Evaluation of LSRB Pathfinding Performance in an Autonomous Obstacle-Avoiding Robot

Usran[1], **Muh. Rafly Rasyid**[2], **Wawan Firgiawan***[3]
Universitas Sulawesi Barat, Majene, Sulawesi Barat, Indonesia
E-mail : usran@unsulbar.ac.id[1], muh.rafly@unsulbar.ac.id[2], wawanfirgiawan@unsulbar.ac.id[3]*
**Corresponding author*

Received 10 April 2025; Revised 24 April 2025; Accepted 28 April 2025

Abstract - This study presents an evaluation of the performance of the Left-Straight-Right-Back (LSRB) algorithm implemented in an autonomous obstacle-avoiding robot. The LSRB algorithm operates based on a fixed priority rule in pathfinding decisions: turn left, go straight, turn right, and finally perform a 180-degree turn if no paths are available. The robot is equipped with ultrasonic sensors and a servo motor to scan obstacles on the left and right sides, and utilizes an 8×8 dot matrix display to indicate its navigation status. Testing was conducted in a custom-built maze environment featuring branches, dead ends, and narrow paths to simulate real-world navigation scenarios. Performance evaluation parameters include travel time, number of maneuvers, and path accuracy. Experimental results show that the LSRB algorithm achieved 100% path completion accuracy across all test cases, with consistent travel time and efficient obstacle avoidance. The findings demonstrate that LSRB is a reliable and lightweight navigation strategy, particularly suitable for low-cost, microcontroller-based robots used in educational or semi-structured environments. Limitations regarding power supply stability and the absence of memory-based path tracking are also identified, offering opportunities for future improvements.

Keywords - Obstacle-Avoiding Robot, Robot Navigation, LSRB Algorithm, Ultrasonic Sensor

1. INTRODUCTION

The development of robotics technology over the past few decades has demonstrated significant progress, particularly in the field of autonomous mobile robots. These types of robots possess the ability to move and navigate independently without direct human control and can adapt to changing environmental conditions [1], [2]. One of the primary challenges in the development of mobile robots is the ability to navigate effectively, especially when encountering obstacles in real time [3].

Obstacle-avoiding robots are among the most important applications in the field of robotics, utilized in various sectors such as hazardous area exploration, driverless vehicles, and automated logistics systems [4], [5]. These robots are typically equipped with sensors such as ultrasonic [6], infrared [7], or cameras [8], which are used to detect surrounding objects and determine safe directions for movement. The success of navigation is not solely determined by the accuracy of the sensors but also by the efficiency of the decision-making algorithms that process sensory information.

Various algorithms have been developed to support robot navigation systems, ranging from simple approaches like wall-following and Dijkstra [4], [8], to more complex algorithms such as genetic algorithms and approaches based on neural networks [10]. While these advanced algorithms offer high precision and optimization, their implementation often requires powerful hardware and complete environmental mapping, which may not be feasible for simple or low-cost robotic systems.

As an alternative, rule-based navigation algorithms such as LSRB (Left-Straight-Right-Back) offer a lighter and more practical approach. The LSRB algorithm operates based on a priority sequence in determining movement direction: turn left, go straight, turn right, and finally perform a 180-degree turn if all directions are blocked [11]. This strategy allows the robot to make localized decisions without relying on a global map, making it highly suitable for unknown environments or systems with limited resources [12].

The main advantage of the LSRB algorithm lies in its simplicity, which facilitates integration with microcontrollers such as Arduino. This approach is particularly suitable for robots designed for basic exploration or navigation in confined spaces without digital maps. Moreover, the algorithm supports fast and responsive decision-making by relying directly on real-time sensor readings.

Several previous studies have explored various approaches in the development of obstacle-avoiding and line-follower robots. For instance, Andi et al. (2020) developed a line-following robot using PID control for stable navigation through complex paths [13]. Another study by Nanditta et al. (2021) implemented infrared sensors for simple navigation in obstacle-avoiding robots [7], while Izumi et al. (2006) evaluated the combination of ultrasonic sensors and cameras to support dynamic navigation [14]. Wahyudi et al. (2022) emphasized the efficiency of the wall-following algorithm in miniature robots for confined-space exploration [15]. These four studies serve as important references for understanding the position of the LSRB algorithm in the context of simple robot navigation.

This study aims to analyze the performance of the LSRB algorithm in an obstacle-avoiding robot navigation system. The algorithm is implemented in a microcontroller-based robot equipped with ultrasonic sensors for obstacle detection. Evaluation is conducted through testing in various artificial environments that simulate real-world conditions, such as mazes, branching paths, and randomly placed obstacles. The evaluation parameters include travel time, path efficiency, number of maneuvers, and the robot's ability to respond to obstacles effectively.

Compared to existing studies that implement basic wall-following or sensor-triggered obstacle avoidance algorithms, this research introduces a lightweight and fully autonomous navigation strategy using the LSRB (Left-Straight-Right-Back) method, which operates without requiring environmental mapping or memory storage. The novelty of this work lies in the integration of a complete real-time decision-making logic that activates all four directional priorities, tested in a structured maze scenario that reflects realistic exploration challenges. Unlike prior implementations, the proposed system combines ultrasonic scanning through servo actuation, real-time visual feedback using a dot matrix display, and a consistent set of performance metrics including path accuracy, maneuver count, and travel time, providing a more comprehensive assessment of algorithm effectiveness. This makes it particularly suitable for low-cost, microcontroller-based robotic platforms used in constrained or semi-structured environments, where simplicity and responsiveness are prioritized.

Through this research, it is expected that a clearer understanding of the effectiveness of the LSRB algorithm in robot navigation will be obtained, along with its advantages and limitations compared to other approaches. Furthermore, the results of this study may serve as a foundation for the development of simple, efficient, cost-effective, and easily implemented robot navigation systems for both educational and small-scale industrial applications.

2. RESEARCH METHOD

This study aims to analyze the performance of the Left-Straight-Right-Back (LSRB) algorithm in obstacle-avoiding robot navigation. The research method consists of system design, algorithm implementation, experimental testing in a controlled environment, and performance evaluation based on specific parameters.

An experimental approach was employed to ensure the algorithm could be effectively implemented on a real robot and tested across various environmental scenarios. Thus, the outcomes of this research can serve as a reference for the development of efficient and low-cost autonomous robot navigation systems.

2.1. Equipment and Software

To support the experiment, several hardware and software components were utilized, as summarized in the following table:

Table 1. Robot Components

Component	**Specification**
Microcontroller	Arduino Uno
Distance Sensor	3 x HC-SR04 Ultrasonic Sensor
Drive Motor	2 x Motor DC
Motor Driver	L298N
Servo Motor	SG90
Battery	Li-ion 7.4V
Display Module	8x8 Dot Matrix
Test Environment	Dinding karton / akrilik
Software	Arduino IDE

The table above outlines the components used in this study. The Arduino Uno microcontroller serves as the central control unit, while ultrasonic sensors are used to detect obstacles.

2.2. Robot System Design

The robot used in this study is a mobile robot based on the Arduino Uno microcontroller, equipped with ultrasonic sensors, an 8x8 dot matrix display module, and a servo motor for specific mechanical movements. This design allows the robot to detect obstacles, display navigation status on the dot matrix, and perform autonomous navigation. The core structure of the robot includes:

1) Arduino Uno microcontroller, serving as the processing and navigation control center.
2) HC-SR04 ultrasonic sensors, used to detect obstacles in front, left, and right of the robot.
3) Servo motor, used to drive certain mechanical components.
4) 8x8 dot matrix module, used to display status and navigation information.
5) 7.4V battery power supply, as the main power source.

To provide a clearer overview of the connections between components, the robot's electronic schematic is illustrated in Figure 1 below:

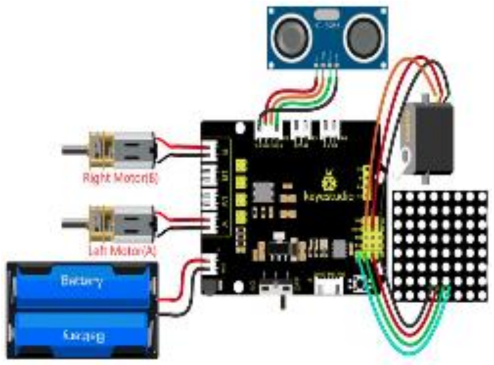

Figure 1. Electronic Schematic of the Obstacle-Avoiding Robot [16]

The diagram in Figure 1 shows the relationships between the microcontroller, ultrasonic sensors, dot matrix module, servo motor, and power supply. It facilitates understanding of data flow and power distribution within the robot system. For a deeper understanding of the schematic, Table 2 presents the detailed component connections.

Table 2. Obstacle-Avoiding Robot Component Connections

Component	**Connection Description**
8x8 Dot Matrix Module	GND and VCC connected to GND and 5V on the Arduino expansion board SDA and SCL connected to pins A4 and A5 of the expansion board
HC-SR04 Ultrasonic Sensor	VCC to 5V Trig to D12 Echo to D13 GND to GND
Servo Motor	GND to GND VCC to 5V Signal (orange wire) to D10
Power Supply	Battery connected to the BAT port of the expansion board

This robot design emphasizes energy efficiency and sensor responsiveness to ensure optimal navigation in various environments. Additionally, sensor placement has been tested to prevent interference caused by ultrasonic signal reflections, which could result in inaccurate detection.

2.3. LSRB Algorithm Implementation

The Left-Straight-Right-Back (LSRB) algorithm is a rule-based navigation strategy [12], [17]. Decisions are made based on readings from ultrasonic sensors in the following priority order:

1) Left: If the left path is clear, the robot turns left.
2) Straight: If the left is blocked but the front is clear, the robot continues straight.
3) Right: If both left and front are blocked but the right path is open, the robot turns right.
4) Back: If all directions are blocked, the robot performs a 180-degree turn to search for an alternative path.

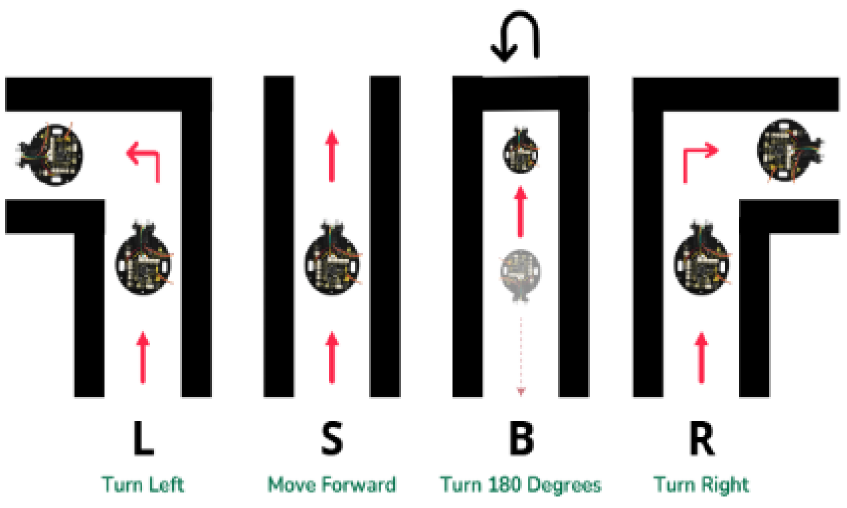

Figure 2. LSRB Algorithm Workflow

Navigation decisions are based on a distance threshold from the ultrasonic sensor (set at 10 cm). Sensor data is processed using the Arduino IDE and transmitted to the navigation system to be executed in real-time. This implementation also considers the possibility of measurement errors due to factors such as uneven surface reflections or interference from other ultrasonic sources. Therefore, a moving average filtering method is applied to ensure sensor readings are accurate before navigation decisions are made.

2.4. Performance Evaluation Parameters

To assess the effectiveness and efficiency of the Left-Straight-Right-Back (LSRB) algorithm in obstacle-avoiding robot navigation, a systematic and measurable evaluation approach is necessary. Performance evaluation is a key aspect of this study, enabling the researchers to understand how well the algorithm produces optimal navigation results, particularly in complex, dynamic, and unpredictable environments.

In robotic systems, performance is not solely evaluated based on whether the robot reaches its target, but also on how quickly, efficiently, and accurately it navigates. This study employs quantitative parameters to enable objective comparison and reproducibility in future research. The performance metrics used to evaluate the LSRB algorithm include:

1) Travel Time (seconds): The total time required to complete the path.
2) Number of Maneuvers: The total number of turns or direction corrections made.
3) Path Accuracy (%): The extent to which the robot reaches the goal without collisions or veering off course.

The data collected from testing is analyzed to evaluate the algorithm’s effectiveness. To increase validity, each scenario is repeated five times. The results are presented in the form of performance graphs that display path completion speed, number of maneuvers, and efficiency, providing a comprehensive overview of how well the algorithm performs under different navigation scenarios.

3. RESULTS AND DISCUSSION

This section presents the experimental results from implementing the LSRB algorithm in an obstacle-avoiding mobile robot. The testing phase was conducted across multiple environmental scenarios specifically designed to resemble a maze, incorporating a variety of directional options, dead ends, and obstacle placements. These scenarios were intended to rigorously assess the robot’s ability to detect and respond to obstacles in real-time, as well as to autonomously determine the most appropriate path based on the algorithm’s predefined decision logic.

By simulating diverse spatial configurations, the experiments aimed to evaluate not only the functional correctness of the LSRB algorithm but also its robustness, consistency, and adaptability when deployed in physical environments with constrained paths and varying levels of complexity. The findings from this section serve as a foundation for understanding how well the algorithm performs in practical applications, particularly in resource-constrained systems where real-time navigation and simplicity of control are essential.

3.1. Test Track Design

To comprehensively evaluate the effectiveness of the LSRB algorithm, the robot was tested in a variety of controlled environments, each designed to represent specific navigational challenges. These scenarios included right turns, left turns, straight paths without obstacles, and dead ends—each corresponding to one of the four decision-making branches of the LSRB algorithm. In particular, the dead-end scenario was constructed to test the robot's ability to detect the absence of available paths and respond appropriately by executing a 180-degree turn to retrace its steps and find an alternative route. This configuration provides a rigorous assessment of the robot’s logical response to real-time sensory data and its ability to autonomously adapt to complex, constrained spaces.

Figure 3 illustrates the layout of the test track used in the experiment, which combines all these elements into a cohesive maze-like structure to simulate realistic and challenging navigation conditions.

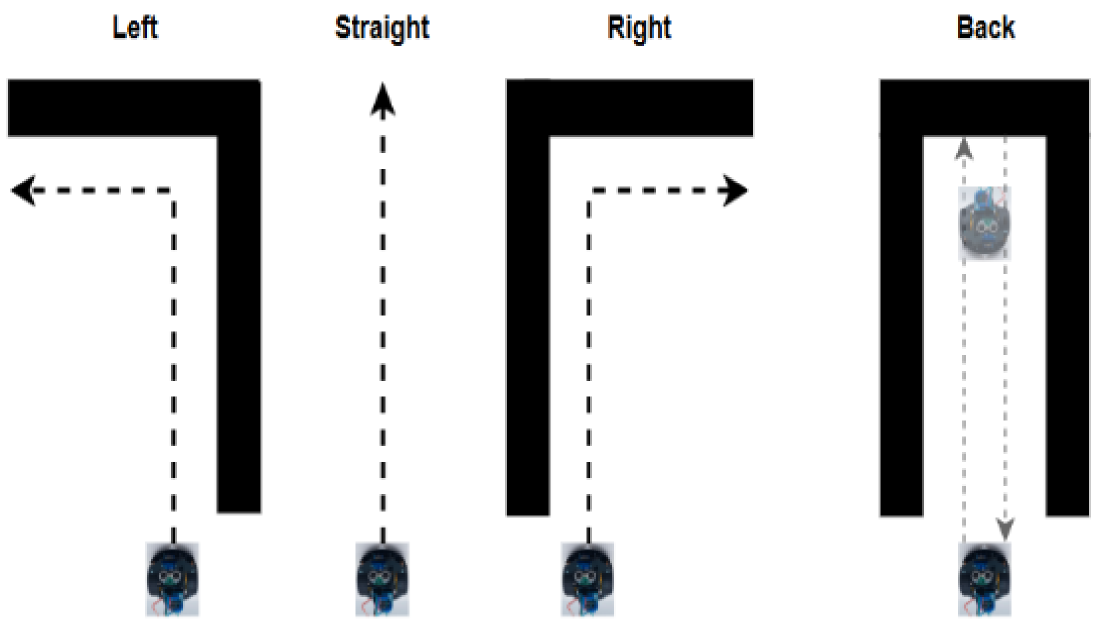

Figure 3. Obstacle-Avoiding Robot Test Track Based on the LSRB Algorithm

3.2 Pseudocode of the Robot Navigation Program Using the LSRB Algorithm

To provide a general overview of the robot's navigation decision-making process, the following is the LSRB algorithm implemented in the form of pseudocode:

Table 2. Robot Avoider Pseudocode

```
WHILE (true) DO
    distanceFront = readFrontSensor()
    IF distanceFront < 8 AND distanceFront ≠ 0 THEN
        stop()
        displayIcon("STOP")

        rotateServo(170)
        delay(700)
        distanceLeft = readLeftSensor()

        rotateServo(10)
        delay(700)
        distanceRight = readRightSensor()
        rotateServo(90) // reset to center

        IF distanceLeft < 15 AND distanceRight < 15 THEN
            turnRightSmooth() // perform 180° turn
            displayIcon("RIGHT")
            delay(2100)
        ELSE IF distanceLeft > distanceRight THEN
            turnLeftSmooth()
            displayIcon("LEFT")
            delay(1400)
        ELSE
            turnRightSmooth()
            displayIcon("RIGHT")
            delay(1000)
        END IF
    ELSE
        moveForward()
        displayIcon("FORWARD")
    END IF
END WHILE
```

This pseudocode represents the actual logic in the Arduino program, where directional decisions are made based on ultrasonic sensor readings through servo motor rotation. The robot turns toward the wider side, and if both sides are narrow, it performs a 180-degree right turn. The pseudocode illustrates the direction priority: left → straight → right → back. This strategy is selected because the left-hand rule is commonly used in solving mazes without loops. By adding logic for moving forward and turning right, with the final option being a reversal, the algorithm can navigate through various path structures.

The strength of the LSRB algorithm lies in its simplicity. It does not require path mapping or memory storage, making it suitable for low-spec robots. However, this also becomes a limitation when applied in dynamic environments or complex branches where the optimal route may not align with the predefined direction priorities.

3.3 Robot Navigation Test Results

The robot navigation tests were conducted five times for each scenario to obtain average performance data across three primary parameters: travel time, number of maneuvers, and path length. The testing environment was constructed as a structured maze with multiple configurations that activate each logical branch of the LSRB algorithm, namely left turn (L), straight path (S), right turn (R), and backtracking (B). This configuration was deliberately chosen to evaluate the algorithm's effectiveness in handling various real-world-like navigation situations.

To illustrate how the robot responds in each decision-making condition, the experimental documentation includes four visual scenarios, as shown in Figures 4 through 7. These figures provide clear examples of the robot's autonomous behavior and demonstrate the LSRB logic in action.

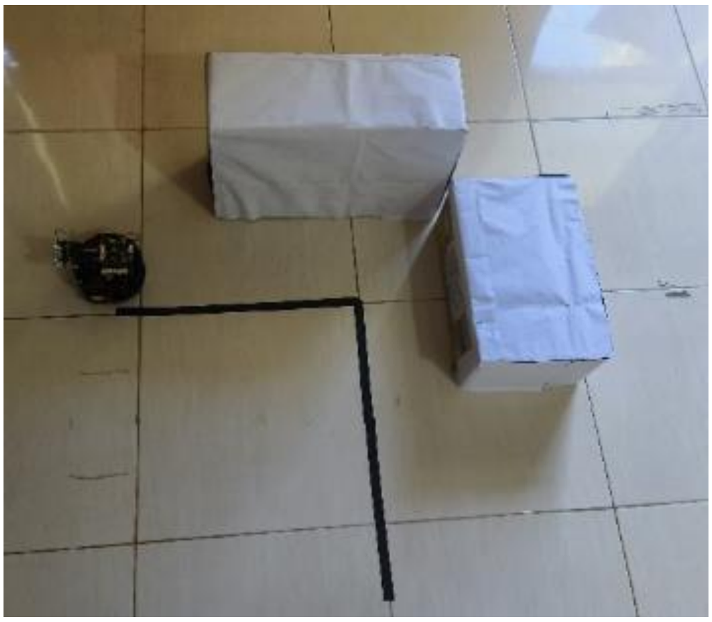

Figure 4. Robot Executes Left Turn (L - Left)

In this condition, the robot detects a clear left path while both the front and right are blocked. According to the LSRB priority rule, the robot chooses to turn left. This behavior confirms the proper implementation of the algorithm's logic hierarchy and the robot's responsiveness to sensor input.

Figure 5. Robot Moves Straight (S - Straight)

This figure illustrates a situation where the front path is unobstructed and the left path is blocked. The robot proceeds straight ahead without deviation. This demonstrates the robot's ability to maintain forward motion when no immediate obstacle exists, validating the algorithm's handling of basic traversal.

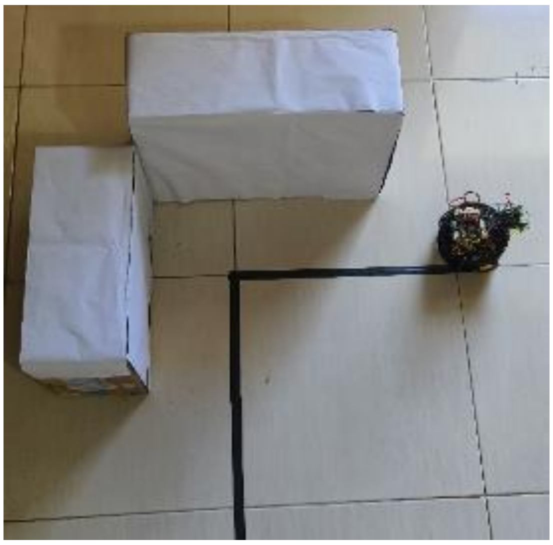

Figure 6. Robot Turns Right (R - Right)

When both the left and front directions are blocked, but the right path is clear, the robot performs a right turn as per the LSRB algorithm’s hierarchy. This action reflects the algorithm's capability to adapt to narrowing path options and still maintain navigation without requiring environmental mapping.

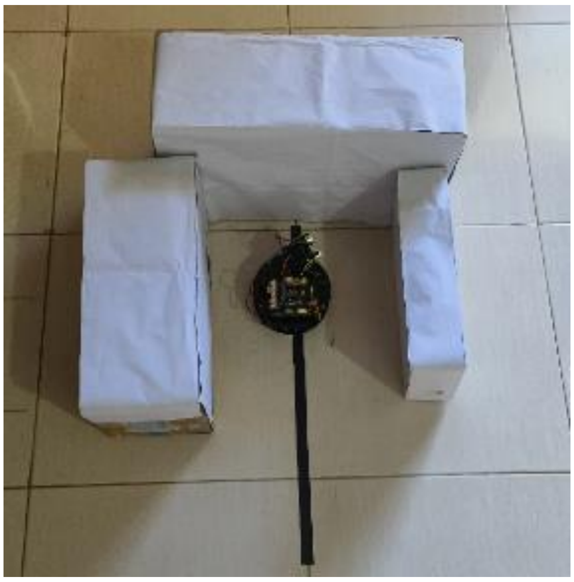

Figure 7. Robot Performs Backtrack Maneuver (B - Back)

This scenario represents a dead-end, where the robot detects that all directions—left, front, and right obstructed. The LSRB algorithm responds by executing a 180-degree rotation, allowing the robot to retreat and explore an alternate route. This behavior demonstrates the robot’s adaptability to failure scenarios and its ability to recover from navigational deadlocks.

The results from each scenario were compiled into a performance summary, which is presented in Table 3. This table details the average travel time, number of maneuvers, and the path length the robot traversed under each type of condition.

Tabel 3. Test Results

Scenario	Travel Time (seconds)	Number of Maneuvers	Path Length (cm)
L - Left	27	1	90
S - Straight	24	1	60
R - Right	22	0	90
B - Back	70	7	120

From the table above, it is evident that the travel time remained relatively consistent for the L, S, and R scenarios, with minor variations due to the robot's positioning and real-time sensor interpretation. However, the B (Back) scenario resulted in a significantly longer travel time and a higher number of maneuvers, as the robot had to recognize a dead-end and execute a reversal maneuver, including re-scanning the area to select a new path. Despite the increase in time and motion in the backtracking scenario, the robot successfully completed the navigation task in all cases.

These findings confirm that the LSRB algorithm is both functional and reliable in guiding mobile robots through various obstacle-filled environments. The consistent outcomes across multiple test trials indicate that the algorithm maintains stable performance under diverse path configurations. Moreover, its low computational demands make it highly suitable for microcontroller-based systems, especially in applications where affordability, simplicity, and real-time responsiveness are critical.

Despite its strengths, the LSRB algorithm exhibits certain limitations—particularly the absence of memory-based path recording or environmental mapping. This constraint may reduce its effectiveness in more complex or dynamic environments where route optimization or adaptive obstacle handling is required. To address this, future developments may consider integrating basic memory mechanisms or hybridizing LSRB with more advanced navigation strategies, thereby retaining its simplicity while enhancing its adaptability and path-planning efficiency.

To contextualize the effectiveness of the proposed algorithm, comparisons were made with several related studies. Nanditta et al. (2021), for instance, implemented an infrared sensor-based robot with a basic reactive algorithm. While capable of detecting nearby objects, the system lacked directional prioritization and adaptability in intricate scenarios, often leading to suboptimal navigation decisions when multiple obstacle options were present.

By contrast, the LSRB algorithm proposed in this study leverages ultrasonic sensors combined with servo-driven scanning, enabling more detailed environmental perception. This allows the robot to evaluate left and right directions before making movement decisions, resulting in improved path selection—particularly at intersections and dead ends.

Similarly, Selvakumar et al. (2022) examined the application of the LSRB approach in a maze-solving robot. However, their study focused primarily on completion success, without including detailed performance metrics. The present research extends that work by incorporating quantitative indicators—such as travel time, number of maneuvers, and path accuracy—thus providing a more comprehensive evaluation of the algorithm's efficiency.

Wahyudi et al. (2022) adopted a wall-following approach, which is effective in simple corridor navigation but less adaptable in environments with complex branches or dead ends. In comparison, the LSRB algorithm offers greater flexibility by adjusting navigational decisions based on real-time sensor input and a clearly defined directional hierarchy, allowing for better maneuvering in unpredictable settings.

In conclusion, the proposed LSRB algorithm demonstrates superior consistency, adaptability, and accuracy compared to previous methods, particularly in low-cost, microcontroller-based platforms where computational resources and hardware capabilities are limited.

4. CONCLUSION

The results clearly demonstrate that the LSRB algorithm is capable of navigating maze-like environments with high reliability. In all test scenarios, the robot successfully completed the path with 100% accuracy, indicating consistent performance in recognizing obstacles and selecting appropriate maneuvers. The robot also recorded an average travel time of approximately 31.3 seconds and an average of seven maneuvers per trial, highlighting its ability to adapt quickly and efficiently to changes in the environment’s structure.

A notable strength of the LSRB algorithm lies in its ability to handle dead-end situations without the need for global mapping. The robot was able to detect blocked paths and respond autonomously by executing a 180-degree backtracking maneuver, effectively allowing it to return to a previous node and continue navigation. This confirms the algorithm’s robustness in constrained and unknown environments, particularly in scenarios where simplicity, low resource consumption, and real-time responsiveness are crucial.

Although the algorithm is not designed to produce the shortest or most optimized path, its ability to consistently reach the target with perfect accuracy across all trials validates its reliability in exploration tasks. This makes it well-suited for general-purpose obstacle avoidance in robots with limited computational resources, especially in educational or low-cost robotic systems where algorithmic simplicity is favored over optimization.

Despite its effectiveness, the current implementation of the LSRB algorithm presents some limitations. One of the key areas for improvement is the lack of route memory or path recording functionality. In its current state, the robot is unable to recall previously traversed paths, and if required to return to the starting point, it must restart the navigation process from scratch. Future developments could address this limitation by integrating basic memory features, such as path logging or node recognition, enabling the robot to operate more intelligently in complex or dynamic environments.

In addition, the study highlights the reliability of ultrasonic sensors, particularly when optimally positioned and calibrated. Throughout the experiments, no significant cases of false detection were observed. This can be attributed to the use of non-reflective wall materials and the implementation of a simple filtering mechanism specifically, a moving average filter to improve the stability and accuracy of distance measurements.

Another critical challenge encountered during testing was related to battery capacity and power stability. The robot’s ability to execute precise maneuvers, especially during directional changes, was found to be highly dependent on consistent voltage levels. When the battery charge dropped below optimal levels, the motors exhibited reduced torque, which in turn affected the robot’s ability to complete sharp or accurate turns. This occasionally led to delayed or incomplete maneuvers, potentially compromising overall navigation performance.

This observation underscores the importance of reliable power delivery in mobile robotic systems. For future implementations, it is recommended to adopt enhanced power management strategies, such as regulated power supply modules, higher-capacity rechargeable batteries, or the addition of real-time battery monitoring systems. These improvements would allow the system to detect voltage drops and adapt its behavior accordingly. For example, by reducing motor speed or temporarily avoiding complex movements when battery performance is insufficient.

Addressing this issue will not only improve the robot's endurance and movement precision but also ensure more consistent and dependable navigation across varying conditions. Consequently, power management and battery health should be considered fundamental design elements in the development of autonomous mobile robots, particularly for long-duration or mission-critical tasks.

REFERENCES

[1] Pohan MAR. Algoritma Perencanaan Jalur Kendaraan Otonom Berbasis Hibridisasi Algoritma BFS dan Path Smoothing. TELEKONTRAN. 2020; 8(1): 1-8.

[2] Zhao J, Liang B, Chen Q. The key technology toward the self-driving car. International Journal of Intelligent Unmanned Systems. 2018; 6(1): 2-20.

[3] Ušinskis V, Nowicki M, Dzedzickis A, Bucinkas V. Navigasi Berbasis Sensor-Fusion untuk Robot Bergerak Otonom. Sensors. 2025; 25(4): 1248.

[4] Alshammrei S, Boubaker S, Kolsi L. Improved Dijkstra Algorithm for Mobile Robot Path Planning and Obstacle Avoidance. Computers, Materials & Continua. 2022; 72(3): 5939-5954.

[5] Cornelia N, Pratama DJ, Zandiyano A, Mahrijal, Dwisaputra I. Obstacle Avoiding Berbasis Remote Control. Edu Elektrika Journal. 2024; 12(1): 10-15.

[6] Ummah MAFAB. Rancang Bangun Robot Penghindar Halangan Dengan Metode PID. Jurnal Teknik Mesin, Industri, Elektro dan Informatika (JTMEI). 2023; 2(3): 212-222.

[7] Nanditta RV, Venkatesan A, Rohit R, Gowtham R, Nagulash NB, Das NK, Stephen JDG. Autonomous Obstacle Avoidance Robot using IR Sensors Programmed in Arduino UNO. IJERECE. 2021; 8(9): 10-14.

[8] Yu H, Zhang F, Huang P, Wang C, Yuanhao L. Autonomous Obstacle Avoidance for UAV based on Fusion of Radar and Monocular Camera. IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS). Las Vegas. 2020: 5954-5961.

[9] Fahmizal, Arrofiq M, Mayub A. Identifikasi Pemodelan Matematis Robot Wall Following. JNTETI. 2018; 7(1): 651-658.

[10] Katona K, Neamah HA, Korondi P. Obstacle Avoidance and Path Planning Methods for Autonomous Navigation of Mobile Robot. Sensors. 2024; 24(11): 3573.

[11] Pandian JA, Karthick R, Karthikeyan B. Obstacle Avoidance for Mobile Robots. National Conference on Architecture, Software Systems and Green Computing (NCASG). Tiruvannamalai. 2021: 92-100.

[12] Selvakumar R, Abhiram RVS, Reddy KP. Experimental Analysis of Maze Solving Robot using LSRB Algorithm. International Conference on Computer Communication and Informatics (ICCCI). Coimbatore. 2022: 01-10.

[13] Andi IGMA, Setiyono B, Sinuraya EW. Robot Mobil Line Follower dengan Kendali PID sebagai Pengembangan BRT Trans Semarang. TRANSIENT. 2020; 9(4): 651-658.

[14] Izumi K, Watanabe K, Shindo M, Sato R. A Sensor Fusion Technique Using Visual and Ultrasonic Information to Acquire Obstacle Avoidance Behaviors for Quadruped Robots. SICE-ICASE International Joint Conference. Busan. 2006: 5120-5125.

[15] Antoun S, McKerrow PJ. Wall Following with a Single Ultrasonic Sensor. Intelligent Robotics and Applications - 3rd International Conference (ICIRA). Shanghai. 2010; 6425: 130-141.

[16] Keyestudio. KS0464 Keyestudio Smart Little Turtle Robot V3. [Accessed: Mar. 16, 2025]. https://wiki.keyestudio.com/KS0464_KEYESTUDIO_Smart_Little_Turtle_Robot_V3.

[17] Islam A, Ahmad F, Sathya P. Shortest Distance Maze Solving Robot. IJRET. 2016; 5(7): 253-257.