NYU TANDON SCHOOL OF ENGINEERING

Final Project Report

Putting ChoiRbot to the Test

Networked Robotics Systems, Cooperative Control and Swarming
ROB-GY 6333

Alejandro Ojeda
Nishant Pushparaju
Santiago Bernheim
Vivekananda Swamy

NYU TANDON SCHOOL OF ENGINEERING

Table of content

NYU TANDON SCHOOL OF ENGINEERING

1. Abstract

This project extends the capabilities of the CHOIRBOT framework by integrating vision-based localization with two physical differential-drive robots and one simulated TurtleBot3. The physical robots, named Pandebono and Mate, are equipped with ArUco markers for pose tracking and receive motion commands via BLE using a custom ROS 2 bridge. A USB camera mount detects the ArUco markers in real time, providing global pose estimates that allow the central controller to assign navigation tasks and coordinate paths. The system demonstrates a hybrid swarm of real and simulated robots operating cohesively within the CHOIRBOT distributed robotics architecture — showcasing a scalable, centrally coordinated multi-robot system for real-world applications."

2. Introduction

Autonomous robotic systems are revolutionizing industries by optimizing movement, improving efficiency, and reducing reliance on manual labor. A key challenge in these systems is enabling multiple robots to navigate shared workspaces without collisions while efficiently reaching assigned destinations. These technologies have applications in logistics, industrial automation, agriculture, and smart infrastructure, where autonomous coordination is essential for streamlining operations.

In this project, we build upon the CHOIRBOT framework, a ROS 2-based architecture designed for distributed multi-robot coordination. CHOIRBOT enables robots to execute complex tasks such as task assignment and feedback control through modular components that support both simulation and hardware integration. Our implementation extends this capability by integrating BLE-based communication for physical robots and ArUco-based vision tracking for localization, while maintaining compatibility with CHOIRBOT's distributed control structure. The system supports a hybrid team of real and simulated robots operating in coordination to complete assigned navigation goals, providing a testbed for exploring real-time swarm behavior and task distribution.

The two robots developed, one with an arduino BLE named 'Pandebono' and the Propeller-ESP32 named 'Mate'. These robots process the commands from the ROS BLE node that is performing a bridge function where it broadcasts the messages for each robot using the topic cmd_vel, same as the Turtlebot robot.

The localization of this system is performed through the detection of ArUco markers using a central computer and an Arducam camera module. The Host system runs Ubuntu 20.04 with ROS2 Foxy. The project was built on top of the ChoiRbot framework [4]

3. Related Work

This paper [1] focuses on improving the relative localization of a two-wheeled differential drive robot using odometry. The researchers considered parameters such as weight, velocity, wheel perimeter, and wheel thickness to optimize the robot's positioning. They developed a mathematical model for odometry error and used Response Surface Methodology (RSM) to determine optimal conditions. The odometry error calculation based on ticks for expected value vs position was used to estimate the error for our robot.

The document by Figueiredo [2] demonstrates the feasibility of using a single camera and ArUco [5] markers for precise indoor robot navigation when GPS is unavailable, detailing a system with differential steering, a Raspberry Pi for image processing and localization, and an Arduino with PID control for wheel movement. This approach can serve as a reference for developing a similar autonomous navigation system for two differential robots utilizing ArUco markers for localization and potentially extending the docking application for coordinated multi-robot tasks.

4. System Architecture

4.1. Conceptual Design

The core concept of the project is to develop an integrated and scalable multi-robot system capable of testing an application of ChoiRbot library for container handling within a designated workspace. From that point on, two physical robots were implemented, and a third simulated Turtlebot 3 was used to have 3 agents.

4.2. Hardware

In this subsection the hardware component of each robot and the central station are detailed for a better understanding of each element's role in the system development. Pandebono is the robot based on Arduino Nano BLE, Mate is the robot based on Propeller with ESP32 companion and the central station is run by a Pavilion x360.

4.2.1. Pandebono:

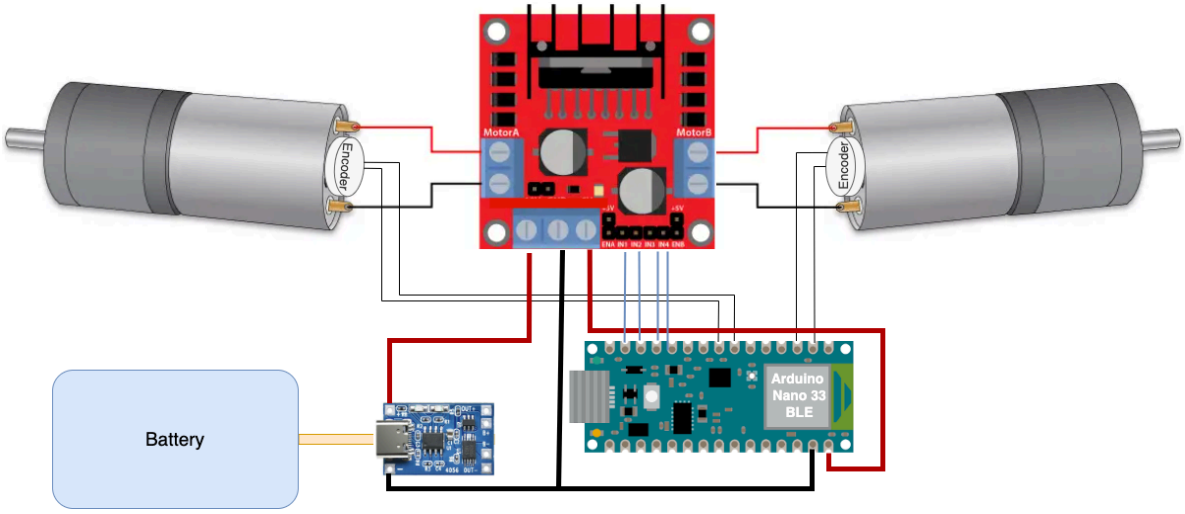

Fig 2. Pandebono's Electrical Diagram.

4.2.2. Mate:

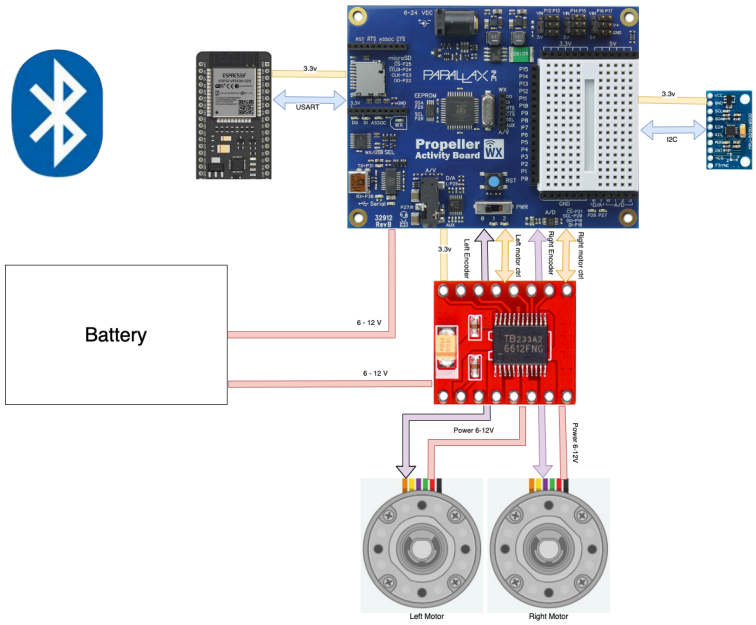

Fig 2. Mate's Electrical Diagram

4.2.3. Central Station

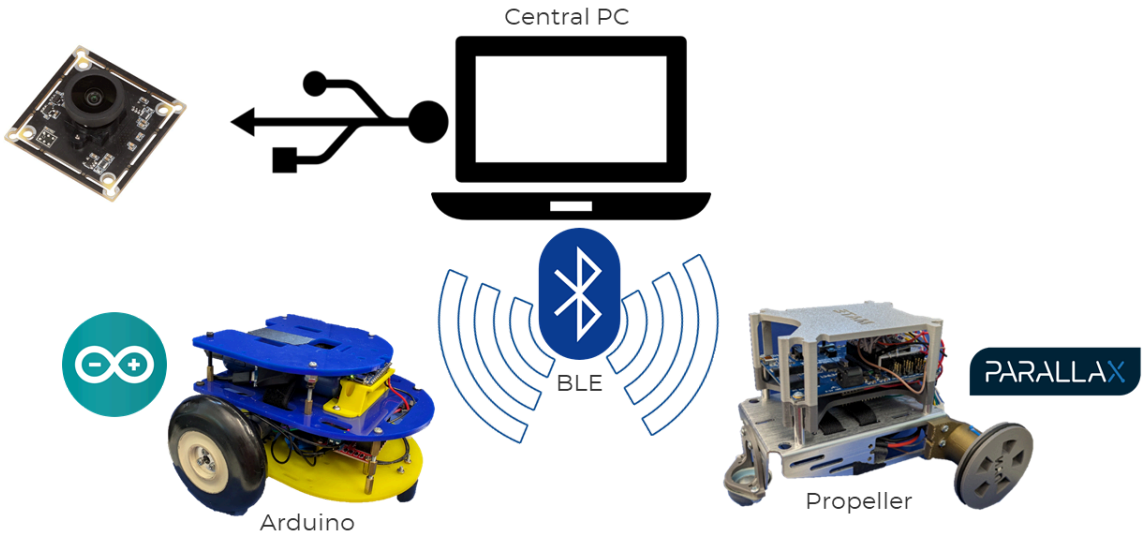

Fig 3. Central Station Component Diagram

- **Main Board - HP Pavilion x360:**
  - Microprocessor: Intel i3

- RAM: 8GB (maximum available configuration)
- Key Features for this Project: Can be replaced by a Raspberry Pi 5GB, but a 16GB model would be recommended for faster performance.

- **Camera - Arducam USB Camera**

4.3. Software and Communication Protocol

The whole system is running on ROS2 as mentioned before and thus the main communication framework is the ROS messaging system. To communicate with the remote robots we developed a BLE bridge that is capable of receiving cmd_vel commands and forwarding them to each robot accordingly. The following fixture shows the flow of messages between all the software components.

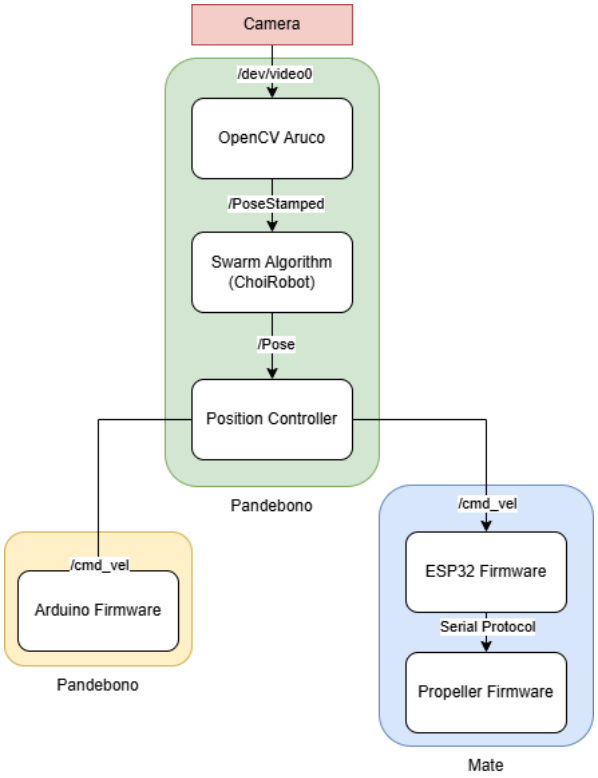

Fig4: System overview

4.4. Vision-Based Localization with AruCo Markers.

To localize the robots, 3 AruCo markers were used from the dictionary DICT_7X7_250. The AruCo Marker with ID 0 was designated as the global frame reference, serving as the origin of

the coordinate system for localization. Meanwhile, ID 4 and ID 5 were assigned to the Pandebono and Mate robots, respectively.
The localization process relies on detecting these markers within the camera's field of view. Each marker has a unique ID, allowing the system to differentiate between the robots and the global reference frame. By leveraging the geometric properties of the markers and the known size of each marker, the system can calculate the pose (position and orientation) of each robot in the global reference frame.

The following steps outline the localization process:

1. Marker Detection:
    - The USB camera captures images of the workspace.
    - The AruCo detection algorithm identifies the markers present in the image and extracts their IDs and corner coordinates.
2. Pose Estimation:
    - Using the intrinsic parameters of the camera (obtained through camera calibration), the system computes the translation (x, y, z) and rotation (roll, pitch, yaw) of each detected marker relative to the camera.
3. Global Frame Transformation:
    - The pose of each marker is transformed into the global reference frame defined by ID 0. This allows the system to localize the Pandebono and Mate robots relative to the global origin.
4. Robot Localization:
    - The system associates the poses of ID 4 and ID 5 with their respective robots (Pandebono and Mate). Thus, the robots' positions and orientations are continuously updated as they move.

Benefits of Using AruCo Markers

- High Accuracy: The markers provide precise localization information when the camera is properly calibrated.
- Robustness: AruCo markers are resistant to variations in lighting conditions and slight occlusions.
- Ease of Implementation: The OpenCV library offers built-in functions for detecting and estimating the pose of AruCo markers, simplifying the implementation.
- Faster processing than April Tags.

Considerations

1. Camera Calibration:
    - Accurate camera calibration is crucial for reliable pose estimation. Ensure that the camera's intrinsic parameters (focal length, principal point, distortion coefficients) are correctly determined. Given the ArduCam model uses manual focus, that was determined as the final calibration for the robot.
2. Marker Placement:

- Markers should be placed in a way that ensures they are visible to the camera throughout the operation.
- The global reference marker (ID 0) should be positioned in a location that is visible from all angles of the workspace.

3. Workspace Size:
   - The distance between the camera and markers should be within the range where the camera can reliably detect and decode the markers. Larger markers may be required for larger workspaces.

4.5. Task Assignment (ChoirBot)

The Centralized Task Assignment system for the ChoirBot is designed to efficiently manage and allocate tasks among multiple robots within a collaborative environment. This approach ensures that tasks are executed in a synchronized manner, optimizing resource utilization and enhancing overall system performance. Here's an overview of how the centralized task assignment operates:

System Overview

The centralized task assignment module serves as the brain of the ChoirBot system, coordinating the activities of individual robots (e.g., Pandebono and Mate) to achieve collective goals. The key components of this system include:

1. Task Queue Management:
   - A centralized task queue is maintained where all tasks are listed and prioritized based on urgency, complexity, and resource requirements.
   - Tasks can include navigation to specific locations, data collection, or interaction with other system components.
2. Robot Status Monitoring:
   - The system continuously monitors the status of each robot, including their current tasks, battery levels, and operational conditions (e.g., errors or malfunctions).
   - This information is vital for making informed decisions about task assignments and ensuring that no robot is overloaded or underutilized.
3. Task Assignment Algorithm:
   - A decision-making algorithm evaluates the current state of the robots and the task queue to assign tasks effectively.
   - Factors considered include:
     - Robot Capabilities: Each robot's specific skills and hardware capabilities are taken into account to ensure tasks are matched to the most suitable robot.
     - Proximity: Robots closer to the task location may be prioritized to reduce response time and improve efficiency.
     - Load Balancing: The algorithm strives to evenly distribute tasks among robots to avoid bottlenecks and ensure smooth operation.

5. Implementation

5.1. Robot Platform

Each of the robots required an additional platform to add the Aruco Markers on the top. The structure was designed to host a 15x15cm (the maximum the distance between the robot wheels) marker. Each of the robots can be seen with the structures on top, in the following figures.

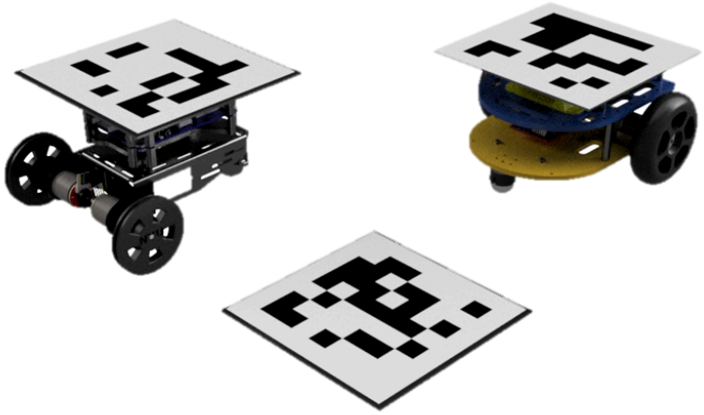

Fig5: Marker Attachment

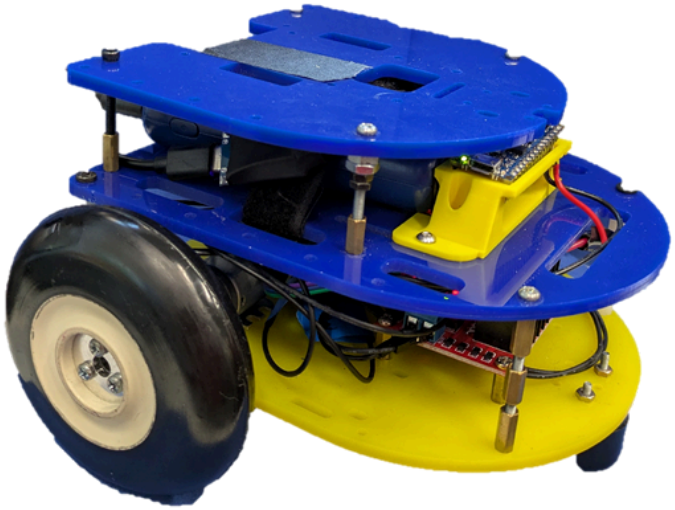

Fig6: Pandebono Attachment

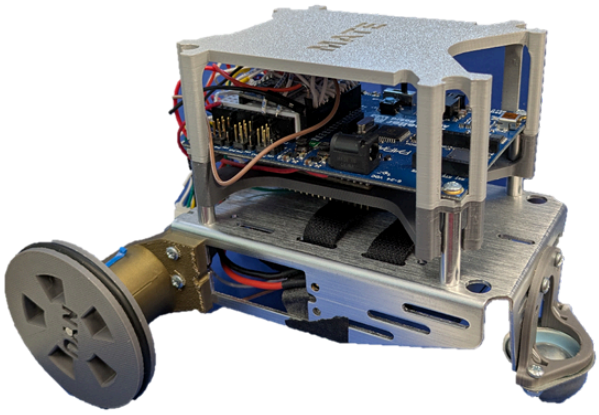

Fig7: Mate Attachment

5.2. Camera System and Computer Setup

For the camera system, two cameras were tested. First, an Android remote camera was tested for ArUCo marker detection using DroidCam as an IP Camera, but the brightness of white surfaces was too high, so the ArduCam camera was selected instead. As well, the delay in image capturing through the IP Cam was too high for an optimal control. The camera is placed in a tripod position for a general overview of the location of testing. Initially, the setup was placed in a room with windows for natural light, but the differences in testing different day times caused issues, therefore the setup was relocated to a room without windows and using only artificial light. This camera is connected to the computer using a USB port, and OpenCV libraries.

For camera calibration in OpenCV, the standard checker board was used, and 40 calibration images were selected. In this stage it is important to mention that multiple camera heights were tested for an optimal focus, as the camera uses manual focus. Finally, the camera calibration is performed with the final focus, which is above 1.70m.

5.3. Control Algorithms

5.3.1. **Navigation:**

For robot navigation, standard commands used in differential mobile robots are utilized. Specifically, the goal was to use topics widely used in the ROS community.

Twist messages from the geometry_msg package are received on the /cmd_vel topic in ROS 2. These commands are sent directly through the BLE bridge node, which is enabled using the

bleak library. The firmware then decodes the velocity commands, which include linear and angular velocities, into PWM signals for each motor.

For Path Planning, the local planner estimates positions to get to the desired goal, in general, this planner uses the guidance node. The local planner identifies the desired point, aligns the robot's orientation with the goal, and then estimates the euclidean distance.

5.3.2. **Task assignment control:**

The robot constantly receives its current position and orientation — either from the /odom topic (in simulation) or from the ArUco marker tracking system (in the physical setup). This information tells us where the robot is and which way it's facing.

Each robot subscribes to a goal topic (e.g., /agent_0/goal) where a target point is published. Once a goal is received, the robot starts calculating how to reach that position.

Once a goal is received, the robot calculates both how far away the goal is and in which direction it lies. It figures out the straight-line distance to the goal — essentially, how much it needs to drive forward — and also calculates how much it needs to turn to face that direction.

Using this information, the controller decides how fast the robot should move and how quickly it should rotate. If the robot is far from the goal, it will move faster to cover ground efficiently. As it approaches the target, it gradually reduces its speed to allow for a smooth and controlled stop. Both the forward and turning speeds are published together in a Twist message, which contains the linear and angular velocity values that drive the robot toward its goal.
As the formation moves, these calculations are done continuously so the robots can adapt and adjust their positions dynamically, even if one speeds up or slows down.

5.3.3. **Formation Control:**

The formation_control.py node is used in simulation to coordinate multiple robots to move while maintaining a specific formation — such as a line, triangle, or other geometric pattern. This is useful for swarm behaviors like area coverage or synchronized group movement.

Each robot in the simulation is aware of its own position and the positions of its neighbors, which are available through shared topics in the simulated environment. The formation is defined by expected relative distances between the robots.

As the robots move, the controller checks whether each robot is maintaining its correct position in the formation. If not, it calculates a velocity command to correct the position — essentially nudging the robot forward or adjusting its orientation slightly. These commands are also published as Twist messages to /cmd_vel.

Formation control runs continuously, allowing the robots to adapt dynamically and maintain their arrangement as they navigate the environment. This module is currently used only in the Gazebo simulation and is not dependent on ArUco markers.

6. Lessons Learned

6.1. Usability and Flexibility of Task Assignment

The implementation of ChoirBot demonstrated effective coordination in simulation, leveraging a dynamic task allocation among multiple virtual robots. This architecture facilitated straightforward task reassignment and adaptation to changing simulated scenarios. However, this approach also highlighted potential real-world limitations, including the laptop as a single point of failure and the necessity for reliable, low-latency communication, issues that became apparent when transitioning the controller designed in simulation to physical robot implementations.

6.2. Over-Reliance on ChoirBot ROS Package and Simulation Results

A key lesson was the over-reliance on simulation and the ChoirBot ROS package during development on the laptop. While simulation was crucial for initial validation, it did not fully account for real-world complexities. This included potential computational limits on the laptop when processing data from multiple physical robots, unmodeled physical effects like varying lighting and camera calibration drift, and hardware-specific integration challenges. This discrepancy between the simulated and physical environments underscored that deploying ChoirBot on real robots would not be a simple plug-and-play process and would necessitate significant adjustments and troubleshooting beyond the initial simulation-based understanding. Noticeably, the controller had to be completely tweaked for real robots, in comparison to simulated ones.

6.3. System Scalability and Testing Limitations

Although the system was designed with scalability in mind, most testing was performed with two robots, and the ChoirBot ROS package was primarily validated for a fixed number of agents (N=6). As a result, the system's behavior for other robot counts ($N \neq 6$) was not thoroughly characterized. Some edge cases and bugs only appeared during physical multi-robot tests, highlighting the importance of comprehensive hardware validation across a range of configurations.

6.4. Integration and Validation: Key Project Management Lessons

Reflecting on the overall project workflow, several important lessons emerged:

- Validate theory in real environments as early as possible. Early integration of hardware and software components can reveal practical issues that are not apparent in simulation or isolated subsystem tests.
- Perform multi-robot integration with identical hardware. Differences in robot platforms (e.g., sensors, chassis, firmware) complicated integration and

debugging. Standardizing hardware across the swarm would streamline development and reduce unforeseen incompatibilities.
- Iterative, incremental testing—from single-robot to multi-robot scenarios—proved essential for identifying and resolving issues before full system deployment.

7. Conclusion

In conclusion, this project successfully demonstrated the implementation of the CHOIRBOT framework, integrating vision-based localization with two physical differential-drive robots and a simulated TurtleBot3. The journey to achieve a functional system, however, was not without its challenges, particularly due to the reliability issues encountered with the ChoirBot framework. The transition from simulation to real-world application revealed discrepancies that necessitated extensive troubleshooting and adaptation. This experience underscores the critical importance of rigorous hardware validation and real-world testing, which often expose challenges not evident during initial simulations.

Moreover, while the original ChoiRbot system was primarily tested with a configuration of six robots (N=6), the limitations of this approach became apparent. The lack of comprehensive testing across varying robot counts restricted our understanding of the system's scalability and performance in diverse configurations. Edge cases and bugs emerged when transitioning to physical tests, highlighting the necessity for a more robust validation process that includes a broader range of operational scenarios. Future work should focus on expanding the testing framework to include various robot counts and hardware configurations to ensure the system's adaptability and resilience in real-world environments.

Ultimately, the insights gained from this project provide a valuable foundation for future research in multi-robot systems. While simulations serve as a beneficial starting point, the complexities of real-world implementation require careful consideration and iterative testing. Moving forward, it will be essential to standardize hardware across the robotic swarm and to prioritize early integration of all components, thereby enhancing the robustness of the system and reducing unforeseen integration challenges. This project not only contributes to the ongoing development of the CHOIRBOT framework but also lays the groundwork for advancements in autonomous robotic coordination and swarm behavior.

8. References

1. Ravikumar, T. M., & Saravanan, R. (2014). Reduction of odometry error in a two-wheeled differential drive robot. *Int. J. Eng*, 27(3), 359-366.
2. Krajewski, J., & Khosravi, M. (2020). Simultaneous Localization and Mapping of a small-scale vehicle using low-cost IMU, optical wheel encoders and. DiVA portal.
3. Figueiredo, F.A.V., Pereira, E.G.C., Vasques, C.M.A. (2023). Indoor Navigation of an Autonomous Guided Vehicle Using ArUco Markers. In: Almeida, F.L., Morais, J.C., Santos, J.D. (eds) Multidimensional Sustainability: Transitions and Convergences. ISPGAYA 2022. Springer Proceedings in Earth and Environmental Sciences. Springer, Cham. https://doi.org/10.1007/978-3-031-24892-4_20
4. ChoiRBot: https://github.com/OPT4SMART/ChoiRbot
5. AruCo: https://docs.opencv.org/4.x/d5/dae/tutorial_aruco_detection.html