iJRASET
International Journal For Research in Applied Science and Engineering Technology



INTERNATIONAL JOURNAL FOR RESEARCH

IN APPLIED SCIENCE & ENGINEERING TECHNOLOGY

Volume: 14 **Issue:** X **Month of publication:** October 2026

DOI: https://doi.org/10.22214/ijraset.2026.85427

www.ijraset.com

Call: 08813907089 | E-mail ID: ijraset@gmail.com

International Journal for Research in Applied Science & Engineering Technology (IJRASET)
ISSN: 2321-9653; IC Value: 45.98; SJ Impact Factor: 7.538
Volume 14 Issue X Oct 2026- Available at www.ijraset.com

Design and Development of an AI-Based Voice-Controlled Robot for Intelligent Human-Robot Interaction

Rajnish Kumar[1], Md. Sahil[2], Akash Kumar[3], Sonu Kr. Chaudhary[4]
[1,2,3,4]*Student, Electronics Engineering, Government Polytechnic Sitamarhi, Sitamarhi, India.*

Abstract: ***Most low-cost voice-controlled robots accept only a handful of fixed words such as "forward" and "stop", and depend on a smartphone app for speech recognition. This paper presents the design and development of an AI-based voice-controlled mobile robot that holds natural spoken conversations and turns free-form requests into safe movements. The robot is built around an ESP32 microcontroller with an INMP441 I²S MEMS microphone, a MAX98357A I²S class-D amplifier and speaker, an L298N dual H-bridge driving two DC gear motors with wheels, a servo for head and gesture movements, and a 2S pack of 18650 lithium-ion cells with a main switch. A hybrid architecture is used: a compact keyword-spotting model on the ESP32 detects the wake word and a local "stop" command without any network, while the spoken request is streamed to an AI server where speech-to-text, a large language model and text-to-speech produce a structured JSON action and a spoken reply. A command validator on the ESP32 checks every action against a whitelist, speed limits and a time-out before the motors move, so the language model can never drive the robot outside its safe envelope. A review of 19 related studies shows that earlier robots offered either fixed-vocabulary control on cheap hardware or natural-language interaction on expensive platforms, but rarely both. Design calculations show that a 4-second request needs 128 kB of audio, well within the ESP32's memory, and that the battery pack supports an estimated 13.4 h of mixed use. An experimental protocol with metrics for recognition accuracy, latency, safety response and user experience is provided for prototype validation.***

IndexTerms: Human–robot interaction, voice control, ESP32, large language model, keyword spotting, INMP441, MAX98357A, L298N, speech recognition.

I. INTRODUCTION

Human–robot interaction (HRI) studies how people and robots communicate and work together [1]. Speech is the most natural channel for that communication: it needs no screen or controller, works when the user's hands are busy, and is usable by children, elderly people and people with limited mobility. Spoken interaction with robots, however, raises problems that ordinary voice assistants do not face, such as grounding words in the physical world, handling noise from the robot's own motors, and acting safely on ambiguous requests [2].

Student and hobby voice-controlled robots typically use an Android app to convert speech to text and send a single word over Bluetooth to an Arduino [4], [5]. They understand only a fixed list of commands, cannot reply, and stop working when the phone is out of range. At the other end, recent research robots use large language models (LLMs) to understand open-ended instructions and hold conversations [13]–[19], but run on expensive hardware with full computers on board.

This paper bridges the two. It designs a robot built entirely from low-cost modules — an ESP32 microcontroller, an INMP441 I²S microphone, a MAX98357A amplifier with speaker, an L298N motor driver with two DC motors and wheels, a servo motor, and a pack of 18650 rechargeable cells with a switch — that nevertheless supports natural, two-way spoken interaction through an AI server. The contributions are:

1) A hybrid on-device/server architecture in which wake-word and emergency-stop detection run on the ESP32, while speech recognition, language understanding and speech synthesis run on an AI server.
2) A structured-output interface in which the LLM returns a JSON action drawn from a fixed set of robot primitives together with a spoken reply.
3) An on-robot safety layer that validates every action and stops the motors on a spoken "stop", a time-out or a lost connection, independent of the AI server.

©IJRASET: All Rights are Reserved

4) A complete hardware design with pin assignments and power path, a review of 19 related studies, design calculations for memory, latency and battery life, and an experimental protocol for validation.

Section II reviews related work. Sections III to V describe the architecture, hardware and software. Section VI sets out the experimental method, Section VII presents the results and discussion, and Section VIII concludes the paper.

II. LITERATURE REVIEW

The reviewed studies fall into four groups: foundations of spoken HRI, low-cost voice-controlled vehicles, on-device speech recognition, and LLM-based robot control. Table 1 summarises the 19 studies.

A. Foundations of spoken human–robot interaction

Goodrich and Schultz surveyed HRI and identified the level of autonomy, the nature of information exchange and the structure of the human–robot team as its central design dimensions [1]. Marge et al. reported the outcome of a U.S. National Science Foundation workshop on spoken language interaction with robots and made 25 recommendations across eight themes, including robust speech recognition in noisy, mobile settings and grounding language in the robot's perception and actions [2].

B. Low-cost voice-controlled robots and vehicles

Pleshkova et al. developed a speech recognition algorithm and a LabVIEW model on a myRIO embedded device to control the motion of a mobile robot by voice [3]. Umar et al. built a four-wheel-drive vehicle on an Arduino Uno with a Bluetooth module, in which an Android app converts speech to text; the vehicle responded correctly to all tested commands within a Bluetooth range of 8.5–12 m [4]. Azahar et al. used the AMR Voice app, an HC-05 module and a Maker Drive motor driver for five commands, and measured a stable delay of about 100 ms with a link range of 50 m [5]. Pratama et al. built an ESP32 goods-carrying robot with two L298N drivers, Mecanum wheels and an MG996R servo lift; commands passed through Google Assistant and IFTTT, and the robot failed to respond to “left” and ended only two of five transfers straight [6].

Voice-controlled wheelchairs show the same pattern. Bakouri et al. recognised five commands with a CNN in a Flutter Android app and drove a wheelchair through a Raspberry Pi, reaching about 87.2% accuracy for some commands [7]. Mendoza-Montoya et al. drove an Arduino and L298N wheelchair with Google's Spanish speech recognition and found that some words were recognised only half the time until they were replaced with acoustically distinct alternatives, which raised accuracy to about 85–95% [8].

C. On-device speech recognition

Zhang et al. showed that keyword-spotting neural networks can be fitted to microcontroller memory and compute limits, with a depthwise-separable CNN reaching 95.4% accuracy [10]. Warden released the Speech Commands dataset of short spoken words, which has become the standard benchmark for such models [11]. Adetoyi and Folami deployed a 1-D CNN on MFCC features with TensorFlow Lite on an ESP32-S3 with an I²S microphone to recognise “open” and “close” for a door lock, at about 97.5% accuracy [9]. For open-vocabulary transcription, Radford et al. trained the Whisper model on about 680,000 hours of weakly supervised audio and achieved robust recognition without task-specific fine-tuning [12].

D. Large language models for robot control and interaction

Ichter et al. (SayCan) grounded LLM plans in the robot's learned skills so that only feasible actions are chosen [13], and Liang et al. had LLMs write short robot programs that call perception and control functions [14]. Vemprala et al. described design principles for using ChatGPT in robotics through a high-level function library and prompt engineering [15]. Koubaa's ROSGPT converted natural-language instructions into JSON commands for ROS 2 robots, but observed occasional undefined actions and hallucinated responses [16].

Kim et al. compared an LLM-powered social robot with text and voice agents in a 32-participant study, finding that the robot was good at rapport-building but raised expectations for non-verbal cues and could cause anxiety [17]. Wang et al. (LaMI) let an LLM coordinate speech, gaze and expressive movements on a physical robot [18], and Lakhnati et al. used GPT-4 with function calling to let users talk to several robots in a VR teaming simulation [19].

Table 1 Summary of reviewed literature

Ref.	Year	Platform / hardware	Speech method	Key result or limitation

Ref.	Year	Platform / hardware	Speech method	Key result or limitation
[1]	2008	Survey	—	Defines core HRI dimensions: autonomy, information exchange, teaming
[2]	2022	Workshop report	—	25 recommendations for spoken interaction with robots
[3]	2018	myRIO, LabVIEW	Custom recognition algorithm	Voice control of mobile-robot motion
[4]	2023	Arduino Uno, 4WD, Bluetooth	Android speech-to-text	All commands correct; range 8.5–12 m
[5]	2023	Arduino, HC-05, Maker Drive	AMR Voice app	5 commands; ~100 ms delay; 50 m range
[6]	2025	ESP32, 2 × L298N, servo lift	Google Assistant + IFTTT	"Left" not recognised; cloud-dependent
[7]	2022	Raspberry Pi wheelchair	CNN in Android app	~87.2% accuracy for some of 5 commands
[8]	2023	Arduino, L298N wheelchair	Google speech (Spanish)	Word choice raised accuracy to 85–95%
[9]	2025	ESP32-S3, I²S microphone	1-D CNN on MFCC, TFLite	~97.5% accuracy; on-device
[10]	2017	Microcontrollers	DS-CNN keyword spotting	95.4% accuracy within MCU limits
[11]	2018	Dataset	Speech Commands	Standard benchmark for keyword spotting
[12]	2023	Server / GPU	Whisper ASR	Robust transcription from 680k h training
[13]	2023	Mobile manipulator	LLM + learned affordances	Grounds plans in feasible skills
[14]	2023	Robot arms, mobile robots	LLM writes policy code	Language-to-program control
[15]	2023	Drones, arms, simulators	ChatGPT + function library	Design principles for LLM robotics
[16]	2023	ROS 2 robots	ChatGPT → JSON commands	Occasional undefined or hallucinated actions
[17]	2024	Social robot (32 users)	LLM dialogue	Good rapport; weak logic; can cause anxiety
[18]	2024	Bimanual robot with neck	LLM + speech, gaze	Coordinated speech and expressive motion
[19]	2024	VR simulation	GPT-4 function calling	Conversational multi-robot control

E. Research gap

Two separate lines of work emerge. Low-cost robots [3]–[8] recognise a few fixed words, rely on a phone or cloud assistant, and cannot talk back; their accuracy falls when words sound alike [6], [8]. LLM-based systems [13]–[19] understand open requests and converse, but run on costly platforms, and their outputs can include undefined or hallucinated actions [16] that must not reach the motors unchecked. On-device keyword spotting [9]–[11] is accurate and works offline but covers only a small vocabulary. Table 2 shows how the proposed design combines the strengths of each line.

Table 2 Gaps in existing work and how the proposed robot addresses them

Gap	Evidence in reviewed work	Approach in this paper
Fixed vocabulary, no spoken reply	Phone apps send single words [4], [5], [7]	Server ASR + LLM understand free speech; TTS reply through MAX98357A
Dependence on phone or cloud for every command	Bluetooth app [4], [5]; Google Assistant + IFTTT [6]	Wake word and "stop" recognised on the ESP32 with no network
Unsafe or undefined LLM actions	Hallucinated actions in [16]	JSON schema, whitelist, limits and time-out enforced on the robot
Costly platforms for natural interaction	Full computers on board in [13], [18]	All on-board parts are low-cost modules; heavy AI runs off-board

III. PROPOSED SYSTEM ARCHITECTURE

The system has two parts connected over Wi-Fi (Fig. 1). The robot captures speech, plays replies, drives the motors and servo, and enforces safety. The AI server — a laptop on the same network or a cloud service — runs the computationally heavy models: speech-to-text, the language model and text-to-speech. The ESP32's two cores divide the on-board work: core 0 runs audio capture and keyword spotting continuously, and core 1 runs networking, motion control and the safety layer.

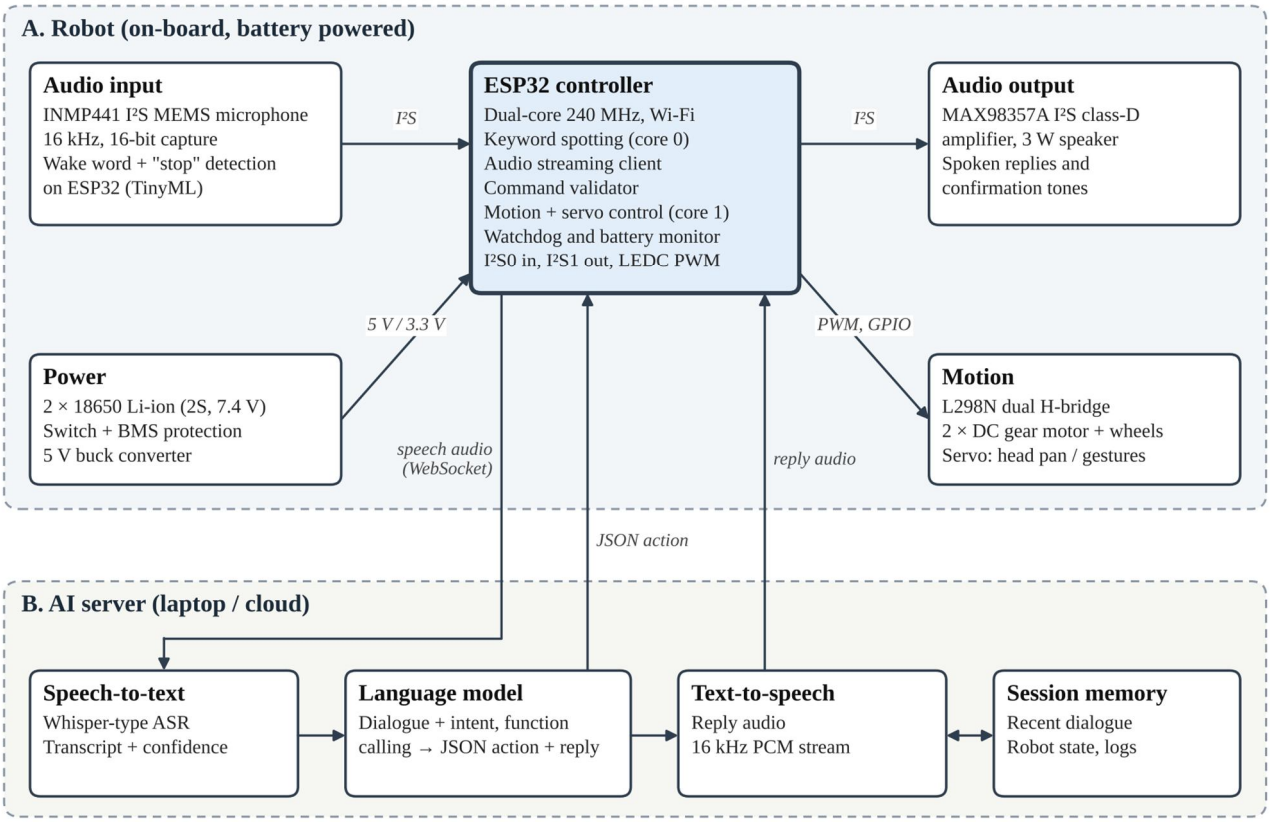


Fig. 1 System architecture of the AI-based voice-controlled robot

A WebSocket connection [25] carries audio from the robot to the server and carries the JSON action and reply audio back. A persistent connection avoids the delay of opening a new connection for each request and lets reply audio start playing before the whole reply has been synthesised.

IV. HARDWARE DESIGN

Table 3 lists the components and their roles, and Fig. 2 shows the wiring and power path. All components are inexpensive, widely available modules.

Table 3 Hardware components of the robot

Component	Key specification	Role in the robot
ESP32 (dual-core Xtensa LX6) [20]	Up to 240 MHz, 520 KB SRAM, Wi-Fi + Bluetooth, two I²S peripherals, LEDC PWM	Main controller: audio, keyword spotting, networking, motion, safety
INMP441 MEMS microphone [21]	Omnidirectional, 24-bit I²S output, SNR 61 dBA	Captures the user's speech
MAX98357A amplifier [22]	I²S input, filterless class-D, up to 3.2 W into 4 Ω at 5 V	Drives the speaker for spoken replies and tones
Speaker	4–8 Ω, about 3 W	Voice output
L298N motor driver (L298 IC) [23]	Dual full-bridge, up to 46 V and 2 A per channel	Speed and direction of the two DC motors
DC gear motors with wheels (×2) + caster	Typically 3–6 V geared motors	Differential-drive locomotion
Servo motor (SG90 class)	5 V, 0–180°, 50 Hz PWM	Head pan to face the speaker; nod and gesture movements
18650 Li-ion cells (×2) with 2S BMS	7.4 V nominal, 8.4 V full; about 2600 mAh per cell (cell-dependent)	Rechargeable power source with over-charge and over-discharge protection
Switch (SPST)	Rated above peak motor current	Main power on/off
5 V buck converter	3 A output	Supplies ESP32, amplifier and servo

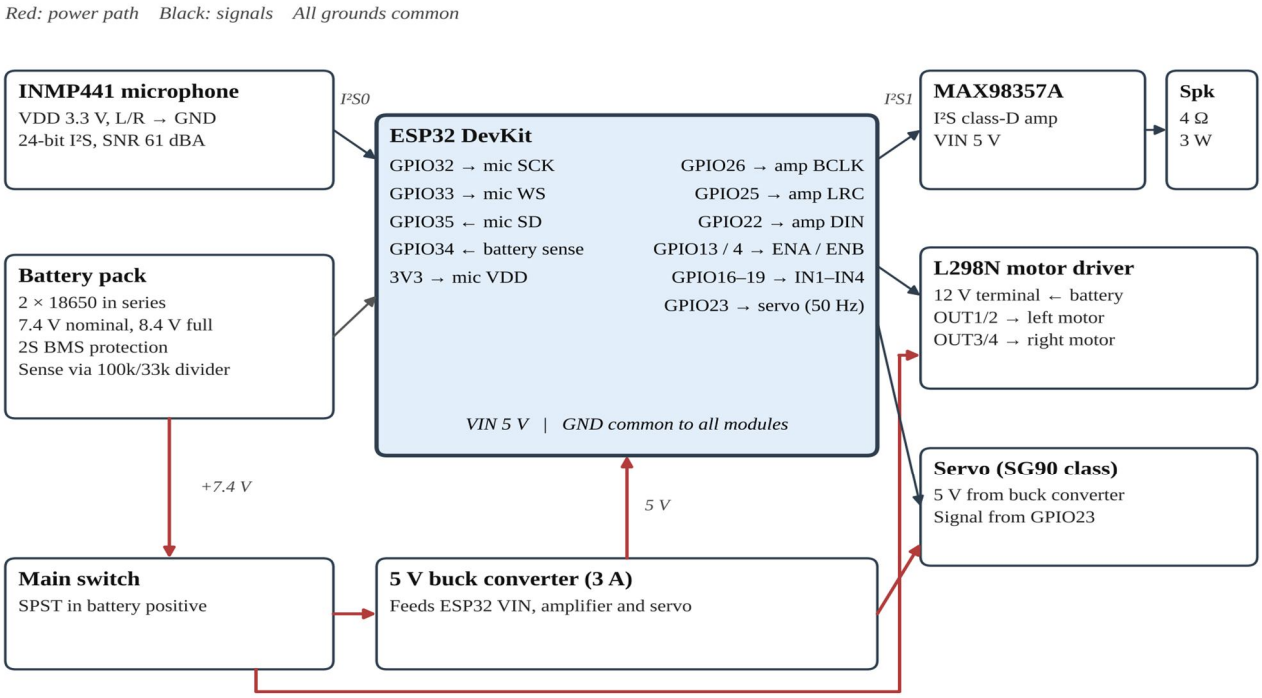


Fig. 2 Wiring and power distribution of the robot

The microphone and the amplifier use the ESP32's two separate I²S peripherals [24], so the robot can listen for "stop" while it is speaking. The INMP441 outputs 24-bit samples in 32-bit frames; the firmware keeps the upper 16 bits. Its L/R pin is tied to ground so that it transmits on the left channel.

The battery feeds the L298N motor terminal directly and the 5 V buck converter through the main switch. The servo and the amplifier draw current peaks, so they are powered from the buck converter rather than from the ESP32's on-board regulator, and the INMP441 is powered from the ESP32's 3.3 V output to keep its supply quiet. A 100 kΩ/33 kΩ divider scales the full-charge voltage of 8.4 V to about 2.1 V for the ESP32's analogue input, so the firmware can warn the user and park the robot when the battery is low. The L298's bipolar output stage drops roughly 2 V or more at typical motor currents [23], so the motors see about 5 V from the 7.4 V pack.

V. SOFTWARE DESIGN

A. Interaction Pipeline

Fig. 3 shows the interaction loop. The ESP32 continuously analyses one-second windows of microphone audio with a compact keyword-spotting model of the kind described in [10] and trained on data such as [11]. When the wake word is detected, the servo turns the head and a short tone confirms that the robot is listening. The ESP32 then streams audio to the server until a voice-activity detector finds 0.6 s of silence or 4 s have elapsed.

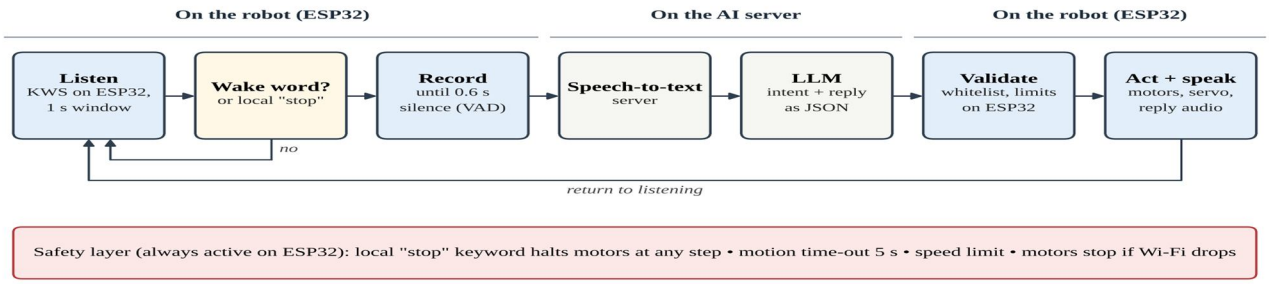


Fig. 3 Voice interaction pipeline and always-on safety layer

On the server, a Whisper-type model [12] transcribes the audio. The transcript, the robot's current state and the recent dialogue are passed to the LLM with a system prompt that defines the robot's personality and its available functions, following the function-library approach of [15], [16], [19]. The LLM must answer in the JSON format below, and its reply text is converted to speech and streamed back as 16 kHz audio.

{"say": "Sure, moving forward.", "action": "move", "params": {"direction": "forward", "speed": 60, "duration_ms": 1500}}

B. Robot primitives and safety layer

The robot exposes a small set of primitives (Table 4). The LLM can combine them into sequences, such as "turn around and say hello", but cannot invent new ones. The validator on the ESP32 parses the JSON and rejects any action that is not in the whitelist, clips speed and duration to their limits, and replies with a spoken error instead of moving if the message is malformed.

Table 4 Robot action primitives and limits enforced on the ESP32

Action	Parameters	Limit enforced on robot
move	direction (forward/backward), speed (%), duration (ms)	Speed ≤ 70%; duration ≤ 5000 ms; ramp-up 300 ms
turn	direction (left/right), angle (°)	Angle ≤ 360°; turn speed ≤ 50%
stop	—	Immediate; overrides all other actions
look	angle (°)	Servo range 20–160°
gesture	nod, shake, wiggle	Pre-defined servo/motor patterns only
say	text (reply audio from server)	Volume limit; interrupted by "stop"
status	—	Reports battery, Wi-Fi strength, last action

The safety layer runs independently of the server. A local "stop" keyword halts the motors at any point, including while the robot is speaking. Every motion ends after its validated duration even if no further message arrives. If the WebSocket connection drops during a motion, the motors stop immediately, and if the battery voltage falls below 6.4 V, motion is disabled and the robot announces that it needs charging.

C. Motor Control

Each motor is driven by an L298N channel: two GPIO pins set the direction, and an LEDC PWM signal on the enable pin sets the speed. For a differential-drive robot with wheel separation b, left and right wheel speeds v_L and v_R give the forward speed v and turning rate ω in (1). A turn of angle θ at equal and opposite wheel speeds v_w takes the time t_{turn} in (2), which the firmware uses to convert the "turn" angle into a motor run time; the effective v_w is calibrated once for the floor surface.

$$v = (v_R + v_L) / 2, \quad \omega = (v_R - v_L) / b \tag{1}$$

$$t_{turn} = \theta\, b / (2\, v_w) \tag{2}$$

VI. EXPERIMENTAL METHODOLOGY

A. Test set-up

The robot is tested indoors on a smooth floor. The AI server is a laptop on the same Wi-Fi network; tests are repeated with a cloud server to measure the effect of internet delay. A sound-level meter sets background noise, and a smartphone camera at 60 frames per second records the robot so that response times can be measured from the moment the user stops speaking to the first motor or audio output.

B. Test scenarios

1) Wake-word and keyword accuracy: at least 10 speakers say the wake word and "stop" 20 times each at 0.5 m, 1 m and 2 m, in quiet conditions (about 35 dBA) and with background noise (about 60 dBA).

2) False wake-ups: play one hour of speech and music with no wake word and count activations.

3) Fixed commands: "move forward", "turn left", "look right" and similar, 20 trials each, scored for correct action.

4) Free-form requests: a set of 50 natural requests (for example, "come a little closer", "turn around and say hello") scored for correct intent and appropriate reply.

©IJRASET: All Rights are Reserved | SJ Impact Factor 7.538 | ISRA Journal Impact Factor 7.894 |

5) Safety: say "stop" during motion and during speech; disconnect Wi-Fi during motion; send malformed and out-of-range JSON from the server.
6) Battery life: run a scripted cycle of listening, speaking and driving until the low-battery cut-off.
7) User experience: participants rate ease of use, naturalness and trust on a five-point scale after a free interaction session.

C. Evaluation metrics

Recognition accuracy is computed from (3), where $N_{correct}$ is the number of correctly executed commands out of N_{total} trials. For transcription, the word error rate (WER) in (4) counts substitutions S, deletions D and insertions I against N reference words.

$$\text{Accuracy} = 100 \times N_{correct} / N_{total} \quad (\%) \tag{3}$$

$$\text{WER} = (S + D + I) / N \tag{4}$$

Table 5 Performance metrics and targets

Metric	How it is measured	Target
Wake-word detection rate	Detections ÷ utterances, per distance and noise level	≥ 95% at 1 m, quiet
False wake-ups	Activations per hour of non-target audio	≤ 1 per hour
Fixed-command accuracy	Eq. (3)	≥ 95%
Free-form intent accuracy	Correct intent ÷ 50 requests	≥ 85%
Transcription WER	Eq. (4) on recorded requests	≤ 10% (quiet)
Local "stop" latency	End of word → motors off (video)	≤ 0.5 s
End-to-end response time	End of speech → first reply audio or motion	≤ 3 s (local server)
Unsafe actions executed	Out-of-range or unknown actions that moved the robot	0
Battery runtime	Scripted mixed-use cycle	≥ 8 h

VII. RESULTS AND DISCUSSION

A. Audio data rate and memory

At 16 kHz and 16 bits per sample, speech produces 256 kbit/s, or 32 kB per second. The longest request of 4 s therefore needs 128 kB, which fits comfortably in the ESP32's 520 KB of SRAM alongside the keyword-spotting model and network buffers [20]. Streaming the audio in 100 ms chunks as it is recorded reduces the buffer further and lets the server begin transcription before the user has finished speaking. At 256 kbit/s the audio uses only a small fraction of a typical Wi-Fi link.

B. Latency budget

Table 6 sets out the latency budget used to design the pipeline. The stages on the robot are fixed by the design parameters; the server stages are targets that depend on the models and hardware chosen and must be measured. The local "stop" path avoids the server altogether, so its response is limited only by the keyword-spotting window and motor braking.

Table 6 Latency budget (design values and targets)

Stage	Where	Budget
End-of-speech detection (silence)	ESP32	0.6 s (design parameter)
Upload of final audio chunk	Wi-Fi	< 0.1 s
Speech-to-text	Server	Target ≤ 0.8 s
LLM response (first JSON)	Server	Target ≤ 1.0 s
Text-to-speech, first audio chunk	Server	Target ≤ 0.4 s
Validation and motor start	ESP32	< 0.05 s
Total to first reply or motion		Target ≤ 3 s
Local "stop" (no server)	ESP32	≤ 0.5 s

©IJRASET: All Rights are Reserved

C. Power budget and battery life

Battery life was estimated from typical power levels: about 0.6 W while listening with Wi-Fi active, 1.1 W while speaking, and 3.0 W while driving, including the losses in the L298N. Two 2600 mAh cells at 7.4 V store 19.2 Wh; allowing 85% usable capacity gives 16.3 Wh, and the buck converter is assumed to be 85% efficient. With 15% of the time speaking, the estimated runtime falls from 20.6 h with no driving to 13.4 h at 15% driving and 6.6 h at 60% driving (Fig. 4).

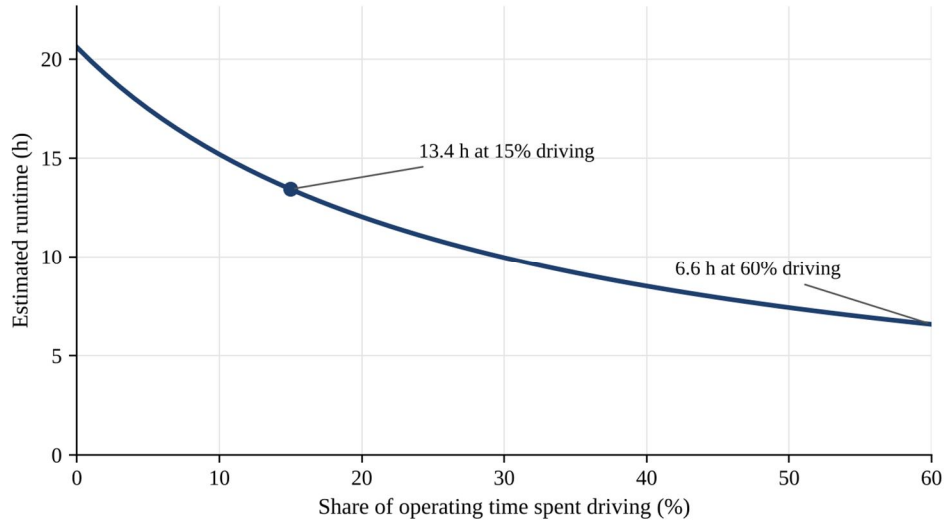


Fig. 4 Estimated battery runtime against the share of time spent driving

Driving dominates energy use, and the L298N contributes much of it: a drop of about 2 V at 7.4 V wastes roughly a quarter of the motor power as heat. A MOSFET driver such as the TB6612FNG would reduce this loss substantially and is recommended for a future version

.

D. Prototype measurements

Measured results are to be entered in Table 7 after the tests in Section VI are completed. Report the number of speakers and trials for each metric, and record failures and unsafe events as well as successes.

Table 7 Measured performance of the prototype (to be completed after testing)

Metric	Quiet (≈ 35 dBA)	Noisy (≈ 60 dBA)	Notes
Wake-word detection rate (%)	[Insert]	[Insert]	Speakers: [ ], trials: [ ]
False wake-ups (per hour)	[Insert]	[Insert]	
Fixed-command accuracy (%)	[Insert]	[Insert]	
Free-form intent accuracy (%)	[Insert]	[Insert]	50 requests
Transcription WER	[Insert]	[Insert]	
Local “stop” latency (s)	[Insert]	[Insert]	Mean / max
End-to-end response time (s)	[Insert]	[Insert]	Local / cloud server
Unsafe actions executed	[Insert]	—	Should be 0
Battery runtime (h)	[Insert]	—	Scripted cycle

E. Discussion

Compared with the low-cost robots in Table 1, the proposed design replaces a fixed command list with natural conversation while keeping the cheap hardware and adding offline stop control. Compared with the LLM-based systems, it moves all heavy computation off the robot and adds a hard safety boundary that the language model cannot cross, addressing the undefined actions reported in [16]. The servo-driven head that turns towards the user and the spoken confirmations provide the kind of non-verbal and verbal feedback that users expect from a talking robot [17], [18].

©IJRASET: All Rights are Reserved | SJ Impact Factor 7.538 | ISRA Journal Impact Factor 7.894 |

The design has limitations. Natural-language interaction needs a network connection and an AI server, and cloud services add internet latency and raise privacy questions about recorded speech, which a local server avoids. Motor and gear noise reaches the microphone during motion, so recognition while driving is expected to be weaker than when stationary. Without wheel encoders, distances and angles are estimated from time and will drift with battery voltage and floor surface. The battery-life and latency figures in Sections 7.2–7.3 are design estimates and must be confirmed by the measurements in Table 7.

VIII. CONCLUSION AND FUTURE SCOPE

This paper presented the design of an AI-based voice-controlled robot built from an ESP32, an INMP441 microphone, a MAX98357A amplifier with speaker, an L298N driver with two DC motors and wheels, a servo and a 2S 18650 battery pack with a switch. A hybrid architecture keeps wake-word and stop detection on the robot and sends free-form requests to a server for speech recognition, language understanding and speech synthesis. A JSON action interface and an on-robot validator let the robot hold natural conversations while every movement stays within fixed safety limits. A review of 19 studies showed that this combination of low cost, natural interaction and enforced safety has not been addressed together before.

Future work will complete the prototype measurements, add wheel encoders and an ultrasonic or time-of-flight sensor for obstacle avoidance, replace the L298N with a MOSFET driver, run speech recognition on an ESP32-S3 or a small local server for fully offline operation, and study long-term interaction with users such as children and elderly people.

IX. ACKNOWLEDGMENT

[Insert acknowledgment of sponsors, institutions or individuals who supported this work.]

REFERENCES

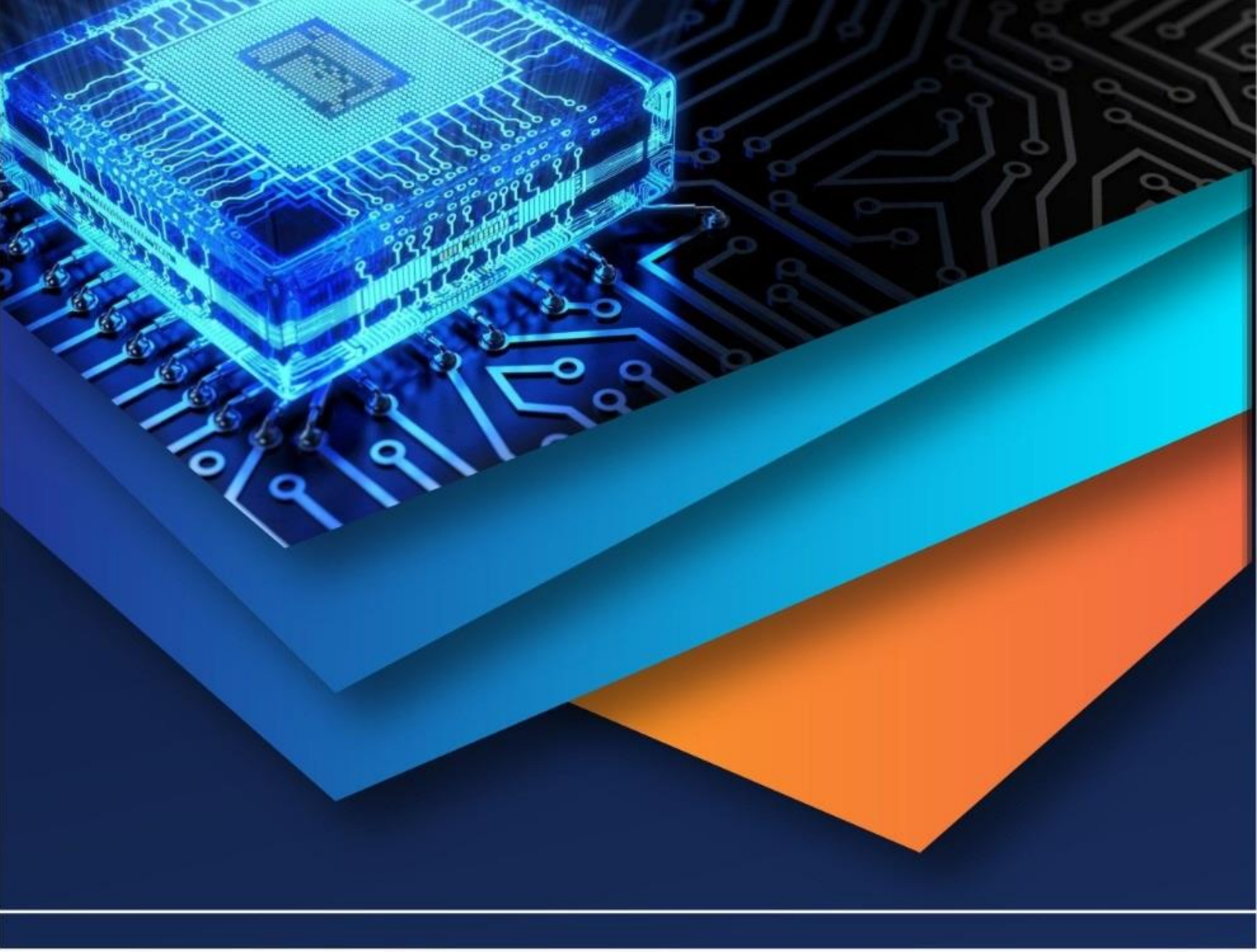
[1] M. A. Goodrich and A. C. Schultz, “Human–robot interaction: A survey,” Foundations and Trends in Human–Computer Interaction, vol. 1, no. 3, pp. 203–275, 2008, doi: 10.1561/1100000005.
[2] M. Marge et al., “Spoken language interaction with robots: Recommendations for future research,” Computer Speech & Language, vol. 71, Art. no. 101255, 2022, doi: 10.1016/j.csl.2021.101255.
[3] S. P. Pleshkova, Z. Zahariev, and A. B. Bekiarski, “Development of speech recognition algorithm and LabVIEW model for voice commands control of mobile robot motion,” in Proc. International Conference on High Technology for Sustainable Development (HiTech), Sofia, Bulgaria, 2018.
[4] A. Umar, M. A. Giwa, A. Y. Kassim, M. U. Ilyasu, I. Abdulwahab, E. E. Agbon, and M. T. Ogedengbe, “Development of an Android-based, voice-controlled autonomous robotic vehicle,” Engineering Proceedings, vol. 58, Art. no. 48, 2023, doi: 10.3390/ecsa-10-16026.
[5] A. Azahar, K. Rosda, N. Mustafa, and A. Alias, “Development of voice command control prototype for vehicle mobility,” Enhanced Knowledge in Sciences and Technology, vol. 3, no. 2, pp. 250–254, 2023, doi: 10.30880/ekst.2023.03.02.029.
[6] D. S. Pratama, Y. C. H. Yuwono, and M. R. Sufandi, “Prototipe mobil robot pemindah barang dengan voice control berbasis ESP32,” Jurnal LITEK: Jurnal Listrik Telekomunikasi Elektronika, vol. 22, no. 2, pp. 89–93, 2025, doi: 10.30811/litek.v22i2.76.
[7] M. Bakouri, M. Alsehaimi, H. F. Ismail, K. Alshareef, A. Ganoun, A. Alqahtani, and Y. Alharbi, “Steering a robotic wheelchair based on voice recognition system using convolutional neural networks,” Electronics, vol. 11, no. 1, Art. no. 168, 2022, doi: 10.3390/electronics11010168.
[8] J. Mendoza-Montoya, K. E. Angulo-Oviedo, E. Y. Turpo-Apaza, and R. Alfonte-Zapana, “Design of a prototype mechatronic wheelchair for people with motor disabilities,” in Proc. CITIE 2023: International Congress on Education and Technology in Sciences, CEUR Workshop Proceedings, vol. 3691, 2023.
[9] O. E. Adetoyi and O. P. Folami, “Spoken command recognition system for hands-free door control,” Journal of Science, Technology and Innovation Research, vol. 1 (Special Issue), pp. 413–420, 2025, doi: 10.51459/jostir.2025.1.Special-Issue.067.
[10] Y. Zhang, N. Suda, L. Lai, and V. Chandra, “Hello Edge: Keyword spotting on microcontrollers,” arXiv:1711.07128, 2017.
[11] P. Warden, “Speech Commands: A dataset for limited-vocabulary speech recognition,” arXiv:1804.03209, 2018.
[12] A. Radford, J. W. Kim, T. Xu, G. Brockman, C. McLeavey, and I. Sutskever, “Robust speech recognition via large-scale weak supervision,” in Proc. 40th International Conference on Machine Learning (ICML), PMLR, vol. 202, pp. 28492–28518, 2023.
[13] B. Ichter, A. Brohan, Y. Chebotar, C. Finn et al., “Do as I can, not as I say: Grounding language in robotic affordances,” in Proc. 6th Conference on Robot Learning (CoRL), PMLR, vol. 205, pp. 287–318, 2023.
[14] J. Liang, W. Huang, F. Xia, P. Xu, K. Hausman, B. Ichter, P. Florence, and A. Zeng, “Code as Policies: Language model programs for embodied control,” in Proc. IEEE International Conference on Robotics and Automation (ICRA), 2023, pp. 9493–9500.
[15] S. Vemprala, R. Bonatti, A. Bucker, and A. Kapoor, “ChatGPT for robotics: Design principles and model abilities,” Microsoft Research, Tech. Rep. MSR-TR-2023-8, 2023 (arXiv:2306.17582).
[16] A. Koubaa, “ROSGPT: Next-generation human–robot interaction with ChatGPT and ROS,” Preprints, 2023, doi: 10.20944/preprints202304.0827.v1.
[17] C. Y. Kim, C. P. Lee, and B. Mutlu, “Understanding large-language model (LLM)-powered human–robot interaction,” in Proc. ACM/IEEE International Conference on Human–Robot Interaction (HRI), 2024 (arXiv:2401.03217).
[18] C. Wang, S. Hasler, D. Tanneberg, F. Ocker, F. Joublin, A. Ceravola, J. Deigmoeller, and M. Gienger, “LaMI: Large language models for multi-modal human–robot interaction,” in Extended Abstracts of the CHI Conference on Human Factors in Computing Systems (CHI EA ’24), Honolulu, HI, USA, 2024 (arXiv:2401.15174).
[19] Y. Lakhnati, M. Pascher, and J. Gerken, “Exploring a GPT-based large language model for variable autonomy in a VR-based human–robot teaming simulation,” Frontiers in Robotics and AI, vol. 11, 2024, doi: 10.3389/frobt.2024.1347538.

©IJRASET: All Rights are Reserved

[20] Espressif Systems, “ESP32 series datasheet,” technical documentation.
[21] InvenSense (TDK), “INMP441 omnidirectional microphone with bottom port and I²S digital output,” datasheet.
[22] Analog Devices (Maxim Integrated), “MAX98357A/MAX98357B tiny, low-cost, PCM class D amplifier with class AB performance,” datasheet.
[23] STMicroelectronics, “L298 dual full-bridge driver,” datasheet.
[24] Philips Semiconductors, “I²S bus specification,” Feb. 1986, revised Jun. 1996.
[25] I. Fette and A. Melnikov, “The WebSocket protocol,” IETF RFC 6455, Dec. 2011.

Copyright & License:

© Authors retain the copyright of this article. This work is published under the Creative Commons Attribution 4.0 International License (CC BY 4.0), permitting unrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited.



10.22214/IJRASET

45.98

IMPACT FACTOR: 7.129

IMPACT FACTOR: 7.429

INTERNATIONAL JOURNAL FOR RESEARCH

IN APPLIED SCIENCE & ENGINEERING TECHNOLOGY

Call : 08813907089 (24*7 Support on Whatsapp)