REPORT ON

SWARMBOTS: AUTONOMOUS MULTI-AGENT TASK EXECUTION

SUBMITTED BY
MADOORI SOWMITH

Under The Guidance of
G.V.V SHARMA

FUTURE WIRELESS COMMUNICATION

INDIAN INSTITUTION OF TECHNOLOGY, HYDERABAD (IITH)

JULY 2025

Contents

ABSTRACT

This project presents SwarmBots, an embedded system designed for the autonomous coordination of multiple Unmanned Ground Vehicles (UGVs) using time-aware task distribution and fully offline communication. The system enables users to draw custom paths via a web interface hosted on the master ESP32, which segments the path and distributes sub-tasks among slave UGVs based on estimated execution time. Each UGV performs real-time position estimation using motor runtime and wheel specifications, eliminating the need for GPS or encoders.

Communication between units is handled using the ESP-NOW protocol, allowing fast, peer-to-peer messaging without requiring internet or a router. The master UGV also provides live feedback and visual tracking of all bots through the same web interface.

This approach demonstrates that complex swarm behavior can be achieved with minimal hardware, low power, and no external infrastructure. The system is scalable, low-cost, and suitable for real-world applications such as warehouse automation, precision agriculture, and educational robotics.

Chapter 1

Introduction

1.1 Background

In the era of autonomous systems, Unmanned Ground Vehicles (UGVs) are increasingly used in surveillance, delivery, and search-and-rescue operations. Swarm robotics where multiple robots collaborate to execute a task offers scalability, fault tolerance, and efficiency. However, most existing swarm systems depend heavily on GPS, centralized servers, or expensive sensors like LIDAR for mapping and coordination.

Our project presents a fully autonomous swarm UGV system that operates entirely offline using only ESP32 modules, RPM based dead reckoning, and a custom web interface. Unlike traditional approaches, our system begins by collaboratively mapping the environment each UGV autonomously explores a designated region. Once the map is formed, the user can select target points directly on the map, triggering a sequence of operations including task distribution, optimal path planning, and obstacle aware execution.

This layered approach starting from mapping, then to dynamic task execution proves that intelligent swarm coordination can be achieved with minimal hardware and no reliance on GPS, encoders, or internet infrastructure

1.2 Motivation

The idea for this project emerged from a powerful, yet simple question - What if you could just explore or scan an area with robots, and then assign them tasks by simply clicking on a map? That vision led us to rethink how multiple robots could collaborate autonomously, without relying on GPS, internet, or high-end sensors. We aimed to combine embedded control, peer-to-peer wireless networking, and an intuitive web-based interface into one seamless swarm solution.

In contrast, most real-world systems whether drone fleets, warehouse bots, or agricultural vehicles depend on costly infrastructure: GPS for localization, LIDAR or vision systems for tracking, and cloud services for coordination. These setups not only increase cost but become unusable in remote or post-disaster zones where connectivity is lost.

So we challenged the norm and asked:

- Can UGVs map and localize themselves without GPS or encoders?
- Can they be controlled and coordinated without the internet?
- Can we visually draw, select, and assign tasks and see it all work, live and offline?

Our answer: **SwarmBots** a lightweight swarm framework where UGVs first explore and map an environment collaboratively, and then dynamically receive tasks, calculate optimal paths, and avoid obstacles all using only RPM-based dead reckoning, a web dashboard, and ESP-NOW communication. This vision of doing more with less became our driving force focused on minimal hardware, full offline capability, and scalable swarm behavior.

1.3 Problem Statement

Traditional multi-robot systems often rely on centralized control and GPS-based navigation, which introduces several limitations:

- **Dependence on External Infrastructure:** GPS fails indoors, and cloud services require stable internet connectivity. LIDAR and vision systems demand heavy computation and cost.
- **Rigid Task Assignment:** Most setups require pre-programmed task distribution or manual control per robot, making them unsuitable for dynamic environments.
- **System Scalability Issues:** Adding new robots typically requires reconfiguring the entire system, reducing flexibility.

These limitations form a major bottleneck when designing a swarm system for offline, real-time applications, especially in network-restricted or post-disaster zones.

The key challenges we targeted were:

- How to collaboratively map an unknown area without GPS, LIDAR, or encoders?
- How to enable point selection on the map after mapping, and then assign tasks dynamically based on user input?
- How to calculate optimal sub-paths and avoid obstacles using only onboard logic and basic sensors?
- How to achieve all of the above using peer-to-peer ESP-NOW communication, without Wi-Fi routers or cloud APIs?

Our goal was to build a fast, fully offline, low-cost, and scalable system where UGVs could not only localize and map environments, but also autonomously split tasks and navigate intelligently all without traditional infrastructure.

1.4 Objective

The primary objective of this project was to design, develop, and demonstrate a fully offline swarm UGV system capable of autonomous mapping, task assignment, and coordinated path execution, using only embedded hardware and local communication.

Specifically, the system enables multiple UGVs to:

- Collaboratively explore and map an unknown environment by dividing the area (e.g., left/right) and estimating their positions using RPM-based dead reckoning without GPS, encoders, or LIDAR.
- Select points directly from the generated map using a web interface, allowing users to assign tasks visually.
- Distribute tasks intelligently using a Time-Aware Path Distribution Algorithm, ensuring balanced workload among all bots based on estimated execution time and distance.
- Use ESP-NOW, a low-latency, peer-to-peer protocol, for all master-slave communication, ensuring fully offline operation.
- Navigate and execute tasks using only onboard logic while avoiding obstacles and updating the master in real-time.
- Visualize everything through an intuitive, self-hosted web dashboard that includes:
  - Canvas-based map/path drawing
  - Click-based target selection
  - Live UGV position tracking
  - Execution status and logs
  - Manual override buttons

Secondary objectives included:

- Ensuring modularity and scalability easily adding more UGVs with minimal configuration
- Maintaining low-power and low-cost hardware constraints
- Using only open-source tools and avoiding cloud/internet dependency

Chapter 2

Literature Survey

The field of swarm robotics has been extensively studied, focusing on decentralized coordination inspired by natural systems like ant colonies or bee swarms. Notable systems such as Amazon's Kiva robots, Starship delivery bots, and ROS-based multi-robot platforms have demonstrated scalable swarm behaviors but they depend heavily on centralized servers, GPS for localization, or expensive sensors like LIDAR for mapping and path tracking.

These high-cost systems often require:

- Central cloud servers for coordination
- GPS for localization (which fails indoors)
- High-performance processors for SLAM or vision-based navigation
- Wi-Fi infrastructure for communication

In contrast, our work addresses a technological gap: achieving similar functionality in fully offline environments using only embedded microcontrollers, without GPS, LIDAR, encoders, or internet.

Previous studies on multi-agent task allocation often rely on:

- Auction-based protocols or Contract-Net protocols, which require constant two-way communication and computing overhead
- Centralized SLAM-based mapping, which is resource-heavy and not feasible on low-cost embedded systems

Our system takes a lightweight, embedded-first approach:

- Collaborative RPM-based mapping is performed before any task assignment, enabling a virtual coordinate space to be created using dead reckoning.
- Once mapped, the user can select target points on the map and the system automatically handles task distribution and obstacle-aware path planning.
- All coordination is done using ESP-NOW, which offers low-latency, peer-to-peer messaging without needing routers or cloud APIs.
- The web-based control panel hosted on the master ESP32 offers full offline interaction, from path drawing to execution feedback.

Chapter 3

Methodology

To develop the SwarmBot system, we conducted an extensive review of research papers, industrial practices, and technical resources related to swarm robotics, embedded motion tracking, and task distribution algorithms. The following key areas guided the design and implementation of our system.

3.1 Swarm Robotics and Multi-Agent Coordination

Swarm robotics is based on principles derived from collective behavior observed in biological systems such as ant colonies, bird flocks, and bee swarms. In engineering, this translates into decentralized control, fault tolerance, and parallel task execution. Key literature, including works by Brambilla et al. (2013) on swarm behavior modeling and optimization, emphasizes the importance of simple local rules enabling complex group behavior. We adapted these ideas to a small-scale, embedded implementation where a master UGV allocates sub-tasks and slave UGVs operate semi-independently. Instead of using AI-based swarm logic, we created a deterministic and predictable distribution mechanism based on task time and distance.

3.2 Task Distribution Strategies in Multi-Robot System

Various algorithms are used in multi-agent robotics for task allocation, including Round Robin, First-Come-First-Serve (FCFS), Auction-Based, and Contract-Net Protocols. These often assume full bidirectional communication and high-level computing infrastructure. We instead designed a Time-Sliced Path Distribution Algorithm, which breaks a user-drawn path into equidistant points, estimates time to travel each segment, and distributes them across UGVs to balance workload. This was inspired by dynamic load balancing methods used in parallel computing, where tasks are assigned based on computational effort or execution time.

3.3 Collaborative Mapping Mechanism

Before task allocation, the system enters Map Mode, where the environment is divided between UGVs. This collaborative scan builds a virtual map, enabling the user to click target points, after which the system performs task distribution, path planning, and obstacle-aware navigation all offline and without GPS, LIDAR, or encoders.

3.4 RPM-Based Odometry and Dead Reckoning

Dead reckoning is the process of estimating a robot's current position using previous position, speed, and heading data. We studied encoder-based odometry used in basic robotics where motor RPM, wheel diameter, and time are used to calculate displacement:

$$\text{Distance} = \frac{\text{RPM} \times \pi \times \text{Diameter} \times \Delta t}{60}$$

We adapted this method for ESP32-based UGVs by reading encoder pulses, calculating RPM in real-time, and updating (x, y) positions using simple kinematic equations. Resources from ROS documentation, open-source Arduino libraries, and MIT's OpenCourseWare provided foundational knowledge.

3.5 Embedded Communication Protocols – ESP-NOW

For swarm-level communication, we needed a protocol that:

- Does not rely on a Wi-Fi router
- Supports many-to-many communication
- Has low latency and energy efficiency

ESP-NOW, developed by Espressif for the ESP32, enables fast, lightweight peer-to-peer messaging without the need for Wi-Fi infrastructure. Each ESP32 is addressed using its MAC address, and messages are sent in broadcast or unicast mode. Our system uses ESP-NOW for master-to-slave communication of tasks and slave-to-master acknowledgment of completion.

Datasheets, Espressif's official documentation, and community forums like GitHub and RandomNerdTutorials helped in understanding and customizing ESP-NOW for our swarm.

3.6 Web-Based User Interface for Robotics Challenges

For intuitive control and visualization, we implemented an HTML + JavaScript-based webpage hosted on the master UGV via SPIFFS. It includes:

- Canvas-based path drawing
- Coordinate extraction and path segmentation
- Live status display and control buttons

Reviewed several open-source robot dashboards and JavaScript libraries for drawing and DOM manipulation. Inspiration came from MIT App Inventor logic blocks, ROS WebTools, and real-time UI examples using websockets and AJAX.

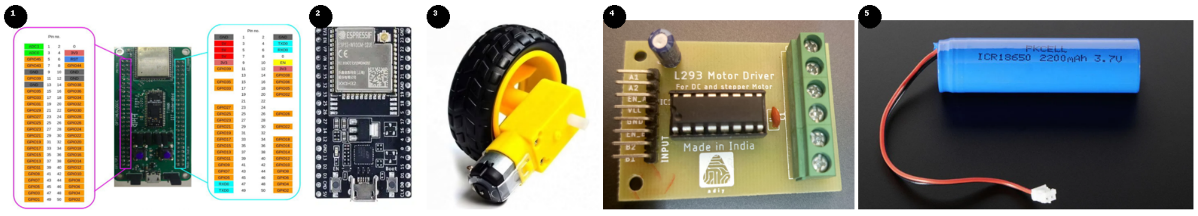
3.7 Real-World Swarm Use Cases and Challenges

We analyzed real-world systems like Amazon Kiva robots (warehouse bots), Starship delivery robots (campus delivery), and SwarmDrones (AI-based swarm control) to understand practical design considerations:

- Simple, robust hardware
- Offline capability in poor connectivity zones
- Scalable and modular architecture

While these systems rely on expensive hardware (e.g., LIDAR, SLAM, GPS), our project was designed to replicate the behavior using embedded techniques proving that basic hardware can also achieve useful swarm functionality with clever logic.

3.8 Component Specification



1. A VAMAN LC-1 Development Board

A VAMAN LC-1, development board which has **ESP32**, **ARM CORTEX M4**, and **PYGMY FPGA** in it.

2. ESP32 Microcontroller

Each UGV is powered by an ESP32 microcontroller.Its dual core processing and built-in Wi-Fi make it ideal for decentralized swarm control.

3. DC Geared Motors with Wheels

Each UGV uses two DC geared motors with attached rubber wheels to enable differential drive motion.The third point of support is provided by a castor wheel for balance and smooth turns.

4. Motor Driver Module L293N

Receives control signals from the ESP32 and drives the two DC motors accordingly. It handles current amplification and directional control, enabling the UGVs to move forward, backward, or rotate.

5. Battery Pack

A 9V battery or Li-ion pack is used to power both the ESP32 and motor driver.

Chapter 4

System Architecture

4.1 Web-Based Control and Mapping Layer

At the top of the system is a web-based interface hosted directly on the ESP32 master using SPIFFS. When users enter “Map Mode,” the interface allows them to trigger a collaborative mapping operation, where the master and slave UGVs are each assigned a region (e.g., left/right half of the map) to explore autonomously.

Each UGV updates its position using RPM-based dead reckoning as it moves. These live positions are displayed in real time on a grid-based canvas, allowing users to observe the ongoing scan visually. Once mapping is complete, the user can click on the virtual map to define target locations for task execution.

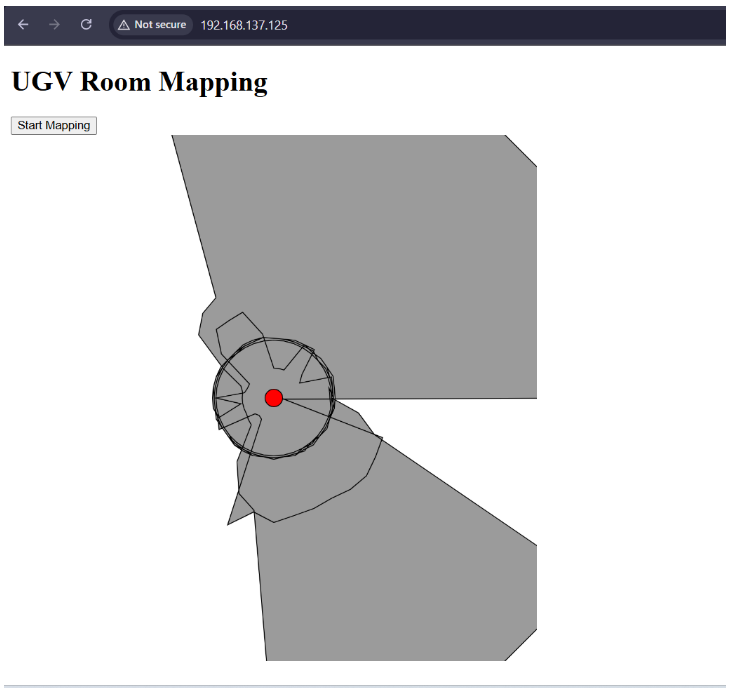
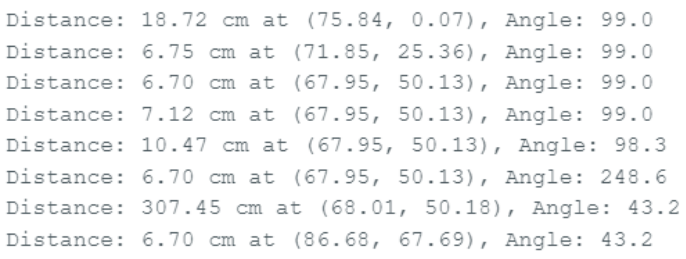


Figure 4.1: The image shows the web-based interface hosted on the master ESP32

The grey shapes represent the walls and boundaries identified by the robot using its RPM-based position tracking and motion updates. The red dot denotes the UGV’s current position, and the black path traces its movement history. As the UGV rotates and moves, each detected boundary point is plotted to create a virtual map of the surroundings.



```
Distance: 18.72 cm at (75.84, 0.07), Angle: 99.0
Distance: 6.75 cm at (71.85, 25.36), Angle: 99.0
Distance: 6.70 cm at (67.95, 50.13), Angle: 99.0
Distance: 7.12 cm at (67.95, 50.13), Angle: 99.0
Distance: 10.47 cm at (67.95, 50.13), Angle: 98.3
Distance: 6.70 cm at (67.95, 50.13), Angle: 248.6
Distance: 307.45 cm at (68.01, 50.18), Angle: 43.2
Distance: 6.70 cm at (86.68, 67.69), Angle: 43.2
```

Figure 4.2: Real-Time Distance and Coordinate Logs

This data directly feeds into the virtual map displayed on the left side, allowing accurate reconstruction of rooms and spaces even without GPS, LIDAR, or encoders.

4.2 4.2 Task Allocation and Distribution Layer

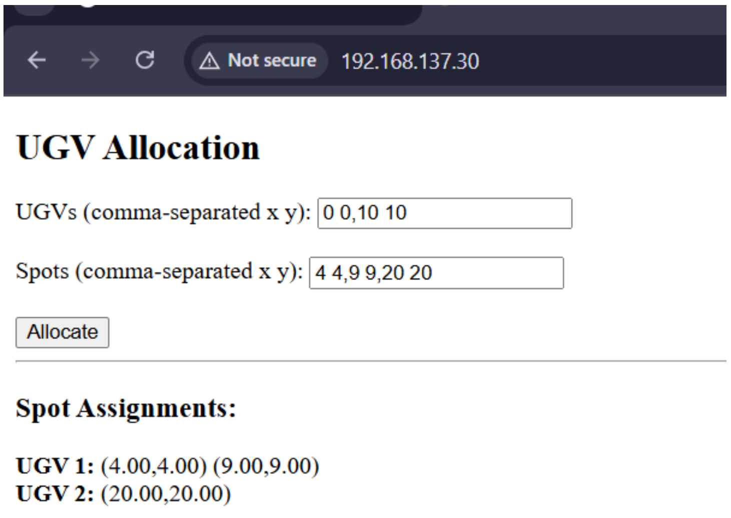
After mapping, users can either:

- Click target coordinates, or
- Draw custom paths on the map

The system then applies a Time-Aware Path Distribution Algorithm, which:

- Segments the selected path or points
- Estimates execution time per segment

Assigns segments to each UGV based on proximity and load balancing. The assignments are communicated over ESP-NOW, a low-latency, peer-to-peer protocol that requires no router or internet.



UGV Allocation

UGVs (comma-separated x y): 0 0,10 10

Spots (comma-separated x y): 4 4,9 9,20 20

Spot Assignments:

UGV 1: (4.00,4.00) (9.00,9.00)
UGV 2: (20.00,20.00)

Figure 4.3: UGV Spot Allocation Interface with Time-Aware Task Distribution

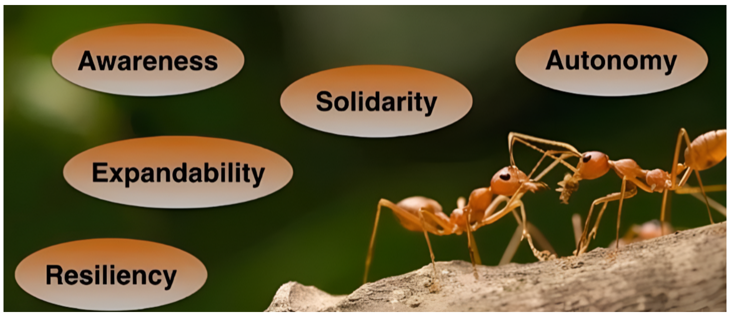


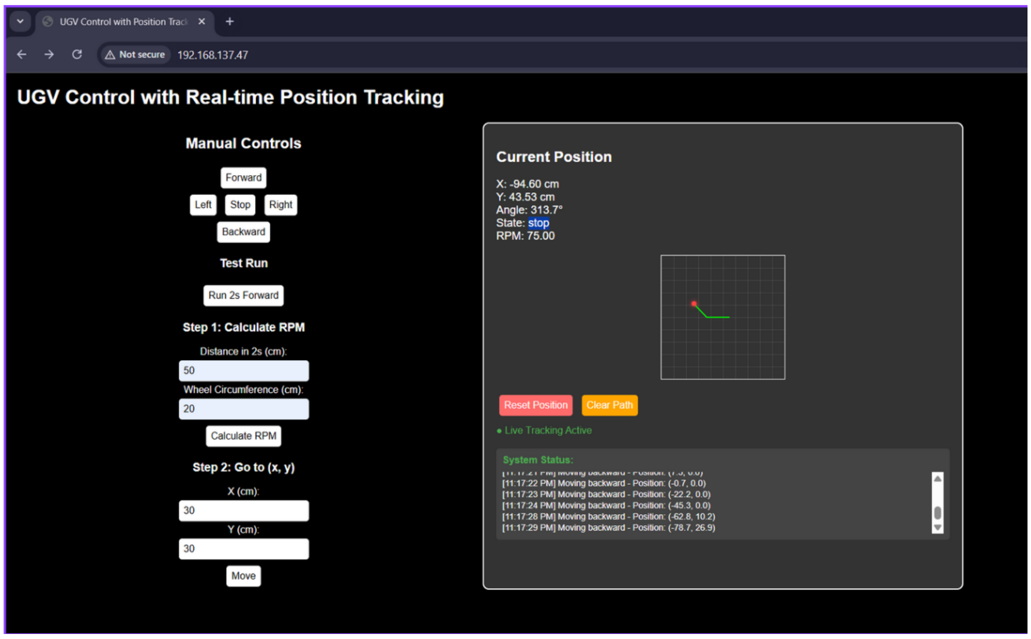
Figure 4.4: Core Traits of Swarm Intelligence Reflected in SwarmBot Architecture

4.3 4.3 Motion Execution and Feedback Layer

Upon receiving their tasks, each UGV:

- Uses motor run-time, RPM, and wheel geometry to calculate displacement
- Follows the assigned (x, y) waypoints autonomously
- Sends live position updates and completion acknowledgments back to the master

Motor direction and speed control are handled through the L293N motor driver, powered by a 9V battery or Li-ion pack. All execution and feedback happen entirely offline, using only local hardware and onboard software.



UGV Control with Real-time Position Tracking

Manual Controls

Test Run

Step 1: Calculate RPM

Distance in 2s (cm): 50

Wheel Circumference (cm): 20

Step 2: Go to (x, y)

X (cm): 30

Y (cm): 30

Current Position

X: -94.60 cm
Y: 43.53 cm
Angle: 313.7°
State: stop
RPM: 75.00

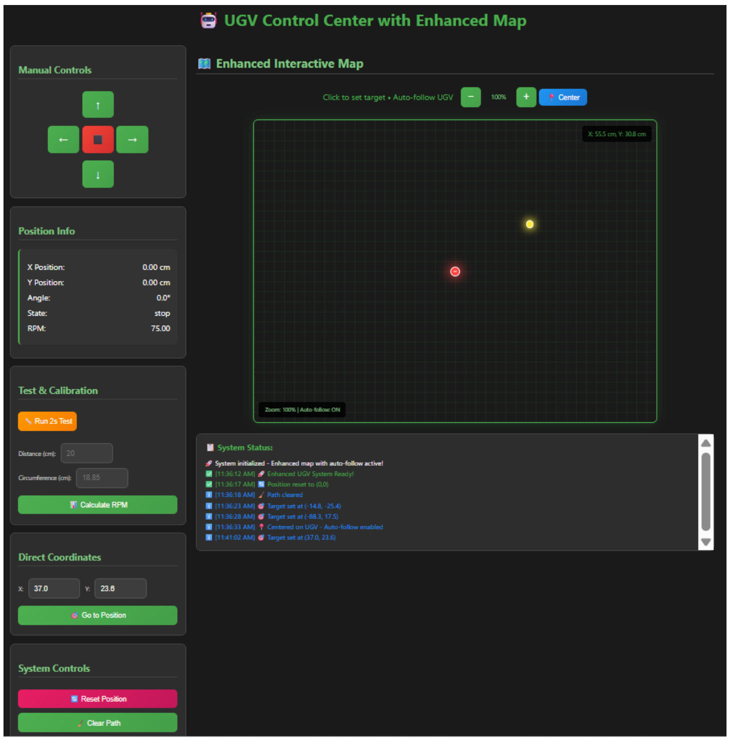
System Status:

```
[11:17:21 PM] Moving backward - Position: (7.3, 0.0)
[11:17:22 PM] Moving backward - Position: (-0.7, 0.0)
[11:17:23 PM] Moving backward - Position: (-22.2, 0.0)
[11:17:24 PM] Moving backward - Position: (-45.3, 0.0)
[11:17:28 PM] Moving backward - Position: (-62.8, 10.2)
[11:17:29 PM] Moving backward - Position: (-78.7, 26.9)
```

Figure 4.5: Web Interface for UGV Control and Real-Time Position Tracking

This interface, hosted on the master ESP32, allows users to control and monitor the UGV in real time. Manual control buttons are provided for movement, along with a test run option to calculate **RPM** using wheel specs and distance. Users can also enter target (x, y) coordinates for autonomous movement.

On the right, the interface displays the UGV's current position, angle, RPM, and state. A small grid shows the live path traced by the bot, while a system log (like serial monitor) records each movement. The entire system runs offline and enables full user interaction without any external server.


UGV Control Center with Enhanced Map

Manual Controls

Position Info

X Position: 0.00 cm
Y Position: 0.00 cm
Angle: 0.0°
State: stop
RPM: 75.00

Test & Calibration

Distance (cm): 20
Circumference (cm): 18.85

Direct Coordinates

X: 37.0 Y: 23.6

Enhanced Interactive Map

X: 55.5 cm, Y: 30.8 cm
Zoom: 100% | Auto-follow: ON

```
System Status:
System initialized - Enhanced map with auto-follow active!
[11:36:12 AM] Enhanced UGV System Ready!
[11:36:17 AM] Position reset to (0,0)
[11:36:18 AM] Path cleared
[11:36:23 AM] Target set at (-14.8, -25.4)
[11:36:28 AM] Target set at (-88.3, 17.5)
[11:36:33 AM] Centered on UGV - Auto-follow enabled
[11:41:02 AM] Target set at (37.0, 23.6)
```

Figure 4.6: Enhanced Web Interface with Interactive Map Control

This updated interface allows users to directly click on any point within the map area to set a target position for the UGV. The selected (x, y) coordinates are immediately sent to the UGV via ESP-NOW, and the bot autonomously moves to the location using real-time position tracking. The map features zoom, centering, and auto-follow options to monitor the bot's motion. Position info, manual controls, and system logs are also displayed for complete interaction and feedback all hosted offline on the ESP32.

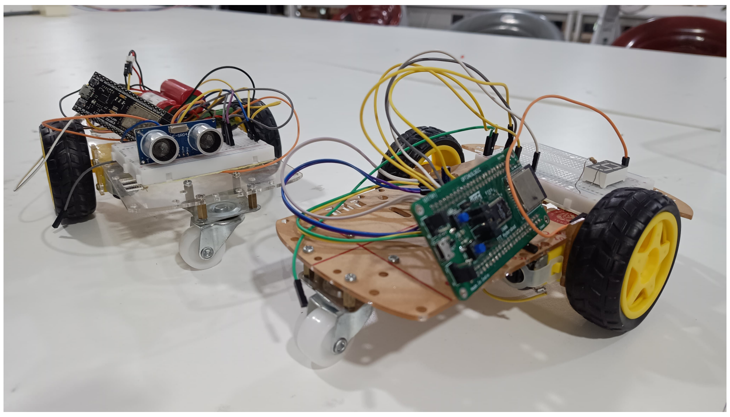


Figure 4.7: Master and Slave UGV Prototypes Used for Swarm Mapping and Coordination

The Communication Layer handles sending sub-paths to each slave UGV using the ESP-NOW protocol. Each slave UGV is identified using its unique MAC address, allowing targeted communication.

The data packet sent to each UGV contains:

- A list of assigned (x, y) coordinates
- Motion instructions
- A start signal to begin execution

Since ESP-NOW is connectionless and low-latency, it allows direct, fast communication between devices without needing a Wi-Fi router or internet.

Once a slave receives its task, the Motion Execution Layer becomes active:

- Position estimation is performed using dead-reckoning based on:
  - Known motor RPM
  - Wheel circumference
  - Motor run-time
- No encoders are used all movement is calculated using basic kinematic equations
- Motor control is handled through the L298N motor driver
- After reaching its assigned destination, each UGV sends a completion acknowledgment back to the master via ESP-NOW

The web interface is updated in real-time, showing:

- Each UGV's path
- Current status
- Completion updates

The entire architecture is:

- Fully offline (no internet required)
- Scalable (easily supports more UGVs)
- Hardware-minimal (no GPS, no encoders, no routers)

All modules operate independently and reliably, making the system robust and suitable for real-world deployment.

Chapter 5

Results Obtained

The SwarmBots system was successfully implemented and tested using multiple UGVs in a structured indoor environment. The following outcomes demonstrate the core capabilities of the system.

Collaborative Mapping Execution
Upon initiating Map Mode, UGVs explored assigned regions of the environment (e.g., left/right zones) using RPM-based dead reckoning. The master ESP32 divided the space and communicated zone boundaries to each bot. The system generated a real-time virtual map displayed on the web interface, allowing complete spatial visualization without any GPS, LIDAR, or encoders.

User-Driven Target Selection Post-Mapping
Once mapping was complete, users could directly click on any location within the virtual map to assign task points. This formed the bridge between environment exploration and task execution, enabling dynamic interaction through a fully offline dashboard.

Autonomous Execution by Multiple UGVs
Each UGV independently executed its assigned segment using dead reckoning and on-board timing. Tasks were performed in parallel, with no manual intervention, leveraging ESP-NOW communication for coordination and status updates.

Accurate Position Estimation Without GPS or Encoders
UGVs estimated their live positions using motor RPM and wheel geometry. Test runs showed a maximum position deviation of ±5 cm, which was sufficient for reliable operation in indoor mapped environments.

Real-Time Tracking and Visual Feedback
The ESP32-hosted dashboard updated the position, angle, and state of each UGV in real time. Features like auto-follow, zoom, and path trace allowed users to monitor all bots throughout mapping and execution phases with complete visibility.

Scalability and Load Balancing
The system was tested with up to 3 UGVs operating concurrently. Each UGV mapped and executed its assigned region independently, with no overlap, demonstrating the scalability of both the mapping and task distribution algorithms.

Note: All technical documentation supporting these results are available in"GitHub".

Chapter 6

Practical Applications

The SwarmBot system demonstrates a scalable and affordable platform for real-world multi-UGV coordination. Its offline capability, autonomous task distribution, and web-based control make it ideal for a range of applications:

Warehouse Automation
UGVs can be assigned to transport goods to various shelves or pickup zones. The master can dynamically divide storage tasks among bots without any human supervision or central server.

Precision Agriculture
The system can divide crop fields into regions and assign monitoring or spraying tasks to different UGVs. The time-aware allocation ensures each bot handles a balanced portion of the field.

Disaster Response and Search Missions
In GPS-denied or infrastructure-damaged environments, SwarmBot can deploy UGVs to explore different zones, mark visited areas, or deliver supplies all coordinated offline.

Educational Demonstrations
The visual nature of the interface and swarm behavior makes SwarmBot an excellent educational tool for teaching embedded systems, robotics, and autonomous systems.

Factory Floor Patrol or Inspection
UGVs can be tasked with patrolling separate sections of a factory or lab, inspecting parameters, or performing basic deliveries within the building.

Custom Path Execution Systems
Any user-defined path such as shapes, letters, or predefined patrol routes can be divided and executed by the swarm, making it suitable for signage robots or even light shows using UGVs.

Chapter 7

Future Scope

SwarmBot can be extended in several ways to handle more complex and large-scale tasks:

Incorporating Basic Obstacle Avoidance
The current system assumes relatively open spaces or static boundaries. Future work can integrate basic IR or ultrasonic sensors for real-time obstacle detection and avoidance.

Improving Position Accuracy
Adding wheel encoders or gyroscopic sensors can improve dead reckoning accuracy. Alternatively, simple camera-based visual odometry can enhance localization.

Dynamic Mapping and Updating
The system currently performs static mapping before task assignment. Future upgrades can allow dynamic re-mapping, obstacle updates, and real-time path re-planning.

Scaling Beyond Three UGVs
The system can be tested and extended to handle larger swarms, possibly using a multi-layer communication hierarchy to reduce congestion.

Integrating More Complex Path Planning Algorithms
Currently, simple straight-line and point-to-point pathing are used. Advanced algorithms like A* or Dijkstra's algorithm can further optimize navigation, especially in more cluttered spaces.

Exploring Outdoor Capabilities
While designed for indoor use, ruggedizing the system with better motors, weather-resistant bodies, and solar power could enable outdoor operation.

Chapter 8

Conclusion

SwarmBot demonstrates that complex multi-robot coordination does not necessarily require GPS, LIDAR, or cloud-based servers. By leveraging RPM-based dead reckoning, peer-to-peer ESP-NOW communication, and a self-hosted web dashboard, we successfully created a scalable, low-cost, fully offline swarm system.

Through collaborative mapping, dynamic task assignment, and autonomous path execution, SwarmBot provides a proof of concept for resource-efficient swarm robotics. Its real-time visualization, click-based control, and lightweight hardware requirements open up new possibilities for deploying autonomous fleets in environments where GPS or Wi-Fi infrastructure is unavailable.

The project also serves as an excellent educational platform for understanding embedded systems, web interfacing, and decentralized control while laying the foundation for future expansion into more advanced swarm behaviors.

Chapter 9

References

1. Brambilla, M., Ferrante, E., Birattari, M., & Dorigo, M. (2013). Swarm robotics: A review from the swarm engineering perspective. *Swarm Intelligence, 7*(1), 1–41.

2. Kalra, N., Ferguson, D., & Stentz, A. (2005). Hoplite: A Markov Decision Process-Based Multirobot Coordination System. *International Journal of Robotics Research, 24*(10), 971–1000.

3. Espressif Systems. ESP-NOW User Guide. [Online].

4. Arduino. HC-SR04 Ultrasonic Distance Sensor Tutorial. [Online].

5. Kumar, M., & Michael, N. (2008). Opportunistic Task Allocation in Multi-Robot Systems. *Robotics: Science and Systems IV*.

6. Random Nerd Tutorials. ESP32 Web Server using SPIFFS (Host HTML Files). [Online].

7. Gadepalli, V. (2023). Future Wireless Communication: Embedded Systems and Robotics Tutorials. [GitHub Repository].