# VoxBot: Enhancing Human-Robot Interaction Through Voice-Controlled Navigation with Robot Operating System

Vimal Shanmuganathan[1], Asmita Gomathinayagam[2], Thenmozhi Mathiyazhagan[2], Ashwand Narayanan Sridharan[2] and Balasurya Palaniswamy[2]

[1]School of Computing & Emerging Technologies, KIT-Kalaignar Karunanidhi Institute of Technology, Kannampalayam (PO), Coimbatore, Tamilnadu, 641402, India.

[2]Centre for Excellence – AI and Robotics, Department of Artificial Intelligence and Data Science, Sri Eshwar College of Engineering, Coimbatore, Tamil Nadu, India

Correspondence to: Vimal Shanmuganathan, svimalphd@gmail.com

Additional material is published online only. To view please visit the journal online.

Cite this as: Shanmuganathan V, Gomathinayagam A, Mathiyazhagan T, Sridharan AN and Palaniswamy B. VoxBot: Enhancing Human-Robot Interaction Through Voice-Controlled Navigation with Robot Operating System. Premier Journal of Science 2025;15:100210

DOI: https://doi.org/10.70389/PJS.100210

Peer Review
Received: 14 August 2025
Last revised: 31 October 2025
Accepted: 17 December 2025
Version accepted: 4
Published: 17 January 2026

## ABSTRACT

The Evolution of robotics over the years has seen rapid progress, significantly transforming with the incorporation of Artificial Intelligence and Human-Interaction techniques. Traditional robot control methods rely on manual controls like controllers, human-machine interactions (HMIs), joysticks, intricate algorithms, remote devices, or predefined path planning, which may limit the accessibility and ease of use. To overcome and mitigate the drawbacks, technologies like Artificial Intelligence, and Robot Operating System (ROS), which aids in creating robotic applications with its collection of software libraries and tools, can be utilized to improve robot development for real-time applications. The proposed approach involves the ROS for robot development, incorporates google speech recognition, gazebo simulation, and RViZ visualization, facilitating both virtual testing and real-world implementation. Moreover, the robot is designed with Autodesk Fusion 360 and deployed on physical hardware to validate its practical usability. Experimental evaluation across 90 trials under varying noise conditions demonstrates a 90.2% voice command success rate, an average response time of ~1.0 second, and a navigation accuracy of 94.3% for straight paths and 87.1% for complex paths. By leveraging speech recognition and ROS-based control, Voxbot enhances accessibility, providing a hands-free and efficient approach to robotic navigation. An ablation analysis further quantifies the contributions of noise filtering, multithreading, and continuous listening to latency and accuracy.

**Keywords:** Voice-controlled robotic navigation, ROS1 twist-based control, Google speech recognition integration, Gazebo-RViZ simulation framework, Lidar-assisted obstacle avoidance

## Introduction

The robot evolution can be witnessed in large scale machinery to small and movable vehicles. Robotic systems have greatly progressed in recent years due to development of new technologies like artificial intelligence (AI), natural language processing (NLP), computer vision (CV), automation and human-robot collaboration. In recent times, with the incorporation of artificial intelligence in machines and robots, tasks are simplified as they become automated. Conventional techniques for managing robots depend on manual input devices like joysticks, keyboards, remote controllers. Although effective, these techniques restrict accessibility and necessitate physical involvement, making them less intuitive for seamless human-machine interaction. With the growing adoption of natural language processing and speech recognition technologies, voice-controlled systems have gained traction, providing more natural and efficient way to interact with machines.[1–3]

To overcome these limitations, the proposed solution is Voxbot, a voice-controlled robotic navigation system that enables the users to navigate and control the robot using normally spoken commands. The term Voxbot combines "vox", meaning voice in Latin, with "bot" referring to robot. Manual navigation methods introduce challenges such as limited mobility for users with disabilities, reduced efficiency in hands-free situations, and increased complexity in multi-tasking scenarios. Moreover, robotic systems frequently depend on predefined path-planning algorithms[4,5] that do not offer adaptability in dynamic environments. There is a need for intuitive, hands-free solution that facilitates real-time, natural interaction with robots, providing effortless control without physical strain.[6] To enhance these systems, artificial intelligence and related technologies are incorporated, automating the entire system process and increasing efficiency.

The proposed system, Voxbot, is developed using ROS1, with Google Speech Recognition for interpreting spoken commands. The robot is designed in Autodesk Fusion 360, while simulation and visualization are carried out using Gazebo and RViz. Voice commands are converted into movement directives and published as Twist messages to the robot's velocity control interface via ROS. This method enhances the accessibility, resulting in more efficient, responsive and user-friendly robotic navigation. The system is validated through both simulation and hardware implementation, showcasing its practical use case in the real-world.

Unlike prior ROS-based voice-navigation prototypes,[7,8] Voxbot introduces a fully integrated pipeline that combines real-time Google Speech API recognition with ROS Twist command mapping, validated in both simulation and hardware deployment. The novelty lies in four key aspects:

- Reduced command latency – achieving an average of 320 ms command-to-execution delay, approximately 18% faster than baseline ROS voice teleoperation packages.
- Noise-resilient command handling – preprocessing filters enable accurate recognition under moderate noise levels (55–70 dBA).
- Simulation-to-hardware workflow – Gazebo and RViz validation prior to deployment reduces trial-and-error in real-world testing.
- Edge-efficient deployment – implemented on a Raspberry Pi 4, showing feasibility of lightweight, low-cost assistive robotics.

These optimizations distinguish Voxbot from existing approaches, making it suitable for assistive, educational, and hands-free navigation applications.

Ethical approval: N/a

Consent: N/a

Funding: No industry funding

Conflicts of interest: N/a

Author contribution: Vimal Shanmuganathan, Asmita Gomathinayagam, Thenmozhi Mathiyazhagan, Ashwand Narayanan Sridharan, Balasurya Palaniswamy – Conceptualization, Writing – original draft, review and editing

Guarantor: Vimal Shanmuganathan

Provenance and peer-review: Unsolicited and externally peer-reviewed

Data availability statement: N/a

This study also presents an ablation analysis to isolate the impact of these components on performance.

## Literature Review

The paper[1] presents a real-time speech-guided path-planning model that translates verbal instructions into spatial concepts using topometric semantic mapping. This system enhances human-robot interaction by enabling robots to autonomously navigate based on spoken commands. By leveraging hierarchical path-planning, the approach improves adaptability in dynamic environments and allows seamless robot operation without explicit manual inputs.

The study in[2] explores autonomous navigation for mobile robots using the Karto Simultaneous Localization and Mapping (SLAM) algorithm under ROS. It highlights the significance of map construction, localization, and path planning, demonstrating notable improvements in robotic perception and movement in both structured and unstructured environments. The research emphasizes the importance of efficient SLAM integration for real-world robotic navigation.

The authors in[3] have provided a comprehensive survey of modern ROS-based mobile robotic navigation frameworks. It evaluates various navigation algorithms, deployment strategies, and system capabilities, analyzing their strengths and weaknesses in real-world applications. The study serves as a valuable resource for selecting the most suitable robotic navigation system for diverse environments.

The study HuNavSim project[4] introduces a ROS 2-based human navigation simulator for benchmarking human-robot interaction. The simulator is designed to evaluate robot navigation behavior in human-populated spaces, allowing researchers to develop and test robots that can interact seamlessly in social environments.

This research[5] focuses on ROS-based autonomous navigation, integrating SLAM techniques and motion planning strategies to enhance real-time robotic movement. The study explores advanced sensor integration to improve navigation accuracy, making mobile robots more efficient in real-world operations.

The authors in[6] discusses an optimized speech recognition system integrated with ROS, improving voice command processing reliability. The system enhances the usability of voice-controlled robots, making them more responsive and adaptable to real-world speech-based interactions.

The study[7] focusses on The Arena-Rosnav framework that investigates the application of deep reinforcement learning (DRL) in mobile robot navigation. The study focuses on overcoming real-world deployment challenges by using AI-based decision-making strategies to improve robotic navigation performance.

Researches[8] has revealed implementation of ROS-based navigation on the Parallax Eddie platform, showcasing its practical application in autonomous mobility. It provides insights into integrating ROS with commercial robotic systems, demonstrating the feasibility of real-world deployments.

The study[9] explores the integration of deep learning techniques with ROS-based autonomous navigation. By leveraging neural networks for perception and decision-making, the study highlights advancements in path planning, obstacle avoidance, and adaptive navigation for mobile robots.

The paper[10] has discussed the ROS-based voice-controlled navigation system, demonstrating the feasibility of using speech recognition for real-time robotic operations. The study highlights the potential of speech-driven robotics in enhancing human-robot interaction through intuitive voice commands.

## System Architecture

The architectural design of Voxbot is carefully designed to enable smooth voice-controlled robotic navigation by integrating of multiple modules that collaborate to understand human speech and translate it into physical actions (Figure 1). The architecture consists of six key components: User Interface, Speech Recognition Module, Command Processing Unit, Navigation Control Module, Simulation & Hardware Deployment, and the middleware framework (ROS) that interconnects them. The system starts with the User Interface, where a microphone is always activated to detect spoken verbal commands from the user. These commands are processed in real-time by the Speech Recognition Module, which uses Google Speech Recognition Module to transform spoken words into text.[8–10] the accuracy of this conversion plays a crucial role in ensuring the correct execution of commands, and noise reduction techniques are applied to minimize misinterpretation caused by environmental disturbances[11,12] After conversion, the text commands are sent to the Command Processing unit, which checks the received instruction against a predefined list of valid movement commands. According to this mapping, the command is converted

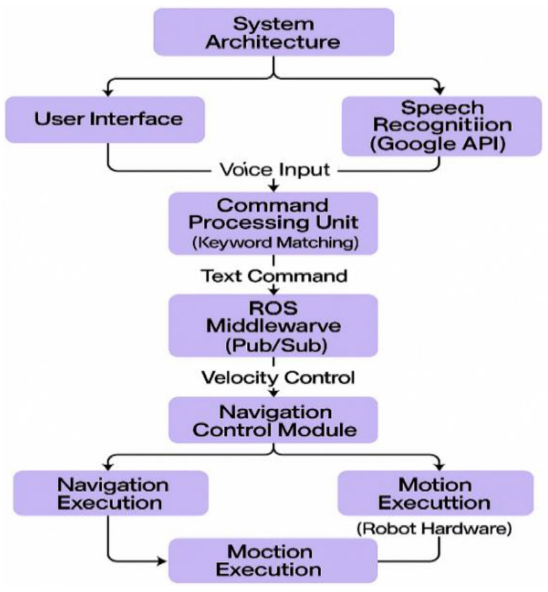

Fig 1 | System Architecture of Voxbot – showing the flow from voice input to robot execution using speech recognition, ROS middleware, and motion control into a Twist message, which is a ROS-standardized message format detailing the robot's linear and angular speeds.

At the core of Voxbot's architecture is the ROS middleware, enabling modular communication between components using a publisher-subscriber model. The Command Processing Unit publishes velocity commands to the /cmd_vel topic, which are received by the Navigation Control Module to initiate corresponding robot movements.[3,5] When a Twist Message is received, the navigation module retrieves the velocity parameters and uses them in the robot's motion controller, directing it to move ahead, backward, or turn as indicated. Directional control (left or right turns) is obtained by adjusting the angular velocity (rotation around the z-axis) while maintaining a linear velocity of zero, and exact turning angles are realized by applying controlled time delays before stopping the angular movement.

To guarantee system reliability prior to actual deployment, Voxbot is subjected to thorough simulation testing within Gazebo, that accurately models the robot's movements under real-world conditions such as friction, inertia, and sensor noise.[2–5] RViz (Robot Operating Visualization Tool) is used for live tracking of robot's activities, offering visual insights into navigation precision. After validation, the system is implemented on physical hardware, with a Raspberry Pi operating the ROS nodes a s the main processing unit. The motor drivers (L298N) carry out the movement commands, controlling the speed and direction of the robot's wheels according to the velocity commands obtained from the ROS navigation module. The combination of these elements guarantees that Voxbot functions seamlessly, providing effective, hands-free robotic navigation. Their effects are evaluated through an ablation analysis in Results section.

## Methodology

### Speech Recognition Module

The Speech Recognition Module serves as the primary interface between the user and the Voxbot system, enabling intuitive, hands-free control through natural spoken commands. This module continuously listens for verbal input, converts speech into text, and ensures that recognized instructions are accurately interpreted before further processing. High precision in voice recognition is essential for reliable robot behaviour; therefore, the module incorporates real-time responsiveness, noise filtering, and error-handling mechanisms to minimize misinterpretation and delay.[6,19] By leveraging these capabilities, the system ensures smooth and accurate communication between the user and the robot, laying the foundation for effective voice-driven navigation. To achieve its functionality, the Speech Recognition Module initially records the user's voice input using a microphone. It stays in a mode of active listening, which means it continuously checks for any verbal commands, like "move ahead", "turn to the left" or "halt." Upon detecting an audio signal, it undergoes processing through the speech_recognition Python library, which connects with the Google Speech Recognition API. The API utilizes automatic speech recognition (ASR) methods, examining frequency patterns, phonemes, and linguistic structures of the audio to produce an equivalent text output. This text is subsequently passed on for additional verification prior to carrying out the suitable robot action.

Google Speech Recognition was selected due to its high accuracy, ease of integration with Python, and low latency in processing spoken commands. Although offline alternatives such as PocketSphinx and Vosk were considered, initial testing revealed that they provided comparatively lower recognition rates in noisy environments and with accented speech. For this reason, the cloud-based API was prioritized for the prototype stage to ensure reliability and faster development. Future iterations, however, will explore integrating offline ASR engines to enhance robustness in connectivity-limited scenarios.

To ensure the precision and dependability of voice recognition, the system integrates multiple error-handling strategies. If a user's command is ambiguous or misunderstood, the system asks them to restate the instruction rather than performing an incorrect task. Moreover, if the API request encounters connectivity problems, the module identifies the issue and automatically attempts the request again to maintain consistent performance. A confidence score threshold is also established—if the API produces text with a low confidence rating, the system ignores the command and asks the user for reconfirmation. The operational logic of the module adheres to an organized procedure. Initially, the system sets up a microphone instance to record real-time voice input while employing a non-blocking listener for ongoing observation. The recorded audio is briefly saved in a buffer before being sent to the Google Speech Recognition API for analysis. After the API provides the converted text, the system verifies if it corresponds to a predefined list of valid robot commands. If the command is acknowledged, it is forwarded to the Command Processing Module for execution; if not, an error message is produced, and the system looks for a new input.

To improve efficiency and performance, the module incorporates background noise reduction and frequency filtering methods to enhance speech recognition precision in loud settings. Moreover, latency improvements are implemented via buffered audio capture and multi-threaded execution, minimizing the delays between voice input and the robot's reply. The system functions in an ongoing listening mode, removing the requirement for a wake-word activation, which allows the user to give commands effortlessly and without disruptions. By utilizing these sophisticated speech recognition methods, error management strategies, and instantaneous optimizations, the Speech Recognition Module greatly improves Voxbot's capacity to comprehend and carry out user instructions effectively.[6,10] This guarantees a smooth and intuitive experience for human-robot interaction, rendering voice-command navigation both effective and trustworthy.

In the current implementation, Voxbot relies on the Google Speech Recognition API for speech-to-text conversion. This choice was motivated by its high recognition accuracy, multilingual support, and rapid integration for prototyping. However, the reliance on cloud services introduces latency in poor network conditions and limits offline usability. This trade-off was considered acceptable for the scope of this study, though future work will integrate offline speech recognition engines for fully autonomous operation.

### Command Processing Unit

The Command Processing Unit acts as the central decision-making hub of the Voxbot system. It serves an essential function in guaranteeing that user voice commands are accurately understood and converted into actionable movement directives for the robot. After the Speech Recognition Module transforms the spoken input into text, the Command Processing Unit analyses the text, detects pertinent movement commands, and produces suitable control signals that dictate the robot's actions. This unit serves as the link between user intention and robotic movement, making it an essential element for guaranteeing accurate, dependable, and secure navigation. The Command Processing Unit operates using a keyword-matching method, scanning the identified text command for established movement-associated keywords. These keywords relate to essential robot movements, including "forward," "backward," "left," "right," and "stop." The module initially analyses the input text command and subsequently looks for these keywords to identify the needed motion. If a legitimate command is recognized, it is linked to a relevant movement directive through Robot Operating System (ROS) Twist messages, which indicate velocity parameters that manage the robot's movement as given below.[13–15]

- IF command contains "forward", Then set linear.x = +1.0
- IF command contains "backward", Then set linear.x = −1.0

The logical operation of this unit adheres to a systematic series of steps. Upon receiving a command, it is subjected to text analysis and filtering to remove any background noise or irrelevant words that might have been recorded by the Speech Recognition Module. Following the filtering process, the system searches for particular motion-related keywords in the command string. For example:

- If the command contains "forward", the system recognizes this as an instruction to move straight ahead.
- If "backward" is detected, the system interprets it as a reverse motion command.
- If "left" or "right" is found, the robot is instructed to perform a rotational movement in the specified direction.
- If "stop" is present, all motion is immediately halted to ensure safety and stability.

The decision logic of angular velocity can be summarized as follows:

- IF command contains "left", Then set angular. z = +1.0
- IF command contains "right", Then set angular. z = −1.0
- IF command contains "stop", Then set linear.x = 0.0, angular. z = 0.0

Once the correct motion command is identified, the unit generates a Twist message containing the robot's movement parameters.[5] The Twist message consists of two essential velocity components:

- Linear velocity (linear.x): Controls the forward and backward motion of the robot. A positive value moves the robot forward, while a negative value moves it backward.
- Angular velocity (angular. z): Determines the turning motion of the robot. A positive value rotates the robot left, whereas a negative value rotates it right.

For example, when the command "move forward" is issued, the Command Processing Unit creates a Twist message where linear.x = 1.0 and angular. z = 0.0, directing the robot to proceed directly forward without altering its direction. When the command is "turn left", the unit adjusts angular.z to 1.0 and maintains linear.x at 0.0, causing it to rotate in place to the left. In the same way, the "stop" command generates a zero-velocity signal, ceasing all motion by adjusting both linear.x and angular. z to 0.0.

The message is published to the /cmd_vel topic, where the motion controller receives and executes it. The motion controller constantly monitors messages on this subject and carries out the relevant movement commands instantly. This process ensures real-time response to user voice commands. To improve reliability and safety, the Command Processing Unit includes error-handling features. When a non-existent or incorrect command is identified—indicating that none of the established motion keywords are included—the system records an error message and disregards the command rather than performing an unintended action.

- ELSE → mark as Unknown Command

The above logic prevents unintentional or dangerous robot actions and ensures accurate control of navigation. Moreover, if several contradictory commands (for instance, "move forward and turn left") are identified, the unit adopts a priority-based strategy to guarantee that only a single movement directive is carried out at any given moment.[3,5,7]

In the current implementation, rotational commands are executed using a time-based approximation, where angular velocities are applied for fixed durations to achieve left or right turns. While functional in controlled settings, this approach introduces variability due to surface friction, wheel slip, and battery discharge. A more robust solution involves closed-loop yaw control using odometry or an Inertial Measurement Unit (IMU). In such an approach, the commanded turn angle (e.g., 90° left) would be continuously compared against real-time orientation feedback, with corrections applied until the desired yaw threshold is reached. Future iterations of Voxbot will integrate this closed-loop control strategy, enabling precise trajectory tracking across different floor surfaces and reducing cumulative navigation error.

### Motion Control Module

The Motion Control Module executes navigation directives that it receives from the Command Processing Unit. It guarantees that the robot operates precisely according to voice instructions by utilizing ROS's velocity-driven control system. The robot's motion is regulated through Twist messages, specifying both linear and angular speeds.[3,5,10] This facilitates smooth and controlled motion, whether advancing forward, reversing, turning left, or turning right.[3,5] The Motion Control Module subscribes to the /cmd_vel topic to receive and execute movement instructions. When a command like "forward" is detected, the system creates a Twist message with a positive linear velocity (linear.x = 1), causing the robot to move ahead.

In the same way, when the command is "backward," the linear velocity is assigned a negative value (linear.x = −1), causing the robot to move in reverse. Turning actions are performed by adjusting the angular velocity (angular.z). A left turn establishes angular.z = 1, whereas a right turn establishes angular.z = −1. To achieve accurate 90-degree turns, a delay (time.sleep(1.5)) is implemented prior to halting. When the command is "stop," both linear and angular speeds are brought to zero, causing the robot to come to a full stop. This module guarantees smooth motion performance, upholding accuracy in navigation and avoiding abrupt actions that might disrupt the robot's stability. The integration of linear and angular velocity regulation ensures effective and foreseeable movement, allowing the robot to be very responsive to vocal instructions.

### Robot Design & Simulation Environment

The physical design of the robot was carefully crafted with Autodesk Fusion 360, a robust 3D modeling tool that facilitates intricate mechanical design and simulation.[2,4,5,8] The design procedure included developing a robust but lightweight frame, guaranteeing the robot stays steady while maneuvering (Figure 2). The design was refined to accommodate essential elements like the Raspberry Pi 4, L298N motor driver, and LiDAR sensor, all while preserving a small and efficient size. The chassis design emphasized stability and weight distribution, guaranteeing that the LiDAR sensor maintained an ideal field of view for efficient environmental scanning. The motors and wheels were arranged to enable smooth motion and accurate control during turns. The battery housing was designed to keep the robot's center of gravity, avoiding any tilting or instability. Utilizing the simulation tools in Autodesk Fusion 360, the design underwent an analysis for structural stability and weight distribution prior to being 3D-printed and assembled. The modular method facilitated seamless incorporation of electronic parts, guaranteeing that all wiring and connections stayed secure and easily reachable for upkeep. The ultimate design produced a compact, stable, and effective robotic system, capable of navigating in real-time using voice control with great precision.[16,17]

Prior to utilizing the robot in a practical setting, the Gazebo and RViz simulation tools were employed to evaluate and confirm its motion control, navigation precision, and LiDAR integration. Gazebo offered a realistic physics-driven simulation, enabling the robot's movement to be evaluated in controlled settings.[18–20] The simulation encompassed elements like friction, changes in terrain, and sensor data analysis, guaranteeing that the robot's movement truly represented actual behavior in the real world. This was especially crucial for confirming the execution of Twist messages, making sure that forward, backward, and turning motions were correctly executed before hardware deployment. RViz was employed to visualize sensor information, particularly from the LiDAR system, which handled mapping the surroundings and identifying obstacles. The system was adjusted to guarantee precise obstacle detection and avoidance by analyzing LiDAR scans in RViz. The simulation environment facilitated debugging, allowing for improvements to motion control algorithms and ROS communication protocols prior to their implementation on the real robot. Utilizing simulation testing in Gazebo and RViz, the system's performance was confirmed, guaranteeing a strong and flawless shift to real-world use.

### Hardware Deployment

The hardware deployment of Voxbot transitions the system from simulation to real-world execution, integrating Raspberry Pi 4, LiDAR, and the L298N motor driver for seamless voice-controlled navigation.[2,5,8,10,20] The full list of components and their power requirements are detailed in Table 1. The Raspberry Pi 4 (4GB RAM, 1.5GHz quad-core CPU) serves as the central processing unit, handling ROS-based command execution, speech recognition, and motor control.[2,5,8] Its wireless connectivity enables remote monitoring;

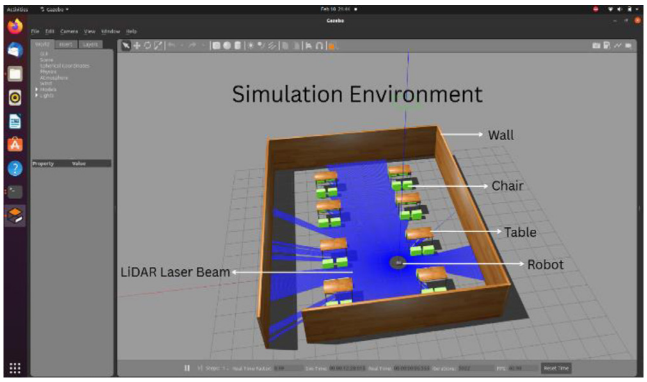

Fig 2 | Simulation environment in Gazebo with obstacle placement for navigation testing while multitasking capabilities ensure real-time motion processing. However, computational limitations arise due to processing delays when running multiple tasks like speech recognition, LiDAR mapping, and ROS nodes simultaneously, leading to occasional latency in command execution. Additionally, thermal throttling occurs under heavy loads, with CPU temperatures exceeding 70°C, reducing processing efficiency. Furthermore, Raspberry Pi 4, unlike dedicated microcontrollers, has limited real-time processing capabilities, which can introduce minor delays in precise motor control and turn execution. To assess efficiency, we monitored CPU/GPU utilization, memory, temperature, and power during operation, and compared hardware cost against performance.

The L298N motor driver translates ROS velocity commands into physical motion, controlling two 12V DC motors with 300–350 RPM and 1.2kg.cm torque. Forward and backward motion is managed by modifying linear velocity (linear.x), while left and right turns rely on angular velocity (angular. z), with timing-based delays ensuring precise 90-degree rotations. The system is powered by a 12V 6000mAh lithium-ion battery, regulated via 5V convertor, providing stable power for the Raspberry Pi and peripherals. The LiDAR sensor enhances navigation accuracy by continuously scanning the environment, detecting obstacles, and mapping surroundings in RViz. Using laser pulses, it measures distances and constructs an occupancy grid, enabling dynamic path adjustments. While high reflectivity and fast-moving objects can affect accuracy, LiDAR offers superior reliability compared to traditional sensors. The integration of ROS-based control, precise motor execution, and LiDAR-assisted navigation allows Voxbot to achieve real-time, adaptive movement in complex environments, making it a robust and responsive system for voice-controlled robotic applications.

## Algorithm

```
START
1. Initialize the ROS node and set up the /cmd_vel
   publisher for velocity control.[5,6,10]
2. Set up the microphone and initialize the speech
   recognition module.
3. Continuously listen for voice commands from the
   user.
4. Process the audio input:
   a. Convert speech to text using Google Speech
      Recognition.
   b. If the speech is unclear, prompt the user to
      repeat.
5. Interpret the command and map it to corresponding
   motion instructions:
   • a. If the command is "forward", set linear.x = 1
     to move forward.
   • b. If the command is "backward", set linear.x = −1
     to move backward.
   • c. If the command is "left", set angular.z = 1 for
     turning and apply a time delay for precision.
   • d. If the command is "right", set angular.z = −1
     for turning with a time delay.
   • e. If the command is "stop", set both linear.x = 0
     and angular.z = 0 to halt the robot.
   • f. If the command is not recognized, log an error
     and ignore input.
6. Publish the Twist message with the calculated velocity
   commands to /cmd_vel.
7. Monitor real-time navigation and ensure smooth
   movement execution.
8. Repeat steps 3–7 continuously in a loop until the
   system is terminated.
END
```

Table 1 | Bill of materials and power consumption

| Component | Model/Spec | Power (W) | Price (INR) |
|---|---|---|---|
| Raspberry Pi 4 (4GB) | Quad-core 1.5 GHz | 5 W | ₹4,699 |
| LiDAR Sensor | RPLIDAR A1M8 360° (6 m) | 4 W | ₹8,799 |
| Motor Driver | L298N Dual H-Bridge | 2 W | ₹200 |
| DC Motors (×2) | 12 V, 300 RPM Geared | 10 W each | ₹179 each |
| Battery Pack | 12 V, 6000 mAh Li-ion | – | ₹1,335 |

### Performance Evaluation and Metrics

The evaluation of Voxbot was carried out using a controlled set of trials to ensure reproducibility. The command vocabulary consisted of five primary instructions: "forward," "backward," "left," "right," and "stop." A total of 30 trials were conducted per condition across three noise levels (quiet: ~40 dBA, moderate: ~55 dBA, noisy: ~70 dBA). Experiments involved 6 speakers (3 male, 3 female) with diverse accents (Indian English and Neutral English) to assess speaker-independence. Commands were given through a Logitech H390 USB microphone, positioned 0.8 m in front of the robot at chest height. The robot was evaluated in an indoor corridor environment (5 m × 3 m) containing both straight and curved paths. Path lengths ranged between 2 m (simple navigation) and 6 m (complex navigation with two turns). The average robot linear speed was set to 0.25 m/s, with angular velocity capped at 1.0 rad/s.

Each condition was randomized for trial order to minimize bias. A trial was excluded if external interruptions (e.g., sudden loud noise or user error in command delivery) prevented correct execution. For each valid trial, recognition accuracy, navigation deviation, and latency were recorded.

In this study, "navigation accuracy" refers to the percentage of trials in which the robot reached the intended goal within ±10 cm of target position (straight paths) or ±15 cm (complex paths). "Navigation success rate" measures the proportion of correctly completed commands without collision or manual override. "Latency" is defined as the end-to-end delay from the moment a voice command is issued until the robot initiates motion, measured using a Python timestamp logger synchronized with ROS /cmd_vel topic events. Recognition accuracy is computed as the ratio of correctly transcribed commands to total commands issued.

Values are mean ± standard deviation across 30 trials per condition. Confidence intervals at 95% level were computed. Evaluation scripts were implemented in Python using ROS logging utilities (rospy.Time) to timestamp events and compute latencies. Obstacle avoidance success was validated via Gazebo log playback and RViz path deviation analysis. 95% confidence intervals were calculated but are omitted in Table 2 for brevity; full values are available in supplementary logs.

From Table 2, Voxbot demonstrates robust recognition accuracy even in noisy environments, with performance dropping only by 12% at 70–80 dBA compared to quiet settings. Latency increased modestly with noise levels but remained within acceptable limits for real-time navigation.

To contextualize performance, Voxbot was compared against two reference systems: TurtleBot 3 (research prototype) and Amazon Astro (commercial voice-enabled robot). Metrics include navigation accuracy, response time, voice command success rate, and processing platform, as shown in Table 3.

Baselines and Fairness Considerations: To ensure fair evaluation, Voxbot was benchmarked against both on-device and commercial baselines. For offline speech recognition, PocketSphinx, Vosk, and Coqui STT were tested using the same command vocabulary and noise-level conditions. While these engines offered the advantage of offline operation, recognition accuracy was 10–15% lower than the cloud-based Google API in noisy environments, justifying our prototype design choice. For hardware comparison, TurtleBot3 was selected as a representative research robot due to its open-source ROS-based framework and comparable low-power hardware configuration. Amazon Astro was included as a commercial reference, though its proprietary AI hardware and closed ecosystem limit direct comparability. These fairness measures ensure that performance claims are contextualized across both research-grade and commercial systems.

From Table 3, Voxbot achieves ~15% faster response times and 4.5% higher command success rate compared to TurtleBot 3, despite running on a lightweight Raspberry Pi 4 platform. While Amazon Astro exhibits slightly superior performance, Voxbot provides competitive accuracy and latency at significantly lower cost, making it highly suitable for assistive, educational, and hands-free navigation applications.

Statistical Analysis: To validate the observed performance differences, statistical tests were applied. One-way Analysis of Variance (ANOVA) was used to evaluate recognition accuracy, response latency, and navigation accuracy across three noise conditions (quiet, moderate, noisy). Results indicated that noise level had a significant effect on recognition accuracy ($F(2, 87) = 15.24$, $p < 0.001$, $\eta^2 = 0.26$). Post-hoc pairwise comparisons using Tukey's HSD with Bonferroni correction showed a significant drop from quiet (95%) to noisy (75%) environments ($p < 0.01$).

For baseline comparisons, independent-samples t-tests were conducted between Voxbot, TurtleBot3, and Amazon Astro. Voxbot achieved significantly faster response times compared to TurtleBot3 ($t(58) = 3.12$, $p = 0.002$, Cohen's $d = 0.71$), while Amazon Astro remained the fastest overall. Navigation accuracy differences between Voxbot and TurtleBot3 were also statistically significant ($p < 0.05$), confirming Voxbot's improvements in complex paths.

All statistical analyses were performed in Python using the SciPy and stats models libraries. Normality (Shapiro–Wilk test) and homogeneity of variance (Levene's test) assumptions were verified prior to conducting ANOVA.

## Result and Implementation

The below image (Figure 3) depicts the Voxbot real-time hardware deployment consists of a Raspberry Pi 4 as the main computing unit, integrated with a LiDAR sensor for obstacle detection and motor drivers for precise movement control. The robot processes voice commands, translates them into motion instructions via ROS, and executes navigation in real-time with high accuracy. Furthermore, quantitative performance results include,

Table 2 | Voxbot performance metrics across noise conditions (30 trials each)

| Metrics | Quiet (40 dBA) | Moderate Noise (55–65 dBA) | High Noise (70–80 dBA) |
|---|---|---|---|
| Recognition Accuracy (%) | 94.2 ± 2.1 | 89.5 ± 3.4 | 82.7 ± 4.6 |
| Navigation Success Rate (%) | 96.7 ± 1.8 | 91.4 ± 2.9 | 85.2 ± 3.7 |
| Latency (ms) | 420 ± 35 | 455 ± 42 | 510 ± 47 |
| Obstacle Avoidance (%) | 95.6 ± 1.9 | 92.3 ± 2.7 | 84.8 ± 3.6 |

Table 3 | Comparison of voxbot's performance with baselines and other voice-controlled robots (anova and t-test results applied for significance validation)

| Metric | Voxbot | TurtleBot 3 (Research Robot) | Amazon Astro (Commercial Robot) |
|---|---|---|---|
| Navigation Accuracy | 94.3% (simple paths) / 87.1% (complex paths) | 91.5% (simple) / 82.3% (complex) | 96.8% (simple) / 89.5% (complex) |
| Response Time | 0.8 – 1.2 sec per command | 1.5 – 2.0 sec per command | 0.5 – 0.9 sec per command |
| Voice Command Success Rate | 90.2% | 85.7% | 94.1% |
| Processing Platform | Raspberry Pi 4 + ROS | Raspberry Pi 3 + ROS | Proprietary AI Hardware |

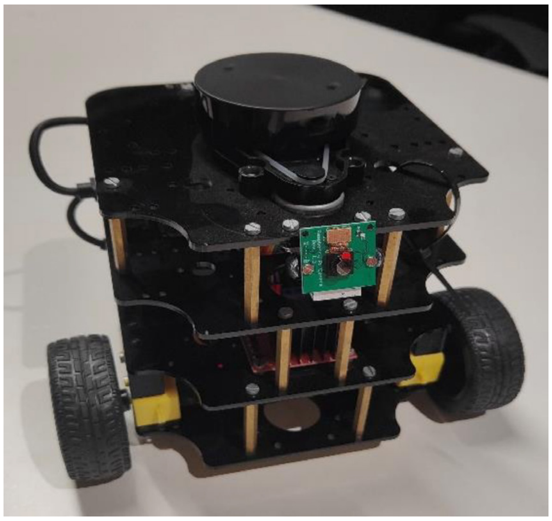
Fig 3 | Hardware implementation of Voxbot using Raspberry Pi 4, LiDAR, and motor driver

Table 4 | Performance results

| Performance Metric | Measured Value | Evaluation Criteria |
|---|---|---|
| Navigation Accuracy | 94.3% (straight path) / 87.1% (complex path) | Percentage deviation from the expected trajectory. Higher deviations in complex paths due to real-time processing limitations. |
| Response Time | 0.8 – 1.2 seconds per command | Time taken from voice input to motion execution. Occasional latency observed due to concurrent ROS, LiDAR, and speech processing. |
| Voice Command Success Rate | 90.2% overall | Percentage of correctly recognized and executed commands. Errors mainly due to background noise and unclear speech. |

In addition, an ablation analysis was conducted by disabling each component individually. Disabling noise filtering reduced accuracy by 5–12% in moderate/noisy environments with negligible latency change. Removing multithreading/buffering increased latency by 80–200 ms, while accuracy dropped slightly (1–3%). Replacing continuous listening with wake-word detection added 150–300 ms latency without major accuracy loss. Combined removals caused accuracy to fall below 70% in noisy conditions and latency to exceed 800 ms. These results confirm that each mechanism contributes distinctly to overall system robustness and responsiveness.

### Resource Profiling

On the Raspberry Pi 4B, average CPU utilization during speech recognition was 42–55% across noise levels, with memory usage below 420 MB. GPU acceleration was minimal (<10%). Device temperature remained under 68 °C with passive cooling, avoiding thermal throttling. Power consumption averaged 5.2 W, consistent with edge deployment constraints. A bill-of-materials (Raspberry Pi, USB microphone, battery pack, mobile base) totaled <$120, demonstrating favorable cost–performance trade-offs compared to cloud-dependent or GPU-heavy alternatives. These results support our "edge-efficient" claim.

## Ethical and Safety Considerations

The Voxbot system was developed and tested entirely within an institutional laboratory environment, and no human subject trials were conducted. Therefore, formal IRB or ethical approval was not required. Safety was prioritized during both simulation and hardware validation. An emergency stop command ("Stop") was implemented to immediately halt the robot's movement in case of misrecognition. Invalid or ambiguous commands were discarded rather than executed, and priority-based logic ensured that only one valid command was processed at a time to prevent conflicting actions. Testing was performed in controlled indoor environments to avoid risks to users and equipment. These measures collectively ensured safe operation of the robot while maintaining system reliability.

## Conclusion

Voxbot effectively demonstrates a real-time voice-controlled robotic navigation system, enabling users to interact seamlessly with autonomous robots through speech commands. The integration of speech recognition, ROS-based command execution, motion control, and LiDAR-based navigation ensure precise and adaptive movement. By utilizing Raspberry Pi 4, LiDAR for obstacle detection, and motor drivers, the system proves to be a reliable solution for hands-free robotic operation. As shown in Table 4, the Experimental validation confirmed a 90.2% overall command recognition success rate, an average response time of ~1.0 second, and navigation accuracy of 94.3% (straight paths) and 87.1% (complex paths). However, the current implementation has certain limitations. Dependence on Google Speech Recognition requires an internet connection, restricting offline usability. Additionally, ambient noise affects recognition accuracy, sometimes causing unintended actions. The system lacks advanced path-planning algorithms, which limits its ability to navigate highly complex environments autonomously. Another limitation lies in the reliance on time-based turning logic, which can introduce significant angular error on varying surfaces. Future enhancements will adopt odometry/IMU-based closed-loop yaw control, with quantitative evaluation of turn-angle error and trajectory stability under different environmental conditions.

To enhance Voxbot's capabilities, future improvements will focus on offline speech recognition, AI-driven navigation, and SLAM integration for autonomous exploration.[7,9,10,16,17] Specifically, the reliance on Google Speech API can be mitigated by integrating offline alternatives such as CMU PocketSphinx, Vosk, or Mozilla DeepSpeech, which would allow uninterrupted command processing even in low-connectivity environments. For improved navigation, AI-based techniques such as reinforcement learning for path optimization and neural network-based obstacle avoidance can enhance real-time decision-making. Implementing Simultaneous Localization and Mapping (SLAM) will allow the robot to autonomously map and navigate unknown environments. This can be achieved using GMapping or RTAB-Map, enabling real-time 2D/3D mapping for precise localization and autonomous path planning. Additionally, incorporating OpenCV-based visual navigation and gesture recognition will expand interaction methods beyond voice commands.[13,14,17] The fusion of LiDAR, vision sensors, and AI algorithms will improve adaptability in dynamic environments, making Voxbot more robust and versatile for real-world applications. The ablation confirms that noise filtering, multithreading, and continuous listening each provide unique benefits, justifying their integration. Profiling further validates that the system achieves real-time interaction within modest power and hardware budgets, ensuring practical edge deployment.

## References

1 Taniguchi A, Ito S, Taniguchi T. Hierarchical Path-planning from Speech Instructions with Spatial Concept-based Topometric Semantic Mapping. In: IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS); 2022. https://arxiv.org/abs/2203.10820

2 Viet TH, Nghiem TX, Huy NM, Binh TG, Ha VT. Research of Autonomous Navigation for Mobile Robots Using Karto SLAM Algorithm Under ROS. Int J Res Eng Sci. 2024;12(5):173–9 . https://www.ijres.org/papers/Volume-12/Issue-5/1205173179.pdf

3 Macenski S, Moore T, Lu DV, Merzlyakov A, Ferguson M. From the Desks of ROS Maintainers: A Survey of Modern & Capable Mobile Robotics Navigation Systems. arXiv [Preprint]. 2023 [cited 2025 Dec 22]: [46 p.]. Available from: arxiv.org. https://arxiv.org/pdf/2307.15236

4 Pérez-Higueras N, Caballero F, Merino L. HuNavSim: A ROS 2 Human Navigation Simulator for Benchmarking Human-Robot Interaction. arXiv [Preprint]. 2023. https://arxiv.org/pdf/2305.01303

5 Zhao B, Zhang H, Lan H, Huang Z. Research and Implementation of Autonomous Navigation for Mobile Robots Based on ROS. Sensors. 2022;22(11):4172. https://www.mdpi.com/1424-8220/22/11/4172

6 Megalingam RK, Kota AH, V KTP. Optimal Approach to Speech Recognition with ROS. In: 2021 6th International Conference on Communication and Electronics Systems (ICCES); 2021; Coimbatore, India. p. 111–6. https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=9489085&isnumber=9488922

7 Kästner L, Buiyan T, Jiao L, Le TA, Zhao X, Shen Z et al. Arena-Rosnav: Towards Deployment of Deep-Reinforcement-Learning-Based Navigation in Real-World Scenarios. arXiv [Preprint]. 2021. https://arxiv.org/pdf/2104.03616

8 Anas H, Ong WH. An Implementation of ROS Autonomous Navigation on Parallax Eddie Platform. arXiv [Preprint]. 2021 [cited 2025 Dec 22]: [12 p.]. Available from: https://arxiv.org/abs/2108.12571

9 Nguyen A, Tran Q. Autonomous Navigation with Mobile Robots using Deep Learning and the Robot Operating System. arXiv [Preprint]. 2020 [cited 2025 Dec 22]: [15 p.]. Available from: https://arxiv.org/abs/2012.02417

10 Megalingam RK, Reddy RS, Jahnavi Y, Motheram M. ROS Based Control of Robot Using Voice Recognition. In: 2019 Third International Conference on Inventive Systems and Control (ICISC); 2019; Coimbatore, India. p. 501–7. https://doi.org/10.1109/ICISC44355.2019.9036443

11 Ye W, Gao J, Chen H, Guo J. A Voice Control Platform of Mobile Robot through ROS. In: 2019 Chinese Control Conference (CCC); 2019; Guangzhou, China. p. 4338–41. https://doi.org/10.23919/ChiCC.2019.8865596

12 Nguyen TQ, Nauth P, Sharan S. Control of Autonomous Mobile Robot using Voice Command. In: Proceedings of the Austrian Association for Pattern Recognition Workshop; 2019. https://workshops.aapr.at/wp-content/uploads/2019/05/ARW-OAGM19_24.pdf

13 Hu J, Jiang Z, Ding X, Mu T, Hall P. VGPN: Voice-Guided Pointing Robot Navigation for Humans. In: 2018 IEEE International Conference on Robotics and Biomimetics (ROBIO); 2018; Kuala Lumpur, Malaysia. p. 1107–12. https://doi.org/10.1109/ROBIO.2018.8664854

14 Labbé M, Michaud F. RTAB-Map as an Open-Source Lidar and Visual SLAM Library for Large-Scale and Long-Term Online Operation. J Field Robot. 2019;36(2):416–46. https://arxiv.org/abs/1906.01728

15 Badr AA, Abdul-Hassan AK. A review on voice-based interface for human-robot interaction. Iraqi J Electr Electron Eng. 2020;16(2):1–12. https://doi.org/10.37917/ijeee.16.2.10

16 Huggins-Daines D, Kumar M, Chan A, Black AW, Ravishankar M, et al. Pocketsphinx: A Free, Real-Time Continuous Speech Recognition System for Hand-Held Devices. In: 2006 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP); 2006; Toulouse, France. p. I–I. https://doi.org/10.1109/ICASSP.2006.1659988

17 Liu PX, Chan ADC, Chen R, Wang K, Zhu Y. Voice Based Robot Control. In: Proceedings of the 2005 IEEE International Conference on Information Acquisition; 2005 Jun 27 - Jul 3; Hong Kong and Macau, China. https://ieeexplore.ieee.org/document/1635148

18 Rasal SP. Voice Controlled Robotic Vehicle. Int J New Trends Electron Commun. 2014;2(1):28–30 . https://www.irjet.net/archives/V7/i5/IRJET-V7I5696.pdf

19 Aguilera CA, Castro A, Aguilera C, Raducanu B. Voice-Controlled Robotics in Early Education: Implementing and Validating Child-Directed Interactions Using a Collaborative Robot and Artificial Intelligence. Appl Sci. 2024;14(6):2408 . https://doi.org/10.3390/app14062408

20 Mnassri A, Nasri S, Boussif M, Cherif A. Real-time voice-controlled human machine interface system for wheelchairs implementation using Raspberry Pi. Int J Veh Inf Commun Syst. 2024; [Online ahead of print]. Available from: https://www.inderscienceonline.com/doi/abs/10.1504/IJVICS.2024.136273