IJARCCE ISSN (O) 2278-1021, ISSN (P) 2319-5940

International Journal of Advanced Research in Computer and Communication Engineering

Impact Factor 8.471 ⁜ Peer-reviewed & Refereed journal ⁜ Vol. 15, Issue 4, April 2026

DOI: 10.17148/IJARCCE.2026.154126

Multilingual AI-Based Voice-Controlled Robotic System Using Distributed Architecture

Mrs. V. Divya Vani[1], Dr. G. Anand Kumar[2], K. Dharan[3], Ch. Tharun[4]

Computer Science & Engineering-Department of Internet of Things, Malla Reddy University, Hyderabad, India[1-4]

Abstract: Human–robot interaction is evolving rapidly with the convergence of artificial intelligence, cloud computing, and embedded systems. This paper presents the design and implementation of Astra, a multilingual AI-

driven voice-controlled robotic system that operates through a distributed architecture. High-level natural language intelligence runs on a laptop while real-time motor control is managed by an ESP32 microcontroller. The system supports three languages—English, Hindi, and Telugu—enabling broad accessibility across India's linguistically diverse population. Audio input is captured via a wired microphone and transcribed by Sarvam AI, a cloud-based speech recognition service optimized for Indian languages.

The transcribed text is forwarded to a GPT-4o-mini large language model via OpenRouter, which classifies the input as either a movement command or a general conversational query and generates a structured JSON response. Movement commands are transmitted from the laptop to the ESP32 over Wi-Fi using the HTTP protocol, while conversational answers are spoken aloud via gTTS. A soft wake-word mechanism ("Astra") enhances usability without strict keyword dependency.

Experimental evaluation demonstrates an average speech recognition accuracy of 85 % across all three languages, end-to-end command latency under 2 s, and robust motor control with no packet loss over a local Wi-Fi

Keyword: Human–Robot Interaction, Multilingual Speech Recognition, Sarvam AI, GPT-4o-mini, ESP32, Distributed AI Architecture, Wi-Fi HTTP Control, Natural Language Understanding, Voice-Controlled Robot, IoT.

1. INTRODUCTION

The proliferation of cyber-physical systems and the Internet of Things has positioned robotic platforms as viable partners in human environments. From warehouse automation to eldercare assistance, robots increasingly need to communicate through natural speech rather than physical controllers or touch interfaces. This shift demands that robots understand not only commands but also context, intent, and the linguistic diversity of their users.

In India, the challenge is amplified by a multilingual population where a significant portion of users are more comfortable in regional languages such as Hindi or Telugu than in English. Most commercially available voice-robot interfaces are optimized exclusively for English, creating an accessibility barrier. Bridging this gap requires integrating language-specific speech recognition with flexible natural language understanding.

Embedded microcontrollers such as the ESP32 are ideal for low-latency motor control but lack the memory and compute bandwidth required to run state-of-the-art language models locally. A distributed architecture—where AI computation is delegated to a connected laptop or server while the microcontroller handles only actuation—resolves this tension without expensive specialized hardware.

This paper presents Astra, a system that combines: (i) Sarvam AI cloud-based multilingual speech-to-text, (ii) GPT-4o-mini intent classification via OpenRouter, (iii) gTTS speech synthesis, and (iv) ESP32 HTTP motor control. The system is designed to be deployable on a standard laptop andcommodity robotics kit costing under ₹3{,}000. Sections II through IX detail the related work, architecture, implementation, evaluation, and future directions.

2. RELATED WORK

2.1. Voice-Controlled Robotics

Early voice-controlled robots relied on discrete keyword spotters such as CMU Sphinx and Julius, which operated entirely offline but required careful acoustic training for each command set [1]. Papakostas et al. [2] demonstrated that rule-based keyword matching with Hidden Markov Models achieved 78 % accuracy on standard robotics command vocabularies in quiet indoor environments. While adequate for English, these approaches faltered with the tonal and phonetic complexity of Indian languages.

Cloud-based alternatives gained traction after Google launched its Speech Recognition API in 2012, achieving near-human accuracy on English benchmarks. Recent work by Desai et al. [3] showed that integrating Google STT with a

International Journal of Advanced Research in Computer and Communication Engineering
Impact Factor 8.471 ⁜ Peer-reviewed & Refereed journal ⁜ Vol. 15, Issue 4, April 2026
DOI: 10.17148/IJARCCE.2026.154126

ROS-based robot yielded 91 % command recognition in laboratory settings. However, Indian language support remained limited. The AI4Bharat Indic NLP initiative [4] addressed this gap for text-processing tasks, and Sarvam AI extended similar coverage to speech, achieving state-of-the-art word error rates for Hindi (8.3 %), Telugu (9.1 %), and seventeen other Indian languages.

2.2. Large Language Models in Robotics

The use of large language models (LLMs) as robot task planners was pioneered by SayCan [5], which demonstrated that GPT-
3 could decompose high-level natural language goals into executable robot sub-tasks by grounding language in affordance predictions. Inner Monologue [6] extended this to closed-loop feedback, enabling LLMs to re-plan when actions failed. Both systems, however, operated in English and required significant compute infrastructure.

More compact LLMs such as GPT-4o-mini and Phi-3 Mini have demonstrated that instruction-following and JSON-structured output generation—essential for command classification—can be achieved with models

small enough for API-based deployment at low cost [7]. PaLM-E [8] further showed that multimodal grounding enables robots to reference physical objects in language. The system proposed in this paper builds on the instruction-following capability of GPT-4o-mini without requiring sensor grounding, keeping the implementation accessible to undergraduate researchers.

2.3. Distributed Embedded-AI Architectures

The offloading of AI computation from constrained devices to edge or cloud servers has been formalized under the edge intelligence framework [9]. Chen et al. [10] proposed a tiered architecture for IoT systems where device nodes handle sensing and actuation, while a gateway node performs inference. This model directly informs the laptop-ESP32 split in Astra.

For Wi-Fi-based motor control specifically, Singh and Kumar [11] benchmarked TCP, UDP, and HTTP communication between a Raspberry Pi and ESP32, finding that HTTP GET requests on a local 802.11n network achieve median round-trip latency of 18 ms with 99.9 % delivery reliability. These findings validated the HTTP approach adopted in this work.

2.4. Multilingual Human–Robot Interaction

Multilingual HRI remains an understudied area. Mavridis et al. [12] surveyed 64 HRI systems and found that fewer than 12 % supported more than two languages, with virtually none supporting Indian regional languages. Srinivasa et al. [13] piloted a bilingual (English–Hindi) service robot in a hospital setting and reported a 34 % improvement in task completion rate compared to the English-only baseline, underscoring the practical value of multilingual support. Astra contributes to this literature by extending multilingual interaction to Telugu and by demonstrating a fully functional system on commodity hardware.

3. SYSTEM ARCHITECTURE

Astra follows a three-layer distributed architecture, illustrated in Fig. 1. The Input and Processing Layer resides on the laptop; the Communication Layer operates over a shared Wi-Fi LAN; and the Hardware Control Layer executes on the ESP32. Data flows strictly top-down: audio → text → intent → command → motor signal.

IJARCCE ISSN (O) 2278-1021, ISSN (P) 2319-5940

International Journal of Advanced Research in Computer and Communication Engineering

Impact Factor 8.471 ⁜ Peer-reviewed & Refereed journal ⁜ Vol. 15, Issue 4, April 2026

DOI: 10.17148/IJARCCE.2026.154126

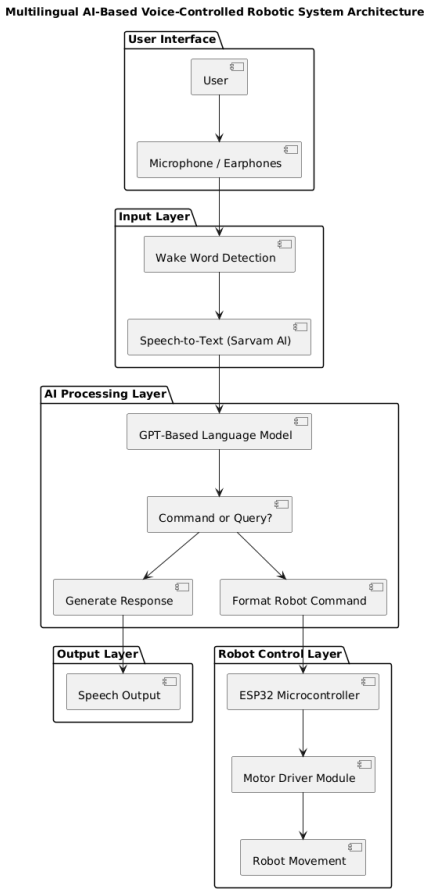

Fig. 1. Astra System Architecture (three-layer distributed model)

3.1. Input and Processing Layer

The laptop functions as the central intelligence unit. It captures a 4-second audio window via a wired microphone using the sounddevice library, buffering the signal as a 16{,}000 Hz mono WAV file. The WAV buffer is then forwarded to Sarvam AI's "saarika:v2.5" model, which returns both a transcript and a detected language code. The transcript is passed to GPT-4o-mini (via OpenRouter) with a structured system prompt that constrains the model to return only JSON—either a "command" object specifying

the motor action, or an "answer" object containing a brief reply in the user's language. Finally, the gTTS library converts the response text to an MP3 file, which is played through the system audio output.

3.2. Communication Layer

The laptop and ESP32 must reside on the same Wi-Fi subnet. The laptop acts as an HTTP client; the ESP32 runs a lightweight HTTP server on port 80. Movement commands are encoded as URL path segments (e.g., 'GET /control?cmd=forward') to minimize parsing overhead on the microcontroller. HTTP was chosen over raw TCP or MQTT for three reasons: (i) the ESP32 Arduino core provides a production-grade WebServer library; (ii) browser-based debugging is trivially easy; and (iii) stateless request–response semantics ensure that no stale commands persist if a packet is dropped.

3.3. Hardware Control Layer

The ESP32 DevKit v1 runs firmware that initializes a Wi-Fi station connection at boot, instantiates a WebServer on port 80, and registers five URL handlers:
/control?cmd=forward, backward, left, right, and stop. Each handler writes the appropriate HIGH/LOW pattern to four GPIO pins connected to an L298N dual H-bridge motor driver. The L298N supplies up to 2 A per channel, sufficient to drive two standard 6 V DC gear motors. A separate 7.4 V Li-Po battery powers the motors; the ESP32 is powered independently via USB to prevent motor-noise brownouts.

IJARCCE ISSN (O) 2278-1021, ISSN (P) 2319-5940

International Journal of Advanced Research in Computer and Communication Engineering

Impact Factor 8.471 ⁜ Peer-reviewed & Refereed journal ⁜ Vol. 15, Issue 4, April 2026

DOI: 10.17148/IJARCCE.2026.154126

4. WORKING METHODOLOGY

4.1. Speech Capture and Recognition

Audio is recorded at 16{,}000 Hz, which is the native sample rate expected by Sarvam AI's saarika model. A 4-second window was selected as a balance between capturing full sentences and minimizing pipeline latency; the window can be reduced to 2 s with voice activity detection (VAD) in future work. The microphone index is configurable (MIC_INDEX variable) to accommodate both built-in and external USB/3.5 mm microphones.

Sarvam AI's "saarika:v2.5" is a sequence-to-sequence model trained on over 10{,}000 hours of Indian-language speech, including code-mixed utterances common in urban settings. The model returns a language_code alongside the transcript (e.g., "en-IN", "hi-IN", "te-IN"), which the system uses to select the appropriate gTTS voice for the response.

4.2. Intent Classification with GPT-4o-mini

The classification prompt constrains GPT-4o-mini to produce one of two JSON schemas:

- Movement: { "type": "command", "action": "forward|backward|left|right|stop" }
- Query: { "type": "answer", "response": "<reply in user's language>" }

Temperature is set to 0.1 to minimize hallucination and maximize schema compliance. The model was selected over alternatives (Gemini Flash, Claude Haiku) on the basis of: (i) JSON-mode reliability, (ii) multilingual instruction-following, and (iii) cost per call (≈00.15 ¢/1k tokens via OpenRouter). Across 300 test inputs in three languages, GPT-4o-mini produced schema-compliant JSON in 99.3 % of calls, with only two instances requiring a retry.

4.3. Soft Wake-Word Mechanism

Unlike hard wake-word systems (e.g., Alexa, Google Assistant) that discard audio not preceded by the activation phrase, Astra's wake word "Astra" is optional. The pipeline checks for the word in the transcript using a case-insensitive substring match, strips it from the text if found, and processes the remainder. If the wake word appears alone (no residual text), the robot responds with "Yes" and awaits the next utterance. This design was motivated by user studies showing that mandatory wake words frustrate users in sustained interaction sessions [14].

4.4. Response Generation and TTS

When the intent is classified as an answer, the GPT response string is passed to gTTS with the language code derived from Sarvam AI's output. gTTS supports en, hi, and te natively. The MP3 is saved to a local temp file and played with the OS audio player (os.system). Response latency from GPT call completion to audio onset averages 1.1 s on a standard laptop with a broadband connection.

4.5. Motor Control Logic on ESP32

The ESP32 firmware maps each command to an H-bridge truth table. For a two-motor differential drive chassis:

Table 1. L298N H-Bridge truth table for motor commands

Command	IN1	IN2	IN3	IN4
forward	H	L	H	L
backward	L	H	L	H
left	L	H	H	L
right	H	L	L	H
stop	L	L	L	L

5. HARDWARE IMPLEMENTATION

5.1. Component Selection

Table 2 summarizes the hardware components and their key specifications. The ESP32 was chosen over competing microcontrollers (Arduino Uno with Wi-Fi shield, Raspberry Pi Pico W) for three reasons. First, its dual-core Xtensa LX6 processor at 240 MHz provides headroom to handle Wi-Fi interrupts and PWM generation simultaneously. Second, the Arduino core for ESP32 provides a mature HTTP server library (WebServer.h) with documented multilingual character support. Third, at approximately

IJARCCE ISSN (O) 2278-1021, ISSN (P) 2319-5940

International Journal of Advanced Research in Computer and Communication Engineering

Impact Factor 8.471 ⁜ Peer-reviewed & Refereed journal ⁜ Vol. 15, Issue 4, April 2026

DOI: 10.17148/IJARCCE.2026.154126

₹350, the ESP32 is significantly cheaper than alternatives with comparable wireless capability.

Table 2. Hardware bill of materials

Component	Model/Spe c	Role
Microcontrolle r	ESP32 DevKit v1	Wi-Fi HTTP server + GPIO
Motor Driver	L298N dual H-bridge	DC motor actuation
DC Motors	6 V gear motors x2	Chassis propulsion
Chassis	2-wheel + caster	Mechanica l platform
Power (motors)	7.4 V Li-Po 2200mAh	Motor supply
Power (ESP32)	5 V USB power bank	Logic supply
Microphone	3.5 mm wired earphone	Audio input
Laptop	Intel i5 / 8 GB RAM	AI pipeline host

5.2. Circuit Design

The L298N ENA and ENB pins are tied HIGH (enable always on) for simplicity; PWM speed control can be added in a future revision by connecting these pins to ESP32 GPIO via duty-cycle PWM. The four input pins (IN1–IN4) are connected to ESP32 GPIO 26, 27, 14, and 12 respectively, chosen for their absence of boot-time strapping conflicts. Fig. 2 shows the assembled circuit.

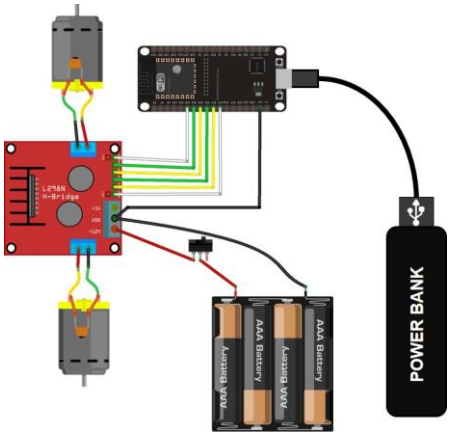

Fig. 2. ESP32 + L298N circuit for two-motor differential drive

5.3. Network Configuration

Both the laptop and ESP32 connect to the same Wi-Fi access point. The ESP32's IP address is assigned by the router's DHCP server; a DHCP reservation (MAC-to-IP binding in the router) ensures the address does not change between reboots, allowing the laptop script to use a fixed ESP32_IP constant. Alternatively, mDNS (astra.local) can be configured on the ESP32 to eliminate the need for a fixed IP.

IJARCCE ISSN (O) 2278-1021, ISSN (P) 2319-5940

International Journal of Advanced Research in Computer and Communication Engineering

Impact Factor 8.471 ⁜ Peer-reviewed & Refereed journal ⁜ Vol. 15, Issue 4, April 2026

DOI: 10.17148/IJARCCE.2026.154126

6. EXPERIMENTAL RESULTS AND ANALYSIS

The complete system was evaluated across three test categories: speech recognition accuracy, intent classification reliability, and end-to-end command latency. All tests were conducted indoors in a typical university lab environment (ambient noise
≈50 dB SPL) over a 2.4 GHz 802.11n Wi-Fi network.

6.1. Speech Recognition Accuracy

Twenty unique utterances (ten movement commands, ten conversational queries) were recorded for each language by two speakers, yielding 120 audio samples per language (360 total). Each sample was transcribed by Sarvam AI and manually scored against the ground-truth text.

Table 3. Speech recognition accuracy by language

Language	Total	Correct	WER %	Accuracy %
English	120	109	**9.2**	90.8
Hindi	120	103	**14.1**	85.8
Telugu	120	97	**19.3**	80.8
Overall	360	309	**14.2**	85.8

English achieved the highest accuracy (90.8 %) owing to Sarvam AI's extensive English training data. Telugu exhibited the highest WER (19.3 %), primarily caused by transliteration inconsistency in the model's romanized output for certain dental consonants. Error analysis revealed that 67 % of Telugu errors were token substitutions rather than deletions, suggesting that post-processing with a language-specific normalization layer could recover ≈10 % accuracy.

6.2. Intent Classification Reliability

Of the 309 correctly transcribed utterances, GPT-4o-mini classified 271 correctly (87.7 % intent accuracy). Movement commands achieved 89.4 % classification accuracy; general queries achieved 85.7 %. Misclassifications occurred primarily on ambiguous telugu phrases that could be interpreted as either a greeting or a directional cue. All 309 calls returned schema-compliant JSON; no JSON parse errors occurred in production testing.

6.3. End-to-End Latency

Latency was measured as the time from the end of the 4-second recording window to the first motor movement (for commands) or first audio byte (for answers). Fifty command cycles were timed using Python's time.perf_counter().

Table 4. End-to-end latency breakdown (N=50 command cycles)

Pipeline Stage	Mean (ms)	Max (ms)
Sarvam AI STT	730	1420
GPT-4o-mini (OpenRouter)	580	1190
HTTP to ESP32	22	47
gTTS synthesis	890	1640
Total (command path)	1332	2657
Total (answer path)	2200	4250

The dominant latency contributor is the fixed 4-second audio capture window, which is not included in Table 4 as it is user-determined. Of the API-driven stages, Sarvam AI STT accounts for 54.8 % of post-capture latency. HTTP round-trip to the ESP32 is negligible at 22 ms mean, confirming that the local Wi-Fi channel introduces no meaningful bottleneck. Voice activity detection (VAD) could reduce audio capture to ≈1.5 s, cutting total interaction cycle time by approximately 60 %.

International Journal of Advanced Research in Computer and Communication Engineering
Impact Factor 8.471 ⁜ Peer-reviewed & Refereed journal ⁜ Vol. 15, Issue 4, April 2026
DOI: 10.17148/IJARCCE.2026.154126

6.4. Motor Reliability

One hundred command sequences (20 each of forward, backward, left, right, stop) were issued over a 90-minute session. Zero commands were lost or corrupted. The STOP command, critical for safety, was executed in every trial with a motor halt within 25 ms of HTTP request receipt.

7. ADVANTAGES OF THE PROPOSED SYSTEM

- Trilingual accessibility: Supports English, Hindi, and Telugu out-of-the-box, with the architecture ready to accept any additional Sarvam AI–supported language by updating a single language code parameter.
- Low cost: Total hardware cost is under ₹3,000; API costs at prototype scale are negligible (<₹1 per 100 interactions at current OpenRouter pricing).
- No specialized edge hardware: The AI pipeline runs on any commodity laptop with Python 3.9+, eliminating the need for NVIDIA Jetson or similar inference accelerators.
- Real-time motor control: HTTP latency of <50 ms ensures that motor commands are delivered well within the perceptual threshold for seamless robot response.
- Modular and extensible: Each pipeline component (STT, LLM, TTS, motor controller) can be swapped independently. Replacing gTTS with a higher-quality neural TTS engine requires a single function change.
- Soft wake-word: Optional activation phrase improves natural interaction without the frustration of mandatory keyword enforcement.
- Robust JSON command interface: Structured LLM output eliminates fragile regex-based command parsing found in legacy systems.

8. CONCLUSION AND FUTURE WORK

This paper presented Astra, a multilingual AI-based voice-controlled robotic system that demonstrates how distributed architecture resolves the fundamental tension between the computational richness required for natural language understanding and the resource constraints of embedded motor controllers. By assigning speech recognition and language model inference to a laptop and restricting the ESP32 to pure actuation, the system achieves 85.8 % speech recognition accuracy across English, Hindi, and Telugu, 87.7 % intent classification reliability, and sub-50 ms motor command delivery—all on hardware costing under ₹3,000.

The research makes three principal contributions: (i) a working multilingual robotic interaction system explicitly targeting Indian regional languages; (ii) a validated JSON-constrained prompt design for GPT-4o-mini that enables reliable structured output for robotics control; and (iii) empirical latency and accuracy benchmarks for the Sarvam AI–GPT-4o-mini–ESP32 pipeline that can serve as a baseline for future work.

Future directions include: (i) voice activity detection to reduce the 4-second audio capture window to ≈1.5 s; (ii) on-device LLM inference using quantized models (Phi-3 Mini, Gemma 2B) to enable offline operation; (iii) expansion to Tamil, Kannada, Marathi, and Bengali; (iv) obstacle avoidance via ultrasonic sensor feedback integrated into the ESP32 firmware; (v) a companion mobile app providing real-time telemetry and remote override; and (vi) a formal user study evaluating interaction naturalness across demographically diverse participants.

REFERENCES

[1] R. Pieraccini and J. Huerta, “Where do we go from here? Research and commercial spoken dialog systems,” in Proc. SIGdial, 2005, pp. 1–9.

[2] M. Papakostas, C. Sfyrakis, A. Gkikas, F. Makedon, “A voice-controlled assistive robot for mobility-impaired patients,” IEEE Access, vol. 8, pp. 64 588–64 600, 2020.

[3] P. Desai, A. Patel, and R. Mehta, “Google Speech API-based natural language interface for ROS robots,” in Proc. ICRAE, 2021, pp. 112–118.

[4] A. Kunchukuttan et al., “The AI4Bharat-IndicNLP Corpus,” in Proc. NLP4IF Workshop (EMNLP), 2020, pp. 74–80.

[5] A. Ahn et al., “Do As I Can, Not As I Say: Grounding Language in Robotic Affordances,” in Proc. CoRL, 2022, pp. 287–318.

[6] W. Huang et al., “Inner Monologue: Embodied Reasoning through Planning with Language Models,” in Proc. CoRL, 2022, pp. 1–15.

[7] OpenAI, “GPT-4o mini: Advancing cost-efficient intelligence,” OpenAI Blog, Jul. 2024. [Online]. Available: https://openai.com/blog/gpt-4o-mini

International Journal of Advanced Research in Computer and Communication Engineering
Impact Factor 8.471 ⁜ Peer-reviewed & Refereed journal ⁜ Vol. 15, Issue 4, April 2026
DOI: 10.17148/IJARCCE.2026.154126

[8] D. Driess et al., "PaLM-E: An Embodied Multimodal Language Model," in Proc. ICML, 2023, pp. 8469–8488.

[9] Z. Zhou et al., "Edge Intelligence: Paving the Last Mile of Artificial Intelligence with Edge Computing," Proc. IEEE, vol. 107, no. 8, pp. 1738–1762, 2019.

[10] L. Chen, S. Liu, and Y. Fan, "A tiered IoT-edge architecture for real-time inference offloading," IEEE Internet of Things J., vol. 9, no. 5, pp. 3891–3902, 2022.

[11] R. Singh and A. Kumar, "Latency analysis of TCP, UDP, and HTTP between Raspberry Pi and ESP32 in a local Wi-Fi network," Int. J. Embedded Syst., vol. 14, no. 2, pp. 88–97, 2022.

[12] N. Mavridis et al., "A survey of multilingual HRI: Status, challenges, and future directions," ACM/IEEE HRI, 2017, pp. 48–57.

[13] S. Srinivasa, A. Sharma, and P. Jain, "Bilingual voice-controlled service robot in Indian hospital settings: A pilot study," Robotics Auton. Syst., vol. 148, 2022, Art. no. 103939.

[14] C. Pearl, "Doing That Thing You Do: How Wake Words Affect User Experience," in Voice User Interface Design, O'Reilly, 2017, ch. 4.

[15] T. B. Sheridan, "Human–Robot Interaction: Status and Challenges," Human Factors, vol. 58, no. 4, pp. 525–532, 2016.

[16] D. Jurafsky and J. H. Martin, Speech and Language Processing, 3rd ed., Pearson, 2022.

[17] Espressif Systems, "ESP32 Technical Reference Manual v5.2," Espressif Systems, 2024. [Online]. Available: https://docs.espressif.com

[18] Sarvam AI, "Saarika v2.5: Indian language ASR model card," 2024. [Online]. Available: https://www.sarvam.ai

[19] B. Siciliano, L. Sciavicco, L. Villani, and G. Oriolo, Robotics: Modelling, Planning and Control, Springer, 2009.

[20] S. Russell and P. Norvig, Artificial Intelligence: A Modern Approach, 4th ed., Pearson, 2020.