IARJSET ISSN (O) 2393-8021, ISSN (P) 2394-1588

International Advanced Research Journal in Science, Engineering and Technology

Impact Factor 8.311 ⁙ Peer-reviewed & Refereed journal ⁙ Vol. 12, Issue 12, December 2025

DOI: 10.17148/IARJSET.2025.121237

Industrial Automated Line Follower Robot

Bhavyashree H D[1], Akash B[2], Charan N[3], Charan V[4], Rajath S[5]

Assistant Professor, Department of Information Science and Engineering, Maharaja Institute of Technology, Mysore, Karnataka, India[1]

Undergraduate student, Computer Science and Engineering, Maharaja Institute of Technology, Mysore, Karnataka, India[2,3,4,5]

Abstract: Industrial material movement in small and medium scale industries in India commonly depends on either low-cost Automated Guided Vehicles (AGVs) that follow fixed paths, or expensive Autonomous Mobile Robots (AMRs) with unnecessary sensing and processing overhead. AGVs lack path adaptability and halt when path conditions change, while AMRs are prohibitively costly for widespread deployment. This paper presents BHEEMA, a semi-autonomous navigation robot positioned between AGVs and AMRs. BHEEMA uses a modified region-focused Dijkstra algorithm running on a low-cost ESP32, along with a 5-sensor IR junction detection array for local navigation and Wi-Fi based coordination through a local server. The robot intelligently selects a subsection of the full map and performs local shortest-path planning, reducing computation while allowing dynamic routing. The system supports multi-robot coordination, collision prediction, and deadlock reduction inspired by the Braess Paradox. The design is highly cost-effective, scalable and suitable for Indian warehouse and cottage industry environments.

Keywords: AGV, AMR, Path Planning, ESP32, Dijkstra Algorithm, Swarm Robotics, Industrial Automation

I. INTRODUCTION

A. Problem Statement

Traditional line-following robots, commonly used in small and medium-scale industries, textile units, pharmaceutical packing floors, and electronics assembly warehouses, are typically limited to following fixed predefined tracks marked by colored lines or reflective tape. These systems lack the ability to understand the dynamic structure of the warehouse, and therefore cannot make intelligent route adjustments when encountering obstacles or path blockages.
Similarly, commercially available Automated Guided Vehicles (AGVs) provide predictable navigation but offer little flexibility, while Autonomous Mobile Robots (AMRs), though fully autonomous, require costly sensors (e.g., LiDAR, depth cameras), SLAM computation, and high-end processors. These systems are therefore economically impractical for smaller industries where layout changes are rare and operational budgets are constrained. Because of this disparity, industries are forced to choose between: Low-cost but unintelligent line-followers that require manual intervention, or Highly capable but financially prohibitive AMRs. This leads to inefficiencies such as frequent halts, need for human supervision, delays in material movement, and an overall reduction in productivity. As a result, there exists a clear need for a cost-effective, intelligent, adaptive navigation robot that can operate efficiently in structured but dynamic industrial environments.

B. Proposed Solution

This paper presents BHEEMA, a semi-autonomous industrial navigation robot designed as a balanced alternative between AGVs and AMRs. BHEEMA is built on: A modified Dijkstra's path planning algorithm executed on a low-power ESP32 microcontroller. A 5-sensor IR junction-detection array for precise local navigation and node classification. Ultrasonic obstacle detection for safety-based stopping behavior. TCP/IP Wi-Fi communication with a local server to enable real-time multi-robot path coordination and collision prediction. Unlike standard Dijkstra implementations, the proposed region-focused modification limits computation to the local area of interest within the warehouse map, significantly reducing processing time and memory usage, making it suitable for microcontroller-level hardware. BHEEMA is therefore capable of: Real-time shortest-path planning within a warehouse network. Dynamic rerouting based on predicted robot traffic .Operating as part of a cooperative swarm, preventing deadlocks through server-based trajectory prediction and application of Braess Paradox principles for congestion reduction.

C. Contributions:

- Modified Dijkstra Path Planning on ESP32:
  A region-restricted graph search method enabling real-time navigation on lightweight hardware.

- Multi-Sensor Navigation Hardware Architecture:
  Integration of a 5-point IR array and ultrasonic proximity sensor on a minimal, maintainable chassis for reliable node-based localization.
- Multi-Robot Cooperative Navigation Protocol:
  TCP-based communication with a centralized coordination server for intelligent routing, traffic prediction, and deadlock avoidance without using SLAM or GPS.

Paper Structure:
"The remainder of this paper is organized as follows: Section II details the Literature Survey. Section III describes the System Architecture including implementation of the navigation. Section IV presents the experimental results, and Section V concludes the paper with a discussion on future work."

II. LITERATURE SURVEY

Modern industrial robotics is evolving with multi-robot coordination, embedded intelligence, and advanced path planning. Systems employ leader-follower protocols and MQTT-based schedulers. Planning strategies range from classical (Dijkstra's, A*) to adaptive (PRM, RRT*). The ESP32 is widely adopted for its integrated Wi-Fi/Bluetooth and performance. Sensor input relies on a mix of IR, ultrasonic, and vision, often combined via sensor fusion. Control loops, PID controllers, and predictive models are common. Simulation environments like Gazebo and MATLAB are used for virtual prototyping.
However, notable limitations persist. IR sensors fail under variable lighting, and ultrasonic sensors can misinterpret angled reflections. Centralized control systems suffer from communication lags. Many studies lack practical validation in real industrial conditions. Most robots still depend on predefined paths and lack dynamic re-routing. Battery-efficient path planning and the integration of AI on low-cost platforms remain underexplored, highlighting the need for scalable, intelligent, and energy-efficient multi-robot systems.
[1] R. Phani Vidyadhar et al. (2024): This work on an IoT-based line follower using an ESP32 and obstacle detection validates the choice of the ESP32 as a capable core for integrating IoT features, sensor processing (ultrasonic), and control logic in a single, low-power package. [2] A. Mahant et al. (2024): Their design for an autonomous line follower with obstacle avoidance, though using an Arduino, highlights the fundamental challenges of PID control for sharp turns and sensor accuracy. This informs our decision to use a more powerful microcontroller (ESP32) and a robust sensor array. [3] A. Sharma and S. G. Kulhalli (2023): This paper is highly relevant, as it directly addresses warehouse automation using an ESP-32 and Dijkstra's algorithm. It provides a direct precedent for our system, which we build upon by adding multi-robot communication and a more robust mechanical design. [4] N. Saqib and M. M. Yousuf (2021): Their use of decision-making algorithms (e.g., Wall Follower) for a shortest-path rover, while simpler, demonstrates the feasibility of implementing path-finding logic on low-cost microcontrollers. Our work scales this concept up to the more complex and efficient Dijkstra's algorithm. [5] H. Kareem and D. Dunaev (2021): This comparative analysis provides a strong justification for selecting the ESP32 over other modules like Arduino, citing its superior processing power, integrated Wi-Fi/Bluetooth, and overall value for complex, connected IoT applications. [6] K. A. Raza and W. Monnet (2019): This research on using an HC-SR04 ultrasonic array for object detection and direction-finding informs our multi-sensor approach to obstacle avoidance, confirming the sensor's capability for reliable detection in a cost-effective manner. [7] S. E. A. Askar et al. (2024): Their evaluation of dynamic task allocation strategies in multi-agent systems highlights the trade-offs between "greedy" algorithms (faster but unbalanced) and probability-based approaches. This informs our project's need for a balanced central server logic to manage job distribution efficiently among multiple robots. [8] A. M. J. Skulimowski and W. Szulc (2024): This work on modeling robot interactions within IIoT environments supports our architectural choice of integrating robotic agents with a broader digital infrastructure. It emphasizes the utility of digital twins and high-level interaction models for optimizing robot-cloud communication. [9] H. Marah et al. (2023): By modeling a multi-robot warehouse as a Cyber-Physical System with centralized control, this paper validates our design choice of using a central coordination server to manage trajectories and robot coordination. It underscores the importance of defining clear operational roles and control strategies in warehouse automation. [10] H. Zeghida et al. (2023): This study on securing MQTT protocols using machine learning addresses the critical aspect of cybersecurity in industrial IoT. It highlights the necessity of robust, secure communication channels for reliable fleet management, relevant to our Wi-Fi-based client-server architecture.

III. SYSTEM DESIGN AND ARCHITECTURE

The proposed system is a modular, low-cost robotic platform centered on the ESP32 microcontroller, selected for its integrated Wi-Fi capabilities and computational power.

The high-level design as shown in *Figure 1* uses a client-server architecture where a Central Coordination Server manages high-level job scheduling and swarm traffic control. The "smart" Robot (client) receives a job (e.g., "go to node X") and then uses its onboard ESP32 to calculate its own path by running a modified Dijkstra's algorithm. The robot executes this path using a 5-point IR array for line navigation and a separate range sensor for local obstacle avoidance.

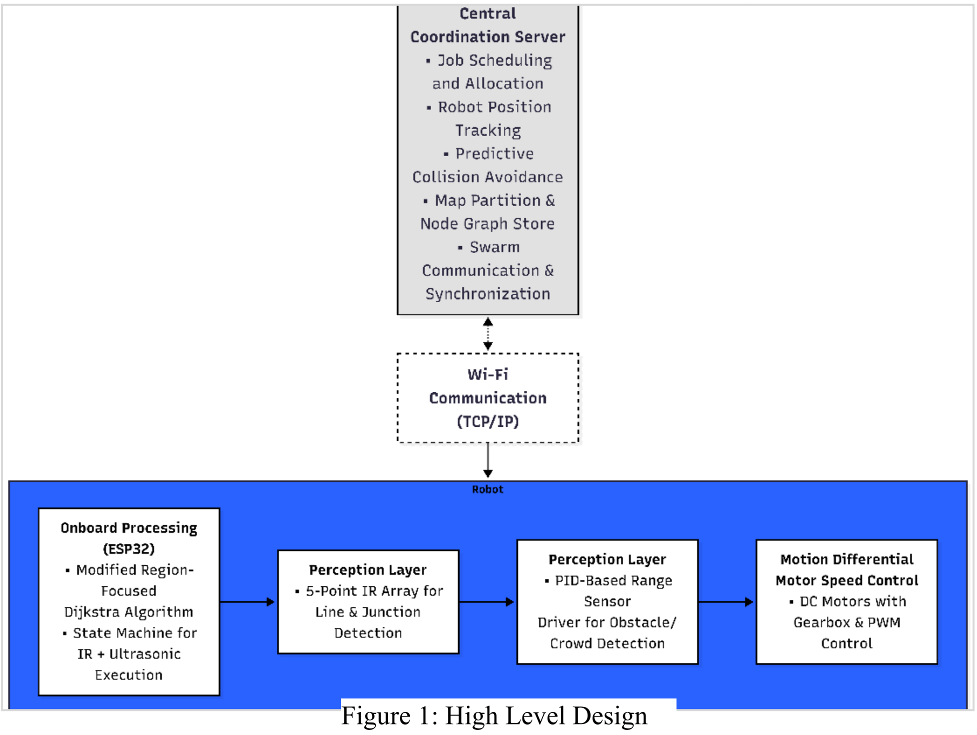
Central Coordination Server
• Job Scheduling and Allocation
• Robot Position Tracking
• Predictive Collision Avoidance
• Map Partition & Node Graph Store
• Swarm Communication & Synchronization
Wi-Fi Communication (TCP/IP)
Robot
Onboard Processing (ESP32)
• Modified Region-Focused Dijkstra Algorithm
• State Machine for IR + Ultrasonic Execution
Perception Layer
• 5-Point IR Array for Line & Junction Detection
Perception Layer
• PID-Based Range Sensor Driver for Obstacle/ Crowd Detection
Motion Differential Motor Speed Control
• DC Motors with Gearbox & PWM Control

Figure 1: High Level Design

A. Hardware Design

This Design includes DC gear motors for locomotion, controlled by an L298N/L293D motor driver, and powered by a central Li-ion battery pack. The sensor suite is designed for efficient, line-based navigation without expensive sensors like GPS or SLAM; it employs a 5-sensor IR array for precise line tracking and junction detection, supplemented by an ultrasonic sensor for real-time safety obstacle detection. The software, developed in Embedded C or MicroPython, implements a PID-based controller for smooth motor control and a Region-Focused Dijkstra's algorithm for dynamic path planning.
This architecture supports multi-robot coordination through socket-based communication and a server API, preventing collisions and deadlocks. This design is inherently scalable, more adaptive than traditional AGVs, and significantly cheaper than AMRs, making it an effective solution for real-world warehouse layouts.

B. Sensor Suite

1) 5-Sensor IR Junction Detection Array

The robot uses a custom 5-sensor IR reflective array configured to detect straight line segments, 90° left or right turns, T-junctions and cross-junctions, and endpoint or dead-end nodes.

Table 1: Index of sensors and purpose

Sensor Index	**Position**	**Purpose**
BL	Back-Left	Detects left-turn availability / exit confirmation
FL	Front-Left	Detects branch availability while moving forward
FS	Front-Center	Maintains line alignment
FR	Front-Right	Detects branch availability on right
BR	Back-Right	Detects right-turn exit or cross-node signature

DOI: 10.17148/IARJSET.2025.121237

C. Software Architecture

The system's software architecture is built upon the ESP32 platform, utilizing Embedded C for performance-critical tasks and MicroPython for high-level application logic. At the foundational level, a PID-based motor control algorithm is implemented to ensure precise line-following and velocity regulation using sensor feedback. The core navigation intelligence is managed by a Region-Focused Dijkstra's Path Planning Logic, enabling the robot to compute the optimal path from a source to a destination node within the warehouse's graph model.
For multi-robot collaboration, the system employs socket-based communication to interface with a central Server API. This coordination layer is critical for deconflicting routes, managing traffic at intersections, and preventing deadlocks in a dynamic environment.

D. Localization and Node Recognition

Localization does not use GPS or camera-based methods. Instead, navigation is discretized into nodes as shown in *Figure 2*, where the robot:

- Tracks its current node ID
- Updates its orientation (N, E, S, W)
- Reads the junction signature from the IR pattern to confirm the node shape

This provides low-cost, deterministic localization well-suited for warehouse layouts that rarely change.

E. Path Planning: Modified Region-Focused Dijkstra Algorithm

Traditional Dijkstra's algorithm, which computes shortest paths on the entire map – Areana *Figure 2*, is inefficient for microcontrollers. To address this, a region-limited path search is used:

- Identify the start node and destination node.
- Extract a local subgraph limited to the area within *R* hops from the starting node and the area within *R* hops from the destination.
- Merge both regions and retain only bridging edges.
- Run Dijkstra's algorithm on this reduced graph.

This modification significantly reduces time complexity, RAM usage, and CPU load, allowing real-time shortest path planning on an ESP32 without external computation.

F. Multi-Robot Coordination (Swarm Layer)

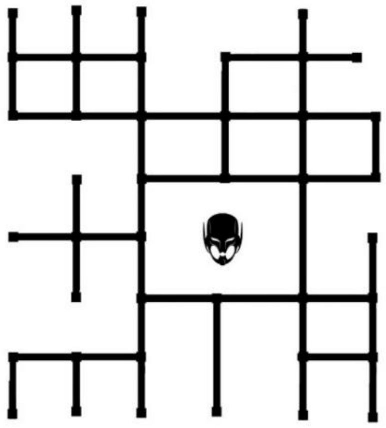

Figure 2: Areana

Coordinated fleets are supported through a Wi-Fi connected central server which:

- Tracks each robot's current edge and node
- Predicts path conflicts before they occur
- Suggests reroutes or timed waiting to avoid head-on collisions, congestion loops, and deadlocks at junctions.

The server employs principles inspired by the Braess Paradox, where closing certain "shortcut" paths can increase total fleet efficiency. *Figure 3* Shows the Line follower robots which are under the Swarm layer.

IARJSET ISSN (O) 2393-8021, ISSN (P) 2394-1588
International Advanced Research Journal in Science, Engineering and Technology
Impact Factor 8.311 ⁜ Peer-reviewed & Refereed journal ⁜ Vol. 12, Issue 12, December 2025
DOI: 10.17148/IARJSET.2025.121237

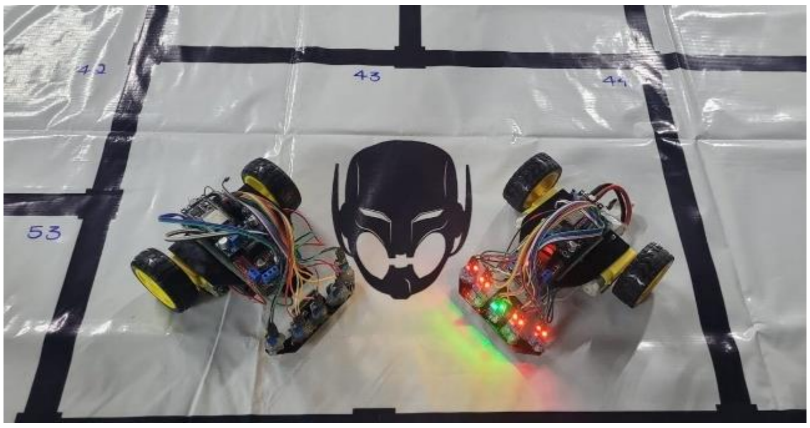

Figure 3: Line Follower Robots

G. Motion Control and Actutation

- Motors: Two DC gear motors are configured for differential drive.
- Control: PWM-based speed modulation is used.
- Line Tracking: Proportional correction (motor speed biasing) is applied.
- Turn Execution: Time-based actuation is combined with IR confirmation events.

This ensures smooth navigation and repeatable maneuvering at nodes.

IV. EXPERIMENTATION / TESTING

The *Table 2* shows the necessary Experimentation which made sure to add the details of the proposed system for industrial navigation

Table 2: Experimentation / Testing

Experimentation /Testing	Objective	Methodology	Result
1	**Motion Layer Testing** Verify 150 RPM motors, drivers, and encoder functionality.	Command motors forward/backward at different speeds; spin manually to check encoder data.	Motors spin smoothly in the correct direction; encoder values change relative to speed.
2	**Perception Layer Testing** Validate accuracy of the 5-channel IR array and HC-SR04 ultrasonic sensors.	Print IR binary values over lines/junctions; measure objects at fixed distances (10-100cm).	IR outputs reliable binary (e.g., '01110'); Ultrasonic reports distance within a small margin of error.
3	**Pathfinding Module** Verify the Modified Dijkstra's algorithm calculates the shortest path correctly on the ESP32.	Call pathfinding function with a defined graph and print the command list.	Algorithm consistently returns the mathematically shortest path and correct command list.
4	**Line Following** Test integration of IR sensors, logic, and motors for smooth navigation.	Place robot on a track with curves and straight lines; activate line-following state.	Robot follows the line smoothly without heavy oscillation or derailing.
5	**Node Handling** Test the ability to detect junctions and execute turns.	Input a path (e.g., "Turn Right"); place robot on a track leading to a T-junction.	Robot follows straight, stops at the junction ('11111'), and executes the correct turn.
6	**Full Autonomous Mission** Validate job reception, path calculation, and navigation to destination.	Send "Go to Node G" from the server to the robot positioned at "Home".	Robot receives job, calculates path, and successfully navigates to "Node G".

7	**Dynamic Re-routing** Validate detection of obstacles and calculation of detours.	Place an unexpected obstacle on the line while the robot is navigating a long distance.	Robot detects object, stops, recalculates a new path, and takes a detour to the destination.
8	**Battery Life Test** Validate the claim of 3-4 working hours of uptime.	Run the robot on a continuous loop with a fully charged 2200mAh LiPo battery.	Robot operates for a minimum of 3 hours before voltage drop.
9	**Economic Constraints** Validate that the Bill of Materials (BOM) is under 10,000 Rs.	Sum the cost of the final parts list (ESP32, motors, chassis, etc.).	Total sum of all components is less than 10,000 Rs.

V. FUTURE IMPLEMENTATIONS

Future work will focus on enhancing the system with motor encoders to improve distance accuracy and reduce drift during long runs. The IR junction detection system can be upgraded to infrared line cameras for finer path correction on uneven surfaces. Integration of battery monitoring and auto-charging docks will allow continuous operation without manual intervention. Additionally, extending the coordination server to a cloud-based fleet management dashboard will enable tracking and control of larger robot groups across multiple warehouse zones. Finally, lightweight vision markers or QR-based node identifiers may be added to improve localization accuracy in environments with high floor reflectivity.

VI. CONCLUSION

This project successfully developed a cost-effective, industrial-grade automated robot that bridges the gap between basic line followers and expensive autonomous mobile robots (AMRs) for Small to Medium-sized Enterprises. By utilizing an ESP32 microcontroller with a modified Dijkstra's algorithm, the system achieves intelligent, memory-based localization and dynamic path planning without the need for costly LiDAR sensors. The implementation of a custom honeycomb chassis and a four-wheel differential drivetrain ensures structural stability and reliable traction for industrial load carrying while keeping the total cost under 4,500 Rs. A key achievement was validating the robot's ability to autonomously detect obstacles and calculate real-time detours, a significant upgrade over traditional manufacturing robots that simply stop at blockages. Future iterations aim to enhance this platform with computer vision for QR code navigation and mesh networking to create a fully decentralized, self-coordinating warehouse

ACKNOWLEDGMENT

The authors express sincere gratitude to **Dr. Sharath Kumar Y H** – Professor and Head, **Prof. Bhavyashree H D**, Assistant Professor, Department Information Science and Engineering, Maharaja Institute of Technology, Mysore, for in valuable guidance in system architecture design and project mentorship. We acknowledge the support of the institution in providing resources and infrastructure for development and testing.

REFERENCES

[1]. R. Phani Vidyadhar et al., "An IoT-Based Obstacle Detection and Line Follower Robot using LabVIEW," 2024.
[2]. A. Mahant, D. Kashyap, and K. Boora, "Autonomous Line Follower Robot with Obstacle Avoidance Design," 2024.
[3]. I. Opara and M. O. Onibonoje, "Development of a Multi-Robot Model for Target Identification and Tracking in Hostile Environments Using Particle Swarm Optimization Technique," in *IEEE NIGERCON*, 2024.
[4]. S. Jadhav et al., "Comparison of Path Planning Algorithms for Indoor Static Environment," in *IEEE Pune Section International Conference (PuneCon)*, Dec. 13-15, 2024.
[5]. S. Shirmohammadi and F. Baghbani, "Design and Implementation of a Line Follower Robot," in 10th Int. Conf. on Artificial Intelligence and Robotics (QICAR), Feb. 29, 2024.
[6]. S. Wei, J. Chen, M. Zhang, Y. Gan, H. Chen, and D. Huang, "Path-Follower-2: A Narrow Environment Navigation System for Large Mobile Inspection Robot," in *IEEE 13th DDCLS Conf.*, pp. 139, 2024.
[7]. S. E. A. Askar, S. F. S. Eid, and Y. I. Elshaaer, "Dynamic Pickup and Delivery Task Allocation in Multi-Agent Robotic Systems: Vacancy Chain Approaches Evaluation," in Proc. 5th Int. Conf. on Electrical, Communication and Computer Engineering (ICECCE), Kuala Lumpur, Malaysia, Oct. 2024.
[8]. A. M. J. Skulimowski and W. Szulc, "Modelling Autonomous Mobile Robot Interaction with IoT Environments," in Proc. 10th Int. Conf. on Control, Decision and Information Technologies (CoDIT), Valletta, Malta, Jul. 2024.

[9]. H. Marah, R. Paredis, M. Challenger, and H. Vangheluwe, "A Multi-Robot Warehouse System: An Exemplar," in *2023 ACM/IEEE International Conference on Model Driven Engineering Languages and Systems Companion (MODELS-C)*, 2023.

[10]. H. Zeghida, M. Boulaiche, and R. Chikh, "Securing MQTT Protocol for IoT Environment Using IDS Based on Ensemble Learning," *Int. J. of Information Security*, Springer, vol. 22, pp. 1075-1086, 2023.

BIOGRAPHY

Bhavyashree H D is an Assistant Professor in the Department of Information Science and Engineering at Maharaja Institute of Technology, Mysore, Karnataka, India. Her academic and research interests include Image Processing, Data Structure and Algorithms.

Akash B is an undergraduate student in Computer Science and Engineering at Maharaja Institute of Technology, Mysore, Karnataka, India. His areas of interest include IoT, Artificial Intelligence, and Full stack development.

Charan N is an undergraduate student in Computer Science and Engineering at Maharaja Institute of Technology, Mysore, Karnataka, India. His interests include Data Science, Machine Learning.

Charan V is an undergraduate student in Computer Science and Engineering at Maharaja Institute of Technology, Mysore, Karnataka, India. His academic interests include software development, and Full Stack development.

Rajath S is an undergraduate student in Computer Science and Engineering at Maharaja Institute of Technology, Mysore, Karnataka, India. His interests include Full Stack development, system design, IoT and AI/ML.