INTELLIGENT SYSTEMS AND SUSTAINABLE ENERGY
E-ISSN: 3154-7532
VOLUME 1, ISSUE 1, 2026, pages 43 – 52
DOI: https://doi.org/10.15282/isse.1.1.14255

RESEARCH ARTICLE

Development and experimental evaluation of a low-cost ROS2-enabled autonomous carrier robot for structured factory automation

Izz Haziq Ahmat Nasir[1], Amran Abdul Hadi[1,*], Raja Mohd Taufika Raja Ismail[1], Md Rizal Othman[1], Mohamad Rahimi Mohamed Rodzi[2]

[1]Faculty of Electrical and Electronics Engineering Technology, Universiti Malaysia Pahang Al-Sultan Abdullah, 26600 Pekan, Pahang, Malaysia
[2]Academy for Advanced TVET, Universiti Malaysia Pahang Al-Sultan Abdullah, 26600 Pekan, Pahang, Malaysia

Abstract - In this paper a low-cost autonomous carrier robot is developed for structured factory transportation tasks. The system is based on Robot Operating System 2 (ROS2) running on a Raspberry Pi 4. It works together with two distributed ESP32 microcontrollers. Obstacle detection, line-following navigation and platform identification tasks are performed using ultrasonic sensors, infrared sensors and radio frequency identification (RFID) modules. The communication between the carrier robots and the central control node is implemented using ROS2 and Message Queuing Telemetry Transport (MQTT) . Experiments were conducted in an indoor factory-like environment. Navigation performance, obstacle avoidance, RFID detection, and communication behaviour were evaluated. Navigation tests were performed using different track widths and movement conditions. A navigation success rate of up to 90% was recorded during repeated trials. Two obstacle avoidance methods were evaluated. The reverse-and-forward method completed all test trials successfully. RFID platform identification was tested using different line spacing configurations, and a 100% detection rate was obtained at the selected operating condition. Communication tests showed low message delay during task coordination between carrier robots and the central control node. The developed system combines ROS2 communication, MQTT messaging, distributed ESP32 processing, and RFID-assisted navigation within a single carrier platform. The robot was evaluated under structured operating conditions using low-cost hardware components commonly available for embedded robotic applications.

Article history
Received : 11 March 2026
Revised : 18 May 2026
Accepted : 23 May 2026
Published : 6 June 2026

Keywords
Smart product carrier
Factory automation
Robot operating system
Autonomous mobile robot
Industry 4.0

1. Introduction

Material transportation is a common task in manufacturing facilities [1],[2],[3]. Components, semi-finished products, and finished products must be moved between workstations, storage areas, and production lines throughout the manufacturing process. Fixed conveyor systems and manual transportation methods are still widely used for this purpose [4],[5],[6]. These approaches are suitable for repetitive operations but may require layout modification when production requirements change. In reconfigurable manufacturing environments, transport systems with greater flexibility are often required.

Automated Guided Vehicles (AGVs) and Autonomous Mobile Robots (AMRs) have been used for material handling and intralogistics applications in factories and warehouses [7],[8],[9]. These systems can transport materials between locations with minimal operator involvement. Today's mobile robot platforms typically integrate navigation, obstacle detection, task scheduling, and wireless communication [10],[11],12]. The many commercial systems employ LiDAR sensors, high-performance computing hardware, and proprietary software frameworks. These technologies offer enhanced navigation capabilities but also increase system cost and complexity of implementation. ROS2 (Robot Operating System 2) is a software framework for distributed robotics applications [13],[14],[15]. The framework supports communication between software nodes and allows different sensing, control, and monitoring functions to be integrated within a common platform. ROS2 has been adopted in industrial and research applications involving autonomous robots, multi-robot systems, and embedded robotic devices [16],[17],[18]. Low-cost computing platforms such as Raspberry Pi and ESP32 microcontrollers have also been used with ROS2 for mobile robot development [19], [20].

In structured factory environments, navigation routes are often predefined. Under these conditions, infrared-based line-following remains a practical navigation method because it requires only simple sensing hardware and limited processing resources. Additional sensing devices may be incorporated to support obstacle detection and station identification. Wireless communication is also required when multiple robots and supervisory systems exchange task information. MQTT has been used for lightweight machine-to-machine communication and data exchange in distributed automation systems [21], [22]. This work presents a low-cost autonomous carrier robot developed for structured factory transportation tasks. The system uses a Raspberry Pi 4 together with two ESP32 microcontrollers. Infrared sensors are used for line-following navigation, ultrasonic sensors are used for obstacle detection, and RFID tags are used for station identification. Communication between carrier robots and the central control node is implemented using ROS2 and

***CORRESPONDING AUTHOR | Amran Abdul Hadi** | ✉ amranhadi@umpsa.edu.my

MQTT. Experimental work was carried out in an indoor factory-like environment to evaluate navigation behaviour, obstacle avoidance, RFID detection, and communication performance. The main contributions of this paper are summarized as follows:

i. Low-cost autonomous carrier robot design with Raspberry Pi and ESP32 hardware.
ii. Integration of ROS2 and MQTT communication for coordination of carrier robot.
iii. Line following navigation and obstacle avoidance based on RFID station identification.
iv. Experimental validation of navigation performance, obstacle avoidance behaviour, RFID detection and communication latency in structured operating conditions.

2. Materials and Methods

2.1. System Architecture and Hardware Configuration

The autonomous carrier system consists of one Main Robot Carrier and two sub-carrier robots, Robot Bravo and Robot Tango. Figure 1 shows the overall architecture of the system. The Main Robot Carrier coordinates tasks and Robot Bravo and Tango perform transport tasks within the factory environment.

2.1.1. Main Robot Carrier

The Main Robot Carrier is based on a Raspberry Pi 4 running ROS2 Humble. The Raspberry Pi takes care of the task management and the communication to the ROS2 Publish/Subscribe nodes. A Station Selector Switch is provided for manual selection of the desired destination. The Raspberry Pi is interfaced with an ESP32 microcontroller for local device control. The ESP32 communicates with a load cell to measure weight and an L298N motor driver to control two DC motors. The Main Robot Carrier is the central node for coordination in the carrier system.

2.1.2. Robot Bravo and Tango

Robot Bravo and Robot Tango use two ESP32 microcontrollers. The first ESP32 is responsible for navigation and motion control. This controller interfaces with an L298N motor driver, a servo motor, ultrasonic sensors, and infrared sensors. The ultrasonic sensors are used for obstacle detection, while the infrared sensors are used for line-following navigation. The second ESP32 is used for platform identification. An MFRC522 RFID reader is connected to this controller for reading RFID tags placed at factory stations. The separation of navigation and identification functions allows both processes to operate independently during robot movement.

2.1.3. Communication Architecture

Communication between the Main Robot Carrier, Robot Bravo, and Robot Tango is performed using MQTT over a Wi-Fi network. The Raspberry Pi operates as the central coordination node, while the carrier robots exchange task information through MQTT messages. When a robot finishes a task or reaches a station, a status message is sent to the MQTT broker. The updated information is published on the corresponding topic, and other robots subscribed to the topic receive this information. This mode of communication is used for synchronization of tasks and monitoring of status during operation. The distributed architecture separates task management, navigation, motor control, and RFID identification in separate processing units. The main hardware components used in the system are shown in Table 1.

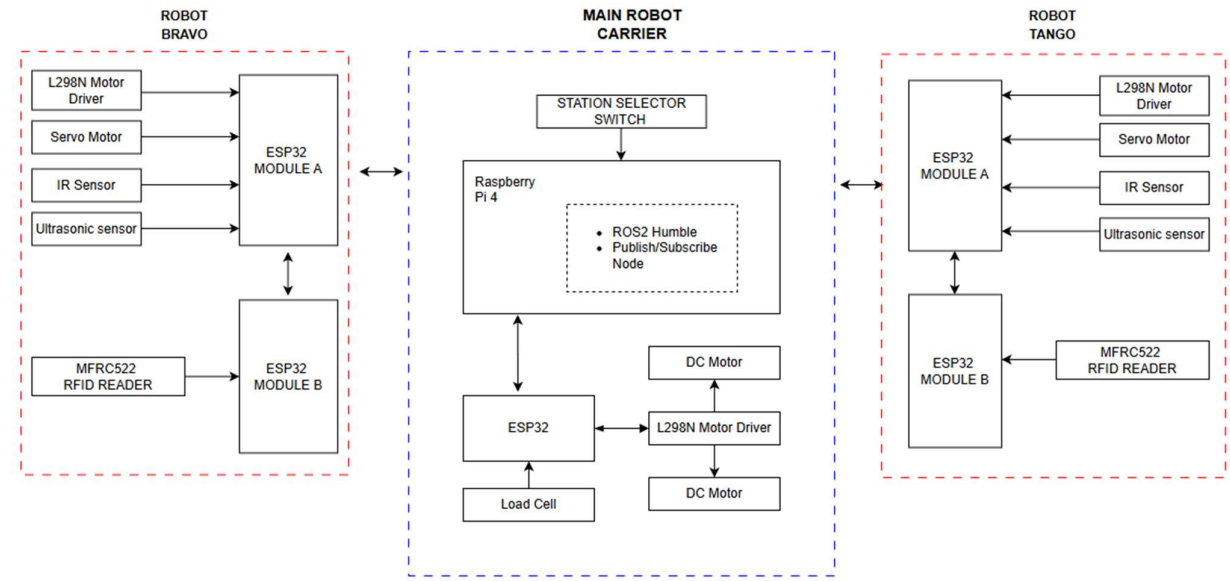

Figure 1. System Block Diagram

Table 1. Main hardware components of the smart product carrier

Component	Model / Type	Function
Central controller	Raspberry Pi	Task coordination, ROS2 node execution
Microcontroller	ESP32	Sensor processing, motor control, communication
Ultrasonic sensor	HC-SR04	Obstacle detection
Infrared sensor	Line sensor array	Path tracking
RFID module	MFRC522	Platform identification
Communication	Wi-Fi (MQTT)	Data exchange between nodes

2.2. Software Framework and Communication Architecture

The carrier system is based on a Raspberry Pi 4 with ROS2 Humble and communicates over Wi-Fi via MQTT. The ROS2 is responsible for the task management, data processing, and coordination functions, while the MQTT is used for message exchange between the carrier robots and the Main Robot Carrier.

2.2.1. ROS2 Node Implementation

The Raspberry Pi 4 runs the main ROS2 nodes of the system. These nodes are responsible for task allocation, station selection, and data logging. The Station Selector Switch sends destination input to the ROS2 application. The current task status and carrier information are also kept while the Raspberry Pi is running. Control of local devices is via ESP32 controllers. These controllers handle sensor input, RFID data acquisition, motor control, and hardware events. The gathered information is sent by the ESP32 modules to the Raspberry Pi when the information is needed.

2.2.2. MQTT-Based Inter-Robot Communication

MQTT communication is used between the Main Robot Carrier, Robot Bravo, and Robot Tango. The protocol operates over a Wi-Fi network and is used to exchange task information and robot status messages. MQTT QoS Level 1 was selected for message delivery during the experiments. The Raspberry Pi publishes task commands to MQTT topics assigned to each carrier robot. Robot Bravo and Robot Tango subscribe to the corresponding topics and execute the received commands. For example, a task command may be transmitted through a topic such as /carrier/bravo/task. Status information is also exchanged through MQTT topics. When a carrier robot completes a task, a status message is published. Other robots subscribed to the same topic receive the updated information and update their operating status accordingly.

2.2.3. Data Flow and Synchronization

Task commands generated by the Raspberry Pi are transmitted through MQTT topics and received by the carrier robots. Sensor information, RFID identification data, obstacle detection events, and task status messages are transmitted in the opposite direction. During operation, each carrier robot periodically publishes its current status. These messages are used by the Main Robot Carrier to monitor task execution and carrier availability. The same communication mechanism is used for platform identification updates and obstacle notifications.

2.3. Navigation and Obstacle Avoidance Strategy

Navigation is performed using infrared line-following sensors connected to ESP32 Module A. Obstacle detection is performed using ultrasonic sensors mounted on the front of the carrier. RFID-based station identification is handled by ESP32 Module B.

2.3.1. Navigation Logic (ESP32 Module A)

ESP32 Module A controls the movement of the carrier robot. Infrared sensors are used to detect the track position. Sensor readings are processed by the ESP32 controller and used to adjust the PWM signals supplied to the L298N motor driver. The navigation path consists of predefined tracks placed on the factory floor. During operation, the carrier follows the track using feedback from the infrared sensor array. Two track widths, 1.8 cm and 3.6 cm, were used during the experiments.

2.3.2. Obstacle Detection and Avoidance (ESP32 Module A)

The Obstacle detection is performed using ultrasonic sensors connected directly to ESP32 Module A. The sensors continuously monitor the area in front of the carrier. Figure 2 shows the circuit diagram of Module A. Two obstacle avoidance methods were implemented and evaluated. The performance of both methods was evaluated experimentally.
Navigate-Around Approach – The carrier attempts to move around the obstacle and return to the navigation path.
Reverse-and-Forward Approach – The carrier reverses for a short distance after obstacle detection. The carrier then moves forward after the obstacle is removed.

2.3.3. Identification-Based Navigation (ESP32 Module B)

ESP32 Module B is used for RFID-based station identification. Figure 3 shows the hardware configuration of Module B. An MFRC522 RFID reader is connected to the ESP32 controller. When the carrier arrives at a station, the RFID tag is read, and the station identifier is obtained. The identification result is transmitted to the Raspberry Pi through MQTT communication. The same information is also available to other carrier robots subscribed to the corresponding MQTT topic. Station identification is used as part of the task execution process. The navigation sequence may continue or stop depending on the task status received from the central coordination node.

2.4. Experimental Setup and Evaluation Procedure

Experiments were performed in an indoor laboratory environment. The test track was a closed-loop path consisting of straight and corner sections. In the experiments, two track widths were used: 1.8 cm and 3.6 cm. For navigation, infrared sensors were mounted under the carrier and were used for line-following. Mounted on the front of the carrier, ultrasonic sensors detected obstacles. The experiments were performed with fixed motor speed settings. Two carrier robots, Robot Bravo and Robot Tango, were used in the system evaluation. These robots were assigned delivery jobs to some platforms. A single platform was chosen as a common target for both robots, to test how the system would respond to simultaneous requests for tasks. They then tested navigation, obstacle avoidance, RFID detection, and communication performance. Each test was repeated ten times under the same setup and operating conditions. A run was considered successful if the robot completed the assigned task without leaving the track, hitting an obstacle, or losing communication. Table 2 summarizes the experimental settings used throughout the study.

2.5. Operational Workflow

Figure 4 shows the operating sequence used by the carrier system. The process starts after the MQTT node and database node are initialized. A delivery task is generated when a platform selection button is pressed.

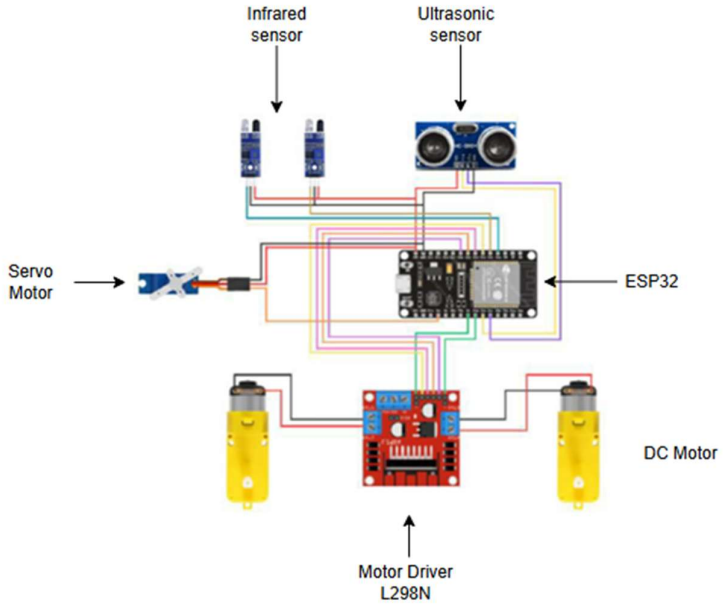

Figure 2. Module A diagram

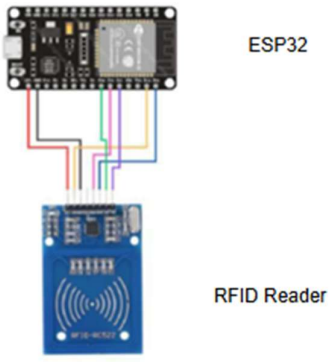

Figure 3. Module B diagram

Table 2. Experimental parameters used for system evaluation

Parameter	Value
Track configuration	Closed-loop with straight and corner segments
Track widths	1.8 cm, 3.6 cm
Obstacle detection method	Ultrasonic sensing
Obstacle detection threshold	Fixed distance (short-range)
Navigation method	Infrared-based line following
Number of smart carriers	2 (Bravo, Tango)
Number of trials per scenario	10
Test environment	Indoor, controlled laboratory setup

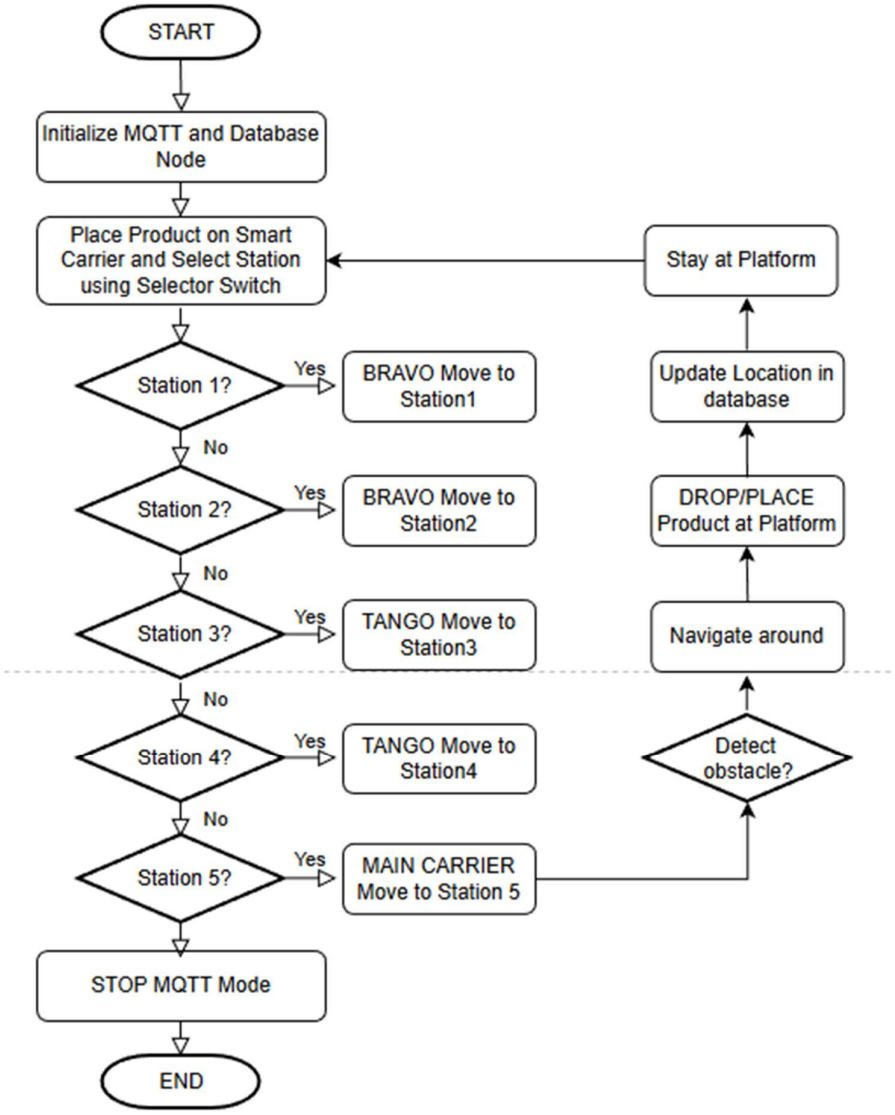

Figure 4. Operation flow chart

2.5.1. Mission Assignment and Navigation

Platform buttons are used to assign destinations to the carrier robots. Platform 1 and Platform 2 are assigned to Robot Bravo. Platform 3 and Platform 4 are assigned to Robot Tango. Platform 5 is assigned as a shared destination for both robots. After a destination is selected, the corresponding robot starts moving along the predefined navigation path. Line-following control is performed by ESP32 Module A using infrared sensor feedback. Motor commands are generated and transmitted to the L298N motor driver. During movement, ultrasonic sensors monitor the path ahead. If an obstacle is detected, the obstacle avoidance routine is executed. The reverse-and-forward method is used during the experiments before navigation resumes.

2.5.2. Arrival Verification and Task Completion

ESP32 Module B reads the RFID tag via the MFRC522 RFID reader when the robot arrives at the destination platform. The location of the platform is identified by an RFID identifier. The RFID information is transmitted through MQTT communication, and it is stored in the database. The task status is then updated. For Platform 5, Robot Bravo and Robot Tango may be assigned the same destination. Once a robot reaches the platform and completes the task, it publishes the status message through MQTT. The second robot gets the status message update and cancels the task it is currently performing. The robot remains in its current position until given a new task. The sequence of operation ends when a stop command is issued. All the carrier robots return to an idle state and wait for the next command of the task.

3. Results and Discussion

3.1. Hardware Prototyping and Assembly

The carrier robot was assembled using the hardware components described in Section 2. Figure 5 shows the arrangement of the main components, while Figure 6 shows the three-dimensional design of the carrier robot. The completed prototype was used for all experiments presented in this study.

3.1.1. Chassis Layout and Component Integration

The carrier robot is based on two-layer circular chassis. Lower layer has motor driver and motor drives and RFID identification system. Station identification is through an attached MFRC522 RFID reader mounted under the chassis. The ESP32 controller, ultrasonic sensor, infrared sensor array and the supporting electronic components are housed in the top layer. On the front of the carrier is mounted an ultrasonic sensor. It is used for obstacle detection. Line following navigation is realized by an infrared sensor array mounted under the chassis. A caster wheel is mounted to a front of the carrier. During movement, the wheel keeps the distance between the sensors and the track surface.

3.1.2. Control Interface Prototyping

Figure 7 shows the Main Robot Carrier control interface. Five push buttons are provided for platform selection. Each button corresponds to a predefined destination platform. Additional Start and Stop buttons are provided for system operation. The buttons are connected directly to the Raspberry Pi and are used during task execution and experimental testing.

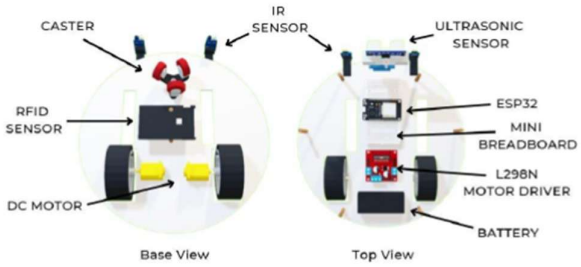

Figure 5. Components arrangement

Figure 6. 3D Design of smart product carrier prototype

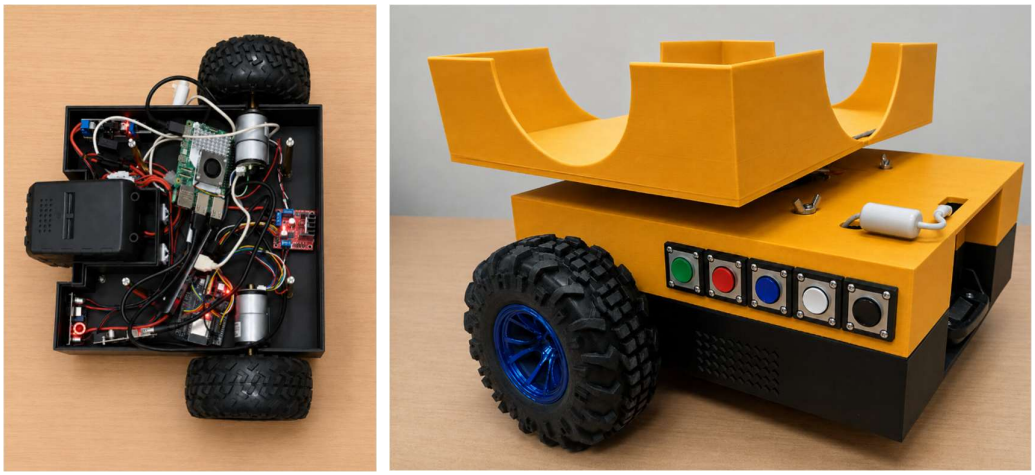

Figure 7. Main Robot Carrier architecture integrating Raspberry Pi 4, ESP32 controller, load cell module, motor driver, and ROS2-MQTT communication framework

3.2. Navigation Performance Evaluation

Navigation experiments were conducted using the track configuration shown in Figure 8. Two track widths, 1.8 cm and 3.6 cm, were evaluated. Straight-path navigation and corner navigation were tested separately. Each experiment was repeated ten times. Table 3 lists the navigation results obtained during the experiments. For straight-path navigation, success rates of up to 90% were recorded. Different results were observed for the two track widths. Corner navigation was also affected by track width. The measured success rates for all test conditions are presented in Table 3.

3.2.1. Obstacle Avoidance Performance

Two obstacle avoidance methods were also evaluated. The first method attempted to move around the obstacle before returning to the navigation path. The second method involved reversing for a short distance and then moving forward after the obstacle was removed. Table 4 lists the obstacle avoidance results. Different success rates were obtained for the two methods. The navigate-around method produced several failures during testing. In these cases, the carrier lost alignment with the navigation track after passing the obstacle. The reverse-and-forward method completed all test trials successfully. Obstacle avoidance experiments were conducted using stationary obstacles placed along the navigation path. The same test arrangement was used throughout the experiments. Dynamic obstacles and moving objects were not included in the present study.

3.3. Analysis of RFID Detection Reliability and Track Geometry

RFID detection experiments were conducted using different line spacing configurations. The objective of the experiment was to evaluate the ability of the MFRC522 RFID reader to detect station tags while the carrier was moving along the navigation path.

3.3.1. Impact of Line Spacing on Detection Success

Three-line spacing configurations were evaluated: 9 cm, 11 cm, and 12 cm. For each configuration, the carrier was allowed to move through the RFID detection area, and the tag detection result was recorded. Table 5 lists the RFID detection results obtained during the experiments. A detection success rate of 0% was recorded for the 9 cm spacing configuration. The 11 cm spacing configuration produced a detection success rate of 100%. For the 12 cm spacing configuration, the detection success rate was 60%. Different detection results were observed for the three spacing configurations. The highest detection rate was obtained at the 11 cm spacing. The measured results for all test conditions are presented in Table 5.

Table 3. Navigation success rate under different track widths

Movement type	Track width (cm)	Successful trials	Success rate (%)
Straight path	1.8	8 out of 10	80
Straight path	3.6	10 out of 10	100
Corner path	1.8	6 out of 10	60
Corner path	3.6	8 out of 10	80

Table 4. Obstacle avoidance success rate

Obstacle avoidance strategy	Successful trials	Success rate (%)
Navigate-around approach	5 out of 10	50
Reverse–forward approach	10 out of 10	100

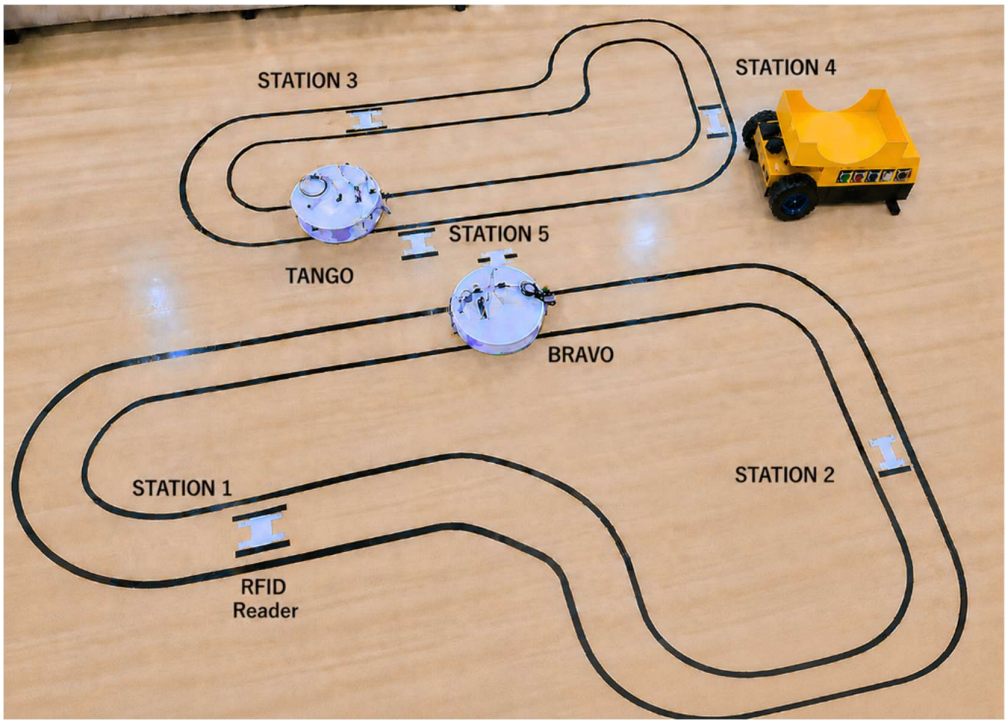

Figure 8. Testing track with different line sizes

3.4. Communication Latency and Control Response

Table 6 presents the communication measurements obtained from the carrier system. Three communication paths were examined, namely command transmission, inter-robot communication, and local obstacle response. The delay between a platform selection command and motor activation averaged 42 ms. Communication between Robot Bravo and Robot Tango through MQTT produced an average delay of 68 ms. The local obstacle response executed by ESP32 Module A was measured at less than 15 ms. No noticeable delay was observed during robot operation. Command execution, status updates, and obstacle responses occurred without interruption throughout the test runs. The results listed in Table 6 were obtained under the same laboratory conditions used for the navigation experiments.

3.5. Effect of Motor Speed and Track Width on Navigation Overshoot

Additional experiments were conducted to evaluate the effect of motor speed and track width on carrier movement. The experiments focused on the behaviour of the carrier during corner navigation. Overshoot was observed when the carrier moved beyond the centre of the navigation path before returning to the track.

3.5.1. Effect of Motor Speed

The effect of motor speed was examined using the 1.8 cm track configuration. During testing, the carrier was observed to overshoot the corner section more frequently at higher speed settings. In several runs, the robot moved beyond the centre of the track before correcting its position. The recorded values are given in Table 7.

Table 5. RFID detection movement evaluation

Distance between lines (cm)	Number of trials	Completed scans	Successful rate (%)
9	10	0	0
10	10	4	40
11	10	10	100
12	10	6	60

Table 6. Summary of system latency and reliability

Communication path	Average latency (ms)	Reliability (%)	Operational significance
Main Carrier to Sub-Carrier (Task)	42 ms	100%	Real-time task dispatching
Sub-Carrier to Peer (Arbitration)	68 ms	100%	Collision & redundancy prevention
Local Sensor to Actuator (Safety)	<15 ms	100%	Immediate obstacle handling

3.5.2. Effect of Track Width

Two track widths were considered in this study, namely 1.8 cm and 3.6 cm. The wider track generally produced fewer navigation failures during cornering. Overshoot was still observed, but the robot was able to recover and continue following the path in most trials. A similar trend appeared during the obstacle avoidance tests. The reverse-and-forward routine completed all runs, whereas the navigate-around routine failed in several cases after the robot lost track alignment. Tables 4 and 7 show the corresponding results.

Table 7. Correlation between speed, track width, and cornering overshoot

Motor speed (PWM %)	Track width (cm)	Avg. overshoot (mm)	Success rate (%)	Reliability assessment
40	1.8	4	80	High stability for narrow paths
60	1.8	12	50	Critical; sensor frequently loses track
40	3.6	5	90	Optimal for high-precision tasks
60	3.6	15	80	Successful due to wider error tolerance

4. Conclusions

This work described the development of a small autonomous carrier robot for structured factory transportation. The prototype consists of a Raspberry Pi 4, several ESP32 controllers, RFID identification, infrared line sensors, ultrasonic sensors, and MQTT communication. Two carrier robots, namely Robot Bravo and Robot Tango, were connected to a Main Robot Carrier for task coordination.

Several laboratory tests were carried out after the prototype was completed. During navigation testing, the robot was able to follow the predefined track and reached a maximum success rate of 90% under the selected operating conditions. RFID experiments showed noticeable differences between the tested line spacing configurations. Among the tested arrangements, the 11 cm spacing produced the most consistent tag detection results. For obstacle avoidance, the reverse-and-forward method completed all trials, whereas failures were observed when the navigate-around method was used.

Communication between the robots was also examined. During communication testing, the delay from a command signal to motor activation averaged 42 ms. Message exchange between the carrier robots required approximately 68 ms. For obstacle handling, the ultrasonic sensors and motor control routines were executed directly on the ESP32 controller. The measured response time was below 15 ms.

All experiments were carried out in a laboratory environment. The robots moved on predefined tracks and obstacle tests used stationary objects placed along the navigation path. Conditions normally found in a production floor were not available during the evaluation. Examples include moving vehicles, human traffic, wireless network congestion, and long-duration operation over multiple shifts. Several areas can still be investigated. Additional sensors may be added to the carrier platform, and tests involving a larger number of robots can be carried out. Communication performance under heavier network traffic and integration with existing factory automation equipment may also be examined in future studies.

Acknowledgement

The authors would like to acknowledge Universiti Malaysia Pahang Al-Sultan Abdullah for providing the facilities, resources, and technical support required to conduct this research.

Funding

This work was supported by Universiti Malaysia Pahang Al-Sultan Abdullah (UMPSA) under a research grant PDU243201.

Declaration of Competing Interest

The authors declare no conflict of interest.

CRediT Authorship Contribution Statement

I. H. A. Nasir (Methodology; Software and Prototype; Writing - original draft; Resources)
A.A. Hadi (Conceptualization; Funding Acquisition; Supervision; Writing – review & editing)
R. M. T. Raja Ismail (Conceptualization; Formal analysis; Validation; Writing – review & editing)
M. R. Othman (Software and Prototype; Resources)
M. R. M. Rodzi (Investigation; Project Administration)

Availability of the Data and Materials

The data used to support the findings of this study are included within the article.

Ethical Declaration

This research did not involve any human participants, animals, or sensitive personal data. Therefore, ethical approval was not required. All experimental evaluations were conducted on a prototype system in a controlled laboratory environment in accordance with institutional research guidelines.

Generative Artificial Intelligence Declarations

The authors used AI-based tools solely for language editing and grammar refinement. No content generation or data interpretation was performed by AI. All intellectual contributions are original and solely attributable to the authors.

REFERENCES

[1] S. Deokar, V. Bansode, S. Wankhede, and P. Kharche, "Industry 4.0 moving towards smart manufacturing: A comprehensive review," Int. J. Sci. Res. Arch., vol. 16, pp. 630–642, Sep. 2025.
[2] M. Attaran, S. Attaran, and B. G. Celik, "The impact of digital twins on the evolution of intelligent manufacturing and Industry 4.0," Adv. Comput. Intell., vol. 3, no. 3, p. 11, Jun. 2023.
[3] K. Höse, A. Amaral, U. Götze, and P. Peças, "Manufacturing Flexibility through Industry 4.0 Technological Concepts—Impact and Assessment," Glob. J. Flex. Syst. Manag., vol. 24, no. 2, pp. 271–289, 2023.
[4] L. Yang, H. Zou, C. Shang, X. Ye, and P. Rani, "Adoption of information and digital technologies for sustainable smart manufacturing systems for industry 4.0 in small, medium, and micro enterprises (SMMEs)," Technol. Forecast. Soc. Change, vol. 188, p. 122308, Mar. 2023.
[5] M. Faccio et al., "Human factors in cobot era: a review of modern production systems features," J. Intell. Manuf., vol. 34, no. 1, pp. 85–106, Jan. 2023.
[6] O. E. Oluyisola, S. Bhalla, F. Sgarbossa, and J. O. Strandhagen, "Designing and developing smart production planning and control systems in the industry 4.0 era: a methodology and case study," J. Intell. Manuf., vol. 33, no. 1, pp. 311–332, Jan. 2022.
[7] A. Bhargava, Mohd. Suhaib, and A. S. Singholi, "A review of recent advances, techniques, and control algorithms for automated guided vehicle systems," J. Braz. Soc. Mech. Sci. Eng., vol. 46, no. 7, p. 419, Jun. 2024.
[8] X. Zhao and T. Chidambareswaran, "Autonomous Mobile Robots in Manufacturing Operations," in 2023 IEEE 19th International Conference on Automation Science and Engineering (CASE), Aug. 2023, pp. 1–7.
[9] G. Fragapane, D. Ivanov, M. Peron, F. Sgarbossa, and J. O. Strandhagen, "Increasing flexibility and productivity in Industry 4.0 production networks with autonomous mobile robots and smart intralogistics," Ann. Oper. Res., vol. 308, no. 1, pp. 125–143, Jan. 2022.
[10] Y. Liu, S. Wang, Y. Xie, T. Xiong, and M. Wu, "A Review of Sensing Technologies for Indoor Autonomous Mobile Robots," Sensors, vol. 24, no. 4, p. 1222, Jan. 2024.
[11] M. Tolani, A. A. Ajasa, A. Balodi, A. Bajpai, Y. AlZaharani, and Sunny, "Advanced Sensor Systems for Robotics and Autonomous Vehicles," in Artificial Intelligence for Robotics and Autonomous Systems Applications, A. T. Azar and A. Koubaa, Eds. Cham: Springer International Publishing, 2023, pp. 439–459.
[12] A. Das and R. K. Jain, "Performance evaluation of mobile robotic system for monitoring and inspection using MQTT protocol and IoT technique," Int. J. Intell. Robot. Appl., vol. 9, no. 4, pp. 1302–1330, Dec. 2025.
[13] K. K, "The Evolution of Robot Operating Systems: A Review of ROS 2," Zenodo, Jun. 2025.
[14] D. Casini, J.-J. Chen, J. Li, F. Reghenzani, and H. Teper, "A Survey of Real-Time Support, Analysis, and Advancements in ROS 2," arXiv preprint arXiv:2601.10722, Dec. 2025.
[15] S. Macenski, T. Foote, B. Gerkey, C. Lalancette, and W. Woodall, "Robot Operating System 2: Design, Architecture, and Uses In The Wild," Sci. Robot., vol. 7, no. 66, p. eabm6074, May 2022.
[16] S. D'Avella, C. A. Avizzano, and P. Tripicchio, "ROS-Industrial based robotic cell for Industry 4.0: Eye-in-hand stereo camera and visual servoing for flexible, fast, and accurate picking and hooking in the production line," Robot. Comput.-Integr. Manuf., vol. 80, p. 102453, Apr. 2023.
[17] N. Demir, P. Demircioğlu, and İ. Böğrekci, "Advancing Industry 4.0 with ROS: A case study on autonomous mobile robot technological advancements," Int. J. 3D Print. Technol. Digit. Ind., vol. 8, no. 1, pp. 130–142, Apr. 2024.
[18] I. H. Savci, A. Yilmaz, S. Karaman, H. Ocakli, and H. Temeltas, "Improving Navigation Stack of a ROS-Enabled Industrial Autonomous Mobile Robot (AMR) to be Incorporated in a Large-Scale Automotive Production," Int. J. Adv. Manuf. Technol., vol. 120, no. 5, pp. 3647–3668, May 2022.
[19] W. Jo, J. Kim, R. Wang, J. Pan, R. K. Senthilkumaran, and B.-C. Min, "SMARTmBOT: A ROS2-based Low-cost and Open-source Mobile Robot Platform," arXiv preprint arXiv:2203.08903, Mar. 2022.
[20] A. Prasad, A. M. Nair, A. K. J. Nair, G. Govind, K. L. Nisha, and J. Divya Udayan, "Design and Implementation of Robot Operating System 2 (ROS 2) Controlled Robotic Arm for Bomb Disposal," in Advances in Data-driven Computing and Intelligent Systems, J. C. Bansal, S. Saha, C. A. C. Coello, and H. Rathore, Eds. Singapore: Springer Nature, 2025, pp. 351–362.
[21] J. Zhang, X. Yu, S. Ha, J. P. Queralta, and T. Westerlund, "Comparison of Middlewares in Edge-to-Edge and Edge-to-Cloud Communication for Distributed ROS2 Systems," J. Intell. Robot. Syst., vol. 110, no. 4, p. 162.
[22] R. Carreira, N. Costa, J. Ramos, L. Frazão, and A. Pereira, "A ROS2-Based Gateway for Modular Hardware Usage in Heterogeneous Environments," Sensors, vol. 24, no. 19, p. 6341, Jan. 2024.