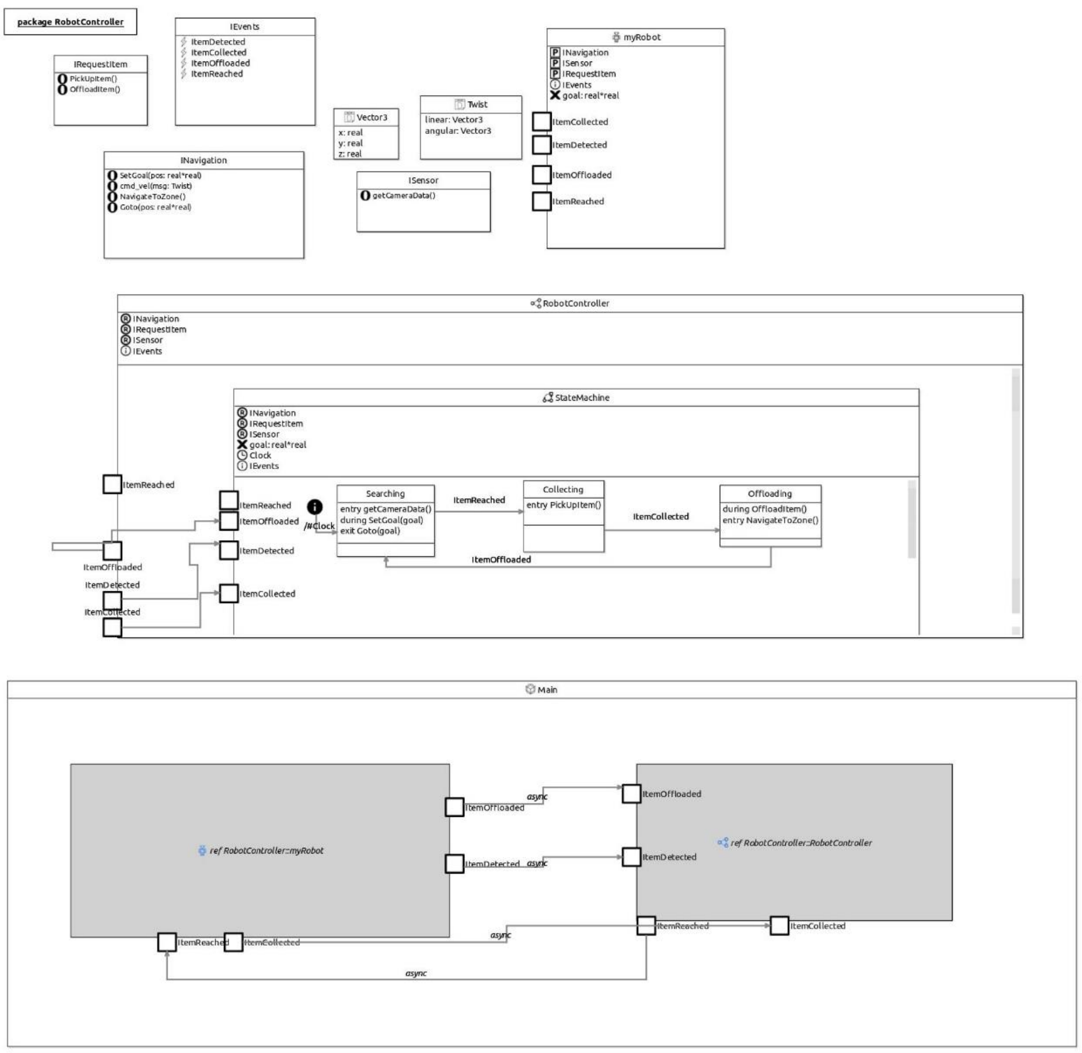


Figure 2

Figure 2 describes the architecture of the solution, which is modeled using a hierarchical state machine approach to manage the robotic behavior for the autonomous retrieval of items. The Controller, RobotController, orchestrates the robot's interactions with its environment by handling the following key behaviors:

- Searching and navigating for items.
- Collecting the items.
- Offloading the items into the appropriate zones.

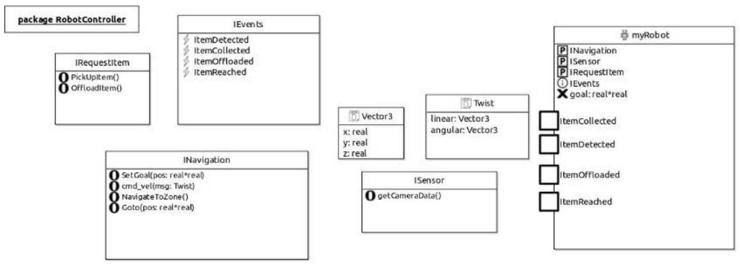


Figure 3

Figure 3 illustrates Interfaces, Operations, Events and the myRobot Platform which has provided interfacesand a variable "goal" is represented as robot's navigation goal.

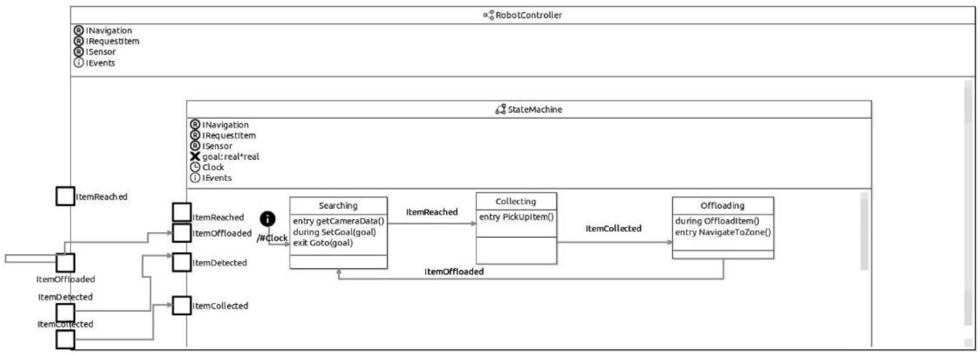


Figure 4

Figure 3 defines a RobotController with StateMachine.

1. **Main Control State Machine (FSM):**
    - **Searching:** Defines the coordinates of the detected item and initiates navigation. Handles the movement toward the detected item, transitioning to **Collecting** when the item is within reach.
    - **Collecting:** Simulates picking up the item upon arrival.
    - **Offloading:** Delivers the item to the zone.

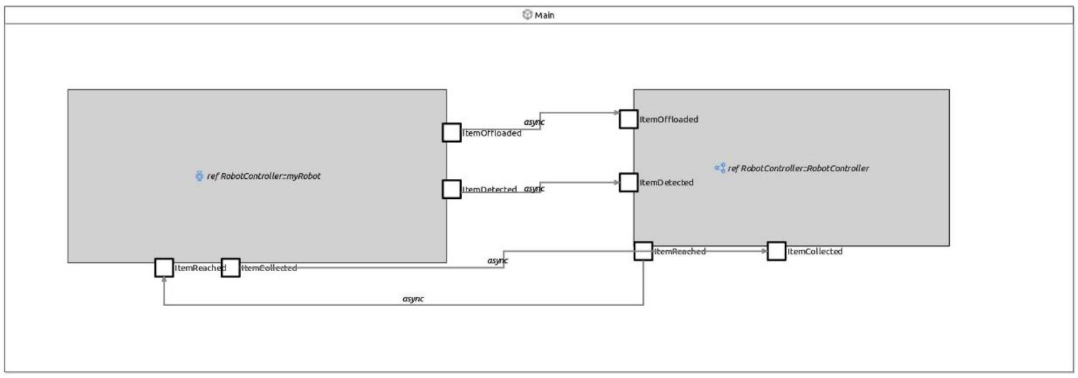


Figure 5

Figure 4 represents Module Main connecting events asynchronously and referencing to the Platform and Controller.

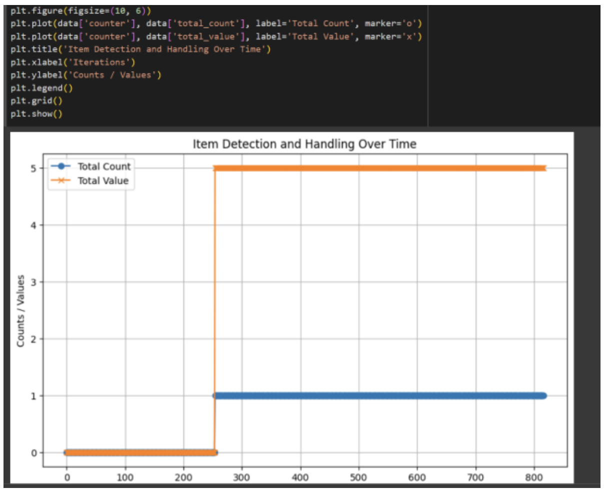
Justification for RoboChart Architecture

1. State-Based Control. The hierarchical state machines in the model provide a logical flow for task execution. It represents the overall sequence of the task, which makes transitioning between states efficient.
2. Reusability. By defining interfaces such as IEvents, Navigation, and RequestItem, the model encourages reusability. For example, the Navigation component can be reused in other robotic systems requiring similar path-planning and goal-setting functionalities.
3. Encapsulation: Each component encapsulates its functionality and communicates with others via defined interfaces (e.g., ItemReached, ItemCollected). This minimizes interdependence and reduces the number of errors propagating through the system.

- **Evaluation**

Our solution was evaluated using a combination of Simulation, Experiment, and Data Analysis methodologies. These approaches tested the functionality, reliability, performance of the robotic system and provided insights into its strengths and weaknesses.

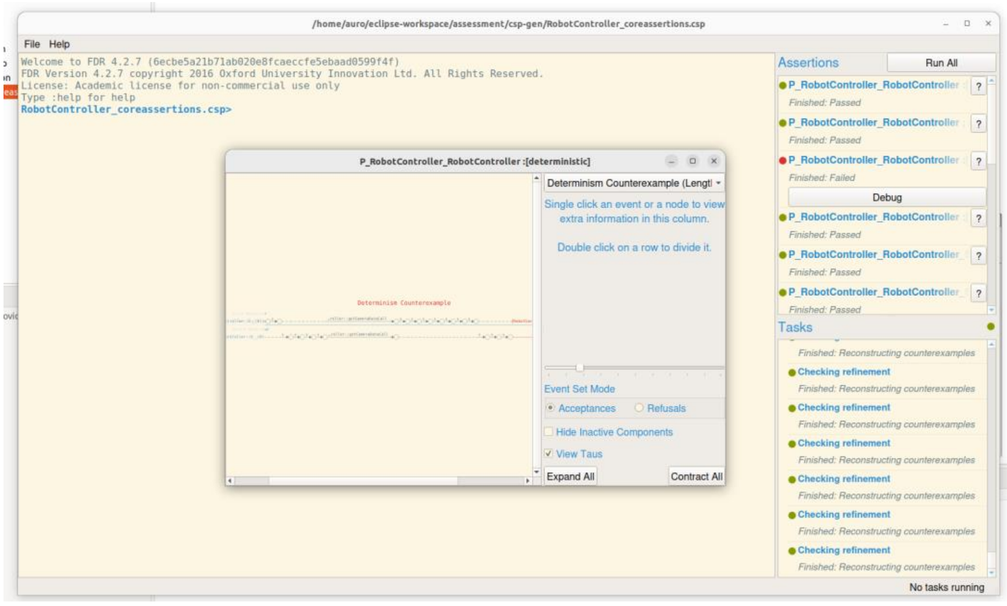
1. **Simulation** was performed using **Gazebo** integrated with **RViz** to emulate real-world scenarios and validate the robot's navigation, manipulation, and perception capabilities in a risk-free virtual environment. We chose Simulation because it makes testing affordable and scalable, especially in the early phases of development when quick iterations are needed.
2. **Experiments** were conducted to determine the most accurate approach for calculating item coordinates for navigation**.** The objective was to compare different methods of calculating the item coordinates relative to the robot for navigation. As a result, computing focal lengths of camera and field of view angles optimized the balance between detection accuracy and computational efficiency.



```
plt.figure(figsize=(10, 6))
plt.plot(data['counter'], data['total_count'], label='Total Count', marker='o')
plt.plot(data['counter'], data['total_value'], label='Total Value', marker='x')
plt.title('Item Detection and Handling Over Time')
plt.xlabel('Iterations')
plt.ylabel('Counts / Values')
plt.legend()
plt.grid()
plt.show()
```

Figure 6

3. **Data Analysis** assessed the performance, reliability, and efficiency of the robotic system across multiple scenarios and analyzed data from simulation to gain insights into the system's performance trends and identify areas for improvement. The purpose was to quantitatively evaluate the system’s efficiency and reliability across multiple runs and scenarios. As defined in Figure 6, after analysing Data Log file, the average total count is 0.69, indicating consistent item detection during the simulation, with a success rate of approximately 69%.



```
/home/auro/eclipse-workspace/assessment/csp-gen/RobotController_coreassertions.csp

Welcome to FDR 4.2.7 (6ecbe5a21b71ab020e8fcaeccfe5ebaad0599f4f)
FDR Version 4.2.7 copyright 2016 Oxford University Innovation Ltd. All Rights Reserved.
License: Academic license for non-commercial use only
Type :help for help
RobotController_coreassertions.csp>
```

P_RobotController_RobotController :[deterministic]

Determinism Counterexample (Lengt

Single click an event or a node to view extra information in this column.

Double click on a row to divide it.

Event Set Mode

(● Acceptances) ○ Refusals

☐ Hide Inactive Components

☑ View Taus

Assertions

● P_RobotController_RobotController
Finished: Passed

● P_RobotController_RobotController
Finished: Passed

● P_RobotController_RobotController
Finished: Failed

● P_RobotController_RobotController
Finished: Passed

● P_RobotController_RobotController
Finished: Passed

● P_RobotController_RobotController
Finished: Passed

Tasks

Finished: Reconstructing counterexamples
● Checking refinement
Finished: Reconstructing counterexamples
● Checking refinement
Finished: Reconstructing counterexamples
● Checking refinement
Finished: Reconstructing counterexamples
● Checking refinement
Finished: Reconstructing counterexamples
● Checking refinement
Finished: Reconstructing counterexamples
● Checking refinement
Finished: Reconstructing counterexamples

No tasks running

Figure 7

4. **Analysing RoboChart Model.** RoboChart’s CSP semantics is used to formally verify the control system’s correctness and focuses on properties:
   - Deadlock Freedom: Ensuring the robot never reaches a state where no further actions are possible.
   - Determinism: Confirming that system behavior is predictable and consistent [9].

   The assertions generated from the model are tested using FDR (Failures-Divergences Refinement). Figure 7 shows the Determinism Counterexample meaning that the behavior might not be consistent.

Strengths and Weaknesses of the Solution

Strengths

1. **Effective Navigation System:** Nav2 Simple Commander provided the robot to autonomously navigate with high accuracy and sets the goal efficiently in a dynamic environment and cameras LiDAR, RGB allow obstacle detection. It enhanced the system's ability to deal with real-time environmental changes.
2. **Adaptive Behavior:** The method used a finite state machine (FSM) to make decisions depending on states. As a result, the robot was able to dynamically adapt to environmental changes, such as unexpected obstacles or missing objects.
3. **Scalability:** With distinct nodes for sensing, manipulation, and navigation, our modular design allows the system to easily scale for multi-robot collaboration or deployment in bigger environments.

Weaknesses

1. **Relying on Simulated Environment:** Although Simulation provides a robust platform for development, it may not fully capture the uncertainties of real-world environments. Issues like sensor noise, hardware limitations, and unpredictable human interactions are underexplored.
2. **Perception Challenges:** Dependency on specific sensors, such as LiDAR may cause inaccuracies in detecting items and zones, particularly under poor lighting or partial occlusions [8].
3. **Inconsistent State Offloading:** The system occasionally fails to correctly detect items after offloading process, leading to incomplete task execution.

Proposed Improvements

1. **Advanced Sensor Fusion:** Integrating additional sensors, such as IMUs or ultrasonic sensors, alongside LiDAR and RGB cameras, could improve environmental perception.
2. **Real-World Deployment:** Introducing iterative testing in real-world environments would help validate system performance under practical conditions, addressing hardware limitations, sensor noise, and human-robot interaction challenges.
3. **Centralized Debugging Tool**: Incorporating a comprehensive debugging framework that visualizes inter-node communications and logs in real-time would simplify troubleshooting in complex multi-node systems.

Safety and ethics

Safety Considerations

1. Collision Avoidance and Navigation

Our autonomous robots are equipped with sensors such as LiDAR or RGB cameras and algorithms to navigate dynamic environments. However, failures in obstacle detection can cause collision with humans, objects or other robots. Mitigations:

- Applying redundant sensors and fallback algorithms to improve robustness [1].
- Integrating industrial safety standards such as ISO 10218 because it would ensure collision avoidance and emergency stops [1].

2. System Reliability

Failures in hardware or software algorithms could lead to unsafe conditions or a loss of functionality, especially when robots operate alongside humans. Mitigations:

- Introducing redundancy in critical hardware, such as LiDAR sensors to prevent failure points [2].
- Using real-time monitoring systems to detect errors and switching to backup systems if needed [3].

3. Data Security

Autonomous mobile robots depend on real-time communication, which is vulnerable to cyberattacks and unauthorized access could deal with safety and sensitive data. Moreover, cybersecurity measures are essential in preventing malicious attempts on navigation and perception systems for the safe robots' operations. Mitigations:

- Protecting data by using encrypted communication protocols, for example TLS for safe data transmission [4].
- Isolating critical robot functions from external network access to reduce the attack surface.

Ethical Considerations

1. Bias in Perception Systems

Due to biases, perception algorithms might misclassify objects or zones which lead to inefficient or unsafe operations. Unbiased objects and zone detection are important for accurate navigation and task execution in environments where diverse objects are handled. Mitigations:

- Training algorithms on diverse datasets that reflect variations in lighting, object color and shapes [5].

2. Privacy Concerns

Robots with cameras may capture sensitive information raising privacy concerns. Privacy measures are relevant in shared spaces to avoid data misuse. Mitigations:

- Anonymize collected data and ensure compliance with GDPR or similar regulations [6].
- Limit data storage to only operationally necessary information, deleting unused data promptly.

3. Employment

Automated systems can displace workers in manufacturing and logistics. Making collaboration between humans and robots can influence efficiency and decrease negative social impact. Mitigations:

- Providing retraining programs to assist employees in assuming positions involving robot supervision [7].
- Using robots to assist by performing dangerous or repetitive activities.

Reflection on Real-World Implementation

These ethical and safety concerns are addressed in multiple ways by our solution:

- Safe Human-Robot Interaction: Implementing international safety standards would promote safe operations in shared environments [1].

- Robust Perception and Decision-Making: Using adaptive algorithms would enhance performance in diverse deployment scenarios [5].
- Ethical Oversight: Providing transparency in decision-making processes with ethical guidelines would build trust in the system [7].

- **Simulation scenario**

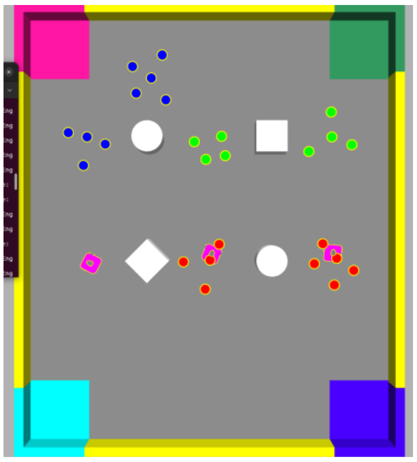
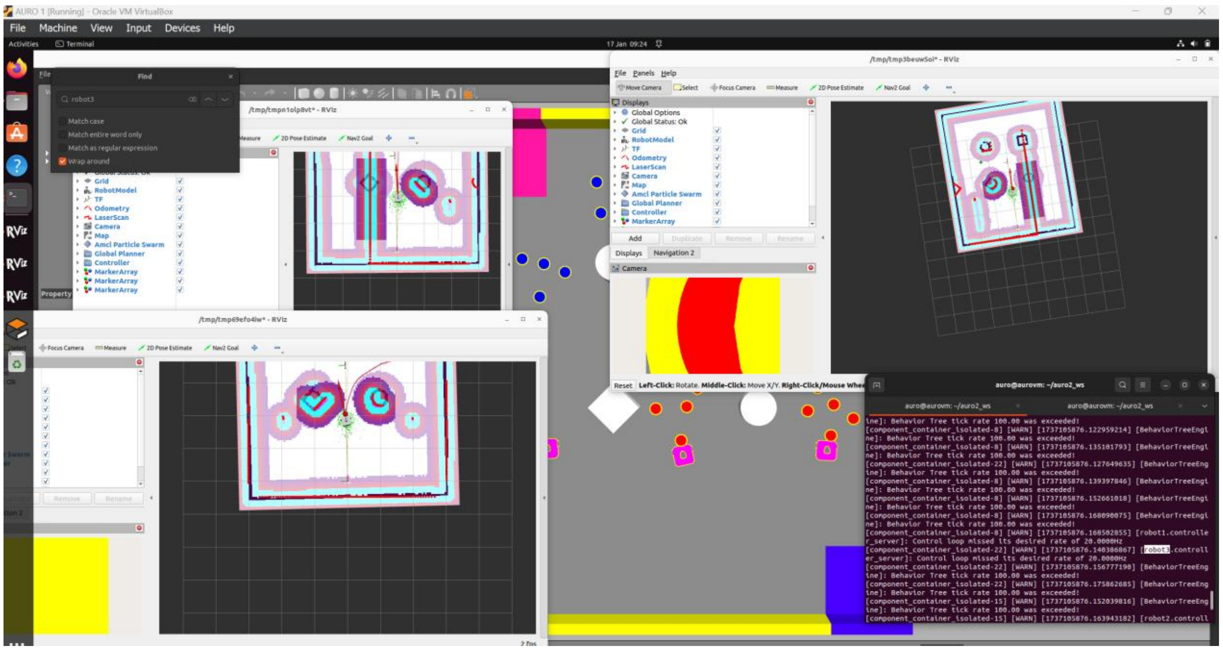


Figure 8



```
ine]: Behavior Tree tick rate 100.00 was exceeded!
[component_container_isolated-8] [WARN] [1737105876.122959214] [BehaviorTreeEngi
ne]: Behavior Tree tick rate 100.00 was exceeded!
[component_container_isolated-8] [WARN] [1737105876.135101793] [BehaviorTreeEngi
ne]: Behavior Tree tick rate 100.00 was exceeded!
[component_container_isolated-22] [WARN] [1737105876.127649635] [BehaviorTreeEng
ine]: Behavior Tree tick rate 100.00 was exceeded!
[component_container_isolated-8] [WARN] [1737105876.139397846] [BehaviorTreeEngi
ne]: Behavior Tree tick rate 100.00 was exceeded!
[component_container_isolated-8] [WARN] [1737105876.152661018] [BehaviorTreeEngi
ne]: Behavior Tree tick rate 100.00 was exceeded!
[component_container_isolated-8] [WARN] [1737105876.160090075] [BehaviorTreeEngi
ne]: Behavior Tree tick rate 100.00 was exceeded!
[component_container_isolated-8] [WARN] [1737105876.168502855] [robot1.controlle
r_server]: Control loop missed its desired rate of 20.0000Hz
[component_container_isolated-22] [WARN] [1737105876.140386867] [robot3.controll
er_server]: Control loop missed its desired rate of 20.0000Hz
[component_container_isolated-22] [WARN] [1737105876.156777190] [BehaviorTreeEng
ine]: Behavior Tree tick rate 100.00 was exceeded!
[component_container_isolated-22] [WARN] [1737105876.175862685] [BehaviorTreeEng
ine]: Behavior Tree tick rate 100.00 was exceeded!
[component_container_isolated-15] [WARN] [1737105876.152039816] [BehaviorTreeEng
ine]: Behavior Tree tick rate 100.00 was exceeded!
[component_container_isolated-15] [WARN] [1737105876.163943182] [robot2.controll
```

Figure 9

Figures 8,9 show the first simulation scenario is setting number of robots as 3 to see how it reacts to performance and effectiveness of simulation. Consequently, robots successfully divided items on their proximity to the items and zones, however, communication using ROS topics led to an increase in task execution latency.

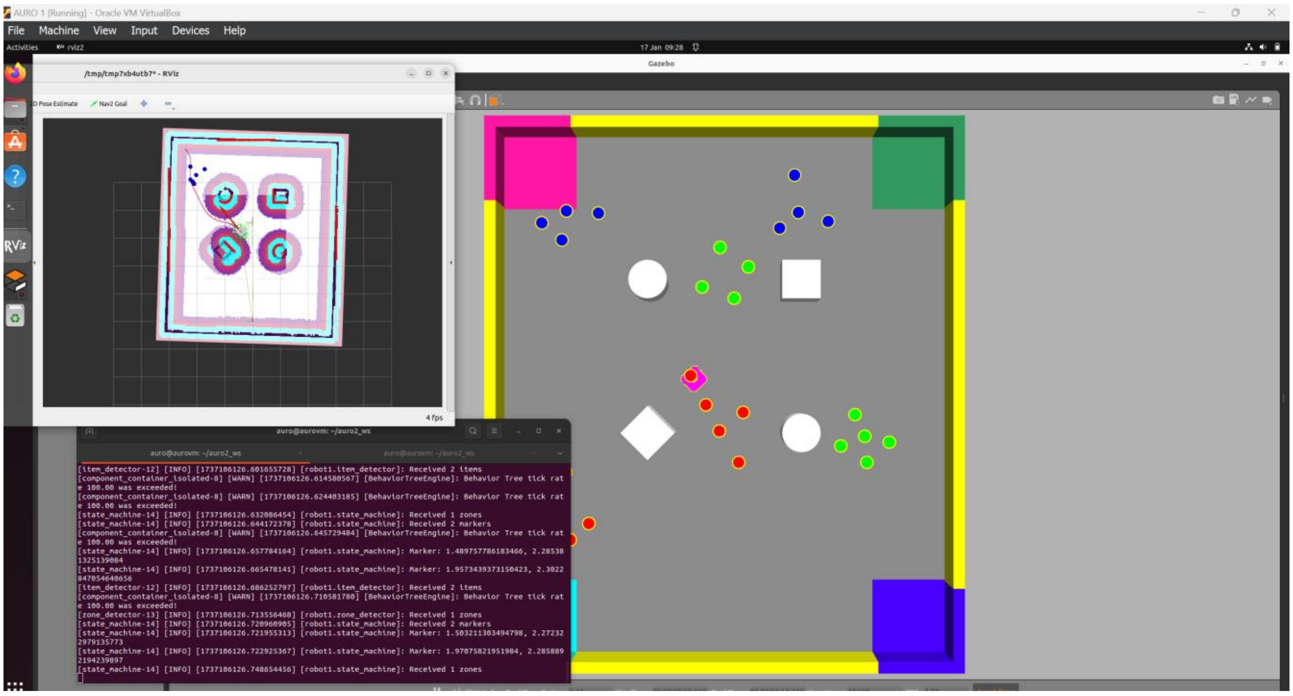

```
[item_detector-12] [INFO] [1737106126.601655728] [robot1.item_detector]: Received 2 items
[component_container_isolated-8] [WARN] [1737106126.614580567] [BehaviorTreeEngine]: Behavior Tree tick rat
e 100.00 was exceeded!
[component_container_isolated-8] [WARN] [1737106126.624403185] [BehaviorTreeEngine]: Behavior Tree tick rat
e 100.00 was exceeded!
[state_machine-14] [INFO] [1737106126.632086454] [robot1.state_machine]: Received 1 zones
[state_machine-14] [INFO] [1737106126.644172378] [robot1.state_machine]: Received 2 markers
[component_container_isolated-8] [WARN] [1737106126.645729484] [BehaviorTreeEngine]: Behavior Tree tick rat
e 100.00 was exceeded!
[state_machine-14] [INFO] [1737106126.657784164] [robot1.state_machine]: Marker: 1.489757786183466, 2.28538
1325139084
[state_machine-14] [INFO] [1737106126.665478141] [robot1.state_machine]: Marker: 1.9573439373150423, 2.3022
847054640656
[item_detector-12] [INFO] [1737106126.686252797] [robot1.item_detector]: Received 2 items
[component_container_isolated-8] [WARN] [1737106126.710581780] [BehaviorTreeEngine]: Behavior Tree tick rat
e 100.00 was exceeded!
[zone_detector-13] [INFO] [1737106126.713556468] [robot1.zone_detector]: Received 1 zones
[state_machine-14] [INFO] [1737106126.720960905] [robot1.state_machine]: Received 2 markers
[state_machine-14] [INFO] [1737106126.721955313] [robot1.state_machine]: Marker: 1.503211303494798, 2.27232
2979135773
[state_machine-14] [INFO] [1737106126.722925367] [robot1.state_machine]: Marker: 1.97075821951984, 2.285889
2194239897
[state_machine-14] [INFO] [1737106126.748654456] [robot1.state_machine]: Received 1 zones
```

Figure 10

In the second scenario, in Figure 10, the random seed argument was set to 3 to introduce controlled randomness into the simulation and affect the item manager, which responsible for generating item locations and properties.

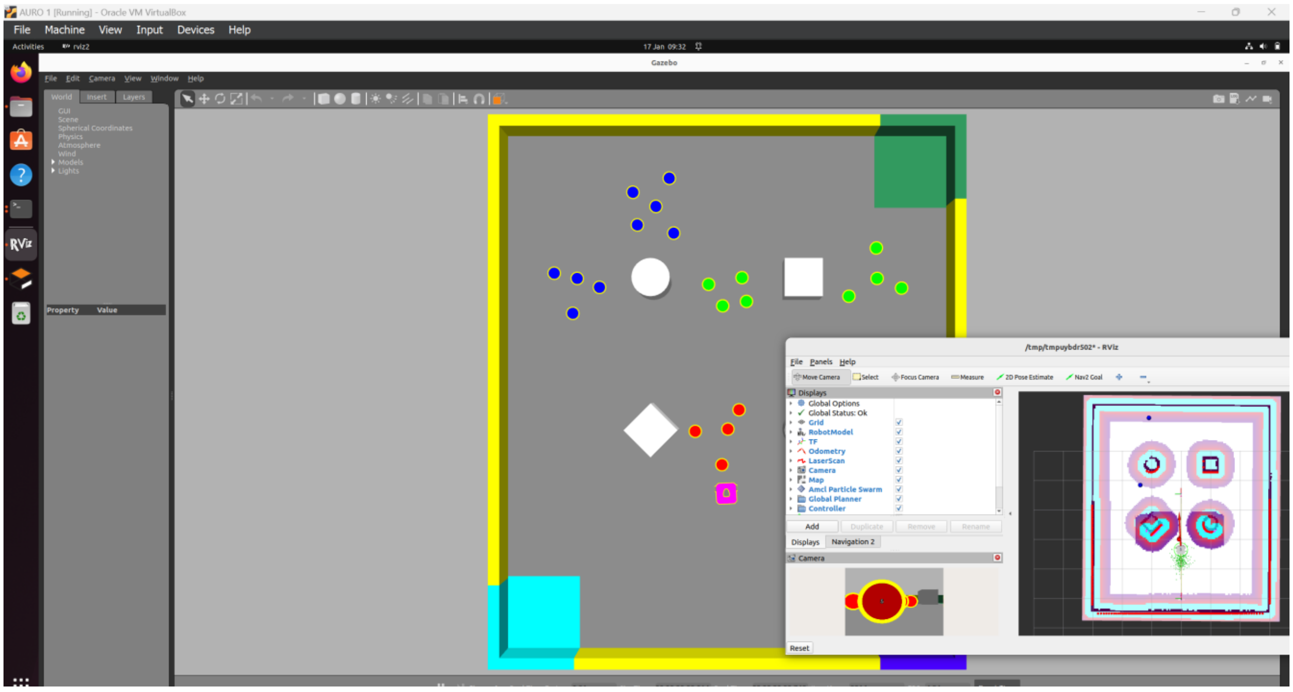


Figure 11

Figure 11 illustrates scenario where Left Zone is false, this setting mirrors practical, real-life scenarios where storage areas may not have predictable or optimal placement.

References

[1] ISO 10218, "Robots and Robotic Devices - Safety Requirements for Industrial Robots," *International Organization for Standardization*, 2011.

[2] M. L. Visinsky, J. R. Cavallaro, and I. D. Walker, "Robotic fault detection and fault tolerance: A survey," *Reliability Engineering & System Safety*, vol. 46, no. 2, pp. 139-158, Jan. 1994. doi: 10.1016/0951-8320(94)90132-5.

[3] IEEE Global Initiative on Ethics of Autonomous and Intelligent Systems, *Ethically Aligned Design*, IEEE, 2019.

[4] European Commission, *Ethics Guidelines for Trustworthy AI*, Brussels, Belgium, 2021.

[5] N. Leveson, "Are you sure your software will not kill anyone?," *Communications of the ACM*, vol. 63, no. 2, pp. 25–28, 2020.

[6] European Commission. (2020). *White Paper on Artificial Intelligence - A European approach to excellence and trust*.

[7] United Nations Environment Programme, *Sustainability in Robotics*, United Nations, 2023.

[8] N. Correll, B. Hayes, C. Heckman, and A. Roncone, *Introduction to Autonomous Robots: Mechanisms, Sensors, Actuators, and Algorithms*, 3rd ed., Cambridge, MA: MIT Press, 2022.

[9] A. Miyazawa, A. Cavalcanti, S. Foster, W. Li, P. Ribeiro, J. Timmis, and J. Woodcock, *Modelling and Verification with RoboChart*.