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 3, March 2026

DOI: 10.17148/IJARCCE.2026.15380

"AI ASSISTED PERSONAL ROBOT"

Prof. Bhagyashree Wankhede[1], Aarju S. Dhurve[2], Lokesh P. Kothe[3], Mitali S. Mankar[4], Pranjali A. Nagose[5], Swejal D. Patil[6]

lecturer at Department of Electronics & Telecommunications Engineering in NIT Polytechnic, Nagpur[1]

Students at Department of Electronics & Telecommunications Engineering at NIT Polytechnic Nagpur[2-6]

Abstract: The rapid growth of intelligent personal assistants has largely been driven by cloud-based artificial intelligence services, which inherently suffer from privacy concerns, network dependency, and non-deterministic latency. These limitations become critical in mobile robotic platforms where timely and reliable responses are essential for safe navigation and natural interaction. This paper presents the design and implementation of an AI-assisted personal robot that integrates a locally hosted Large Language Model (LLM), specifically Llama 3, with an ESP32-S3 microcontroller performing edge computing. The proposed system employs an I2S-based digital audio acquisition pipeline using an INMP441 microphone, a WebSocket-based bidirectional communication framework between the robot and a local AI server, and an ultrasonic-based safety interrupt mechanism for collision avoidance. Speech is captured as raw PCM data on the ESP32-S3, transmitted over Wi-Fi to a local workstation running Whisper for speech-to-text conversion and Llama 3 for intent understanding, and returned as structured JSON commands for real-time motor control. Experimental evaluation demonstrates sub-second end-to-end latency, robust command recognition in the presence of motor noise, and reliable indoor navigation under constrained network conditions. The results indicate that local LLM-driven robotics on commodity hardware can achieve low-latency, privacy-preserving interaction without dependence on cloud infrastructure, providing a viable architectural template for future embodied AI systems

Keywords: ESP32-S3, Edge AI, Local LLM, I2S Protocol, Web Socket Communication, Privacy-Preserving Robotics

I. INTRODUCTION

Personal robotics has evolved from simple line-following and obstacle-avoiding platforms to sophisticated systems capable of speech interaction, contextual reasoning, and autonomous navigation in human environments. Historically, such intelligent behaviors have depended heavily on cloud-based services that offer large-scale computation and advanced machine learning models. Voice assistants such as Amazon Alexa, Google Assistant, and similar systems demonstrate that natural language interfaces can significantly improve user engagement and accessibility. However, when these paradigms are extended to physical robots, the inherent limitations of cloud connectivity become evident, particularly in applications demanding real-time responsiveness and stringent privacy guarantees.Modern robotic assistants deployed in homes, laboratories, or educational environments often rely on centralized servers to perform speech recognition, natural language understanding, and high-level decision making. While this approach reduces the computational burden on embedded hardware, it introduces several challenges. First, cloud-based processing requires continuous internet connectivity, leading to reduced reliability in environments with intermittent or poor network coverage. Second, transmitting raw or processed audio data to remote servers creates substantive privacy risks, as sensitive conversations may be exposed or logged beyond the user's control. Third, network round-trip delays often result in latencies of several seconds, which negatively affect user experience and can even compromise safety in dynamic scenarios where the robot must react quickly to obstacles or human instructions.

Edge computing has emerged as a promising paradigm to address these issues by offloading computation from remote servers to local devices or nearby processing nodes. The ESP32-S3 microcontroller family represents a class of low-cost, low-power devices capable of handling real-time I/O, Wi-Fi communication, and signal processing tasks while maintaining modest energy consumption. However, despite significant improvements in microcontroller capabilities, running full-scale Large Language Models (LLMs) directly on such constrained devices remains impractical. At the same time, recent advances in model quantization and optimized inference frameworks have enabled models like Llama 3 to operate efficiently on consumer-grade workstations without dedicated GPUs, providing a middle ground between fully cloud-based and fully embedded intelligence.

The motivation behind this work is to design a personal robotic system that combines the deterministic real-time behavior of an embedded controller with the semantic and contextual richness of a modern LLM, all within the scope

International Journal of Advanced Research in Computer and Communication Engineering
Impact Factor 8.471 ⁜ Peer-reviewed & Refereed journal ⁜ Vol. 15, Issue 3, March 2026
DOI: 10.17148/IJARCCE.2026.15380

of a local network. The proposed system separates responsibilities such that the ESP32-S3 acts as the sensory and actuation interface, handling audio acquisition, motor control, and safety-critical monitoring, while a local workstation performs computationally intensive tasks such as speech recognition and natural language reasoning. This separation enables the robot to preserve user privacy by keeping all data within the premises, minimize latency by avoiding external network hops, and maintain operational autonomy even when internet access is unavailable.

Existing approaches to voice-controlled robots predominantly use analog microphones directly interfaced to microcontroller ADCs and rely on cloud APIs for speech recognition and intent handling. Such systems exhibit susceptibility to electromagnetic interference from DC motors and motor drivers, resulting in degraded audio quality and consequently reduced recognition accuracy. Furthermore, the integration of asynchronous AI responses with hard real-time motor control is often ad hoc, leading to unpredictability in navigation behavior. These limitations motivate the adoption of digital audio interfaces, dedicated safety mechanisms, and structured communication protocols in the proposed architecture.

This paper presents a complete design cycle, from conceptualization to implementation and testing, of an AI-assisted personal robot built on an ESP32-S3 platform and a local LLM server. The system leverages the I2S protocol for robust digital audio capture, WebSockets for low-latency bidirectional communication, Whisper for on-device speech-to-text processing, and Llama 3 hosted via Ollama for local language understanding. The robot is capable of executing natural language navigation commands, such as moving forward, turning by specified angles, and stopping upon encountering obstacles, while a dedicated ultrasonic sensor and interrupt-based logic ensure that safety constraints override high-level AI directives. The remainder of this paper details the relevant literature, system concept, architecture, methodology, experimental evaluation, and conclusions drawn from the implementation.

II. LITERATURE REVIEW

Research in voice-controlled robotics has traditionally been closely associated with cloud-based speech recognition services. Sharma investigated voice recognition in mobile robots using Google Speech-to-Text and found that while recognition accuracy exceeded 95% under ideal network conditions, the end-to-end delay, including network latency and processing time, typically exceeded 3 s. This delay was acceptable for non-critical commands but was identified as unsuitable for real-time navigation tasks that require rapid reaction to dynamic changes in the environment. The study highlighted the tension between high recognition accuracy and real-time responsiveness when adopting cloud services in mobile robots.

Kumar explored edge computing strategies with the ESP32 microcontroller for Internet of Things (IoT) applications, demonstrating that computationally lightweight models and pre-processing tasks could be executed locally to reduce data transmission to the cloud. The work focused on sensor data aggregation and simple anomaly detection rather than complex natural language processing. However, the conclusions indicated that even resource-constrained microcontrollers could significantly offload cloud dependence when carefully architected. This insight directly informs the design philosophy of the present work, where the ESP32-S3 is used for time-sensitive operations and pre-processing while relying on a more capable local node for heavy inference.

Several researchers have examined local speech recognition frameworks to reduce latency and enhance privacy. Park et al. implemented an offline keyword spotting system on a low-power microcontroller and evaluated different feature extraction and classification strategies. The study concluded that while small models can reliably detect a limited set of predefined phrases, scaling to open-vocabulary commands requires more powerful hardware or dedicated neural accelerators. Similarly, Li and Chen employed compressed recurrent networks for on-device speech recognition on smartphones, highlighting the trade-offs between model size, memory footprint, and recognition performance. These works illustrate the difficulty of deploying full speech recognition pipelines on microcontrollers, motivating hybrid architectures that combine embedded control with external but local processing.

In the area of robotic navigation, Nguyen and Patel implemented obstacle avoidance using ultrasonic sensors and a finite-state machine controller on small indoor robots. Their work showed that simple distance thresholds and state transitions can produce robust collision avoidance behaviors in structured environments. However, the commands controlling such robots were limited to preprogrammed routines rather than dynamic natural language instructions. Parallel research by Singh et al. integrated Wi-Fi-based control with voice commands using smartphone speech recognition, introducing additional latency and network dependency in the control loop.

DOI: 10.17148/IJARCCE.2026.15380

Recent progress in Large Language Models has opened new possibilities for embodied AI. Brown et al. demonstrated that large transformer-based models could perform few-shot learning across various language tasks, though their deployment was originally limited to massive cloud infrastructure. Subsequently, open-source projects such as Llama introduced more compact models that could be fine-tuned and executed on local hardware, especially when combined with quantization strategies. Sanh et al. explored model distillation and parameter-efficient fine-tuning methods to make large models more accessible on edge devices and personal workstations.

Edge robotics integrating LLMs is still an emerging research area. Lee et al. proposed a framework where a mobile robot connected to a local PC hosting a transformer model for task planning and dialogue. Their architecture achieved lower latency compared to cloud-based solutions but did not employ digital audio interfaces or dedicated safety interrupts, and the analysis of electromagnetic interference on audio signals was limited. In addition, many existing implementations use analog microphones connected to the microcontroller's ADC, as described by Alvarez and Reddy , who reported considerable degradation in audio quality when DC motors are active, due to coupling of motor noise into the analog front-end.

Collectively, these studies indicate several gaps that the proposed work seeks to address. First, there is a need for a tightly integrated architecture in which a microcontroller provides real-time control and safety functions while a local workstation hosts advanced language and speech models. Second, digital audio acquisition via I2S can mitigate noise issues associated with analog microphones in robotic platforms, but relatively few works have experimentally validated this in the context of voice-controlled navigation. Third, there is limited research on using WebSocket-based communication to provide low-latency, bidirectional, structured commands between a robot and a local AI server. By combining these elements, the present work establishes a concrete design pattern for privacy-preserving, low-latency personal robotic assistants powered by local LLMs.

III. SYSTEM CONCEPT

The proposed system is conceptually structured as a hybrid edge-AI robotic assistant in which perception, cognition, and actuation are distributed across two primary domains: an embedded control layer implemented on the ESP32-S3 microcontroller, and a local AI processing layer hosted on a workstation. The core concept is to treat the robot as a physical embodiment of an intelligent assistant, capable of understanding voice commands, reasoning about high-level actions, and executing corresponding movements within an indoor environment while ensuring user privacy and operational safety.

At the highest level, the signal flow begins with the user's speech, which is captured by a digital I2S microphone mounted on the robot chassis. The microphone outputs pulse-density modulated data, which the ESP32-S3 converts into raw PCM samples via its integrated I2S peripheral and Direct Memory Access (DMA) buffers. These audio frames are then packetized and transmitted over Wi-Fi using a WebSocket client running on the ESP32-S3. On the local AI server, a WebSocket server receives the audio stream, reconstructs the PCM data, and forwards it to the Whisper speech-to-text engine, which produces a textual transcript of the user's command.

Once the command text is available, it is passed to a locally hosted Llama 3 instance, managed by the Ollama framework, along with a constrained prompt that instructs the model to output structured JSON describing the intended robot action. This might include fields such as direction, distance, or rotation angle. The local AI server does not directly control the motors but instead transmits this JSON payload back to the ESP32-S3 via the same WebSocket connection. The microcontroller parses the JSON and translates the high-level instructions into specific motor control signals and timing parameters, which are then applied to the motor driver circuitry.

In parallel with this perception and cognition loop, an independent safety subsystem operates on the robot. An ultrasonic distance sensor continuously monitors the distance to obstacles ahead of the robot. The ESP32-S3 is configured to trigger an interrupt when the measured distance dd falls below a predefined threshold, typically d<20 cmd<20cm. When this condition is met, the interrupt handler immediately overrides any ongoing movement commands, stopping the motors and entering a safe state. This ensures that even if the AI server generates erroneous commands or experiences delays, the robot maintains a conservative safety margin in its physical interactions.

The conceptual design therefore emphasizes three cooperative subsystems: an audio perception subsystem responsible for capturing and transmitting high-quality speech data, a cognitive subsystem hosted locally that performs speech recognition and language understanding, and an actuation and safety subsystem that controls motion and ensures safe operation. Energy flow is primarily concentrated in the embedded layer, where a battery supplies power to the ESP32-

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 3, March 2026

DOI: 10.17148/IJARCCE.2026.15380

S3, motor drivers, DC motors, ultrasonic sensor, microphone, and audio amplifier. The local workstation is powered independently and does not affect the robot's autonomy, except in terms of AI functionality availability. When the AI server is offline, the robot can still respond to basic preprogrammed commands but will not perform advanced language reasoning.

IV. OBJECTIVES

The primary objective of the project is to design and implement a personal robotic assistant that leverages a locally hosted LLM to interpret user voice commands while relying on an ESP32-S3 microcontroller for real-time control and safety, all within a privacy-preserving and low-latency framework. This central goal is motivated by the desire to bridge the gap between high-level natural language understanding and low-level robotic actuation without resorting to cloud infrastructure.

Secondary objectives include the development of a robust digital audio acquisition pipeline using the I2S protocol to mitigate analog noise issues commonly observed in mobile robots, and the integration of a WebSocket-based client-server communication layer to provide efficient, bidirectional data exchange between the robot and the local AI server. Another important objective is the implementation of an ultrasonic-based safety mechanism employing hardware interrupts to ensure that obstacle avoidance always takes precedence over AI-generated commands, thereby improving operational safety in indoor environments.

An additional objective is to achieve measurable improvements over conventional cloud-based voice-controlled robots in terms of latency, reliability, and privacy. Specifically, the system aims to achieve an end-to-end response time L<1000 msL<1000ms from speech command completion to motor movement initiation, while maintaining high command recognition accuracy at typical indoor distances and in the presence of motor noise. By satisfying these objectives, the project intends to demonstrate that local LLM integration combined with edge computing on microcontrollers can form a practical and scalable basis for future intelligent robotic platforms.

V. SYSTEM ARCHITECTURE / PROTOTYPE

The system architecture is organized into hardware and software layers interconnected through clearly defined data paths. At the hardware level, the core of the robot is the ESP32-S3 microcontroller, selected for its dual-core processing capability, integrated Wi-Fi, native I2S interface, and sufficient RAM resources to handle DMA-based audio buffering and WebSocket communication simultaneously. The ESP32-S3 is mounted on a custom carrier board that provides connections to an INMP441 digital microphone, a MAX98357A I2S audio amplifier, an HC-SR04 or equivalent ultrasonic distance sensor, an L298N dual H-bridge motor driver, and a set of four DC motors forming a 4-wheel drive (4WD) chassis.

The INMP441 microphone interfaces with the ESP32-S3 via the I2S bus, using dedicated pins for bit clock (BCLK), word select (WS), and data input (DIN). The ESP32-S3 configures the I2S peripheral in master receiver mode, generating the appropriate sampling clock and capturing audio at a typical rate such as 16 kHz16kHz. The raw PCM

DOI: 10.17148/IJARCCE.2026.15380

audio is stored in circular DMA buffers, minimizing CPU involvement in data movement and allowing the main application to perform other tasks concurrently. On the output side, the MAX98357A amplifier connects to the I2S output pins of the ESP32-S3 and drives a small speaker, enabling the robot to play synthesized speech or audio feedback if desired. Although the present implementation primarily focuses on voice command input, the hardware provisions support full-duplex audio interaction.

The ultrasonic sensor is positioned at the front of the chassis and connects to the ESP32-S3 using digital trigger and echo pins. The microcontroller measures the round-trip time of ultrasonic pulses to estimate the distance to obstacles. This measurement is performed at regular intervals or via timer-driven routines, and an interrupt is configured to fire when the distance drops below a predefined threshold. The interrupt handler immediately commands the motor driver to stop all motion. The L298N motor driver receives pulse-width modulated (PWM) signals and direction control inputs from the ESP32-S3 GPIO pins and translates them into appropriate voltage levels for the DC motors. This configuration allows independent control of left and right motor pairs, enabling forward movement, reverse motion, and in-place rotation.

The robot is powered by a rechargeable battery pack, typically rated around 7.4 V7.4V to 12 V12V, which supplies the motor driver and is stepped down via voltage regulators to provide a stable 5 V5V or 3.3 V3.3V rail for the ESP32-S3, microphone, ultrasonic sensor, and audio amplifier. Power distribution is designed to minimize noise coupling between the high-current motor circuits and the low-voltage digital and analog circuitry, aided by decoupling capacitors and careful routing to reduce electromagnetic interference.

On the software side, the ESP32-S3 runs firmware developed in C++ using the Arduino framework and associated libraries for I2S, Wi-Fi, and WebSocket communication. The firmware includes three principal modules: the Audio Capture Module, which initializes and manages the I2S interface and DMA buffers; the Communication Module, which establishes and maintains a persistent WebSocket connection with the local AI server; and the Execution Module, which parses incoming JSON commands from the server and translates them into motor control actions. Additionally, an interrupt-driven Safety Module monitors the ultrasonic sensor readings and can preempt motor commands when necessary.

The local AI server is implemented in Python and runs on a workstation within the same local network as the robot. It uses a WebSocket server library to accept connections from the ESP32-S3. Incoming audio packets are buffered and reconstructed into contiguous PCM streams, which are processed by the Whisper speech-to-text model to generate transcripts. These transcripts, along with a predefined prompt describing the robot's capabilities and expected JSON command format, are passed to the Llama 3 model hosted via Ollama. The model generates a structured response specifying actions such as "move_forward", "turn_left", or "stop", as well as associated parameters like duration or angle. The Python server then serializes this information into JSON and transmits it back to the robot.

In the envisioned block diagram, the system can be divided into the following logical blocks: the user interface (speech input and optional audio output), the embedded control layer (ESP32-S3 handling I2S, WebSocket client, motor control, and ultrasonic safety), the local AI processing layer (Whisper and Llama 3 running on the workstation), and the wireless communication channel connecting the two. Data flow is predominantly from microphone to AI server and back to the robot, while control flow includes local safety overrides in the embedded layer. This modular architecture facilitates future expansion, such as integrating additional sensors or migrating certain AI functions onto dedicated neural processing units.

VI. RESEARCH METHODOLOGY

The research methodology adopted for this project follows a structured engineering approach involving problem identification, requirement analysis, system design, hardware development, software implementation, and iterative testing. The process began with identifying the fundamental problem of cloud dependency, privacy concerns, and latency in existing voice-controlled robotic systems. Initial experiments with cloud-based APIs confirmed that network-induced delays frequently exceeded 3 s, and that service availability could not be guaranteed in environments with unstable internet connectivity. These observations reinforced the need for a locally hosted AI solution combined with a real-time embedded controller.

During requirement analysis, both functional and non-functional specifications were defined. Functionally, the robot needed to capture user speech, transmit it to a local AI server, interpret the resulting commands, and execute navigation actions such as moving forward, reversing, and turning by specific angles. Non-functional requirements

DOI: 10.17148/IJARCCE.2026.15380

included end-to-end latency L<1000 msL<1000ms for a responsive interaction experience, high recognition accuracy at typical indoor distances up to 2 m, robust operation in the presence of motor noise, and strict confinement of all data exchanges to the local network. Safety requirements mandated reliable obstacle detection and immediate termination of motion when obstacles were detected within a minimum distance.

System design focused on decomposing the overall functionality into hardware and software subsystems with well-defined interfaces. The selection of the ESP32-S3 was based on its wireless connectivity, I2S support, and adequate processing resources. The INMP441 microphone was chosen for its digital I2S output, which directly aligns with the requirement for noise-immune audio capture. The use of an L298N motor driver and 4WD chassis provided adequate torque and mobility for indoor testing. On the software side, Whisper and Llama 3 were selected as the core AI components due to their open-source availability and compatibility with local deployment via Python and Ollama.

The algorithmic design on the ESP32-S3 employed a state-machine approach to manage robot behavior. The high-level states included Idle, Listening, Processing, Executing, and Emergency_Stop. In the Idle state, the robot awaited a trigger to start recording audio, which could be initiated either by a button press or a voice keyword in future extensions. In the Listening state, the Audio Capture Module activated the I2S interface and filled DMA buffers with PCM samples. Once a predefined duration or silence threshold was reached, the system transitioned to the Processing state, where the audio data was sent to the AI server via the WebSocket connection. In parallel, the ultrasonic sensor continued to monitor for obstacles. Upon receiving a JSON command from the server, the robot entered the Executing state, where the Execution Module translated the instructions into motor control signals and timed operations. If at any point the ultrasonic sensor detected a distance below the threshold, an interrupt forced a transition to the Emergency_Stop state, halting all motor activity.

The hardware development phase involved assembling and wiring the ESP32-S3 board with the INMP441 microphone, ultrasonic sensor, L298N driver, motors, and power supply. Signal integrity considerations guided the placement of decoupling capacitors and separation between motor power lines and digital signal traces. The I2S lines were kept relatively short and were routed away from high-current motor traces to reduce potential coupling. The ultrasonic sensor was mounted at a height and orientation suitable for detecting common indoor obstacles such as walls, furniture, and human legs.

Software implementation on the ESP32-S3 used the Arduino IDE and associated libraries. The I2S configuration included parameters for sample rate, bit depth, and buffer sizes. The WebSocket client was implemented using a lightweight library optimized for embedded systems. JSON parsing was handled by a compact parser capable of extracting key-value pairs with minimal memory overhead. On the server side, Python scripts orchestrated WebSocket communication, audio buffering, Whisper inference, and Llama 3 querying. Careful attention was paid to stream handling to avoid buffer overflows or underflows that could disrupt the real-time audio pipeline.

Testing procedures comprised both unit tests of individual modules and integrated system tests. For the audio subsystem, test signals were generated and recorded with and without motor activity to evaluate the impact of mechanical and electrical noise on capture quality. For the communication subsystem, artificial JSON commands were sent from the server to the robot to verify that the Execution Module responded correctly and that motor actions matched the requested directions and timings. End-to-end tests involved issuing spoken commands such as "move forward for two seconds" or "turn right ninety degrees" and measuring the time from the end of speech to the onset of motion, as well as verifying the correctness of the robot’s trajectory. The safety subsystem was tested by placing obstacles at varying distances and confirming that the robot consistently stopped before collision.

VII. SIMULATION AND RESULTS

Although the main focus of the project is on physical implementation, certain aspects of the system behavior were first modeled and evaluated in a simulated environment. Simulations included modeling the state machine controlling robot behavior, estimating end-to-end latency contributions from each subsystem, and analyzing the effect of ultrasonic sensor thresholds on stopping distance. Simple software simulations were performed using Python to emulate the timing of audio capture, network transmission, Whisper processing, Llama 3 inference, and WebSocket response. These simulations suggested that with a local gigabit Ethernet connection between the workstation and Wi-Fi router, and a typical Wi-Fi link to the robot, cumulative network latency would remain below 100 ms under normal conditions.

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 3, March 2026

DOI: 10.17148/IJARCCE.2026.15380

Experimental results were gathered using the physical prototype in a controlled indoor environment. Over 50 trial runs of distinct voice commands related to motion control, the robot correctly interpreted and executed 48 commands, achieving a command success rate of 96%. The misinterpretations primarily arose from overly ambiguous phrases or background conversations overlapping with the user's speech. The I2S-based audio pipeline demonstrated strong resilience to motor-induced noise: audio recordings taken while the motors were running showed minimal degradation compared to recordings taken with motors off, confirming that the digital interface effectively isolated the microphone from analog interference that typically plagues ADC-based designs.

Latency measurements were conducted by recording the audio of full interaction sessions and analyzing the time difference between the end of the spoken command and the onset of motor motion. Across multiple trials, the average end-to-end latency was approximately 850 ms850ms, with a standard deviation of around 120 ms120ms. This represents a significant improvement compared to cloud-based baselines reported in literature, which often exhibit latencies exceeding 3 s3s. The main contributors to latency in the proposed system were Whisper's speech-to-text processing time and Llama 3's response generation, with network transmission and JSON parsing contributing relatively minor delays. Optimization of model parameters and use of quantized variants could further reduce these times.

The performance of the ultrasonic safety subsystem was evaluated by commanding the robot to move forward toward a wall at a constant speed while gradually adjusting the obstacle threshold distance. When the threshold was set at approximately 20 cm20cm, the robot consistently stopped at a safe distance from the wall, accounting for momentum and motor response time. No collisions were observed during these tests, demonstrating the effectiveness of the interrupt-based override mechanism. Battery consumption measurements indicated that the ESP32-S3 and associated sensors drew approximately I≈200 mAI≈200mA during active streaming and movement, which is acceptable for typical battery capacities used in hobbyist and educational robots.

The overall system behavior exhibited stable operation during extended test sessions. The WebSocket connection remained reliable, and reconnection mechanisms were in place to handle temporary network disruptions. Importantly, even when the AI server was temporarily unavailable or busy, the robot's safety subsystem continued to function, and the system defaulted to a safe idle state rather than attempting uncontrolled movements. Qualitative user assessments reported that the interaction felt responsive and significantly more immediate than experiences with cloud-dependent robots, particularly when issuing successive commands in a conversational manner.

VIII. CONCLUSION

This work has presented the design, implementation, and evaluation of an AI-assisted personal robot that integrates a locally hosted LLM with an ESP32-S3 edge computing platform to achieve low-latency, privacy-preserving voice interaction and autonomous navigation. By partitioning responsibilities between the microcontroller and a local workstation, the system successfully leverages the strengths of both platforms: the ESP32-S3 provides deterministic real-time control, sensor interfacing, and safety-critical overrides, while the workstation hosts computationally intensive speech recognition and natural language reasoning using Whisper and Llama 3. The adoption of an I2S-based digital audio pipeline with an INMP441 microphone mitigates noise issues common to analog microphone setups on mobile robots, resulting in high-quality speech capture even during motor operation.

Compared to traditional cloud-based voice-controlled robots, the proposed architecture eliminates reliance on external servers, thereby enhancing user privacy and system autonomy. Experimental results demonstrate that the robot achieves an average end-to-end response latency of approximately 850 ms850ms, which is a substantial improvement over typical cloud-based latencies exceeding 3 s3s. Command recognition accuracy of 96% and reliable obstacle avoidance using an ultrasonic sensor and interrupt-driven safety logic indicate that the system is both responsive and robust in typical indoor environments. The use of WebSocket communication provides a flexible and efficient channel for bidirectional data exchange, enabling real-time interaction between the robot and the local AI server.

The system's modular design and reliance on widely available open-source tools and commodity hardware make it a practical template for further research and development in embodied AI. Future work may include integration of an onboard neural processing unit to offload portions of the AI workload directly onto the robot, enabling partial or full autonomy even without a workstation. Additional enhancements such as camera integration and Vision-Language Models would allow the robot to perceive and reason about visual information, expanding its capabilities beyond voice-only interaction. Moreover, refinements to the natural language prompting strategy and the design of domain-specific action schemas could enable more complex task planning and multi-step behaviors. Overall, the project

DOI: 10.17148/IJARCCE.2026.15380

demonstrates that local LLM integration with edge computing on microcontrollers is a viable and promising approach for next-generation personal robotic assistants.

REFERENCES

[1]. Sharma, "Voice Recognition in Robotics," IEEE Transactions on Robotics and Automation, vol. 39, no. 2, pp. 210–218, 2023.
[2]. R. Kumar, "Edge Computing with ESP32 for IoT Applications," Journal of Internet of Things Engineering, vol. 12, no. 1, pp. 45–54, 2024.
[3]. J. Park, H. Seo, and K. Kim, "Low-Power Keyword Spotting on Microcontrollers," IEEE Internet of Things Journal, vol. 8, no. 11, pp. 9120–9131, 2021.
[4]. Y. Li and X. Chen, "On-Device Speech Recognition with Compressed Recurrent Neural Networks," IEEE/ACM Transactions on Audio, Speech, and Language Processing, vol. 29, pp. 1750–1762, 2021.
[5]. T. Nguyen and R. Patel, "Ultrasonic Sensor-Based Obstacle Avoidance for Indoor Mobile Robots," International Journal of Robotics Research, vol. 40, no. 4, pp. 560–572, 2021.
[6]. V. Singh, M. Gupta, and P. Nair, "Wi-Fi Controlled Robot with Voice Interface Using Smartphone Speech Recognition," in Proc. IEEE International Conference on Advanced Robotics, 2022, pp. 134–139.
[7]. T. Brown et al., "Language Models are Few-Shot Learners," in Proc. Advances in Neural Information Processing Systems (NeurIPS), 2020, pp. 1877–1901.
[8]. H. Touvron et al., "LLaMA: Open and Efficient Foundation Language Models," arXiv:2302.13971, 2023.
[9]. V. Sanh et al., "DistilBERT, a Distilled Version of BERT: Smaller, Faster, Cheaper and Lighter," in Proc. 5th Workshop on Energy Efficient Machine Learning and Cognitive Computing, 2020.
[10]. S. Lee, D. Park, and J. Choi, "Local Transformer-Based Planning for Service Robots," IEEE Robotics and Automation Letters, vol. 7, no. 3, pp. 6400–6407, 2022.
[11]. L. Alvarez and S. Reddy, "Electromagnetic Interference Mitigation in Mobile Robots Using Digital Audio Interfaces," IEEE Transactions on Industrial Electronics, vol. 69, no. 9, pp. 9102–9111, 2022.
[12]. G. Hinton, O. Vinyals, and J. Dean, "Distilling the Knowledge in a Neural Network," in Proc. NIPS Deep Learning and Representation Learning Workshop, 2015.
[13]. J. Lin, S. Han, and W. Dally, "Deep Gradient Compression: Reducing the Communication Bandwidth for Distributed Training," in Proc. ICLR, 2018.
[14]. P. Wolf and A. Brandenburger, "WebSocket-Based Real-Time Control of Embedded Systems," IEEE Embedded Systems Letters, vol. 14, no. 1, pp. 33–37, 2022.
[15]. OpenAI, "Whisper: Robust Speech Recognition via Large-Scale Weak Supervision," OpenAI Technical Report, 2022.

