SwarmBots: Autonomous Multi-Agent Task Execution with Centralized Ultrasonic Mapping

B. Sumanth, M. Sowmith and G.V.V. Sharma
Department of Electrical Engineering,
Indian Institute of Technology Hyderabad,
Kandi, India 502284
gadepall@ee.iith.ac.in

***Abstract*—This paper presents a novel low-cost, centralized framework for mapping and multi-task execution using multiple Unmanned Ground Vehicles (UGVs). The system leverages ultrasonic sensors for environmental mapping, followed by autonomous task execution using a master-slave architecture. A user-interactive graphical interface allows manual selection of task points on the generated map. The master UGV maps the environment and broadcasts task locations to slave UGVs for autonomous execution. Experimental results demonstrate reliable mapping accuracy and robust coordination among UGVs in a controlled indoor environment.**

1 Introduction

The evolution of robotics has enabled autonomous systems across logistics, agriculture, and surveillance through collaborative multi-agent behavior [1]. Most commercial solutions, however, rely on costly technologies like LiDAR, GPS, and high-performance processors, making them unsuitable for educational and small-scale indoor deployments [2].

1.1 Problem Statement

There is a lack of low-cost, modular systems that can perform indoor multi-UGV mapping and task execution without relying on infrastructure-dependent technologies. Current methods are either too expensive, too centralized, or lack user-friendly interfaces for real-time coordination in constrained environments.
This project addresses that gap by proposing a centralized control system using ESP32 microcontrollers, ultrasonic sensors [3], and ESP-NOW communication [4]. An intuitive web-based GUI built with SPIFFS [5] allows users to assign tasks and direct swarm behavior interactively, without external routing hardware. Our framework supports scalable swarm control for research and prototyping [6], [7].

2 Related Work

Multi-robot coordination has been extensively studied, with emphasis on areas like SLAM, dynamic task allocation, and swarm behavior [1], [2]. For instance, Jin et al. (2024) introduced a deep reinforcement learning model (DDQN) combined with Ant Colony Optimization for dynamic task assignment in UGV swarms [6]. Similarly, hierarchical approaches for UAV-UGV cooperation using global and local planners have been explored in [7].

Despite these advances, minimal focus exists on using affordable sensors (like HC-SR04) in tandem with ESP-based embedded systems to create real-time indoor mapping systems with GUI-driven task delegation [3]–[5]. This work fills that gap by integrating time-aware task distribution, ESP-NOW communication, and manual path annotation, all under a cost-efficient architecture.

3 Proposed Navigation System

The proposed architecture for multi-UGV coordination is detailed in this section. Each UGV contains hardware for communication and autonomous navigation. With a central master UGV, the system can autonomously execute tasks. Key components of the system include the following.

1) *Collaborative Mapping Using Ultrasonic Sensors:* Ultrasonic modules (HC-SR04) mounted on the master UGV scan surroundings using a grid traversal strategy. Position is estimated via RPM-based dead reckoning, and obstacle data is processed into a 2D occupancy grid map [3]. The complete environment map is constructed in real-time and visualized via the GUI.
2) *Dead Reckoning for Localization* Each UGV computes its position based on wheel rotations (RPM), using:
$$\text{Distance} = \frac{\text{RPM} \times \pi \times \text{Diameter} \times \Delta t}{60} \quad (1)$$
where the various parameters are listed in Table 1. This allows real-time navigation feedback without relying on GPS or encoders.

Symbol	Description
d	Distance to be traveled
D	Diameter of the wheel
r	Radius of the wheel
L	Distance between the wheels (wheelbase)
θ	Angle of rotation in degrees
RPM	Rotations Per Minute of the motor
t_{linear}	Time to move forward by d cm
t_{turn}	Time to rotate by angle θ

TABLE 1: List of Symbols Used in Proposed Navigation System

3) *Time-Aware Task Distribution Algorithm:* After the user assigns points via the GUI, the master UGV estimates travel time to each target. Tasks are segmented and distributed proportionally to each slave's expected duration

to maximize parallelism and reduce idle time [6]. A more detailed explanation of the algorithm is provided in Section 3.

4) *Peer-to-Peer Communication with ESP-NOW:* ESP32 modules communicate using ESP-NOW—a low-latency, MAC-based protocol. It eliminates the need for a Wi-Fi router, offering robustness for offline, indoor multi-agent systems [4].
5) *Time-Based Control for Distance and Turning:* To navigate without GPS or encoders, each UGV calculates in advance the motor ON-time required to travel a specific distance or rotate by a desired angle. This is done using motor RPM, wheel diameter, and wheelbase geometry.
For linear motion, the time to travel a distance d is computed as:
$$t_{\text{linear}} = \frac{60 \times d}{\pi \times D \times \text{RPM}} \tag{2}$$
where D is the wheel diameter. The motors remain ON for t_{linear} milliseconds to achieve this displacement.
For rotation, assuming an in-place turn (e.g., left wheel backward, right wheel forward), the time to rotate by an angle θ (in degrees) is:
$$t_{\text{turn}} = \frac{L \times \theta \times 60}{360 \times r \times \pi \times \text{RPM}} \tag{3}$$
where L is the effective turning circumference (typically the distance between wheels), and r is the wheel radius. These time durations are precomputed and hardcoded for commonly used distances and angles (e.g., 10 cm forward, 90° turn), enabling accurate dead reckoning using only timers and motor control loops. This method allows reliable motion in resource-constrained setups without external sensors or feedback mechanisms.
6) *GUI-Based Human Interaction Layer:* The GUI is hosted on the master UGV and serves as the primary human-machine interface [5]. It enables the following functionalities:
   - Canvas-based path drawing Users can draw custom paths or regions directly onto the grid map for task planning.
   - Click-based task selection Individual grid points can be selected to assign discrete tasks to UGVs.
   - Real-time feedback of UGV positions Each UGV's estimated location is updated and visualized on the interface.
   - Task status visualization Task execution states (e.g., pending, in-progress, completed) are reflected dynamically on the GUI.
7) *Embedded Software Architecture:* Each UGV runs modular firmware developed using either the ESP-IDF or Arduino framework, depending on complexity. The software architecture is divided into:
   - Master UGV: Responsible for path segmentation, task scheduling, and GUI synchronization. It acts as a bridge between the user and slave units.
   - Slave UGVs: Execute motion control routines, update position through RPM-based dead reckoning, and report task status back to the master.

   Firmware modules are structured to separate communication, control logic, sensor interfacing, and utility functions for maintainability and reusability. The platform is engineered for deployment in resource-constrained environments, enabling scalable swarm behaviors through compact embedded logic and lightweight communication protocols.

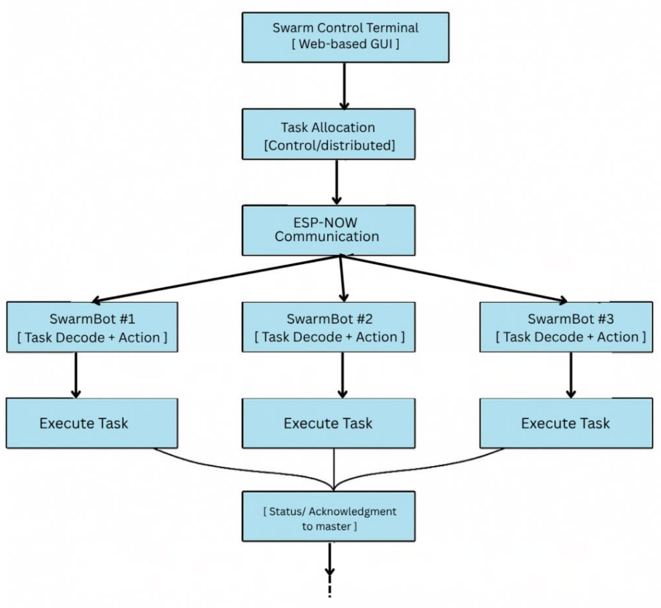

Fig. 1: SwarmBots system architecture: GUI-driven task allocation, ESP-NOW communication, autonomous execution, and status feedback.

4 Methodology

1) *Mapping Phase:* The master UGV performs 2D environmental mapping by traversing the environment in a structured grid pattern. It uses ultrasonic distance sensors [3] to gather obstacle data, which is fused with positional estimates obtained via dead reckoning [2]. A 2D occupancy grid is constructed dynamically to represent free and occupied spaces in the environment.
2) *GUI-Based Task Assignment:* The real-time occupancy grid is rendered on the GUI, allowing users to visually inspect the environment and select target task points, as mentioned in Section 6.
3) *Task Distribution Strategy:* Once the master UGV receives the list of task coordinates from the GUI, it

computes the estimated time each slave UGV would take to reach each target point. This estimation is based on straight-line distance and the kinematic model of the robot.

For each UGV, the expected time to travel a distance d is calculated using:

$$t = \frac{60 \times d}{\pi \times D \times \text{RPM}} \tag{4}$$

where D is the wheel diameter and RPM is the rotational speed of the motor. This time is used as a proxy for cost in task assignment.

The master then applies one of two strategies:

- *Round-robin assignment:* Tasks are distributed sequentially across all available slave UGVs, regardless of estimated time.
- *Time-aware distance-based allocation:* For each task point, the slave UGV with the lowest estimated arrival time (based on precomputed t) is assigned that task. This reduces overall execution time and idle waiting.

This time-aware strategy leads to smarter workload distribution, ensuring parallel execution while minimizing travel time, energy consumption, and path overlaps [6]. The final allocation is communicated to each UGV along with the necessary movement instructions.

4) *Execution Phase:* Upon receiving their assigned task coordinates, slave UGVs begin autonomous navigation. Navigation is achieved through time-based motor control or simple PID algorithms. On reaching the target, each UGV performs the associated operation, such as flashing an LED, taking sensor readings, or signaling task completion to the master.

5 Implementation

The complete source code and setup for the SwarmBots system are available on

https://github.com/gadepall/SWARMBOTS

5.1 Assets

1) UGV chassis with DC motors
2) ESP32 microcontroller with Type-B USB cable
3) L293D Motor Driver IC
4) Breadboard and Jumper Wires
5) Ultrasonic Sensor (HC-SR04)
6) Android Phone or Laptop for GUI (optional if GUI is pre-hosted)
7) (Optional) USB 2.0/3.0 Hub

5.2 Procedure

1) Build the UGVs using ESP32, motor driver, and ultrasonic sensors as per the wiring diagram provided in the repo.
2) Connect each ESP32 to your Android phone or laptop via USB.
3) For firmware compilation, open terminal and run:

```
$ cd SWARMBOTS/master_firmware
$ pio run
```

4) Upload firmware using ArduinoDroid (for Android users) or PlatformIO (on PC):

```
Actions → Upload → Upload Precompiled
```

and choose the firmware file at

```
codes/.pio/build/firmware.hex
```

5) Repeat for slave bots:

```
$ cd SWARMBOTS/slave_firmware
$ pio run
```

6) Access the GUI:
Open browser and navigate to the ESP32-hosted web server.
Draw or click on task points in the occupancy map.
The master UGV will distribute tasks accordingly.
7) Power on the UGVs in an open indoor environment. The master starts mapping; slaves receive task commands and execute them autonomously.
8) Monitor real-time updates and robot positions via the GUI interface.

6 Results

The SwarmBots system was tested in a structured indoor environment using a master-slave configuration of three UGVs. The key outcomes of the experiment are summarized below

Reliable Mapping Accuracy: The master UGV successfully generated a real-time 2D occupancy map using only ultrasonic data and dead reckoning. Spatial features such as walls and obstacles were distinguishable without requiring GPS, LIDAR, or encoders.

6.1 Interactive GUI Performance

The GUI enabled smooth human-robot interaction. Users could assign target points via simple clicks, and receive real-time visual feedback on robot positions and task progress.

6.2 Autonomous Multi-UGV Task Execution

Slave UGVs performed assigned tasks in parallel with high independence. Each UGV navigated to its target using internal logic, and task completion was communicated back to the master without human intervention.

6.3 Position Estimation Precision

Experimental trials demonstrated a positional accuracy within ±5 cm, sufficient for typical indoor navigation and task execution scenarios.

6.4 System Scalability

The framework scaled effectively with multiple UGVs. Task distribution strategies (round-robin and distance-based) helped avoid collisions and ensured balanced workloads.

6.5 *Low-Cost, High-Impact*

All experiments were conducted using off-the-shelf ESP32 boards and basic sensors, underscoring the potential of this architecture in resource-constrained deployments.

7 Conclusion and Future Work

In this work, we developed and demonstrated a low-cost, modular, and scalable system for autonomous task execution using a swarm of Unmanned Ground Vehicles (UGVs). Our approach integrated ultrasonic-based mapping, RPM-based dead reckoning, and a user-friendly GUI for task coordination. The system successfully enabled collaborative mapping and parallel task execution in an indoor setting without relying on GPS, LiDAR, or expensive sensors. The experiments confirmed the reliability of position estimation, real-time feedback, and decentralized task handling using ESP-NOW communication. The architecture proved to be robust, scalable, and efficient for indoor robotic coordination, even with hardware constraints and future work includes.

- Incorporating IMU and optical encoders to enhance localization accuracy
- Implementing advanced obstacle avoidance using IR or vision-based sensors
- Developing adaptive task reassignment algorithms based on real-time status
- Extending the system to semi-structured outdoor environments with longer range communication
- Introducing swarm-level intelligence to support dynamic decision-making and reconfiguration

References

[1] M. Brambilla, E. Ferrante, M. Birattari, and M. Dorigo, "Swarm robotics: A review from the swarm engineering perspective," *Swarm Intelligence*, vol. 7, no. 1, pp. 1–41, 2013.

[2] N. Kalra, D. Ferguson, and A. Stentz, "Hoplite: A markov decision process-based multirobot coordination system," *The International Journal of Robotics Research*, vol. 24, no. 10, pp. 971–1000, 2005.

[3] Arduino, "HC-SR04 Ultrasonic Distance Sensor Tutorial," 2023, [Online]. Available: https://www.arduino.cc/en/Tutorial/UltrasonicSensor.

[4] Espressif Systems, "ESP-NOW User Guide," 2023, [Online]. Available: https://docs.espressif.com/projects/esp-idf/en/latest/esp32/api-reference/network/esp_now.html.

[5] Random Nerd Tutorials, "ESP32 Web Server using SPIFFS (Host HTML Files)," 2024, [Online]. Available: https://randomnerdtutorials.com/esp32-web-server-spiffs.

[6] X. Jin *et al.*, "Multi-ugv path planning and task assignment via ddqn + aco," *IET Intelligent Transport Systems*, 2024.

[7] MDPI Robotics, "Hierarchical plan execution for cooperative uxv missions," *Robotics*, 2022.