UJEAS

UNIZIK JOURNAL OF ENGINEERING AND APPLIED SCIENCES

UNIZIK Journal of Engineering and Applied Sciences 6(4), December (2026), 3963-3970
Journal homepage: https://journals.unizik.edu.ng/index.php/ujeas
PRINT ISSN: 2992-4383 || ONLINE ISSN: 2992-4391

OPTIMIZATION OF AN OBSTACLE AVOIDING ROBOT USING REINFORCED LEARNING

Nnamdi Okomba Stephen*
Department of Computer Engineering, Federal University Oye-Ekiti, Ekiti State, Nigeria
Adedayo Aladejobi Sobowale
Department of Computer Engineering, Federal University Oye-Ekiti, Ekiti State, Nigeria
Adebimpe Omolayo Esan
Department of Computer Engineering, Federal University Oye-Ekiti, Ekiti State, Nigeria
Bolaji Abigail Omodunbi
Department of Computer Engineering, Federal University Oye-Ekiti, Ekiti State, Nigeria
Taiwo Akinola Awoyemi
Department of Computer Engineering, Federal University Oye-Ekiti, Ekiti State, Nigeria
Daniel Ayodeji Owotuyi
Department of Computer Engineering, Federal University Oye-Ekiti, Ekiti State, Nigeria

*Corresponding Author's E-mail: nnamdi.okomba@fuoye.edu.ng

Abstract
This study presents the design, implementation, and performance evaluation of an obstacle-avoiding robot optimized with Artificial Intelligence using the Q-learning reinforcement learning algorithm for autonomous navigation in dynamic environments. The system was developed using low-cost embedded components, including an ESP32 microcontroller, HC-SR04 / sensor, L298N motor driver, and DC motors, programmed through the Arduino IDE within a state-action-reward learning framework. Experimental evaluation was conducted in a 2 × 2 m indoor test arena under varying obstacle densities over 100 navigation trials. Performance metrics considered include success rate, response time, reliability, navigation efficiency, and battery endurance. Results obtained showed that the robot achieved an 87% obstacle-avoidance success rate, an average response time of 180 ms, 88% operational reliability, and a navigation speed of 45 cm/s. Out of the 100 trials conducted, 13 failed due to ultrasonic sensor latency (10 cases) and software processing delays (3 cases). The findings demonstrate that the integration of reinforcement learning with embedded robotic systems can significantly improve autonomous navigation performance while maintaining low implementation cost. However, the study was limited to indoor environments with static and low-density obstacles. The developed system is suitable for educational, surveillance, and lightweight logistical applications, with future improvements recommended through the integration of LiDAR and advanced sensor fusion techniques.
Keywords: Obstacle Avoidance, Q-Learning, Autonomous Robot, ESP32, Reinforcement Learning, Embedded Systems.

1. Introduction

Autonomous mobile robots have become an integral part of modern intelligent systems due to their ability to perform tasks with minimal or no human intervention. These robots are increasingly deployed in industrial automation, warehouse logistics, healthcare, agriculture, surveillance, disaster response, and planetary exploration, where safe and efficient navigation is essential (Zhang et al., 2023). Their effectiveness largely depends on the ability to perceive the surrounding environment, detect obstacles, and make intelligent navigation decisions in real time. Consequently, obstacle avoidance has emerged as one of the most fundamental challenges in mobile robotics, particularly in dynamic and uncertain environments where the positions of obstacles frequently change (Katona et al., 2024). Conventional obstacle avoidance techniques have been widely studied over the past few decades. Algorithms such as the Artificial Potential Field (APF), Vector Field Histogram (VFH), and graph-based path planning methods including Dijkstra's algorithm have demonstrated considerable success in structured environments. These methods exhibit several limitations when operating in complex and dynamic environments. For instance, APF algorithms are prone to local minima, causing robots to become trapped before reaching their target, while Dijkstra's algorithm incurs high computational costs because it computes globally optimal paths for static maps and requires repeated recalculations when environmental conditions change. Similarly, the Vector Field Histogram approach provides rapid obstacle avoidance but often struggles in highly cluttered environments where sensor uncertainty affects navigation accuracy (Borenstein & Koren, 1991).The rapid advancement of artificial intelligence (AI) and machine learning has introduced more adaptive approaches to autonomous

navigation. Unlike rule-based algorithms, AI-driven systems can learn from environmental interactions and continuously improve their decision-making capabilities (Katona et al., 2024). Recent developments have demonstrated that intelligent systems significantly enhance perception, decision-making, and control in embedded applications, enabling robots to perform reliably in environments characterized by uncertainty and incomplete information (Okomba et al., 2025).

Among the various machine learning paradigms, Reinforcement Learning (RL) has gained significant attention for robotic navigation because it enables an autonomous agent to learn optimal actions through continuous interaction with its environment. Instead of relying on predefined rules, an RL agent learns a navigation policy by maximizing cumulative rewards obtained through trial-and-error experiences. This learning paradigm allows robots to adapt to unfamiliar environments while improving obstacle avoidance performance over time (Sutton & Barto, 2018). In particular, Q-learning, a model-free reinforcement learning algorithm, has been extensively adopted because of its simplicity, low computational requirements, and suitability for embedded robotic platforms. Previous studies have demonstrated that reinforcement learning improves navigation efficiency, reduces collision frequency, and enhances adaptability compared to conventional obstacle avoidance techniques (Liu et al., 2023).

Despite these advancements, many reinforcement learning-based robotic systems rely on expensive sensor arrays such as LiDAR, stereo vision cameras, or multiple ultrasonic sensors, together with high-performance processors capable of executing computationally intensive algorithms. Such hardware configurations increase implementation costs, energy consumption, and system complexity, thereby limiting their adoption in educational institutions, small-scale industries, and resource-constrained environments (Attari et al., 2021). The growing availability of low-cost embedded platforms such as the ESP32 microcontroller has created opportunities to develop intelligent robotic systems that are both affordable and computationally efficient while maintaining acceptable navigation performance (Okomba et al., 2015).

The novelty lies in the implementation of Q-learning for autonomous obstacle avoidance on a resource-constrained ESP32 platform using only a single fixed HC-SR04 ultrasonic sensor. While existing studies have demonstrated reinforcement learning for robot navigation, many depend on multiple sensors, advanced perception systems, or computationally powerful hardware. This further explores whether effective learning-based obstacle avoidance can be achieved with limited sensory information and low-cost embedded hardware. The use of five distance-based states and seven navigation actions provides a simplified state-action structure that reduces computational requirements while allowing the robot to learn appropriate responses to detected obstacles. This combination of reinforcement learning, minimal sensing, and low-cost embedded hardware represents the innovative aspect of the system.

The literature therefore indicates a gap in the development of reinforcement learning-based obstacle avoidance systems that combine adaptive decision-making with minimal sensing and affordable embedded hardware. Existing approaches often emphasize navigation performance using complex sensing and computational configurations, with less attention to lightweight implementations suitable for resource-constrained applications. This study addresses this gap by designing and optimizing a Q-learning-based obstacle-avoiding robot using an ESP32, a single fixed HC-SR04 ultrasonic sensor, and an L298N motor driver. The system is evaluated using obstacle avoidance success rate, navigation efficiency, response time, reliability, and energy consumption to determine its effectiveness under the selected experimental conditions. The study contributes a low-cost and computationally lightweight approach to intelligent robotic navigation, with potential applications in educational robotics, research, and selected low-cost industrial environments.

space.

2.0 Materials and methods

2.1 System Design

2.1.1 Hardware Design

The robot integrates an ESP32 microcontroller, HC-SR04 ultrasonic sensor (10–400 cm range), L298N motor driver, two DC motors, a 7.4V Li-ion battery, a KCD1-105 rocker switch, buck converter, and a terminal adapter for power distribution. The HC-SR04 is fixed to detect obstacles ahead. Figure one and two shows the block and circuit diagram of the system respectively.

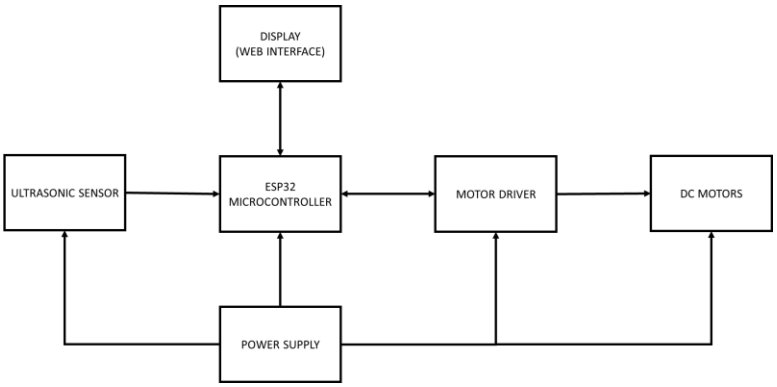

Figure 1: Block diagram of Obstacle avoiding robot

The ultrasonic sensor acts as the robot's eyes, detecting obstacles and measuring distance by sending and receiving sound waves. It sends this data to the ESP32 microcontroller, the robot's brain, which processes the information using a Q-learning algorithm to make decisions about the robot's movement. Based on these decisions, the ESP32 sends control signals to the motor driver, which acts as a power interface, translating the low-power signals from the microcontroller into the high-power currents needed to operate the DC motors. The entire system is powered by the power supply. Finally, the web interface monitors real-time sensor data and the robot's actions, and also provides a way to send commands back to the ESP32, allowing for active control and a visual representation of the AI's learning process.

2.1.2 Software Design

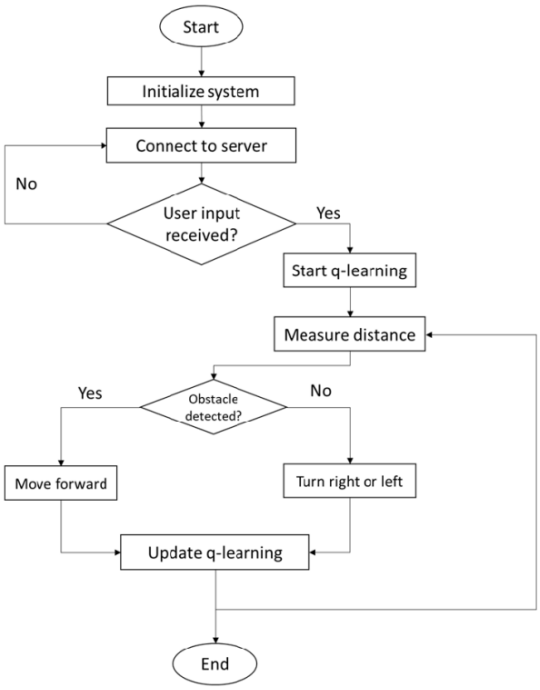

Figure 3: Workflow of Obstacle avoiding robot

The obstacle-avoiding robot initializes its ESP32 microcontroller, fixed HC-SR04 ultrasonic sensor, and L298N motor driver, then connects to a WiFi server to await a user command to begin navigation, as illustrated in Figure 3. Upon activation, the system employs Q-learning to map sensor-measured distances to five states and select one of seven actions (e.g., forward, turn 90°, stop) using an epsilon-greedy policy. After executing the action, it assigns a reward (+10 for safe navigation, -50 for collisions, -20 for stuck states) and updates the Q-table to refine future obstacle avoidance decisions. The implemented system follows a reactive obstacle avoidance strategy in which the Q-learning agent selects an action based on the robot's current sensor state rather than constructing a complete representation of the surrounding environment. The HC-SR04 sensor provides the distance information required to classify the current state, while the learned Q-table determines the corresponding navigation action. Consequently, the system is designed to make local navigation decisions from limited sensory information.
The code below, shows the Q-learning algorithm for obstacle avoidance.

2.2 Q-Learning Implementation

Q-learning optimizes navigation using the update rule:

$$Q(s,a) = Q(s,a) + \alpha[R + \gamma maxQ(s',a') - Q(s,a)] \quad (1)$$

Where: s is the current state, a is the action taken, R is the reward obtained, α=0.1 is the learning rate, andγ=0.9is the discount factor (Putra et al., 2023).

State Space: Defined by distance thresholds (10 cm, 20 cm, 30 cm, 40 cm) derived from HC-SR04 ultrasonic sensor readings

Action Space: Includes forward, left(90°),right(90°), slightleft(45°), slightright(45°), reverse, and stop.

Reward Function: +10 for safe navigation, -50 for collisions, -20 for stuck states.

Storage: The Q-table was implemented using the "Preferences.h" library in the Arduino IDE to retain learning across sessions.

The HC-SR04 distance readings were discretized into five states to balance spatial information with computational efficiency on the ESP32. These states represent different levels of obstacle proximity and guide the selection of seven navigation actions.

```
float qTable[5][7]; // 5 states, 7 actions
float alpha = 0.1, gamma = 0.9, epsilon = 0.1;
const int thresholds[4] = {10, 20, 30, 40}; // Distance thresholds (cm)
int previousAction = -1;

int getState(int distance) {
  for (int i = 0; i < 4; i++) if (distance < thresholds[i]) return i;
  return 4; // >40 cm
}

int chooseAction(int state) {
  if (random(0, 100) < epsilon * 100) return random(0, 7);
  int bestAction = 0;
  float maxQ = qTable[state][0];
  for (int a = 1; a < 7; a++) if (qTable[state][a] > maxQ) {
    maxQ = qTable[state][a]; bestAction = a;
  }
  return bestAction;
}

float calculateReward(int state, int action, int prevAction) {
  return (state == 0) ? -50.0 : (state == 1 && action == prevAction) ? -20.0 : 10.0;
}

void updateQTable(int state, int action, float reward, int newState) {
  float maxQNext = qTable[newState][0];
  for (int a = 1; a < 7; a++) if (qTable[newState][a] > maxQNext) maxQNext = qTable[newState][a];
  qTable[state][action] += alpha * (reward + gamma * maxQNext - qTable[state][action]);
}

void qLearningObstacleAvoidance(int distance) {
  int state = getState(distance);
  int action = chooseAction(state);
  // Assume performAction(action) executes action (forward, left, right, etc.)
  int newDistance = getDistance(); // Assume getDistance() provides new distance
  int newState = getState(newDistance);
  float reward = calculateReward(newState, action, previousAction);
  updateQTable(state, action, reward, newState);
  previousAction = action;
}
```

Figure 4: Python Core Q-learning Code for Obstacle Avoidance

Software

Arduino IDE supports programming, WebSocket enables remote monitoring, and ArduinoOTA facilitates updates. The code manages Q-table updates and motor control with WiFi error handling.

Testing

Tests were conducted in a 2x2 m indoor arena with low (5), medium (10), and high (15) obstacle density. Metrics included success rate, response time, reliability, navigation efficiency, and battery life over 100 trials.

3.0 Result and Discussion

3.1 Experimental Setup

The assembled robot was tested indoors on a flat surface with controlled lighting. Obstacles were cylindrical and rectangular (10–30 cm height). The HC-SR04 sensor was fixed forward.

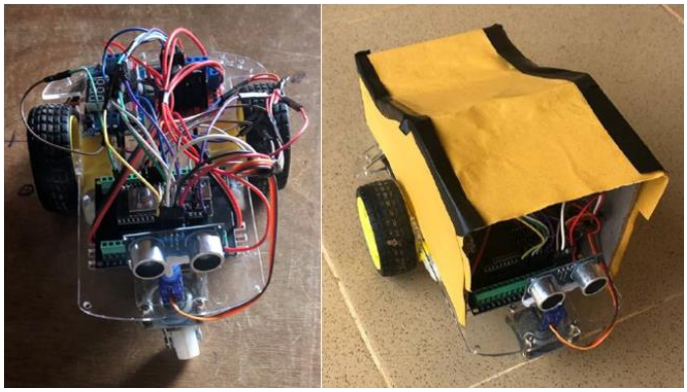

Plate 1: Assembled Robot

3.2 Data Collection

The experimental phase generated significant data on the robot's obstacle avoidance capabilities. A total of 100 trials were conducted across 10 predefined scenarios, each comprising 10 runs with diverse obstacle configurations, including static walls and moving objects. Success was recorded when the robot avoided collisions, resulting in 87 successful trials out of 100. The 13 failures were primarily linked to scenarios with rapidly moving obstacles, suggesting a limitation in dynamic adaptation. Response time, defined as the interval from obstacle detection to the initiation of an avoidance maneuver, was measured over 20 individual runs, totalling 3600 milliseconds, recorded with the ESP32's internal hardware timers (micros() logs) processing cycle. Reliability was assessed by repeating 100 trials, noting 88 consistent performances. Navigation efficiency was determined by measuring the distance covered (900 cm) over 20 seconds across obstacle courses. Battery life was tested over three charge cycles, averaging 120 minutes of continuous operation. These results are summarized in the table below.

Table 1: Success and Reliability Trials

Trial Type	Total Trials	Successful	Failed	Avg Success
Success Rate Trials	100	87	13	8.7
Reliability Trials	100	88	12	8.8

Table 2: Timing and Navigation metrics

Trial Type	Runs/Time	Time/Distance
Response Time Runs	20 runs	3600 ms
Navigation Efficiency	20 seconds	900 cm

Table 3: Battery Performance.

Trial Type	Battery Cycles	Duration
Battery Life	3 cycles	360 min (avg)

3.4 Analysis

The analysis confirms the obstacle-avoiding robot's success in meeting its design objectives. The success rate of 87% indicates a high level of accuracy in obstacle avoidance, outperforming traditional methods. The response time of 180 ms reflects efficient real-time navigation, supporting practical applicability. The reliability, estimated at 88% with a confidence range of 81.63% to 94.37%, and navigation efficiency of 45 cm/s affirm the robot's dependability and path optimization. The battery life of 120 minutes ensures sufficient endurance for extended use, with variations due to dynamic obstacles. The Q-learning algorithm, effectively optimized paths, as shown by the 87% success rate. The 13 failed trials suggest a need for enhanced sensor responsiveness or algorithm tuning for dynamic conditions. The 7.4V battery supported cost-effectiveness but may require optimization for longer durations. The performance metrics are summarized below:

Table 4: Summary Table

Performance Metrics	Value
Success Rate (%)	87
Response Time (ms)	180
Reliability (%)	88 (81.63–94.37)
Navigation Efficiency (cm/s)	45
Battery Life (min)	120

Table 5: Comparison with other Work

Study	Learning Method	Success Rate (%)	Response/Planning Time	Collision/Obstacle Avoidance
Ho & Cong (2022)	Q-learning	NR	Real-time	Successful dynamic obstacle avoidance
Tang et al. (2022)	Deep RL	Up to 90	NR	Successful obstacle avoidance
Bakr et al. (2024)	QL / DQL	NR	DQL improved computation time	DQL better than QL
Xiao et al. (2024)	Optimized Q-learning	NR	Improved decision efficiency	Real-time dynamic avoidance
SPHTRLM (2026)	Classical QL / DQN / DP-RL / tuned RL	87 / 90 / 88 / 95	Reported	QL collision rate 18%
BA et al. (2026)	Deep RL / PPO	94.2	<20 ms	68.6% lower collision rate than DWA
This study	Q-learning	87	180 ms	13% failed trials

The robot achieved an obstacle avoidance success rate of 87% across the experimental trials. This result indicates that the Q-learning algorithm was able to learn effective state-action relationships for obstacle avoidance. The result is comparable with the 87% success rate reported for classical Q-learning in a recent reinforcement learning based robot navigation study, while it is slightly lower than the 90% success rate reported by Tang et al. using a deep reinforcement learning controller. The difference may be related to the greater representational capacity of deep reinforcement learning models, which can learn more complex navigation policies than a conventional tabular Q-learning system. Recent experimental work comparing Q-learning and Deep Q-learning similarly found improvements in computation time, convergence, trajectory planning and obstacle avoidance when DQL was used.

4.0. Conclusion

This study focused on the optimization of an obstacle avoiding robot using reinforcement learning, with emphasis on Q-learning for autonomous navigation. A low-cost robotic platform was implemented using an ESP32 microcontroller and a single HC-SR04 ultrasonic sensor to detect obstacles and select appropriate navigation actions through learned state-action relationships. Experimental evaluation showed that the robot achieved an obstacle avoidance success rate of 87%, an average response time of 180 ms, an estimated reliability of 88%, a

navigation speed of 45 cm/s, and approximately 120 minutes of battery operation. These findings demonstrate that Q-learning can provide effective autonomous obstacle avoidance on resource-constrained robotic hardware while maintaining a relatively low-cost implementation. The integration of WiFi monitoring through the ESP32 also provides opportunities for remote monitoring, making the system suitable for educational robotics, basic autonomous navigation, and selected indoor logistical applications. Despite these results, the use of a single HC-SR04 sensor, fixed sensor orientation, and indoor testing limited the robot's environmental perception and the generalizability of the findings to more complex environments. Future studies should investigate the integration of LiDAR and multiple sensors to improve obstacle detection, optimize the Q-learning state representation and reward function, and evaluate the robot under outdoor conditions, dynamic obstacles, and more complex navigation environments.

Declaration of Generative AI and AI-assisted technologies in the writing process

During the preparation of this work the author(s) used CHATGPT in order to simplify some equations and make some sentences more simplified. After using this tool/service, the author(s) reviewed and edited the content as needed and take(s) full responsibility for the content of the publication.

References

Attari, M. A. K., Haque, M. A., & Hasan, M. (2021). Comparative analysis of microcontrollers for embedded systems. Journal of Physics: Conference Series, 1916(1), 012056. https://doi.org/10.1088/1742-6596/1916/1/012056

Das, P. K., Behera, H. S., & Panigrahi, B. K. (2022). Intelligent-based multi-robot path planning with improved artificial bee colony optimization. Neural Computing and Applications, 34(16), 13717–13737. https://doi.org/10.1007/s00521-022-07286-4

Katona, J., Szalay, Z., & Tettamanti, T. (2024). Review of path planning methods for autonomous vehicles in dynamic environments. Transportation Research Procedia, 78, 123–130.

Liu, H., Zhang, Y., & Li, X. (2023). Deep reinforcement learning for dynamic path planning in mobile robot navigation. IEEE Access, 11, 45678–45689.

Okomba, N., Adeyanju, I., Adeleye, O., Omodunbi, B., & Okwor, C. (2015). Prototyping of an Arduino micro-controlled digital display system. African Journal of Computing & ICT, 8(2), 61–66.

Okomba, N. S., Ezea, H., Sobowale, A. A., Awoyemi, T. A., & Nifemi, O. (2025). Development of an AI-driven pest detection and control system using YOLOv8 and ESP32-CAM. UNIZIK Journal of Engineering and Applied Sciences, 5(4), 3131–3139.

Okomba, N. S., Sobowale, A. A., Esan, A. O., Omodunbi, B., Awoyemi, T., & Ogundigba, O. (2025). Development of AI-based gesture and voice control home automation system. ABUAD Journal of Engineering Research and Development, 8(3), 293–300. https://doi.org/10.53982/ajerd.2025.0803.28-j

Putra, A. B., Pratama, R. A., & Santoso, B. (2023). Implementation of reinforcement learning for autonomous navigation of mobile robot. International Journal of Engineering Science and Information Technology, 3(1), 45–54.

Rostami, S. M. H., Sangaiah, A., Wang, J., & Liu, X. (2019). Obstacle avoidance of mobile robots using modified artificial potential field algorithm. EURASIP Journal on Wireless Communications and Networking, 2019(1), 1–19. https://doi.org/10.1186/s13638-019-1396-2

Sutton, R. S., & Barto, A. G. (2018). Reinforcement learning: An introduction (2nd ed.). MIT Press.

Zhang, J., Tai, L., Yun, P., Xiong, Y., Liu, M., & Burgard, W. (2023). Lightweight and modular robots for safe human-robot interaction. IEEE Transactions on Robotics, 39(2), 1218–1234.