IAES International Journal of Robotics and Automation (IJRA)
Vol. 15, No. 4, December 2026, pp. 820∼831
ISSN: 2722-2586, DOI: 10.11591/ijra.v15i4.pp820-831

Implementation of simulation and architecture of a wheeled autonomous robot using ROS 2 and Gazebo

Tigor Hamonangan Nasution, Soeharwinto, Lucky Sam Pratama Telaumbanua, Stephany, Fahmi Fahmi
Department of Electrical Engineering, Universitas Sumatera Utara, Medan, Indonesia

Article Info

Article history:

Received Dec 2, 2025
Revised Aug 29, 2026
Accepted Nov 10, 2026

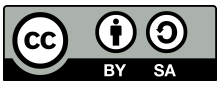
Keywords:

Gazebo
Modular architecture
Obstacle avoidance
ROS 2
Ultrasonic sensor
Wheeled robot

ABSTRACT

This study presents the simulation and physical implementation of a modular obstacle-avoidance architecture for a wheeled robot using Robot Operating System 2 (ROS 2) and Gazebo. The robot uses three HC-SR04 ultrasonic sensors positioned at the front, left, and right, a Raspberry Pi 4B, an L298N motor driver, and two DC motors. The software is organized into separate perception, decision, and actuation nodes that communicate through ROS 2 topics. A rule-based controller determines forward, turning, and recovery maneuvers from the three directional distance inputs. Functional validation was conducted in four representative static-obstacle configurations involving front, left, right, and simultaneously blocked directions. In each configuration, the robot response matched the predefined control rule. The physical experiments also revealed limitations that were not fully represented in simulation, particularly ultrasonic blind spots, measurement fluctuations, and unstable direction changes in dead-end conditions. The study demonstrates a reproducible low-cost architecture for testing ROS 2-based wheeled robot navigation while identifying the technical requirements for future quantitative, map-based, and dynamic-environment evaluation.

This is an open access article under the CC BY-SA license.



Corresponding Author:

Tigor Hamonangan Nasution
Department of Electrical Engineering, Universitas Sumatera Utara
Medan, Indonesia
Email: tigor.nasution@usu.ac.id

1. INTRODUCTION

The development of autonomous mobile robots continues to expand across industrial automation, transportation, logistics, service robotics, and other intelligent-system applications [1]-[4]. Safe local navigation is a fundamental capability that requires coordinated sensing, decision-making, and actuation. robot operating system 2 (ROS 2) provides a modular publisher-subscriber architecture that supports distributed nodes, reusable interfaces, heterogeneous hardware integration, and scalable communication [5]-[8]. Its timing behavior, middleware performance, and suitability for embedded platforms have also been investigated in recent studies [9]-[11]. In parallel, Gazebo is widely used with ROS-based systems to model robot geometry, sensors, actuators, and environmental interactions before physical deployment [12]-[16].

Recent mobile-robot systems increasingly employ LiDAR, simultaneous localization and mapping (SLAM), navigation stacks, sensor fusion, and learning-based control [3], [7], [12], [17], [18]. Although these approaches provide greater environmental awareness, they also require additional sensing, computation, calibration, and validation. For low-cost embedded robots, deterministic reactive control remains useful as a transparent local navigation layer. However, an important gap remains between ROS 2/Gazebo demonstrations

and reproducible physical implementations. Simulation can reproduce robot structure and software interfaces, but equivalent interfaces do not guarantee equivalent sensing or motion behavior [15], [19]. This is particularly relevant for HC-SR04 ultrasonic sensors, whose measurements are affected by target orientation, surface characteristics, sensing geometry, and environmental conditions [20]-[22]. Consequently, physical implementations may exhibit blind regions, measurement fluctuations, actuator asymmetry, and unstable recovery behavior that are less apparent in simulation.

This study addresses this gap through a simulation-to-physical proof-of-concept architecture for a differential-drive robot equipped with three HC-SR04 sensors and a Raspberry Pi 4B. The same high-level obstacle-avoidance process is implemented in Gazebo and on the physical robot using separate ROS 2 perception, decision, and actuation nodes. The contributions are the modular architecture, explicit documentation of simulation-to-physical interface correspondence, formalization of the rule-based controller through pseudocode and a flowchart, and identification of practical sim-to-real limitations. The work is therefore positioned as a reproducible low-cost architectural baseline rather than a claim of superior navigation performance. Because repeated timestamped trials were not recorded, the evaluation is limited to functional validation. Nevertheless, the architecture provides a practical foundation for future quantitative studies and progressive integration of stateful recovery, odometry, sensor fusion, SLAM, and the ROS 2 navigation stack (Nav2).

2. PROPOSED SYSTEM ARCHITECTURE

2.1. Overall perception-decision-actuation architecture

The proposed system consists of a physical platform, a Gazebo simulation platform, and a shared ROS 2 decision layer. On the physical robot, three HC-SR04 ultrasonic sensors provide front, left, and right distance inputs to the ultrasonic_node. The obstacle_avoidance node applies the deterministic decision rules and publishes linear and angular velocity commands through the /cmd_vel topic. The motor_controller_node converts these commands into GPIO and pulse-width modulation signals for the L298N motor driver and the two DC motors.

In simulation, Gazebo sensor plugins provide directional range data to the same decision process. The resulting /cmd_vel message is executed by the simulated differential-drive mechanism. Thus, the low-level sensing and actuation interfaces differ between platforms, whereas the perception-to-decision-to-command sequence remains conceptually equivalent. The revised vector diagram in Figure 1 makes this separation explicit and replaces the low-resolution raster architecture figure in the original manuscript.

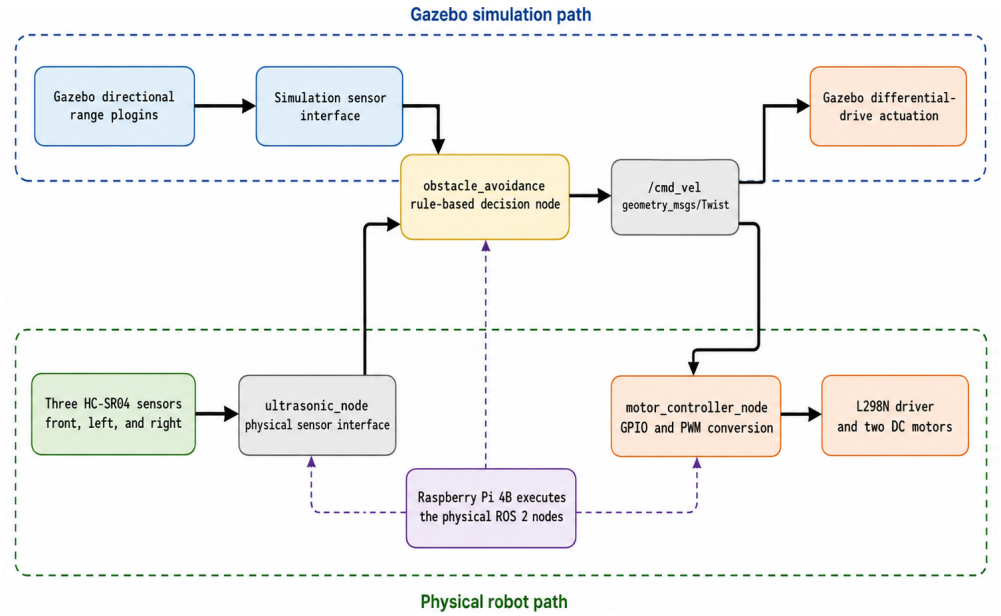


Figure 1. Shared ROS 2 architecture for simulation and physical robot control

The physical hardware uses a Raspberry Pi 4B as the central controller [18], [23]-[26]. Three HC-SR04 ultrasonic sensors are mounted in front, left, and right orientations to provide sparse local obstacle information [20]-[22]. The drive subsystem comprises two DC motors controlled by an L298N dual H-bridge driver [27]-[28]. Power is supplied by a 5200 mAh lithium-polymer battery, with XL4015 and LM2596 step-down converters used for voltage regulation.

2.2. ROS 2 nodes and topic communication

The software is organized into three primary ROS 2 nodes. The ultrasonic_node reads the physical distance sensors and publishes directional range data, the obstacle_avoidance node applies the movement rules and publishes geometry_msgs/Twist commands through /cmd_vel, and the motor_controller_node converts the resulting linear.x and angular.z values into motor direction and speed-control signals. The Gazebo simulation follows the same high-level communication sequence, although sensor data are generated by simulation plugins and velocity commands are executed through a differential-drive plugin. The launch configuration initializes the required nodes and visualization components, with RViz used for inspecting pose, sensor, and topic information and Gazebo representing the three-dimensional environment and robot motion.

Table 1 summarizes the functional correspondence between the simulation and physical platforms. Only parameters documented in the original development records are reported. Complete calibration data for wheel radius, wheel separation, motor-speed limits, sensor-update rates, ROS 2 middleware configuration, control-loop frequency, and simulated sensor-noise parameters were not available. Therefore, Gazebo is treated as a functionally corresponding simulation platform rather than a geometrically and temporally calibrated digital twin. The correspondence reported here is limited to the architecture, decision rules, and communication interfaces and should not be interpreted as evidence of equivalent sensing accuracy, motion dynamics, or timing behavior. Future controlled experiments will explicitly record these parameters to support quantitative reproducibility.

Table 1. Gazebo-to-physical robot correspondence

Aspect	Gazebo simulation	Physical robot	Correspondence and implication
Robot structure	URDF/XACRO model containing the chassis, two drive wheels, and three directional sensor links	Differential-drive chassis with two DC motors and three HC-SR04 sensors	The topology and component placement are represented in both platforms; exact calibration equivalence is not claimed.
Range sensing	Gazebo range or laser-style output represented as directional range data	Acoustic time-of-flight measurements from the HC-SR04 sensors	Both provide front, left, and right decision inputs, but their physical measurement processes differ.
Obstacle threshold	A front clearance threshold of approximately 0.25 m is evaluated by the controller	The same decision threshold is applied to the physical sensor input	The decision rule is shared at the controller level.
Velocity command	/cmd_vel controls the simulated differential-drive mechanism	/cmd_vel is converted into GPIO and PWM signals	The command interface is equivalent, whereas the actuator back end is platform-specific.
Environmental effects	Geometric and repeatable obstacle representation	Object material, orientation, floor friction, motor asymmetry, and battery condition affect behavior	Physical motion and sensing are less idealized than the simulation.

2.3. Rule-based obstacle-avoidance logic

The controller uses three directional distance values and a threshold of approximately 0.25 m. When the front direction is clear, the robot continues forward. When a frontal obstacle is detected, the robot stops and compares the left and right directions. It turns toward an available side; when all three directions are blocked, it reverses and then performs a left-turn recovery maneuver. This logic is deterministic and does not use a global map, obstacle history, or path optimization.

The implemented decision process is formalized in Algorithm 1. Because the controller evaluates a fixed number of three scalar inputs and a fixed sequence of conditional statements, its theoretical time complexity and auxiliary memory complexity are both $O(1)$. This analysis describes the algorithmic structure only; it does not constitute a measurement of processor utilization or communication latency on the Raspberry Pi 4B.

Algorithm 1 Rule-based obstacle-avoidance decision process

Require: Front distance d_f, left distance d_l, right distance d_r
Require: Obstacle threshold d_{th}
while the robot is active **do**
  Read d_f, d_l, and d_r
  if $d_f \geq d_{\text{th}}$ **then**
    Publish a forward command to /cmd_vel
  else
    Publish a stop command
    if $d_l > d_{\text{th}}$ and $d_l \geq d_r$ **then**
      Publish a left-turn command
    else if $d_r > d_{\text{th}}$ **then**
      Publish a right-turn command
    else
      Publish a reverse command
      Publish a left-turn recovery command
    end if
  end if
end while

Figure 2 provides a complementary flowchart for reproducibility. The figure also clarifies why a dead-end condition can result in unstable direction changes: the current logic makes a new decision from the immediate measurements and does not retain a persistent recovery state.

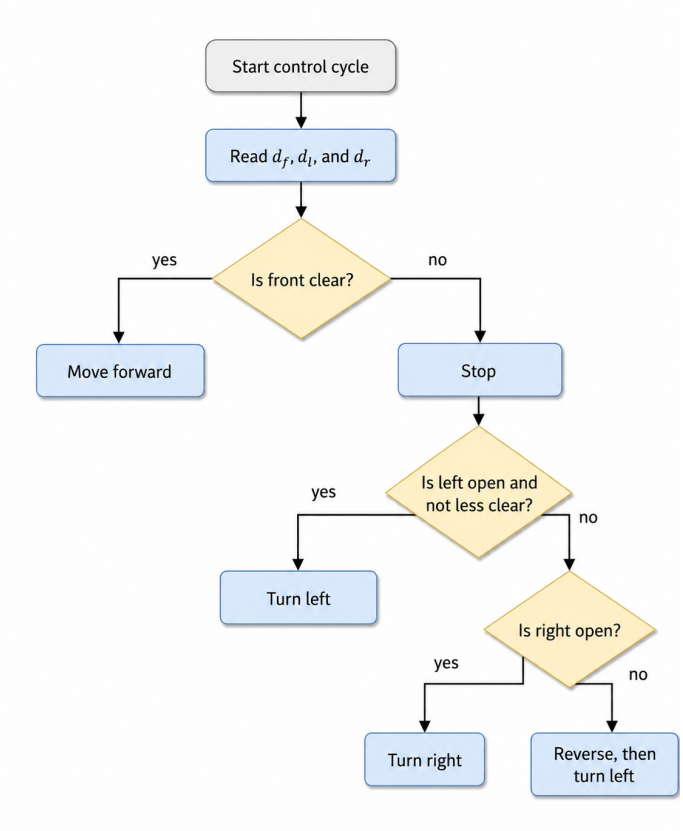


Figure 2. Flowchart of the implemented rule-based obstacle-avoidance

3. METHOD

3.1. Gazebo robot and environment modeling

The simulated robot was modeled using the unified robot description format with XACRO macros, where the robot.xacro file defines the chassis, two drive wheels, component placement, collision and visual

geometry, mass, and inertia, while a separate Gazebo configuration specifies the differential-drive mechanism, joint-state publication, and directional range sensing. The robot model is shown in Figure 3. The simulated sensors use range- or laser-style geometric measurements to reproduce the decision-level behavior of the HC-SR04 sensors; therefore, the output is treated as a functional approximation rather than a precise acoustic model because simulated ray intersections are more deterministic than physical ultrasonic echoes affected by target geometry, surface orientation, material, and environmental interference. The test environment contains representative indoor obstacles such as walls and tables, with Gazebo providing the three-dimensional physics simulation and RViz displaying the robot frame, sensor representation, and movement information. Figure 4 illustrates the directional sensing sectors used during simulation.

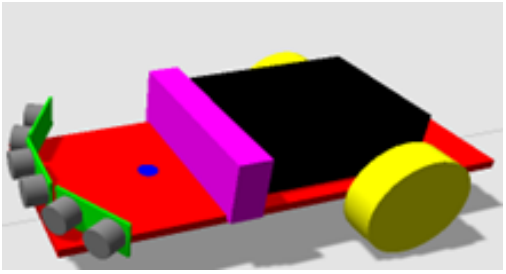


Figure 3. Differential-drive robot model in Gazebo

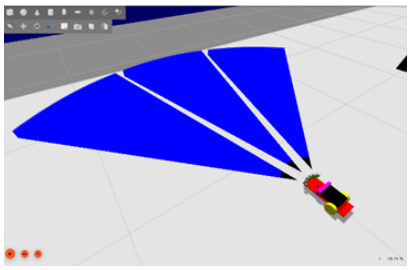


Figure 4. Gazebo robot and directional sensing sectors

3.2. Physical robot implementation

The physical implementation uses the same high-level obstacle-avoidance logic. The HC-SR04 sensors are connected to the Raspberry Pi 4B through GPIO interfaces. Their distance values are processed by the sensing and decision nodes, after which the motor-control node translates the velocity command into signals for the L298N driver. The physical robot therefore preserves the ROS 2 node separation used in the proposed architecture, although the low-level sensor and actuator operations differ from those in Gazebo.

No map, wheel-encoder feedback, inertial fusion, or global planner is used in the current system. The robot acts only on the current front, left, and right distance measurements. This design keeps the controller transparent and computationally simple but limits its ability to reason about previously visited locations, repeated obstacles, and complex dead ends.

3.3. Functional test configurations

The experimental evaluation was structured as a pilot proof-of-concept functional-validation study to verify the correspondence among obstacle perception, controller decisions, and robot motion before a subsequent controlled quantitative experiment. Four representative static-obstacle configurations were evaluated: frontal, left, right, and dead-end obstacles, with the expected robot response determined from Algorithm 1. Simulation behavior was inspected through Gazebo, RViz, and ROS 2 topic or terminal outputs, while the physical robot was observed to determine whether each detected obstacle configuration produced the motion prescribed by the same high-level controller. The primary endpoint was categorical correspondence among detected obstacle configuration, algorithmic decision, and executed motion, while secondary observations identified practical limitations such as ultrasonic blind regions, measurement fluctuations, actuator-related motion differences, and unstable recovery behavior.

The current validation should not be interpreted as a one-group pretest-posttest experiment because no independently measured baseline condition without the obstacle-avoidance controller or repeated time-synchronized trials was recorded during the original experimental campaign. Consequently, the available observations do not support statistical estimation of navigation reliability or retrospective reporting of success rate, response-time distribution, trajectory deviation, processor utilization, ROS 2 communication latency, or energy consumption. This pilot framework therefore establishes only the functional behavior and architectural correspondence required before a more rigorous quantitative evaluation. A subsequent experimental stage will use repeated trials, synchronized ROS 2 timestamps, wheel odometry, and defined baseline conditions to enable statistical comparison of obstacle-response time, successful-maneuver percentage, trajectory deviation, and embedded-platform performance.

4. RESULTS AND DISCUSSION

4.1. Simulation behavior

The Gazebo tests showed that the robot received directional range information and changed its motion according to the implemented rules. When the front distance remained above the obstacle threshold, the robot continued forward; when the threshold was crossed, the forward command stopped and the controller selected a turning direction based on the left and right measurements. If all directions were blocked, the controller issued the predefined reverse-and-turn recovery sequence. The representative terminal output in Figure 5 shows right-obstacle detection together with the corresponding directional distance values, confirming the functional relation between sensor conditions and controller decisions.

```
[lidar_cpp-6] OBSTACLE RIGHT! | right: 0.348345 center: 0.406799 left: 1.80192
[lidar_cpp-6] OBSTACLE RIGHT! | right: 0.348349 center: 0.407376 left: 1.80191
[lidar_cpp-6] OBSTACLE RIGHT! | right: 0.348444 center: 0.407957 left: 1.80085
[lidar_cpp-6] OBSTACLE RIGHT! | right: 0.348559 center: 0.408539 left: 1.79979
[lidar_cpp-6] OBSTACLE RIGHT! | right: 0.348561 center: 0.409122 left: 1.79874
[lidar_cpp-6] OBSTACLE RIGHT! | right: 0.348565 center: 0.409707 left: 1.7981
[lidar_cpp-6] OBSTACLE RIGHT! | right: 0.348569 center: 0.410294 left: 1.79966
[lidar_cpp-6] OBSTACLE RIGHT! | right: 0.348644 center: 0.410883 left: 1.80123
[lidar_cpp-6] OBSTACLE RIGHT! | right: 0.348792 center: 0.411474 left: 1.80245
[lidar_cpp-6] OBSTACLE RIGHT! | right: 0.348795 center: 0.412067 left: 1.80138
[lidar_cpp-6] OBSTACLE RIGHT! | right: 0.3488 center: 0.412663 left: 1.80032
[lidar_cpp-6] OBSTACLE RIGHT! | right: 0.348807 center: 0.413262 left: 1.79926
[lidar_cpp-6] OBSTACLE RIGHT! | right: 0.348864 center: 0.413862 left: 1.79821
[lidar_cpp-6] OBSTACLE RIGHT! | right: 0.349022 center: 0.41447 left: 1.79877
[lidar_cpp-6] OBSTACLE RIGHT! | right: 0.349053 center: 0.415072 left: 1.80033
[lidar_cpp-6] OBSTACLE RIGHT! | right: 0.349057 center: 0.415677 left: 1.8019
[lidar_cpp-6] OBSTACLE RIGHT! | right: 0.349067 center: 0.416288 left: 1.80172
[lidar_cpp-6] OBSTACLE RIGHT! | right: 0.349133 center: 0.416909 left: 1.80065
```

Figure 5. ROS 2 output for right-obstacle detection

The simulation was useful for validating the robot structure, sensor orientation, ROS 2 topic connectivity, and controller behavior before physical deployment. However, the terminal output does not provide a time-aligned latency measurement because synchronized timestamps for sensor reception and command publication were not recorded. Likewise, the repeatability observed in Gazebo should not be interpreted as evidence of equivalent HC-SR04 sensing precision or identical simulated and physical trajectories.

4.2. Physical robot behavior

Table 2 summarizes the observed physical behavior in the four representative configurations. The table reports correspondence with the programmed rules and intentionally replaces the binary "Success" table in the original manuscript. The observations demonstrate that the perception-to-actuation sequence operated as intended in the reported configurations, but they do not constitute a repeated-trial reliability estimate.

Table 2. Physical robot responses to static obstacles

No.	Obstacle configuration	Expected control rule	Observed robot response	Qualitative observation
1	Front direction blocked	Stop forward motion and turn left	The robot stopped and initiated a left-turn maneuver	The response corresponded to the programmed frontal-obstacle rule in the controlled configuration.
2	Left direction blocked	Turn right	The robot changed direction toward the right side	The local maneuver avoided the detected left-side obstacle without using global path information.
3	Right direction blocked	Turn left	The robot changed direction toward the left side	The response corresponded to the predefined right-obstacle rule.
4	Front, left, and right directions blocked	Reverse and perform a left-turn recovery maneuver	The robot reversed and attempted to turn away from the blocked region	Similar side measurements could produce repeated direction decisions and unstable movement in a dead end.

Figure 6 schematically illustrates the expected motion rules for the four functional obstacle configurations, while Figure 7 documents the corresponding physical robot test arrangements and movement responses. The schematic paths represent controller behavior rather than measured trajectories and are therefore not used to

calculate trajectory error or path efficiency. Physical testing revealed two main limitations: the sparse directional coverage of the three ultrasonic sensors created blind regions in which small or narrow objects, such as chair legs, could remain undetected, consistent with the directional and reflection-dependent characteristics of low-cost ultrasonic ranging [20–22]; additionally, unstable direction changes occurred in dead-end conditions because the deterministic controller repeatedly relied on current measurements without maintaining a persistent recovery state, obstacle history, or environmental memory. These observations highlight practical limitations that were less apparent in the idealized simulation environment.

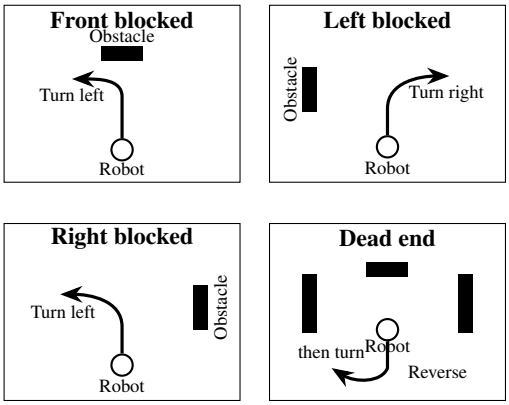


Figure 6. Expected maneuvers for four obstacle configurations; paths are schematic, not measured trajectories

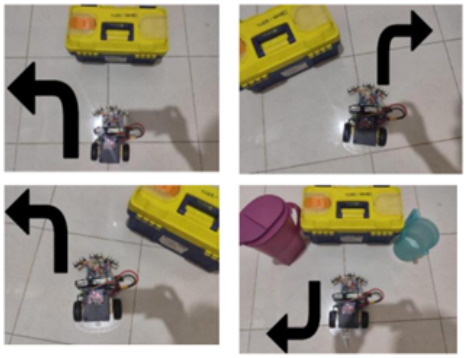


Figure 7. Physical robot and static-obstacle test arrangements

4.3. Simulation-to-physical sensing and motion differences

The simulation and physical robot use directional distance information for the same decision-level purpose, but their sensing mechanisms differ substantially. Gazebo obtains range information from geometric intersections between simulated rays and virtual objects, producing comparatively deterministic measurements. In contrast, the HC-SR04 relies on ultrasonic time-of-flight measurements that are influenced by acoustic reflection, target orientation, object geometry, surface characteristics, sensor alignment, and environmental conditions [20–22]. The three physical sensors also provide only sparse angular coverage, creating blind regions in which narrow objects may remain undetected. Therefore, the Gazebo output should be interpreted as a functional decision-interface approximation rather than a high-fidelity acoustic representation of the physical sensors.

Motion behavior also differs between platforms. Gazebo executes /cmd_vel commands through an idealized differential-drive model, whereas the physical robot is additionally affected by motor asymmetry, wheel slip, floor friction, mechanical alignment, battery condition, and nonlinear PWM-to-wheel-speed characteristics. Consequently, identical high-level commands do not necessarily produce identical trajectories. The present system therefore demonstrates functional correspondence of the ROS 2 decision architecture rather than calibrated simulation-to-real equivalence. Because synchronized poses, wheel-encoder measurements, and

timestamped sensor command pairs were not collected, trajectory error and perception-to-actuation latency are not reported and are reserved for future quantitative evaluation.

4.4. Comparison with alternative navigation approaches

Table 3 positions the implemented controller relative to common mobile-robot navigation approaches. The comparison is conceptual rather than experimental because the alternative algorithms were not implemented as baselines in the present study. The proposed controller is best understood as a low-complexity local reactive layer. It is explainable and compatible with inexpensive directional sensors, but it cannot perform global route optimization or systematic recovery from complex environments.

Table 3. Comparison with common mobile-robot navigation methods

Approach	Map required	Relative complexity	Main advantage	Main limitation
Implemented rule-based controller	No	Low	Simple, explainable, and compatible with three low-cost ultrasonic sensors	Sparse local perception, no environmental memory, and possible oscillation in dead ends.
Artificial potential field	No or partial	Low–medium	Produces continuous reactive motion from attractive and repulsive effects	Can become trapped in local minima and generally benefits from richer obstacle information.
Reactive vector-based control	No	Medium	Can produce smoother steering from a local range distribution	Commonly requires denser sensing than three directional ultrasonic measurements.
SLAM with a navigation stack	Yes	Medium–high	Supports mapping, localization, global planning, and structured recovery	Requires additional sensing, computation, configuration, and map management.
Learning-based navigation	Not necessarily	High during training	Can represent nonlinear and adaptive behavior	Requires training data and introduces validation, generalization, and simulation-to-real concerns.

Compared with SLAM- and navigation-stack-based systems, the implemented method requires neither map storage nor global localization. Compared with learning-based navigation, it does not require training data or neural-network inference. Its limitations are equally clear: it makes decisions only from the current three distances, and its fixed rules cannot guarantee an efficient or globally safe path. The architectural value of the method lies in modularity, transparency, and use as a foundation or emergency local layer rather than superiority over more advanced approaches.

4.5. Practical and educational implications

The proposed architecture has practical value as a low-cost experimental platform because sensing, decision-making, and actuation are separated into independent ROS 2 nodes. This organization allows individual components to be replaced or extended without redesigning the complete robotic system. For example, the current ultrasonic sensing layer can subsequently be complemented or replaced by LiDAR, vision, or fused sensing, while the high-level topic interfaces and actuator layer can remain available for further experimentation. The present configuration is not intended for safety-critical or unstructured industrial deployment; instead, it represents an initial architecture from which more robust service-robot or laboratory-automation prototypes can be developed.

The same modularity also provides educational value. Students can initially examine robot geometry, sensor representation, topic communication, and controller decisions in Gazebo before applying the corresponding high-level logic to a physical robot. This simulation-to-hardware progression makes it possible to study the practical differences between idealized sensor models and real measurements while preserving a common ROS 2 software structure. Individual learning activities can therefore focus separately on ROS 2 communication, sensor interfaces, embedded motor control, reactive navigation, and simulation-to-physical validation.

The architecture can subsequently support progressively more advanced robotics topics. Additional ultrasonic sensors can first be introduced to investigate sensing coverage, followed by wheel encoders and inertial sensing for odometry and sensor fusion. A two-dimensional LiDAR can then support simultaneous

localization and mapping (SLAM), after which the ROS 2 Navigation Stack (Nav2) can be introduced for localization, global planning, local planning, and structured recovery. In this sense, the current system provides both a practical prototyping baseline and a pedagogical bridge between fundamental mobile-robot control and advanced autonomous navigation research.

4.6. Limitations and future development

The present study is limited to functional validation under controlled static-obstacle configurations without repeated quantitative measurements. Therefore, statistical success rate, obstacle-response time, trajectory accuracy, processor performance, communication latency, and energy efficiency are not reported. Several implementation parameters, including ROS 2 and data distribution service (DDS) settings, node publication rates, control-loop frequency, calibration parameters, and simulated sensor-noise characteristics, were also not systematically recorded. In addition, the three ultrasonic sensors provide sparse directional coverage with blind regions, while the controller lacks a global map, motion history, and persistent recovery state. These limitations, together with idealized Gazebo sensing and motion models, restrict quantitative simulation-to-physical equivalence and robustness in complex or dynamic environments.

Future development will focus on improving sensing, recovery behavior, and quantitative evaluation. Two additional ultrasonic sensors and a hysteresis-based persistent recovery state will be introduced to reduce blind regions and dead-end oscillation, followed by wheel encoders and an inertial measurement unit (IMU) for improved motion estimation and sensor fusion. A two-dimensional LiDAR, simultaneous localization and mapping (SLAM) using slam_toolbox, and the ROS 2 navigation stack (Nav2) will subsequently support mapping, localization, planning, and structured recovery. Future experiments will use repeated trials and synchronized ROS 2 logs to quantify obstacle-response time, maneuver success rate, trajectory deviation, communication latency, computational load, and energy consumption under static, narrow-corridor, dense-obstacle, and dynamic-obstacle conditions.

5. CONCLUSION

This study presented a modular simulation-to-physical implementation of a wheeled robot obstacle-avoidance system using ROS 2, Gazebo, three HC-SR04 ultrasonic sensors, and a Raspberry Pi 4B. The architecture separates perception, decision-making, and actuation into interconnected ROS 2 nodes, enabling the same rule-based controller to operate in simulation and on the physical robot while preserving platform-specific sensing and actuation mechanisms. Functional validation across four static-obstacle configurations showed that the robot responses corresponded to the predefined forward, turning, reversing, and recovery rules. Physical implementation also revealed limitations that were less apparent in simulation, particularly ultrasonic blind regions, measurement fluctuations, actuator and surface effects, and unstable recovery in dead-end conditions. Because repeated quantitative measurements and synchronized trajectory records were not collected, the results are interpreted as proof-of-concept functional validation rather than statistical evidence of navigation reliability or calibrated simulation-to-real performance.

The main contribution of this work is a reproducible low-cost ROS 2 architectural baseline that connects simulation-based development with physical embedded implementation while explicitly documenting its practical limitations. Its modular structure allows sensing, navigation, and actuation components to be progressively upgraded and provides a useful platform for both robotics education and further research. Future development will extend the system through additional sensing, stateful recovery, odometry, sensor fusion, simultaneous localization and mapping (SLAM), and Nav2-based navigation, supported by repeated quantitative experiments in more complex and dynamic environments.

FUNDING INFORMATION

This research was carried out with the support of the 2023 Community Service Grant from the Faculty of Engineering, Universitas Sumatera Utara.

AUTHOR CONTRIBUTIONS STATEMENT

This journal uses the Contributor Roles Taxonomy (CRediT) to recognize individual author contributions, reduce authorship disputes, and facilitate collaboration.

Name of Author	C	M	So	Va	Fo	I	R	D	O	E	Vi	Su	P	Fu
Tigor Hamonangan Nasution	✓	✓	✓	✓	✓	✓		✓	✓	✓		✓	✓	
Soeharwinto		✓				✓		✓	✓	✓	✓	✓		
Lucky Sam Pratama Telaumbanua		✓				✓		✓	✓	✓	✓			
Stephany		✓				✓		✓	✓	✓	✓			
Fahmi Fahmi	✓		✓	✓		✓			✓		✓	✓	✓	✓

C : **C**onceptualization
M : **M**ethodology
So : **So**ftware
Va : **Va**lidation
Fo : **Fo**rmal Analysis
I : **I**nvestigation
R : **R**esources
D : **D**ata Curation
O : Writing - **O**riginal Draft
E : Writing - Review & **E**diting
Vi : **Vi**sualization
Su : **Su**pervision
P : **P**roject Administration
Fu : **Fu**nding Acquisition

CONFLICT OF INTEREST STATEMENT

Authors state no conflict of interest.

DATA AVAILABILITY

The ROS 2 configuration files, experimental documentation, and supporting materials associated with this study are available from the corresponding author upon reasonable request. No human-participant data were collected in this research.

REFERENCES

[1] T. H. Nasution, A. M. Prihandoyo, and Seniman, "The implementation of the k-nearest neighbor algorithm to detect the KRSRI robot obstacles," *Bulletin of Electrical Engineering and Informatics*, vol. 14, no. 2, pp. 1330–1342, Apr. 2025, doi: 10.11591/EEI.V14I2.8225.

[2] G. Bathla *et al.*, "Autonomous vehicles and intelligent automation: Applications, challenges, and opportunities," *Mobile Information Systems*, vol. 2022, no. 1, p. 7632892, Jan. 2022, doi: 10.1155/2022/7632892.

[3] J. T. Licardo, M. Domjan, and T. Orehovački, "Intelligent robotics-a systematic review of emerging technologies and trends," *Electronics*, vol. 13, no. 3, p. 542, Jan. 2024, doi: 10.3390/electronics13030542.

[4] S. H. Chung, Q. Zhang, H. L. Ma, X. Wen, and Y. Sun, "Impacts of heterogeneous autonomous mobile robots (AMRs) on smart manufacturing," *International Journal of Production Economics*, vol. 296, Art. no. 109759, Jun. 2026, doi: 10.1016/j.ijpe.2025.109759.

[5] W. Y. H. Adoni, J. S. Fareedh, S. Lorenz, R. Gloaguen, and T. D. Kuhne, "Autotarget: A distributed robot operating system framework for autonomous aerial swarms," in *2024 21st International Conference on Ubiquitous Robots*, 2024, pp. 153–160, doi: 10.1109/UR61395.2024.10597509.

[6] A. Podlubne and D. Gohringer, "A survey on adaptive computing in robotics: Modelling, methods and applications," *IEEE Access*, vol. 11, pp. 53830–53849, 2023, doi: 10.1109/ACCESS.2023.3281190.

[7] F. Yumbla *et al.*, "An open-source multi-robot framework system for collaborative environments based on ROS2," *IEEE Access*, vol. 13, pp. 16288–16302, 2025, doi: 10.1109/ACCESS.2025.3530391.

[8] S. Macenski, T. Foote, B. Gerkey, C. Lalancette, and W. Woodall, "Robot operating system 2: Design, architecture, and uses in the wild," *Science Robotics*, vol. 7, no. 66, May 2022, doi: 10.1126/scirobotics.abm6074.

[9] Y. Ye, Z. Nie, X. Liu, F. Xie, Z. Li, and P. Li, "ROS2 real-time performance optimization and evaluation," *Chinese Journal of Mechanical Engineering*, vol. 36, no. 1, pp. 1–15, Dec. 2023, doi: 10.1186/s10033-023-00976-5.

[10] J. Henle, M. Stoffel, M. Schindewolf, A. T. Nagele, and E. Sax, "Architecture platforms for future vehicles: a comparison of ROS2 and Adaptive AUTOSAR," in *2022 IEEE 25th International Conference on Intelligent Transportation Systems (ITSC)*, pp. 3095–3102, 2022, doi: 10.1109/ITSC55140.2022.9921894.

[11] Y. Tang, X. Jiang, N. Guan, X. Luo, M. Yang, and W. Yi, "Timing analysis of processing chains with data refreshing in ROS 2," *Journal of Systems Architecture*, vol. 155, p. 103259, Oct. 2024, doi: 10.1016/j.sysarc.2024.103259.

[12] H. Yadavari, V. T. Aghaei, and S. Ikizoglu, "Deep reinforcement learning-based control of stewart platform with parametric simulation in ROS and Gazebo," *Journal of Mechanisms and Robotics*, vol. 15, no. 3, Jun. 2023, doi: 10.1115/1.4056971.

[13] J. Collins, S. Chand, A. Vanderkop, and D. Howard, "A review of physics simulators for robotic applications," *IEEE Access*, vol. 9, pp. 51416–51431, 2021, doi: 10.1109/ACCESS.2021.3068769.

[14] M. Marian, F. Stinga, M. T. Georgescu, H. Roibu, D. Popescu, and F. Manta, "A ROS-based control application for a robotic platform using the gazebo 3D simulator," in *2020 21th International Carpathian Control Conference (ICCC)*, 2020, doi: 10.1109/ICCC49264.2020.9257256.

[15] J. Platt and K. Ricks, "Comparative analysis of ROS-Unity3D and ROS-gazebo for mobile ground robot simulation," *Journal of Intelligent and Robotic Systems*, vol. 106, no. 4, pp. 1–23, Dec. 2022, doi: 10.1007/s10846-022-01766-2.

[16] M. Carroll and N. Koenig, "Advancing robotic simulation with the Robotic Operating System and Gazebo," *Computing in Science & Engineering*, vol. 27, no. 4, pp. 52–59, Oct.–Dec. 2025, doi: 10.1109/MCSE.2025.3630762.

[17] Z. Fan, L. Zhang, X. Wang, Y. Shen, et al., "LiDAR, IMU, and camera fusion for simultaneous localization and mapping: A systematic review," *Artificial Intelligence Review*, vol. 58, Art. no. 174, 2025, doi: 10.1007/s10462-025-11187-w.
[18] I. U. Khan and D. R. K. Raja, "IntelliDrive autonomous robot powered by large language model," *IAES International Journal of Robotics and Automation (IJRA)*, vol. 14, no. 3, pp. 339–347, Sep. 2025, doi: 10.11591/ijra.v14i3.pp339-347.
[19] H. Sekkat, O. Moutik, B. El Kari, Y. Chaibi, T. Ait Tchakoucht, and A. El Hilali Alaoui, "Beyond simulation: Unlocking the frontiers of humanoid robot capability and intelligence with Pepper's open-source digital twin," *Heliyon*, vol. 10, no. 14, p. e34456, Jul. 2024, doi: 10.1016/j.heliyon.2024.e34456.
[20] V. N. Sulistyawan, N. A. Salim, F. G. Abas, and N. Aulia, "Parking tracking system using ultrasonic sensor HC-SR04 and NodeMCU ESP8266 based IoT," *IOP Conference Series: Earth and Environmental Science*, vol. 1203, no. 1, p. 012028, Jun. 2023, doi: 10.1088/1755-1315/1203/1/012028.
[21] S. Mexias, V. Fotopoulos, and K. Giannakopoulos, "Comparison of HC-SR04 and TF-LC02 distance sensors for indoor mapping applications," in *2024 Panhellenic Conference on Electronics Telecommunications (PACET)*, 2024, doi: 10.1109/PACET60398.2024.10497028.
[22] U. Jain, V. Kansal, R. Dewangan, G. Dhasmana, and A. Kotiyal, "Performance comparison of HC-SR04 ultrasonic sensor and TF-Luna LIDAR for obstacle detection," *Lecture Notes in Networks and Systems*, vol. 586, pp. 631–640, 2023, doi: 10.1007/978-981-19-7867-8_50.
[23] S. Ajith Kumar, J. S. Kumar, and S. C. Bharadwaj Mahabaleswara, "Efficient diabetic retinopathy detection using deep learning approaches and Raspberry Pi 4," *Bulletin of Electrical Engineering and Informatics*, vol. 14, no. 2, pp. 1063–1072, Apr. 2025, doi: 10.11591/eei.v14i2.8248.
[24] S. Laki, R. Stoyanov, D. Kis, R. Soulé, P. Vörös, and N. Zilberman, "P4Pi: P4 on Raspberry Pi for networking education," *Computer Communication Review*, vol. 51, no. 3, pp. 17–21, Jul. 2021, doi: 10.1145/3477482.3477486.
[25] P. D. P. Adi *et al.*, "Optimization and development of Raspberry Pi 4 model B for the Internet of Things," in *2023 IEEE 9th Information Technology International Seminar (ITIS)*, 2023, doi: 10.1109/ITIS59651.2023.10420261.
[26] K. Kumar *et al.*, "Soft computing based solar follower using Raspberry Pi 4B," *AIP Conference Proceedings*, vol. 2816, no. 1, Mar. 2024, doi: 10.1063/5.0177498.
[27] D. D. Abuan *et al.*, "Exploring variable speed control using leap motion: An integration finger gestures, L298N, and NodeMCU ESP8266 for wireless communication," in *2023 IEEE 15th International Conference on Humanoid, Nanotechnology, Information Technology, Communication and Control, Environment, and Management (HNICEM)*, 2023, doi: 10.1109/HNICEM60674.2023.10589246.
[28] Azhari, T. I. Nasution, and P. F. A. Azis, "MPU-6050 wheeled robot controlled hand gesture using L298N driver based on Arduino," *Journal of Physics: Conference Series*, vol. 2421, no. 1, p. 012022, Jan. 2023, doi: 10.1088/1742-6596/2421/1/012022.

BIOGRAPHIES OF AUTHORS



Tigor Hamonangan Nasution is an associate professor in the Department of Electrical Engineering, Universitas Sumatera Utara, Indonesia. He received his master's degree in electrical engineering and his doctoral degree in computer science from Universitas Sumatera Utara, completing the doctoral degree in 2025. He currently serves as Director of the Directorate of Digitalization of Learning at Universitas Sumatera Utara and is a Senior Member of IEEE. His research interests include embedded systems, robotics, the Internet of Things, artificial intelligence, and digital learning technologies. He can be contacted at: tigor.nasution@usu.ac.id.



Soeharwinto is a lecturer in the Department of Electrical Engineering, Universitas Sumatera Utara, Indonesia. He received his master's degree in electrical engineering from Institut Teknologi Bandung (ITB), Indonesia, in 1998. His research activities include embedded systems, the Internet of Things, wireless sensor networks, automation, and electrical engineering applications. He has contributed to research and publications involving sensor-based monitoring, embedded controllers, wireless systems, and intelligent engineering applications. He can be contacted at: soeharwinto@usu.ac.id.



Lucky Sam Pratama Telaumbanua is an undergraduate student in the Department of Electrical Engineering, Universitas Sumatera Utara. He is involved in student robotics activities and projects related to robotics and automation. He can be contacted at: sptelaumbanua@gmail.com.



Stephany is an undergraduate student in the Department of Electrical Engineering, Universitas Sumatera Utara, Medan, Indonesia. Her academic interests include electrical engineering, robotics, and related technologies. She can be contacted at: stephany021203@gmail.com.



Fahmi Fahmi is a professor in the Department of Electrical Engineering, Universitas Sumatera Utara, Indonesia, and currently serves as Dean of the Faculty of Engineering. He received his master's degree in Sensor System Technology from FH Karlsruhe, Germany, in 2005 and his doctoral degree in Biomedical Engineering from Universiteit van Amsterdam, the Netherlands, in 2015. His research interests include biomedical engineering, embedded and smart systems, artificial intelligence, the Internet of Things, and digital technology. He has been actively involved in multidisciplinary research integrating electrical engineering, biomedical applications, intelligent systems, and digital technologies. He can be contacted at: fahmimn@usu.ac.id