International Journal of Leading Research Publication (IJLRP)

E-ISSN: 2582-8010 ● Website: www.ijlrp.com ● Email: editor@ijlrp.com

ESP32-S3 AI Voice Assistant: A Cloud-Augmented Embedded System for Real-Time Voice Interaction

Neha Agrawal[1], Puja Gupta[2], Shaivi Barwe[3], Chandra Parakash Singar[4], Deepesh Agrawal[5]

[1,2,3,4,5]Department of Information Technology, Shri G.S. Institute of Technology & Science, Indore (M.P.), India

Abstract:
The rapid advancement of embedded systems and cloud-based artificial intelligence has opened new avenues for deploying intelligent, voice-activated interfaces on resource-constrained microcontrollers. This paper presents an ESP32-S3-based AI Voice Assistant that integrates real-time speech recognition, natural language understanding, and voice synthesis into a compact, standalone device costing under INR 1500. The system implements a five-stage pipeline: wake detection via button trigger, audio capture using an INMP441 I2S PDM microphone at 16 kHz/16-bit, cloud-based speech-to-text transcription via the Deepgram Nova-2 API, intelligent response generation using the Grok large language model (xAI), and high-quality text-to-speech synthesis delivered through a MAX98357A I2S Class-D amplifier. A deterministic six-state finite state machine governs pipeline transitions, ensuring predictable behaviour and graceful error recovery. PSRAM-aware dynamic memory allocation enables handling of audio buffers up to 512 KB. All twelve functional test cases passed, with end-to-end latency of 6-8 seconds on a 20 Mbps WiFi connection. The project demonstrates that ultra-low-cost embedded devices can serve as capable endpoints for cloud-augmented AI voice interaction, with potential applications in smart home control, educational technology, and accessible interfaces for differently-abled users.

Keywords: ESP32-S3, Voice Assistant, Speech-to-Text, Text-to-Speech, Large Language Model, Embedded AI, I2S Audio, State Machine, Deepgram, Grok LLM, Edge Computing, IoT

1. Introduction

Voice-activated assistants have become ubiquitous in consumer electronics, ranging from smartphones and smart speakers to automobiles and wearables. However, most commercially available solutions rely on proprietary hardware ecosystems, limiting their adaptability for custom embedded research or industrial applications. Furthermore, solutions capable of integrating state-of-the-art large language models (LLMs) with high-quality text-to-speech (TTS) synthesis typically demand Linux-class hardware such as Raspberry Pi, which entails higher cost, greater power consumption, and increased system complexity.

This paper addresses this gap by presenting a fully functional, open, and extensible AI voice assistant

implemented on the Seeed XIAO ESP32-S3, a sub-dollar dual-core microcontroller. The system integrates three cloud services — Deepgram (STT), Grok/xAI (LLM), and Deepgram (TTS) — with two I2S audio peripherals to achieve a responsive, natural-language voice interaction loop. The firmware is structured around a deterministic finite state machine (FSM) with six states, ensuring predictable execution, simplified debugging, and straightforward extensibility.

1.1 Problem Statement

Existing voice assistant solutions present several limitations for embedded and research applications:

- **Proprietary lock-in:** Commercial assistants (Alexa, Google Assistant) do not permit custom LLM integration or firmware modification.
- **High resource requirements:** Most open-source voice pipelines require Linux-class hardware, increasing cost and power consumption.
- **Fragmented toolchains:** Combining STT, LLM, and TTS on a microcontroller requires careful integration of multiple APIs, audio codecs, and network stacks.
- **Memory constraints:** Buffering raw audio and TTS output simultaneously on devices with limited RAM requires sophisticated memory management.
- **Latency:** Round-trip cloud API calls must be optimised to maintain a responsive user experience on hardware with limited processing capability.

1.2 Objectives

1. Design and implement a complete voice assistant pipeline on the ESP32-S3 microcontroller.
2. Integrate I2S microphone capture and I2S speaker playback using the Arduino framework.
3. Interface with the Deepgram Nova-2 API for accurate, low-latency speech-to-text transcription.
4. Send transcribed text to the Grok LLM and obtain concise, conversational responses.
5. Synthesise responses using Deepgram's Aura TTS model and stream audio to the MAX98357A amplifier.
6. Implement a robust FSM that handles errors and edge cases gracefully.
7. Optimise memory usage through PSRAM-aware dynamic buffer allocation.

1.3 Scope

The scope covers hardware wiring, firmware development, WiFi connectivity, cloud API integration, and audio pipeline optimisation for the Seeed XIAO ESP32-S3. The system is designed for single-user interactive sessions initiated by a push-button trigger. Wake-word detection, multi-user support, and offline inference are identified as future extensions beyond the current implementation.

2. Literature Review

2.1 Historical Context and Evolution of Voice Interfaces

Voice-based human-computer interaction has been an active research area since the 1950s. Early systems such as IBM Shoebox (1961) and Carnegie Mellon's Harpy (1976) demonstrated the feasibility of limited-vocabulary speech recognition. The advent of deep learning — particularly recurrent neural networks (RNNs), connectionist temporal classification (CTC) [7], and transformer architectures — has dramatically improved recognition accuracy and expanded vocabulary to conversational English and beyond.

2.2 Embedded Voice Processing

Several research works have explored voice processing directly on microcontrollers. TensorFlow Lite for Microcontrollers enables keyword spotting on Cortex-M devices with as few as 20 KB of RAM. Edge Impulse provides a platform for training and deploying audio classifiers on Arduino-compatible boards. However, these approaches are constrained to fixed-vocabulary recognition and cannot support open-domain conversation.

The ESP32 family, particularly the ESP32-S3 with its vector instruction extensions, has emerged as a popular platform for edge AI inference. Espressif's ESP-SR library demonstrates on-device wake-word detection and speech command recognition, but relies on pre-trained models that are difficult to customise for arbitrary domains.

2.3 Cloud-Based Speech APIs

Cloud speech-to-text services have matured significantly. Google Cloud Speech-to-Text, Amazon Transcribe, and Microsoft Azure Cognitive Services offer REST APIs with sub-second latency for short utterances. Deepgram differentiates itself through its Nova-2 model, which achieves lower word error rates on conversational speech and supports streaming transcription. Its HTTP POST endpoint is particularly well-suited for microcontroller integration, as it does not require a WebSocket implementation — a significant advantage given limited heap size on embedded platforms [3].

2.4 Large Language Models for Conversational AI

The release of GPT-3 (Brown et al., 2020) [6] marked a turning point in conversational AI, demonstrating that large transformer models trained on diverse text corpora can engage in open-domain dialogue with few-shot prompting. Subsequent models including GPT-4, LLaMA, and Grok have further improved reasoning and instruction-following capabilities. For embedded applications, the key challenge is managing response length: excessively long LLM outputs increase TTS processing time and may overflow audio buffers. This paper addresses the issue by constraining the LLM to 1-3 sentence responses via system prompting.

2.5 Neural Text-to-Speech Synthesis

Traditional TTS systems such as Festival and eSpeak can run on-device but produce robotic-sounding speech that impairs user experience. Modern neural TTS models — WaveNet [8], Tacotron 2, and VITS — produce highly natural speech but require significant compute and memory, precluding direct deployment on microcontrollers. Deepgram's Aura model offers a practical middle ground: neural-quality speech delivered via a simple REST API, with audio returned as raw PCM that can be streamed directly to an I2S amplifier without additional decoding overhead.

2.6 Research Gaps

- No published reference design combines Deepgram STT, Grok LLM, and Deepgram TTS on a single ESP32-S3.
- Existing embedded voice projects use either on-device inference (limited vocabulary) or Linux-class hardware.
- Audio buffer management for simultaneous capture and playback on PSRAM-equipped MCUs is under-documented.
- FSM design patterns for multi-stage cloud API pipelines on resource-constrained embedded systems lack standardised references.

3. System Analysis

3.1 Analysis of Existing Systems

Current consumer voice assistants (Amazon Echo, Google Nest) are closed systems that process audio on proprietary ASICs and route queries to vendor-controlled cloud backends. Academic and hobbyist alternatives typically fall into one of two categories: on-device keyword spotters with fixed command sets, or Linux-based systems (Raspberry Pi with Jasper/Mycroft) that are power-hungry and expensive. Key limitations include no ability to substitute the LLM backend, high idle power consumption from always-on wake-word detection, complex software stacks requiring OS maintenance, and poor audio quality from integrated DACs.

3.2 Proposed System Overview

The proposed system uses a Seeed XIAO ESP32-S3 as the central processing unit. Audio input is captured via an INMP441 I2S PDM microphone at 16 kHz/16-bit mono. The raw PCM is packaged into a WAV container and transmitted over HTTPS to the Deepgram Nova-2 STT endpoint. The returned transcript is forwarded to the Grok LLM, which generates a concise conversational reply. The reply text is then sent to Deepgram's Aura TTS endpoint, and the returned 24 kHz/16-bit PCM audio is streamed to a MAX98357A I2S Class-D amplifier. Key advantages include fully open firmware, hardware cost under INR 1500, modular FSM design, PSRAM-aware memory management, and superior I2S audio quality compared to PWM or sigma-delta alternatives.

3.3 Requirements Analysis

Functional Requirements:

- Capture 5 seconds of audio at 16 kHz/16-bit mono via I2S.
- Connect to a WiFi network and maintain connectivity throughout operation.
- Transmit WAV audio to the Deepgram STT API over HTTPS and receive a transcript.
- Forward the transcript to the Grok chat completion API and receive a text response.
- Send the text response to the Deepgram TTS API and receive 24 kHz PCM audio.
- Play the PCM audio via the MAX98357A I2S amplifier.
- Indicate system status via the built-in LED during recording and speaking states.
- Handle API errors gracefully and return to the IDLE state without hardware reset.

Non-Functional Requirements:

- **Performance:** End-to-end latency under 8 seconds on a stable WiFi connection.
- **Reliability:** FSM recovery from network timeouts without hardware reset.
- **Memory:** Correct operation with and without PSRAM, adapting buffer sizes accordingly.
- **Security:** API keys stored in a config.h file excluded from version control.
- **Maintainability:** Modular codebase with separate functions per pipeline stage.
- **Compatibility:** Firmware compilable with ESP-IDF-based Arduino core for ESP32.

4. System Design

4.1 Overall Architecture

The system follows a linear pipeline architecture governed by a six-state finite state machine controller. Three distinct layers are identified. The hardware layer comprises the ESP32-S3 SoC, the I2S microphone peripheral, and the I2S amplifier peripheral. The firmware layer implements WiFi stack management, I2S driver configuration, HTTP client operations, JSON serialisation/deserialisation (via ArduinoJson v6), and audio buffer management. The cloud layer provides STT, LLM, and TTS services

via HTTPS REST APIs, ensuring that the computationally intensive AI workloads are offloaded from the microcontroller.

4.2 Finite State Machine Design

The FSM defines six states that govern the complete interaction lifecycle. Table 1 summarises each state, its trigger condition, and its corresponding system behaviour.

State	Trigger	Behaviour
STATE_IDLE	System boot / error recovery	Waiting for button press; LED off; WiFi connected
STATE_RECORDING	Button press (debounced 50 ms)	I2S audio capture into record buffer; LED on
STATE_TRANSCRIBING	Buffer full (5 s)	WAV transmitted to Deepgram STT; awaiting transcript
STATE_THINKING	Transcript received	Transcript sent to Grok LLM; awaiting response
STATE_SPEAKING	LLM response received	TTS synthesis; PCM streamed to I2S amplifier
STATE_ERROR	HTTP error / empty response	Diagnostic output; transition to IDLE

Table 1: Finite State Machine State Definitions

4.3 Hardware Design

The hardware design centres on the Seeed XIAO ESP32-S3, chosen for its compact form factor, dual-core Xtensa LX7 processor operating at up to 240 MHz, 8 MB PSRAM, 8 MB Flash, and integrated WiFi/Bluetooth. Two I2S peripherals are utilised:

- **I2S_NUM_0 (Microphone):** Configured in master RX mode for the INMP441 PDM microphone. SCK on GPIO44, WS on GPIO9, SD on GPIO1. Eight DMA buffers of 512 bytes provide low-jitter continuous capture.
- **I2S_NUM_1 (Speaker):** Configured in master TX mode for the MAX98357A amplifier at 24 kHz. BCLK on GPIO7, LRC on GPIO4, DIN on GPIO2. tx_desc_auto_clear suppresses inter-buffer clicks.
- **Trigger Button:** Active-low momentary push button on GPIO21 with internal pull-up resistor. Debounced in firmware with a 50 ms delay to eliminate contact noise.

4.4 Audio Pipeline Data Flow

The complete audio pipeline proceeds as follows: (1) the user presses the button; (2) the firmware transitions to STATE_RECORDING and fills the 160 KB record buffer via I2S DMA; (3) a 44-byte RIFF/WAV header is prepended and the data is POSTed to the Deepgram STT endpoint over HTTPS; (4) the JSON transcript is extracted and forwarded in a chat completions request to the Grok API; (5) the LLM response is POSTed to the Deepgram TTS endpoint, with the returned raw 24 kHz PCM streamed in 512-byte chunks into the TTS buffer; (6) the TTS buffer is written to the MAX98357A via I2S in 4096-byte writes, followed by a 512-byte silence flush to drain the DMA pipeline.

4.5 Technology Stack

Component	Technology
Microcontroller	Seeed XIAO ESP32-S3 (dual-core LX7, 8 MB PSRAM)
Programming Framework	Arduino (ESP-IDF based, esp32 board package v2.0.x+)
Audio Input	INMP441 I2S PDM Omnidirectional Microphone
Audio Output	MAX98357A I2S Class-D Amplifier (3.2 W)
Speech-to-Text API	Deepgram Nova-2 (HTTPS REST POST)
Language Model API	Grok Beta — xAI Chat Completions Endpoint
Text-to-Speech API	Deepgram Aura Asteria (HTTPS REST, 24 kHz PCM)
HTTP Client	Arduino HTTPClient + WiFiClientSecure
JSON Processing	ArduinoJson v6
Audio Codec	WAV/PCM (16-bit, 16 kHz STT; raw PCM 24 kHz TTS)
Version Control	Git and GitHub
IDE	VS Code + Arduino CLI

Table 2: Technology Stack

5. Implementation

5.1 Firmware Overview

The firmware was developed using the Arduino framework targeting the ESP32-S3. The implementation comprises approximately 470 lines of C++ spread across a single main sketch file and a config.h configuration header. The firmware is structured into discrete modules: WiFi initialisation, I2S peripheral configuration, audio capture, STT API integration, LLM API integration, TTS API integration, memory management, and the main FSM loop.

5.2 I2S Microphone Implementation

The microphone I2S port (I2S_NUM_0) is configured in master RX mode with a sample rate of 16,000 Hz, 16-bit samples, mono channel, and standard I2S format. Eight DMA buffers of 512 bytes each provide continuous, low-jitter audio capture. The startRecording() function fills the 160 KB record buffer synchronously using i2s_read() calls of 1,024 bytes each, toggling the built-in LED to indicate active recording status.

5.3 I2S Speaker Implementation

The speaker I2S port (I2S_NUM_1) is configured in master TX mode at 24,000 Hz (matching Deepgram TTS output), 16-bit samples, mono channel, with tx_desc_auto_clear enabled to suppress audible clicks between DMA buffers. The playAudio() function writes the TTS buffer in 4,096-byte chunks via i2s_write(), followed by a 512-byte silence block to flush the DMA pipeline and prevent end-of-playback artefacts.

5.4 Deepgram STT Integration

The transcribeAudio() function constructs a valid 44-byte RIFF/WAV header in RAM, appends the raw

PCM samples, and HTTP POSTs the resulting WAV file to the Deepgram Nova-2 endpoint with smart_format=true and language=en-US parameters. The JSON response is parsed with ArduinoJson to extract the transcript string from results.channels[0].alternatives[0].transcript. HTTP timeout is configured at 15 seconds to accommodate variable network conditions.

5.5 Grok LLM Integration

The callGrok() function builds a chat completions request with a system prompt instructing the model to provide concise, plain-text answers of 1-3 sentences, and a user message containing the STT transcript. The request is POSTed to the xAI API with model=grok-beta and max_tokens=150, constraining response length to prevent TTS buffer overflow. The response content is extracted from choices[0].message.content with a 20-second HTTP timeout.

5.6 Deepgram TTS Integration

The synthesizeSpeech() function POSTs the LLM response text to Deepgram's TTS endpoint specifying model=aura-asteria-en, encoding=linear16, and sample_rate=24000. The response body is raw 16-bit PCM audio, which is streamed into the TTS buffer in 512-byte chunks via the HTTP stream pointer. PSRAM allocation (ps_malloc) provides up to 512 KB of buffer space for longer responses, while heap fallback reduces this to 128 KB on boards without PSRAM.

5.7 Memory Management Strategy

At firmware startup, psramFound() is invoked to detect PSRAM availability. If PSRAM is present, ps_malloc() allocates the record buffer (160 KB) and TTS buffer (512 KB) in external PSRAM, preserving internal SRAM for stack and system use. Without PSRAM, malloc() is used with reduced sizes (160 KB record, 128 KB TTS). Allocation failure at startup triggers an infinite diagnostic loop with serial output, preventing silent runtime failures. All buffers are reused across interactions, eliminating dynamic allocation overhead during the voice pipeline.

6. Testing and Results

6.1 Test Methodology

Testing was conducted on physical hardware using structured test cases covering each pipeline stage individually and the complete end-to-end pipeline. Tests were performed on a 20 Mbps WiFi network in a typical office environment with moderate ambient noise. A total of twelve functional test cases were designed and executed.

6.2 Test Case Results

#	Test Case	Input	Expected Output	Result
1	WiFi Connection	Credentials in config.h	IP assigned; WL_CONNECTED	**Pass**

2	Mic I2S Init	INMP441 on GPIO44/9/1	'Mic I2S initialized' on serial	**Pass**
3	Speaker I2S Init	MAX98357A on GPIO7/4/2	'Speaker I2S initialized' on serial	**Pass**
4	Button Debounce	Rapid tap (no full press)	No state transition to RECORDING	**Pass**
5	Audio Capture	Press button and speak	~160,000 bytes captured	**Pass**
6	WAV Header	Send audio to Deepgram	HTTP 200 from STT endpoint	**Pass**
7	Deepgram STT	'What is the capital of India?'	Transcript matches spoken input	**Pass**
8	Grok LLM	Forward STT transcript	'The capital of India is New Delhi.'	**Pass**
9	Deepgram TTS	Send LLM response to TTS	Non-zero PCM; correct audio plays	**Pass**
10	End-to-End	Press button; ask question	Spoken answer within 8 seconds	**Pass**
11	PSRAM Buffer	Long-answer question	No overflow; audio completes	**Pass**
12	Error Recovery	Disable WiFi mid-session	HTTP error; returns to IDLE state	**Pass**

Table 3: Test Case Results Summary

6.3 Performance Analysis

All twelve test cases passed successfully. The end-to-end pipeline consistently responded within 6-8 seconds on a 20 Mbps WiFi connection, meeting the 8-second latency requirement. Transcription accuracy was high for clear speech in a quiet environment, with occasional errors on heavily accented or noisy input — consistent with Deepgram Nova-2's documented performance characteristics. The error recovery mechanism functioned correctly in all tested failure scenarios, returning the device to IDLE state without requiring a hardware reset. No buffer overflows were observed during PSRAM-equipped operation, even for longer LLM responses approaching the 150-token limit.

7. Conclusion and Future Scope

This paper has presented a complete, cloud-augmented AI voice assistant implemented on the Seeed XIAO ESP32-S3 microcontroller at a total hardware cost under INR 1500. The system successfully integrates I2S audio capture, Deepgram Nova-2 speech-to-text, Grok LLM natural language reasoning, and Deepgram Aura text-to-speech into a coherent five-stage pipeline managed by a six-state deterministic FSM. The state-machine architecture proved to be a reliable design pattern for managing the asynchronous nature of cloud API calls within the single-threaded Arduino loop. PSRAM-aware buffer allocation enabled handling of longer TTS responses without heap fragmentation. All functional requirements were met and all twelve test cases passed, with end-to-end latency of 6-8 seconds.The proposed demonstrates how modern cloud AI APIs can be brought to the embedded domain, enabling sophisticated voice interaction on ultra-low-cost hardware. This has broad implications for accessible technology, IoT voice interfaces, smart home control, and educational embedded systems projects.

REFERENCES:

1. Popovici, A.I. and Anton, F.D., 2026. ESP32-Based Hardware Key for Software Application Protection. Applied Sciences, 16(9), p.4251.
2. Gupta, P. and Kulkarni, N., 2013. An introduction of soft computing approach over hard computing. International Journal of Latest Trends in Engineering and Technology (IJLTET), 3(1), pp.254-258
3. Gupta, P., Sharma, V. and Varma, S., 2021. People detection and counting using YOLOv3 and SSD models. Materials Today: Proceedings.
4. Gupta, P., Kulkarni, A. and Sarda, A., 2013. An embedded health care supervisory systems. International Journal of Latest Trends in Engineering and Technology (IJLTET), 3, pp.379-386
5. Gupta, P., Kumar, S., Singh, Y.B., Singh, P., Sharm, S.K. and Rathore, N.K., 2022. The impact of artificial intelligence on renewable energy systems. NeuroQuantology, 20(16), p.5012
6. Chaturvedi, A. and Gupta, P., 2021. The Cloud: Features, Challenges and Scope. International Journal of Progressive Research in Science and Engineering, 2(8), pp.466-471.
7. Kushwaha, U., Gupta, P., Airen, S. and Kuliha, M., 2022, December. Analysis of CNN Model with Traditional Approach and Cloud AI based Approach. In 2022 International Conference on Automation, Computing and Renewable Systems (ICACRS) (pp. 835-842). IEEE.
8. Gupta, P., Varma, S., Arya, N. and Singh, U., 2023. IoT-Based Smart Chair for Healthcare Supporting System. In Intelligent Sensor Node-Based Systems (pp. 39-52). Apple Academic Press
9. Gupta, P., Varma, S., Arya, N. and Bhagel, R., 2023. Intelligent Security System Based on the Internet of Things (IoT). In Intelligent Sensor Node-Based Systems (pp. 177-191). Apple Academic Press
10. Singh, U, Gupta, P., Shukla, M., Sharma, V., Varma, S. and Sharma, S.K., 2023. Acknowledgment of patient in sense behaviors using bidirectional ConvLSTM. Concurrency and Computation: Practice and Experience, 35(28), p.e7819.
11. Gupta, P., Shukla, M., Arya, N. and Singh, U., 2023. 5 Internet Smart and of Intelligent Things in Healthcare Systems. IoT in Healthcare Systems: Applications, Benefits, Challenges, and Case Studies, p.77
12. Gupta, P. and Singh, U., 2022. Traffic Management for Smart City Using Deep Learning. Autonomous Vehicles Volume 1: Using Machine Intelligence, pp.149-159
13. Rathore, P., Gupta, P., Jain, S. and Shrivastava, Y., 2022. A Study of the Automated Vehicle Number Plate Recognition System. i-manager's Journal on Pattern Recognition, 9(2), p.30.
14. Rajput, A., Gupta, P., Ghodeshwar, P., Varma, S., Sharma, K.K. and Singh, U., 2023, June. Study of Cloud Providers (Azure, Amazon, and Oracle) According To Service Availability and Price. In 2023 3rd International Conference on Pervasive Computing and Social Networking (ICPCSN) (pp. 1177-1188). IEEE.
15. Rathore, P., Gupta, P., Jain, S. and Shrivastava, Y., 2022. A Study of the Automated Vehicle Number Plate Recognition System. i-manager's Journal on Pattern Recognition, 9(2), p.30.
16. Gupta, P, N. Rathore Comparison of cloud and edge computing based face recognition for security enhancement 2025 Journal of Discrete Mathematical Sciences and Cryptography