AURO REPORT

Module Code: COM00052H — Examination Number: Y3898642

I. Design

A. System Overview

In the design of my autonomous robot, I implemented a modular approach with the TurtleBot3, using the ROS2 framework [1]. At the heart of the design is a finite state machine (FSM), which contains the robot's states - FORWARD, TURNING, COLLECTING, and RETURNING. This FSM was chosen for its ability to provide structured yet flexible control, which is essential for navigation and adapting to dynamic task requirements. The keys to perception were the LiDAR and camera sensors, which are essential for effective operation. Their integration within ROS2 is indispensable for the task.

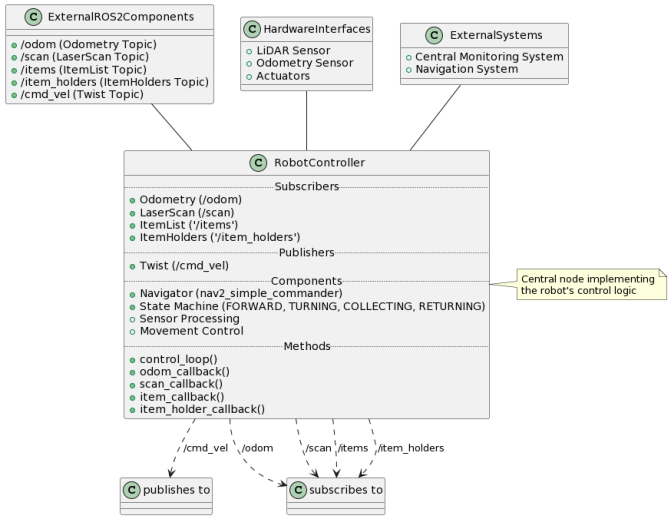

Fig. 1. High-Level System Design

B. Component Interaction

As shown in Fig. 1, the robot controller subscribes to several essential topics for efficient operation:

- **LaserScan**: Provides obstacle navigation data, taken from the LiDAR sensor, enabling the robot to detect and avoid obstacles.
- **ItemList**: Processes information from the camera to identify items within the robot's field of view, facilitating item identification and retrieval.
- **ItemHolders**: Keeps track of the items held by each robot, sourced from internal status monitoring. Helps in preventing task ovelap.
- **Odometry**: Provides data on the robot's position and orientation, taken from odometry sensors. It's essential for accurate navigation and precise maneuvering.

The data from these topics is parsed using the callback methods, a key process in the system's functionality.

II. Implementation

A. ROS Architecture

The implementation of my autonomous robot system is deeply rooted in the ROS 2 framework. It provides a modular and scalable architecture. This architecture is made from multiple ROS 2 nodes, each of which performs specific functions such as sensor data processing, state management, and motion control. The communication among these nodes is facilitated through topics, in which nodes publish and subscribe to messages.

B. Node Structure and Topic Communication

The system is structured around several key nodes. For example, the `robot_controller` node subscribes to the `/scan` topic for LiDAR data and the `/odom` topic for odometry information. It also publishes movement commands to the `/cmd_vel` topic.

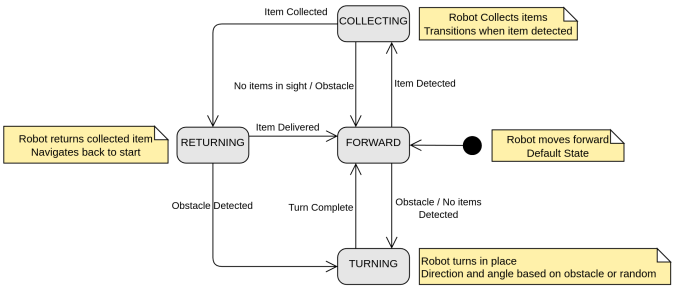

Fig. 2. Finite State Machine Representing Autonomy

C. Autonomy Realization

Autonomy in the system is achieved through a well-defined finite state machine (FSM), as depicted in Fig. 2. The FSM has 4 states in which it can be: FORWARD, TURNING, COLLECTING, and RETURNING. It transitions based on sensor inputs and internal logic, also shown in Fig. 2.

D. Sensor Integration and Data Processing

Integrating sensors like LiDAR and cameras is key to the robot's perception capabilities. Using ROS 2, sensor data is processed in real-time, with custom callback functions handling the data for tasks like obstacle detection and item identification. Without this processed information, navigation decisions and effective task execution would be impossible.

E. Actuation and Movement Control

Movement control within the system is made through the generation of Twist messages. Those messages determine the robot's linear and angular velocities, and are based on the current state of the FSM and the processed sensor data. This shows a direct link between sensory input and actuator output.

F. Challenges and Solutions

During the early phases of the development process, I started with designing and implementing a system for a single robot. This approach, however, later caused issues whilst integrating the multi-robot solution. One such issue was all robots transitioning to the RETURNING state once one of them collected an item. This was due to the lack of optimisation for a multi-robot system. For instance, the `item_holder_callback` method was returning all the bots' information simultaneously, which caused the issue. The solution was to ensure each robot received and acted upon only the information relevant to itself. This change made the system effectively manage the state transitions for each robot independently and further align with the demands of a multi-robot system.

Another issue encountered when first integrating the multi-robot system; In certain instances, a robot would enter the COLLECTING state when seeing another holding an item (RETURNING state), thus chasing it down and prematurely setting off the obstacle avoidance in both. This resulted in delays and slower performance. The solution was to check the vertical coordinates of the item, and determine whether it was being held by another robot based from that information.

III. Analysis

A. Experimental Approach

To evaluate the performance and efficiency of the system, I've taken a comprehensive experimental approach. This involved multiple strategies to gather both qualitative and quantitative data:

- **Data Logging**: Extensive data logging into the terminal was performed to capture real-time operational details of the system - This helped find bugs and solve issues.
- **Visual Inspections**: During simulations, visual inspections / Analysis were done to identify and fix issues in the system's behavior.
- **Simulation Analysis**: Detailed analysis of data logs from 10 different simulations, each lasting 5 minutes in real-time (aprox. 2 minutes sim time). The simulations used different item manager 'seeds' to assess the system's efficiency. This analysis was conducted both before and after implementing a fix for the .

Repeated trials were also conducted to ensure the reliability and consistency of the system's performance across various scenarios.

B. Results

The results of the experiments, both pre- and post-fix, are presented in Table I. A bar plot showcasing the comparative performance is illustrated in Fig. 3.

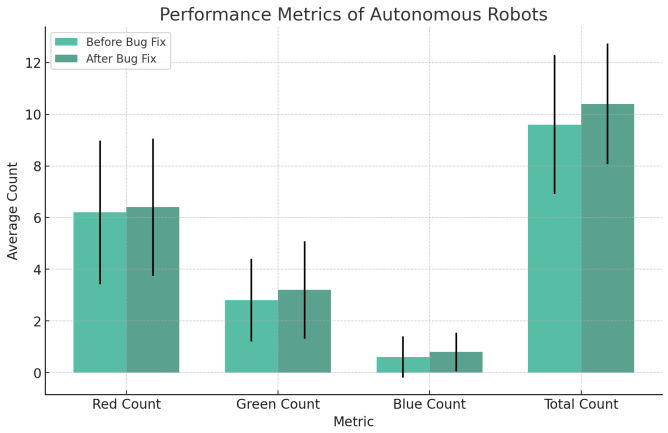

Fig. 3. Bar-Plot Showcasing Performance (pre- and post-fix)

TABLE I
PERFORMANCE METRICS COMPARISON

Metric	Pre-Fix	Post-Fix
Red Count	6.2 (σ2.79)	6.4 (σ2.65)
Green Count	2.8 (σ1.60)	3.2 (σ1.89)
Blue Count	0.6 (σ0.80)	0.8 (σ0.75)
Total Count	9.6 (σ2.69)	10.4 (σ2.33)

Note: σ denotes standard deviation.

C. Results Analysis

Qualitatively, the system showed decent performance in item collection, with an average of approximately 10 items per simulation. Notably, red items were collected most frequently, as they were positioned closest to the home zone. Green items, typically located in the middle, were collected next in frequency, followed by blue items, which were the furthest away. Despite the robots being programmed to prioritize blue, then green, and then red items within their field of view when collecting, this priority order does not reflect itself in the results.

Quantitatively, the reason behind this can be understood through the visual inspections conducted during the simulations. While the robots initially prioritize blue items, they often collect green or red items en route to the blue ones, leading them to transition to the returning state prematurely. Additionally, on their way back to the home zone with a blue or green item, they frequently swap it for more readily accessible red items. This behavior is linked to the functionality of the item manager, though it is unclear whether this is a bug or an intentional design choice. An alternative approach could have been to treat untargeted items as obstacles when the robot is in the returning state, which would prevent item swapping. Doing so would result in a higher count of blue and green items, however, it would also increase the time taken for the robot to return home, decreasing the total count of collected items.

IV. Evaluation

A. Strengths and Weaknesses

During the development and testing phase, several strengths and weaknesses in the system became evident. One key strength is the design's scalability and adaptability which allowed for a smooth integration of multi-robots, capable of efficiently handling multiple tasks such as navigation, item collection and obstacle avoidance. The implementation of the FSM, shown in

Fig. 2, was crucial for the system, significantly contributing to its operational success.

On the flip side, weaknesses were also shown in the system, especially in the context of multi-robot coordination. Notably, in some cases, robots attempted to collect items already held by another. This behaviour stems from an oversight in the COLLECTING state's logic, which didn't account for other robots' activities. The fix involved considering the y-coordinate of the item; since the ground is always level in the simulation, items at a higher y-coordinate are likely being held by another robot. However, this solution might not be feasible in a more complex real-world environment with varying elevations.

Improving the system further could involve advancing the communication between robots. By sharing their current positions and the status of items they're carrying, each robot would be aware of its peers' activities. This enhancement, integrated within the ROS2 framework, would make the system even more efficient, and address a big chunk of its weaknesses.

B. Transferability to Reality

When considering the transfer of the autonomous robot system from simulation to real-world scenarios, several key factors come into play. The simulation provides a controlled environment which significantly differs from the real-world, which can be unpredictable due to the varied terrain, unexpected obstacles, and hardware issues (especially with sensors).

The TurtleBot3's modular design and the ROS2 framework provides a good starting point for this transition, however, adapting the system for real-world use would require additional enhancements such as:

- **Enhanced Sensor data processing**: To deal with faults, and handle errors.
- **Enhanced Navigation Algorithms**: To handle more complex and less predictable environments.
- **Enhanced Obstacle Avoidance**: Since you can't walk through items in real life, they have to be treated as obstacles.
- **Addition of Robot Arm**: To grab and hold the items, since they don't magically get on the robot like in the simulation.

More challenges and unforeseen issues can occur in real-world applications, it would require more development and testing to fully know what's needed.

V. Safety and Ethics

The integration of autonomous robots into real-world scenarios, especially for tasks such as item retrieval, brings forth significant safety implications and ethical considerations.

A. Safety Considerations

The safety of both the robots and humans they may interact with is crucial. For this reason, designing fail-safes and emergency stop mechanisms to prevent accidents is necessary. A robot might malfunction and run into a human (could stem from sensor malfunction, detects human as an item and tries to retrieve it), which could cause damage to both parties involved. The robots might malfunction and run into each other, or into obstacles. This might cause damage to the robots, and environment surrounding them. Later resulting in injuries and financial loss.

B. Ethical Considerations

Ethically, it's crucial to consider the potential impact of these robots on human employment and privacy, especially in contexts where they operate in close proximity to people. Data collected from the robots might have an impact on people's privacy. Camera data, could potentially be used for facial recognition and be later sold to companies, profiting from non-consenting humans is ethically wrong. This could be solved by making sure all data stemmed from the robots is used in an ethical way, and was gathered with consent from all parties involved. The deployment of such systems should be guided by principles that prioritize human welfare, and job security. Doing so ensures that the technology augments human roles, not replace them.

VI. Conclusion

This report has presented the design, implementation, analysis, and evaluation of an autonomous robot system using the TurtleBot3 Waffle Pi and the ROS2 framework. The system, structured around a finite state machine (FSA), demonstrated strengths in scalability, adaptability, and the capability to handle multiple tasks like navigation, item collection, and obstacle avoidance, as shown in the analysis section.

Key findings from the project include:

- The modular design approach facilitated efficient multi-robot integration and task management.
- The implementation of the FSM was instrumental in achieving autonomous operations, despite some challenges in multi-robot coordination.
- Experimental analysis highlighted the system's proficiency in item retrieval, with some limitations noted in prioritizing specific items due to the current item management strategy.
- Safety and ethical considerations underscored the need for robust design, including fail-safes and privacy safeguards, particularly for real-world applications.

Future work for improving the system could focus on several areas such as:

- Enhancing inter-robot communication to optimize task allocation and prevent redundant efforts in item collection.
- Refining sensor integration and data processing techniques to better handle real-world environments.
- Integrating a robot arm for physical item retrieval in real-world scenarios.
- Addressing the ethical implications of autonomous robotics, ensuring that the technology serves to augment human efforts rather than replace them.

Continued development and testing in a variety of environments will be essential to prepare the system for real-world deployment. This is to ensure its reliability, safety, and ethical compliance. The knowledge gained from this project will lay a strong foundation for further advancements in the field of autonomous robotics.

References

[1] Open Robotics, "ROS 2 Humble Hawksbill Documentation," https://docs.ros.org/en/humble/, 2024, [Online; accessed 7-January-2024].