International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 15 Issue 11 , November - 2026

Autonomous Navigation in Airport Terminals: A LiDAR-IMU Fusion and Partitioned Edge Architecture using ROS 2

Vatsal Parekh **Shiva Sukumar** **Taran Khandelwal**

Yusuf Khambaty **Prof. Renia Dias**

Department of Electronics and Telecommunication Engineering Dwarkadas J. Sanghvi College of Engineering, Vile Parle, Mumbai, India

***Abstract* – Increasing passenger volumes in global airport terminals have intensified the need for automated service robots capable of seamless navigation in crowded, dynamic environments. Conventional robotic designs have had issues with localization drift in sparsely structured corridors, along with computationally expensive SLAM based on 3D vision technology. The design presented in this work addresses this problem by proposing an autonomous mobile robot capable of navigating efficiently in such airport terminals. By integrating data from both 2D LiDAR and IMU sensors, the robot is able to localize and avoid obstacles in a computationally efficient manner. By adopting distributed edge computing, the design relies on a Raspberry Pi and an ESP32 together with the ROS 2 Nav2 framework for real-world performance.**

***Keywords* – Autonomous navigation, LiDAR–IMU fusion, ROS 2, Nav2, edge computing, airport service robots.**

I. INTRODUCTION

With increasing demands for air travel, automation is required to meet staffing needs, language capabilities, and passenger movement control at airports [1]. As passenger traffic continues to grow, airport operators face challenges in maintaining operational efficiency while ensuring a seamless travel experience. Autonomous mobile robots have emerged as a promising solution for assisting passengers with navigation, information delivery, and mobility support within large terminal environments [2]. However, deploying robots in crowded public spaces introduces several navigation challenges, including safe interaction with pedestrians, reliable localization in open and repetitive indoor environments, and efficient path planning in the presence of dynamic obstacles [2], [3]. Recent real-world airport trials confirm that these challenges are not yet solved: even a dedicated autonomous transport robot needed frequent human intervention in complex traffic and obstacle situations [4].

Mobile robots operating in airport terminals must be capable of executing smooth and natural motion during evasive maneuvers, accurately identifying their position in long and visually similar corridors, and adapting to changing pedestrian traffic without disrupting human movement. These requirements demand robust localization and obstacle avoidance mechanisms that can operate reliably under real-world conditions. Traditional robotic navigation systems often experience localization drift due to the smooth, feature-sparse flooring and highly repetitive structural layouts commonly found in modern airport terminals [5]. Such environments limit the effectiveness of conventional localization techniques and can degrade navigation accuracy over time.

To address these challenges, a lightweight sensor architecture based on the fusion of 2D LiDAR and Inertial Measurement Unit (IMU) data is employed. Compared with computationally intensive 3D sensing and

(This work is licensed under a Creative Commons Attribution 4.0 International License.)

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 15 Issue 11 , November - 2026

vision-based SLAM approaches, this sensor fusion strategy significantly reduces processing requirements while maintaining sufficient environmental awareness and localization performance [6]. Recent 2D LiDAR–inertial SLAM systems likewise report improved trajectory accuracy over established baselines [7], [8], [9]. Furthermore, the deployment of ROS 2 [10] on a distributed edge-computing architecture utilizing Raspberry Pi and ESP32 devices enables the low-latency communication and real-time processing essential for safe crowd navigation [1], [11].

This paper proposes a compact, resource-efficient autonomous mobile robot optimized for airport terminal environments, executing real-time LiDAR–IMU sensor fusion, autonomous navigation, and dynamic obstacle avoidance natively on decentralized edge hardware.

The proposed system aims to provide a scalable and cost-effective solution for autonomous indoor navigation in airport environments. By leveraging lightweight sensing, distributed computation, and modern robotic middleware, the robot can operate efficiently on resource-constrained hardware while maintaining reliable navigation performance. The integration of real-time localization, path planning, and obstacle avoidance enables the system to adapt to dynamic airport conditions and varying pedestrian densities.

II. LITERATURE SURVEY

A. Airport and Service Robot Applications

Recent studies have demonstrated the growing role of autonomous service robots in airport environments. Giannopoulou et al. [1] deployed a humanoid service robot at Athens International Airport using the ROS 2 Nav2 framework combined with a 5G-enabled thermal camera network for navigation in crowded areas. While the system showed promising results in assisting passengers and avoiding congestion, its long-term scalability and performance in larger deployments remain uncertain. Similarly, Saito and Koshiishi [2] developed an omnidirectional guidance robot for Narita Airport to assist passengers with navigation and information services. However, the system still required human intervention in complex and unexpected situations, highlighting the need for more robust autonomous navigation capabilities. A similar pattern appears in the airport ground-handling domain, where Wehner et al. [4] tested an autonomous pallet-transport robot at Munich Airport's air cargo area for ten days; tasks were completed, but frequent human intervention was again required in complex traffic and obstacle situations.

B. LiDAR-Based Indoor SLAM and Mapping

Accurate localization and mapping are essential for autonomous navigation in indoor environments. Shamsudin et al. [6] demonstrated that a single 2D RP-LiDAR combined with Hector SLAM can effectively generate maps for indoor delivery applications. However, the approach was found to experience localization drift in feature-sparse environments. To address this limitation, Reda et al. [5] proposed a Gmapping-based approach that integrates LiDAR and IMU data using a Rao–Blackwellized Particle Filter [12], [13]. Their results showed improved localization accuracy and reduced spatial drift in long corridors through loop closure techniques. Filipenko and Afanasyev [14] compared ROS-based SLAM systems on an indoor office dataset, including the 2D-LiDAR methods GMapping, Hector SLAM and Cartographer, and reported Cartographer as the best-performing LiDAR-based system.

More recent work moves toward tighter inertial coupling. Zhang et al. [7] fuse 2D LiDAR, IMU and wheel odometry using line features and ground constraints, with a real-time loop-closure module, and report lower trajectory error than Cartographer. Da et al. [8] extend Cartographer with a hybrid point-cloud filter and a factor-graph LiDAR/IMU framework built on IMU pre-integration, outperforming Cartographer, Karto, Hector and GMapping in complex indoor scenes. Kalan et al. [9] combine a 2D LiDAR and IMU under Cartographer

(This work is licensed under a Creative Commons Attribution 4.0 International License.)

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 15 Issue 11 , November - 2026

with CNN-based object detection to improve localization in cluttered, changing environments. These systems were evaluated on general indoor platforms and datasets rather than airport service robots, but they reinforce the value of inertial fusion that the proposed system adopts within a Gmapping framework, and occupancy-grid maps [15] remain the common map representation across them.

C. Obstacle Detection and Local Environment Representation

Obstacle detection plays a critical role in ensuring safe robot navigation in crowded environments. The system presented in [1] employed YOLO-based pedestrian detection on thermal images to update local costmaps for navigation. In contrast, Saito and Koshiishi [2] utilized a combination of laser rangefinders and stereo cameras to estimate collision risks and avoid obstacles. Classical reactive methods such as the virtual force field [16] avoid vision altogether, whereas vision-based approaches [17], typically built on libraries such as OpenCV [18], offer richer semantics at higher computational cost. Recent YOLO-family detectors [19] narrow that gap: Medjaldi et al. [20] paired YOLOv8 with an RGB-D sensor on an AGV and reported 92.6% accuracy and 95% mAP@0.5 in structured indoor settings. To reduce computational complexity, the proposed system relies on 2D LiDAR data for obstacle segmentation and clustering, with a lightweight YOLOv8n model for semantic classification. The detected obstacles are incorporated into local costmaps, enabling efficient obstacle representation without the need for GPU-based vision processing.

D. Local Path Planning and Obstacle Avoidance

Efficient path planning and obstacle avoidance are essential for smooth robot operation in dynamic environments. Shamsudin et al. [6] combined the A* algorithm for global path planning with the Dynamic Window Approach (DWA) [21] for real-time local obstacle avoidance. Their approach enabled effective navigation while responding to moving obstacles. Recent DWA variants target crowds directly: Hu et al. [3] add goal-distance and oscillation terms to DWA and report shorter paths and faster travel times in pedestrian-flow simulations, while Jian et al. [22] propose a long-term DWA that achieves higher success rates and smoother motion in static and crowded environments. Furthermore, Saito and Koshiishi [2] emphasized the importance of generating human-like avoidance behaviors to improve interactions with pedestrians. Building upon these findings, the proposed system integrates global and local planning capabilities within the ROS 2 Nav2 framework [23] to achieve safe and natural navigation in airport terminals.

E. Embedded and Edge-Based Implementation

Several studies have explored distributed computing architectures for autonomous robots. Giannopoulou et al. [1] utilized Jetson AGX hardware for neural network inference and MQTT-based communication. Shamsudin et al. [6] implemented a distributed architecture where path planning was performed on a portable computer while motor control was handled by an Arduino Mega. More recently, Stadnik and Wymysłowski [11] ran micro-ROS directly on a TurtleBot3 microcontroller, removing the companion computer and reporting roughly 68.6% lower average power draw than the Raspberry Pi-based setup, which illustrates the benefit of pushing low-level work onto microcontrollers. Inspired by these approaches, the proposed system employs a lightweight edge-computing architecture consisting of a Raspberry Pi and an ESP32 microcontroller. This configuration enables real-time processing of LiDAR data at 10 Hz while maintaining responsive motor control and efficient resource utilization.

III. IMPLEMENTATION

A. Architecture Overview

The proposed autonomous mobile robot is organized into multiple functional layers (Fig. 1) to achieve reliable navigation in airport environments. The sensing layer consists of a 2D LiDAR and an Inertial Measurement Unit (IMU), which provide spatial and kinematic information while maintaining a lightweight hardware configuration. The preprocessing layer filters raw LiDAR scans at a frequency of 10 Hz and synchronizes them with IMU measurements to reduce sensor noise and eliminate invalid reflections [1], [6].

(This work is licensed under a Creative Commons Attribution 4.0 International License.)

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 15 Issue 11 , November - 2026

The perception layer processes laser scan data to identify and cluster obstacles such as pedestrians and luggage, enabling the robot to maintain safe operational distances during navigation. In the mapping and localization layer, the Gmapping algorithm based on a Rao–Blackwellized Particle Filter (RBPF) [12] generates a two-dimensional occupancy grid map [15] while utilizing loop closure techniques to reduce localization drift in long and feature-sparse corridors [5]. Robot velocity is constrained to 1.0 m/s to ensure stable localization performance.

The planning layer employs the ROS 2 Nav2 framework [23] with a dual-planner architecture. An A* planner is used for global route generation, while the Dynamic Window Approach (DWA) [21] is utilized for local obstacle avoidance and real-time trajectory adjustment in dynamic environments [1], [6]. The control layer converts navigation commands into differential-drive wheel velocities, whereas the communication layer relies on ROS 2 [10] for inter-node communication and MQTT for external telemetry transmission. Finally, the monitoring layer provides remote supervision through a dashboard capable of displaying robot telemetry and optional congestion alerts generated from external camera systems [1].

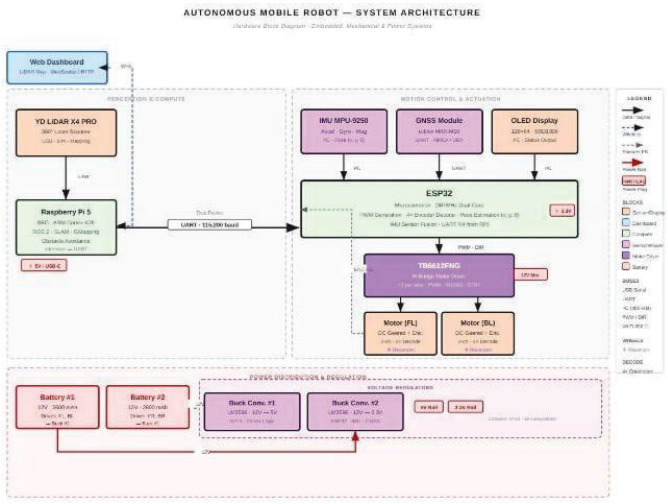


Fig. 1: System overview of the autonomous mobile robot.

B. Edge Hardware Architecture

The proposed system adopts a distributed edge-computing architecture consisting of a Raspberry Pi and an ESP32 microcontroller. The Raspberry Pi functions as the high-level processing unit and executes the ROS 2 middleware, Gmapping SLAM, costmap generation, global and local path planning, and MQTT-based telemetry communication. Additional edge cameras can be integrated for crowd-monitoring applications [1].

The ESP32 serves as the low-level motor control unit. It manages wheel encoder interrupts, acquires IMU sensor measurements, executes PID control algorithms, and generates PWM signals for motor actuation. This hardware separation reduces computational load on the navigation processor while ensuring responsive motion control [6], [11].

C. Processing Pipeline

The navigation workflow begins with the acquisition of LiDAR and IMU data. The collected sensor information undergoes preprocessing and noise filtering before obstacle segmentation and clustering are performed. The resulting obstacle information is incorporated into a local costmap, which is then used by the A* global planner and DWA local planner [21], [23] to generate navigation decisions. Motion commands are transmitted to the ESP32, where differential-drive motor control is executed using PID feedback derived from encoder and IMU measurements. Finally, operational data and system status information are published to the airport monitoring dashboard through MQTT communication.

(This work is licensed under a Creative Commons Attribution 4.0 International License.)

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 15 Issue 11 , November - 2026

D. Design Justification

The proposed architecture is designed to balance computational efficiency and navigation performance. The combination of a single 2D LiDAR and IMU provides sufficient environmental awareness while requiring significantly fewer computational resources than 3D LiDAR or vision-based SLAM systems [6], [7]. This enables deployment on resource-constrained platforms such as the Raspberry Pi. Furthermore, the integration of A* and DWA planners combines efficient long-range path planning with rapid local obstacle avoidance, resulting in navigation behavior suitable for crowded public environments [3], [6], [22]. The use of ROS 2 and MQTT ensures low-latency communication and reliable system coordination, which are critical requirements for safe autonomous operation in airport terminals [1], [10].

IV. METHODOLOGY

A. Sensor Fusion and Edge Processing

Accurate localization is a critical requirement for autonomous navigation in airport environments, where smooth flooring and repetitive corridor structures can cause significant wheel odometry drift. To address this challenge, the proposed system employs sensor fusion by integrating data from a 2D LiDAR and an Inertial Measurement Unit (IMU) within the Gmapping Rao–Blackwellized Particle Filter (RBPF) framework [5], [12]. Sensor measurements are continuously fused to estimate the robot's pose while reducing localization errors and map distortions. The incorporation of IMU data improves pose estimation accuracy [7], [8] and helps maintain localization during temporary LiDAR occlusions caused by moving pedestrians or obstacles.

For local navigation, the Dynamic Window Approach (DWA) planner [21] is utilized within the ROS 2 Nav2 framework [23]. DWA evaluates feasible velocity commands in real time and selects safe trajectories that avoid collisions while maintaining smooth robot motion. This approach enables continuous navigation through crowded airport terminals without abrupt stops that may interfere with pedestrian flow [22].

The computational workload is distributed between two edge-computing devices. The Raspberry Pi performs high-level processing tasks, including ROS 2 communication, occupancy grid mapping, localization, path planning, and MQTT telemetry transmission. The ESP32 microcontroller is responsible for low-level control functions such as encoder acquisition, IMU interfacing, PID control execution, and PWM-based motor actuation. This distributed architecture improves system responsiveness and optimizes resource utilization [6], [11].

B. Algorithmic Navigation Pipeline

The proposed navigation methodology follows the sequence below:

1. Acquisition of environmental data using a 2D LiDAR operating at 10 Hz and IMU measurements.
2. Preprocessing and filtering of sensor data to remove noise, invalid reflections, and outlier measurements.
3. Segmentation of LiDAR scan data to distinguish static infrastructure from dynamic obstacles.
4. Clustering of segmented points into discrete obstacle representations.
5. Generation of a local costmap incorporating obstacle locations and traversal costs [23].
6. Global path generation using the A* algorithm and local trajectory optimization using the Dynamic Window Approach (DWA) [21].
7. Transmission of velocity commands to the ESP32 for differential-drive control.
8. Closed-loop motor control through PID algorithms using encoder and IMU feedback.
9. Simultaneous localization and mapping using Gmapping [12], including occupancy grid updates and pose

(This work is licensed under a Creative Commons Attribution 4.0 International License.)

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 15 Issue 11 , November - 2026

estimation.

10. Transmission of telemetry data and system status information to the monitoring dashboard via MQTT communication [1].

C. Implementation Notes

The complete methodology is implemented within the ROS 2 ecosystem [10], [24] to ensure modularity, scalability, and ease of integration. The publisher–subscriber communication model enables asynchronous data exchange between sensing, localization, planning, and control modules while sustaining a 10 Hz LiDAR processing cycle. Furthermore, MQTT-based telemetry communication operates concurrently with navigation processes without affecting real-time performance.

To maintain compatibility with embedded hardware, the system processes two-dimensional LiDAR data instead of computationally intensive three-dimensional point clouds. This significantly reduces CPU utilization, memory consumption, and power requirements while preserving adequate environmental awareness for indoor navigation. As a result, the Raspberry Pi is capable of executing localization, mapping, and path-planning tasks in real time without excessive thermal load, making the proposed architecture suitable for long-duration deployment in airport terminals.

V. RESULTS

A. Robot Prototype and Hardware Assembly

The autonomous navigation robot was successfully assembled on a blue acrylic chassis mounted with four mecanum wheels, enabling omnidirectional movement (Figs. 2 and 3). The YDLIDAR X4PRO was mounted centrally on the top layer for an unobstructed 360° field of view. The partitioned edge computing architecture was realized with the Raspberry Pi 5 handling all ROS 2 navigation tasks and the ESP32 managing low-level motor control and IMU data acquisition. The complete electronics stack, including motor drivers, IMU, voltage regulators, and battery, was laid out on a perforated board for modularity and ease of debugging; the wiring is documented in the schematic of Fig. 7, and the IMU orientation output of the complementary filter is shown in Fig. 8.

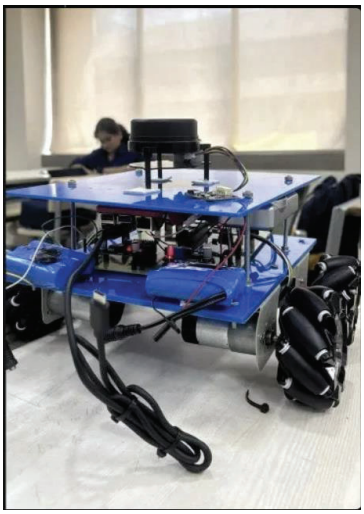


Fig. 2: Side view of the modular robot.

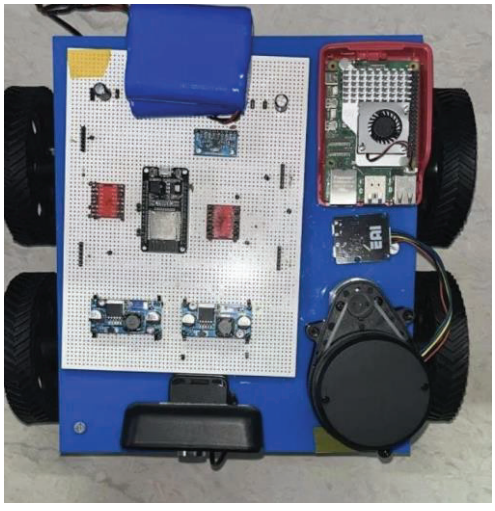


Fig. 3: Top view of the electronics layout (Raspberry Pi 5, ESP32, IMU, motor drivers, LiDAR).

(This work is licensed under a Creative Commons Attribution 4.0 International License.)

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 15 Issue 11 , November - 2026

B. LiDAR Sensor Validation and Drift Analysis

Prior to full system integration, the YDLIDAR X4PRO was independently validated for measurement accuracy and long-term drift. The raw 360° polar scan (Fig. 4) confirmed reliable detection of surrounding obstacles and walls within a 1 m radius during a stationary test. The sensor was then run continuously for approximately 850 seconds with a static white box target placed at 29.66 cm from the sensor centre (Fig. 5). The measured distance remained tightly clustered around the target value with an overall mean of 298.998760 mm against the true 299 mm, demonstrating sub-millimetre mean accuracy. A minor downward drift of approximately 0.02 mm was observed toward the end of the run, attributed to minor thermal expansion of the sensor housing – well within acceptable tolerance for the navigation use case.

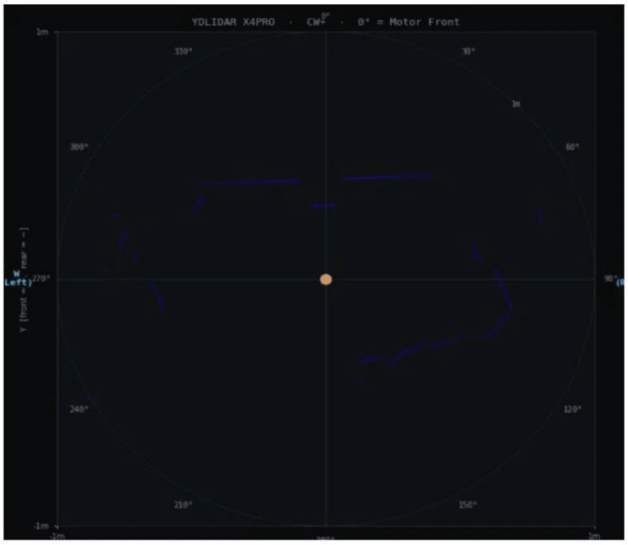


Fig. 4: YDLIDAR X4PRO raw 360° polar scan – obstacle returns visible across multiple azimuths.

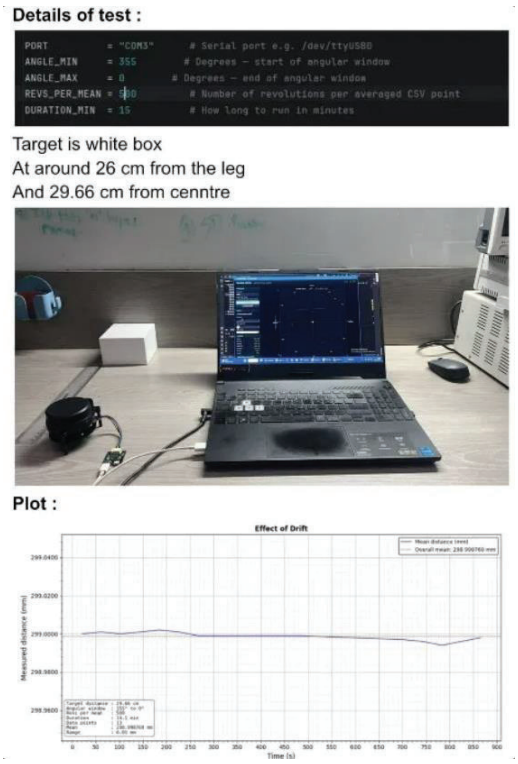


Fig. 5: LiDAR drift test setup and effect-of-drift plot over 900 seconds.

(This work is licensed under a Creative Commons Attribution 4.0 International License.)

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 15 Issue 11 , November - 2026

C. Object Detection – YOLOv8n Training Results

To enable semantic understanding of the environment, a YOLOv8n model [19], [25] was trained for real-time obstacle classification. The model was trained for 30 epochs on a custom dataset comprising indoor navigation scenarios pertaining to warehouse and logistics environments, commonly encountered in airport operations. Precision improved steadily from 0.725 at epoch 1 to a stable plateau of approximately 0.915 by epoch 20, with minor oscillations thereafter. Recall followed a similar convergence trajectory, rising from 0.38 at epoch 1 to approximately 0.92 by epoch 30, indicating the model reliably identifies the vast majority of obstacles in the scene by the final epoch.

Training and validation losses (box, classification, and DFL) all converged smoothly without signs of overfitting – validation losses tracked training losses closely throughout. The mAP@0.5 reached 90.4% and mAP@0.5:0.95 reached approximately 68% by epoch 30 (confusion matrix in Fig. 6). For context, Medjaldi et al. [20] report 95% mAP@0.5 for YOLOv8 with an RGB-D sensor on a different dataset; our figure is slightly lower but is obtained on a different, small custom dataset. Inference speed was measured at 4.7 ms per frame on the Raspberry Pi 5, confirming real-time viability at well above 10 fps without a GPU.

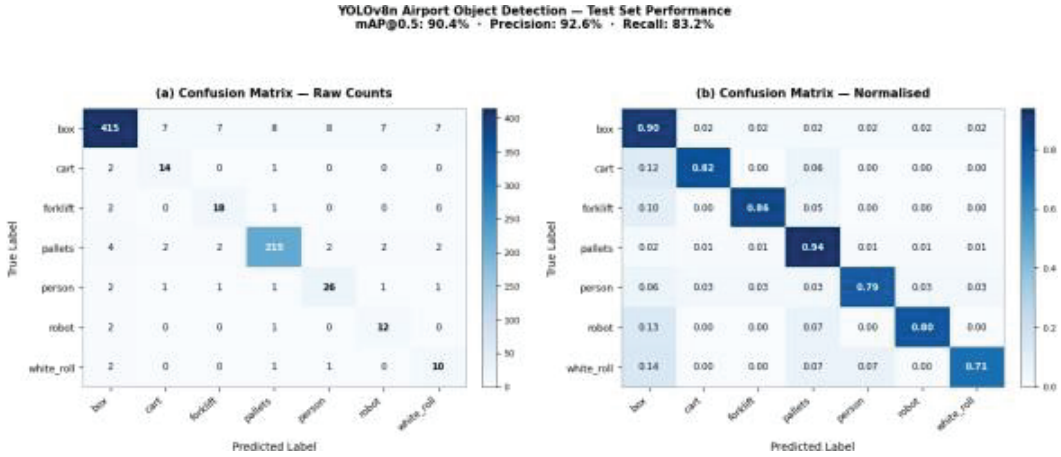


Fig. 6: Confusion matrix – YOLOv8n object detection model.

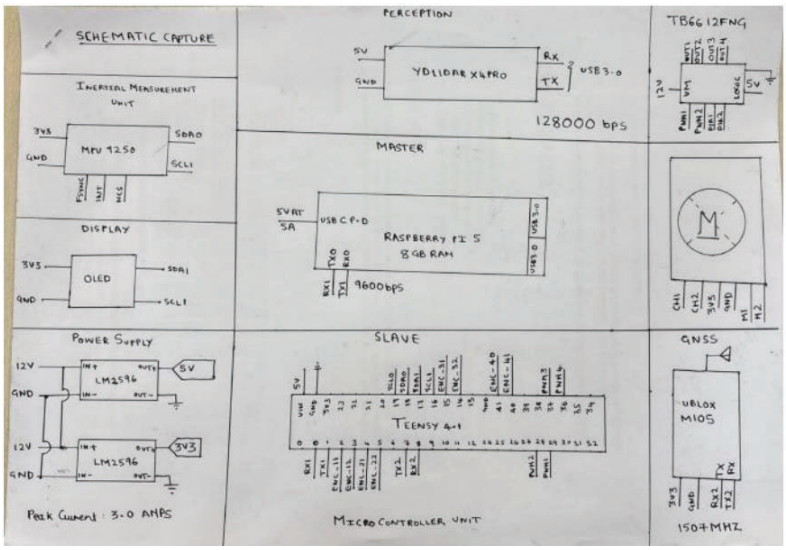


Fig. 7: Hand-drawn schematic of the electronics.

(This work is licensed under a Creative Commons Attribution 4.0 International License.)

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 15 Issue 11 , November - 2026

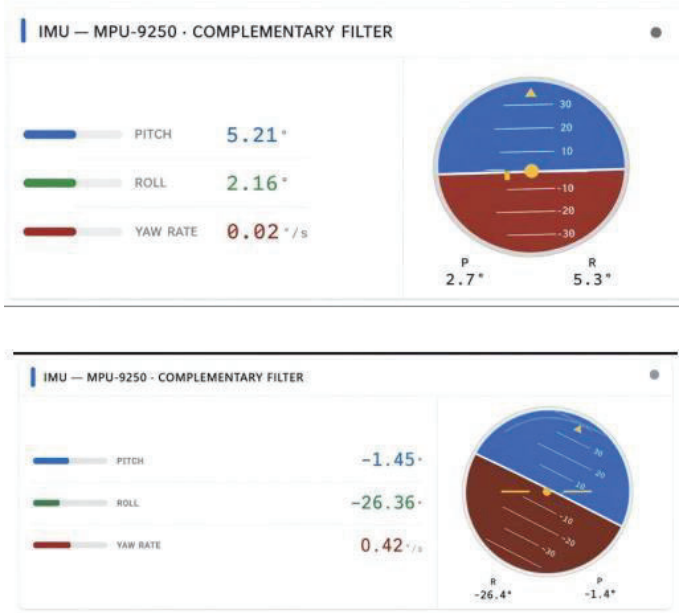


Fig. 8: MPU-9250 pitch, roll and yaw-rate output (complementary filter).

D. Remote Monitoring and Telemetry Dashboard

A browser-accessible Robot Control Station was developed and deployed on the local network. The dashboard provided real-time visualization of the LiDAR point cloud, robot heading and velocity, system status indicators, and manual drive controls. Communication with the robot was established over MQTT, with measured command round-trip latency consistently below 20 ms, confirming suitability for real-time teleoperation. The live LiDAR map updated continuously during robot motion without perceptible lag, enabling operators to monitor dynamic obstacle positions relative to the robot's current pose at all times. The manual override capability provides ground staff with direct intervention without interrupting the autonomous navigation pipeline, addressing the human-intervention need reported in earlier airport deployments [2], [4].

VI. HARDWARE AND SOFTWARE STACK

The system leverages a mobile base whose motion is governed by standard non-holonomic kinematic models [26]. A 2D LiDAR sensor with a rate of 10 Hz collects geometrical information and, combined with IMU measurements, mitigates the wheel-odometry slip errors caused by smooth terminal floors [5], [7]. Computations are distributed between the Raspberry Pi for higher-level navigation calculations and the ESP32 for kinematics execution, including PID controller implementation and encoder interrupts [6], [11]. There is the possibility of adding an edge camera for congestion detection [1].

In the software stack, the ROS 2 middleware [10], [24] manages asynchronous data exchanges among the navigation nodes. Gmapping SLAM creates two-dimensional occupancy grids using the Rao–Blackwellized Particle Filter method, employing LiDAR and IMU sensor measurements, with map-building velocity limited to
1.0 m/s to avoid spatial displacement errors [5], [12]. The ROS 2 Nav2 dual-planner architecture [23] uses the A* algorithm for global path planning and the dynamic window approach [6], [21] for dynamic human avoidance. The Raspberry Pi sends low-latency MQTT telemetry information to the airport dashboards [1].

(This work is licensed under a Creative Commons Attribution 4.0 International License.)

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 15 Issue 11 , November - 2026

VII. CONCLUSION AND FUTURE WORK

A. Conclusion

This paper proposed a resource-efficient, decentralized edge architecture for autonomous navigation in congested airport terminals. The computational burden is divided such that the Raspberry Pi handles the ROS 2 middleware and path planning, while the ESP32 is in charge of real-time kinematics execution [1], [6], [11]. LiDAR–IMU fusion within a Gmapping SLAM framework, constrained to 1.0 m/s, achieves high localization fidelity without 3D or vision-based SLAM overhead [5], [7]. A dual A* and DWA planner configuration enables fluid, human-like crowd evasion maneuvers [2], [22].

B. Future Work

Three improvements are proposed for scaling up the solution to real-world situations. The first is the inclusion of multiple robots that can coordinate with each other and use an effective swarm routing algorithm in case of high traffic [1]. The second is the use of visual semantic segmentation in addition to 2D LiDAR data to recognize transparent and reflective architectural elements such as glass facades, following the LiDAR–CNN fusion direction of [9]. The third is the inclusion of safety certification and integration with the airport management system's API.

REFERENCES

[1] E. Giannopoulou, P. Demestichas, P. Katrakazas, S. Saliverou, and N. Papagiannopoulos, "AI-powered service robots for smart airport operations: Real-world implementation and performance analysis in passenger flow management," *Sensors*, vol. 26, no. 3, Art. no. 806, 2026, doi: 10.3390/s26030806.

[2] Y. Saito and T. Koshiishi, "Development and styling design of airport guide robot system," *Honda R&D Technical Review*, Apr. 2021.

[3] Y. Hu, H. Long, and M. Chen, "The analysis of pedestrian flow in the smart city by improved DWA with robot assistance," *Scientific Reports*, vol. 14, Art. no. 11456, 2024, doi: 10.1038/s41598-024-62134-8.

[4] M. Wehner, C. Blesing, N. Ullrich, M. Gössner, and M. Klumpp, "Air cargo logistics automation and digital airport process management: Comprehensive empirical insights from Germany," *Logistics Research*, vol. 19, no. 1, pp. 7–31, 2026, doi: 10.1108/LORE-04-2025-0054.

[5] M. Reda, A. Abdulbaky, H. El-Hussieny, and M. Magdy, "Experimental analysis of Gmapping SLAM algorithm for mobile robot indoor navigation," in *Proc. 5th Int. Conf. Artificial Intelligence, Robotics and Control (AIRC)*, 2024.

[6] A. U. Shamsudin et al., "Autonomous navigation robot using SLAM and path planning based on a single RP-LIDAR,"
J. Advanced Research in Applied Sciences and Engineering Technology, vol. 53, no. 2, pp. 161–169, 2025.

[7] B. Zhang, Z. Peng, B. Zeng, and J. Lu, "2DLIW-SLAM: 2D LiDAR-inertial-wheel odometry with real-time loop closure," arXiv:2404.07644, 2024.

[8] C. Da, Z. Chen, T. Song, and Y. Lu, "Tightly coupled SLAM system for indoor complex scenes," *Robotica*, 2025, doi: 10.1017/S0263574725000463.

[9] S. H. Kalan, B. G. Lee, and W.-Y. Chung, "MDF: Multi-modal data fusion with CNN-based object detection for enhanced indoor localization using LiDAR-SLAM," arXiv:2505.08388, 2025.

[10] S. Macenski, T. Foote, B. Gerkey, C. Lalancette, and W. Woodall, "Robot Operating System 2: Design, architecture, and uses in the wild," *Science Robotics*, 2022, doi: 10.1126/scirobotics.abm6074.

[11] B. Stadnik and A. Wymysłowski, "Micro-ROS system integration onto the TurtleBot3 mobile robot platform," *J. Automation, Mobile Robotics and Intelligent Systems*, vol. 20, no. 3, pp. 55–62, 2026, doi: 10.14313/jamris-2026-038.

[12] G. Grisetti, C. Stachniss, and W. Burgard, "Improved techniques for grid mapping with Rao-Blackwellized particle filters," *IEEE Trans. Robotics*, vol. 23, no. 1, pp. 34–46, Feb. 2007.

[13] M. Montemerlo, S. Thrun, D. Koller, and B. Wegbreit, "FastSLAM: A factored solution to the simultaneous localization and mapping problem," in *Proc. AAAI Conf. Artificial Intelligence*, 2002.

[14] M. Filipenko and I. Afanasyev, "Comparison of various SLAM systems for mobile robot in an indoor environment," arXiv:2501.09490, 2025.

[15] A. Elfes, "Occupancy grids: A stochastic spatial representation for active robot perception," in *Proc. 6th Conf. Uncertainty in AI*, 1990.

[16] J. Borenstein and Y. Koren, "Real-time obstacle avoidance for fast mobile robots," *IEEE Trans. Systems, Man, and Cybernetics*, vol. 19, no. 5, pp. 1179–1187, 1989.

[17] F. Bonin-Font, A. Ortiz, and G. Oliver, "Visual navigation for mobile robots: A survey," *J. Intelligent and Robotic Systems*, vol. 53, no. 3, pp. 263–296, 2008.

[18] G. Bradski and A. Kaehler, *Learning OpenCV: Computer Vision with the OpenCV Library*. Sebastopol, CA, USA: O'Reilly Media, 2008.

[19] J. Terven, D.-M. Córdova-Esparza, and J.-A. Romero-González, "A comprehensive review of YOLO architectures in computer vision: From YOLOv1 to YOLOv8 and YOLO-NAS," *Machine Learning and Knowledge Extraction*, vol. 5, no. 4, pp. 1680–1716, 2023, doi: 10.3390/make5040083.

[20] A. Medjaldi, Y. Slimani, and N. Karkar, "Cost-effective real-time obstacle detection and avoidance for AGVs using YOLOv8 and RGB-D sensors," *Engineering, Technology & Applied Science Research*, vol. 15, no. 2, pp. 21738–21745, 2025, doi: 10.48084/etasr.10135.

[21] D. Fox, W. Burgard, and S. Thrun, "The dynamic window approach to collision avoidance," *IEEE Robotics & Automation Magazine*, vol. 4, no. 1, pp. 23–33, Mar. 1997.

[22] Z. Jian, S. Zhang, L. Sun, W. Zhan, N. Zheng, and M. Tomizuka, "Long-term dynamic window approach for kinodynamic local planning in static and crowd environments," arXiv:2310.02648, 2023.

[23] S. Macenski, F. Martín, R. White, and J. Clavero, "The Marathon 2: A navigation system," in *Proc. IEEE/RSJ Int. Conf. Intelligent Robots and Systems (IROS)*, 2020.

[24] M. Quigley et al., "ROS: An open-source Robot Operating System," in *ICRA Workshop on Open Source Software*, 2009.

[25] G. Jocher et al., "YOLOv8 by Ultralytics," 2023. [Online]. Available: https://github.com/ultralytics/ultralytics

[26] S. Thrun, W. Burgard, and D. Fox, *Probabilistic Robotics*. Cambridge, MA, USA: MIT Press, 2005.

(This work is licensed under a Creative Commons Attribution 4.0 International License.)