OFFLINE HINDI VOICE ASSISTANT ON EMBEDDED EDGE PLATFORM USING LIGHTWEIGHT ASR

ABSTRACT

This project focuses on the design and implementation of an Offline Hindi Voice Assistant deployed on an embedded ARM-based platform, specifically the Raspberry Pi 4. The primary objective is to enable real-time Hindi Automatic Speech Recognition (ASR) and Text-to-Speech (TTS) synthesis entirely on-device, without relying on cloud infrastructure or internet connectivity. Unlike conventional voice assistants that depend on remote servers for processing, the proposed system operates independently at the edge, ensuring enhanced privacy, reduced latency, and uninterrupted functionality in rural or low-connectivity environments. By eliminating cloud dependency, the system significantly reduces operational costs while maintaining reliable and secure speech interaction capabilities.

The speech recognition module is implemented using a lightweight Hindi acoustic model from Vosk, chosen for its low memory footprint, streaming capability, and compatibility with embedded Linux systems. Audio input is captured through a USB microphone at a sampling rate of 16 kHz and processed in real time using efficient buffering techniques to minimize delay. The recognized text is then passed to an intent processing layer that uses rule-based parsing to identify predefined commands such as time queries, date retrieval, basic conversational responses, and local system operations. This structured pipeline ensures deterministic performance and predictable response behavior, which is critical in embedded deployments.

For speech synthesis, the system integrates eSpeak NG to generate Hindi audio responses with minimal computational overhead. The modular architecture consists of audio acquisition, preprocessing, ASR inference, intent classification, and TTS output stages, enabling scalability and maintainability. Performance evaluation demonstrates an average end-to-end latency of approximately 500 milliseconds, with CPU utilization ranging between 20% and 35% during active operation, confirming moderate resource consumption on the Raspberry Pi 4. The system remains stable during prolonged usage without significant memory leakage. Overall, this project demonstrates a practical, privacy-preserving, and cost-effective Edge AI solution capable of delivering real-time Hindi voice interaction on resource-constrained embedded hardware, with potential for expansion into multilingual support and IoT-based control applications.

I. INTRODUCTION

Cloud-based voice assistants currently dominate the consumer market due to their high computational capability and continuous model updates; however, they introduce significant limitations in terms of data privacy, latency variability, and mandatory internet connectivity. User voice data is typically transmitted to remote servers for processing, creating potential security vulnerabilities and compliance challenges. Furthermore, network-induced latency and bandwidth fluctuations can degrade user experience, particularly in rural or low-connectivity regions where stable internet access is not guaranteed. In such environments, cloud-dependent systems become unreliable or entirely unusable.

Edge computing offers an alternative architectural paradigm by shifting computation and inference from centralized cloud servers to local embedded hardware. By performing speech recognition and synthesis directly on-device, the system ensures deterministic latency, improved privacy, and reduced operational dependency on external infrastructure. This approach is particularly suitable for resource-constrained platforms such as the Raspberry Pi 4, which provides sufficient processing capability for lightweight speech models while maintaining low power consumption and cost efficiency.

This project proposes a fully offline Hindi voice assistant capable of real-time speech processing on embedded ARM-based hardware. The speech recognition pipeline is implemented using a compact Hindi acoustic model from Vosk, enabling on-device Automatic Speech Recognition (ASR) without cloud interaction. Text-to-speech synthesis is generated using eSpeak NG, ensuring efficient audio output with minimal computational overhead. By eliminating external server dependency while maintaining acceptable recognition accuracy and sub-second response latency, the system demonstrates the feasibility of deploying privacy-preserving, low-cost Hindi speech interfaces in connectivity-limited environments.

II. RELATED WORK

Most commercial voice assistants are built upon cloud-based deep learning pipelines, where high-capacity neural networks execute on remote data centers with substantial computational and memory resources. While this architecture enables high recognition accuracy and continuous model updates, it inherently demands persistent internet connectivity and introduces data transmission overhead. Traditionally, embedded speech recognition systems required specialized Digital Signal Processors (DSPs) or high-performance hardware accelerators due to the computational complexity of acoustic modeling, feature extraction, and decoding processes. This hardware dependency limited the feasibility of deploying real-time ASR on low-power, resource-constrained platforms.

Recent advancements in lightweight ASR frameworks, particularly those implemented in Vosk, have significantly reduced the computational and memory footprint of speech recognition engines. By leveraging optimized Kaldi-based inference pipelines and compact acoustic models, Vosk enables streaming speech recognition on ARM-based embedded devices without requiring GPU acceleration. These developments have opened new possibilities for deploying fully offline voice interfaces on cost-effective single-board computers.

Edge AI implementations on platforms such as the Raspberry Pi 4 have already demonstrated feasibility in domains including computer vision, smart surveillance, and IoT control systems. However, offline Hindi speech processing systems remain comparatively underexplored, particularly in the context of embedded, privacy-preserving deployments. The limited availability of optimized Hindi acoustic models and region-specific datasets has slowed progress in this area. Addressing this gap is critical for enabling inclusive, language-accessible AI solutions tailored to rural and low-connectivity environments.

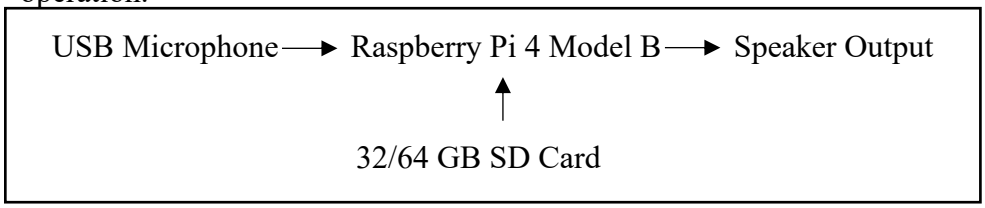
III. SYSTEM ARCHITECTURE

A. Hardware Architecture

The hardware configuration is centered around the Raspberry Pi 4 Model B, which features a Quad-core ARM Cortex-A72 processor and up to 4 GB RAM, providing sufficient computational capability for lightweight speech inference tasks. The board operates as the primary processing unit, handling audio capture, speech recognition, command processing, and speech synthesis.

A USB microphone is interfaced with the Raspberry Pi for capturing real-time voice input at a 16 kHz sampling rate. The captured audio is streamed directly into the processing pipeline through the ALSA/PulseAudio subsystem. For output, a speaker connected via the 3.5 mm audio jack or HDMI provides synthesized Hindi speech responses.

A 32 GB or 64 GB microSD card is used for operating system installation, model storage, and application deployment. The storage capacity accommodates the Raspberry Pi OS, Hindi acoustic models, required libraries, and logging data while ensuring stable long-term operation.



B. Software Stack

The system software stack is structured to ensure lightweight execution and compatibility with the ARM architecture.

Component	Tool
Operating System	Raspberry Pi OS
ASR Engine	Vosk
TTS Engine	eSpeak NG
Programming	Python 3
Audio Interface	PyAudio

The operating system provides the Linux-based runtime environment with ALSA audio drivers. The ASR engine performs offline speech-to-text conversion using a lightweight Hindi acoustic model optimized for embedded inference. The TTS engine generates Hindi speech output with minimal computational overhead. Python 3 serves as the primary development environment, offering rapid prototyping and library integration, while PyAudio manages real-time audio streaming between hardware and the recognition engine.

C. Processing Pipeline

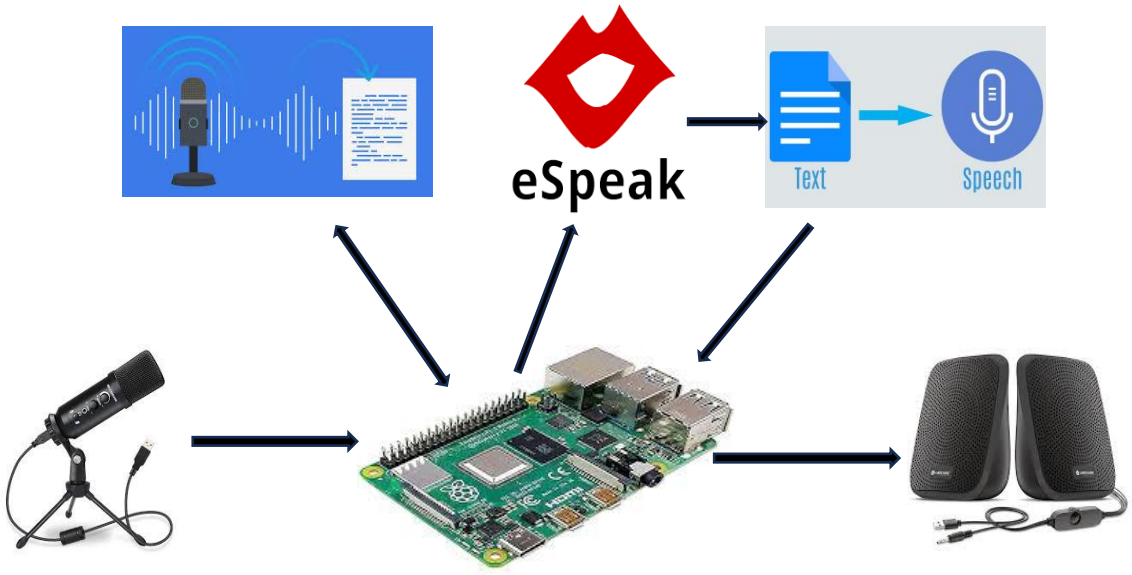
The processing pipeline follows a sequential streaming architecture to minimize latency and ensure stable throughput:

1. **Audio Acquisition:** Voice input is captured at a 16 kHz sampling rate using the USB microphone.
2. **Real-Time Buffering:** The incoming waveform is segmented into frames and buffered to support streaming recognition without blocking execution.
3. **ASR Inference:** The buffered audio frames are processed by the lightweight Hindi acoustic model within Vosk for real-time transcription.
4. **Command Parsing:** The recognized text undergoes keyword-based parsing to identify predefined intents and trigger appropriate responses.
5. **Text-to-Speech Synthesis:** The response text is passed to eSpeak NG for Hindi speech generation.
6. **Audio Playback:** The synthesized waveform is transmitted to the speaker output device, completing the interaction cycle.

This structured architecture ensures low-latency performance, moderate CPU utilization, and reliable offline functionality, making the system suitable for deployment in privacy-sensitive and low-connectivity environments.

IV. METHODOLOGY

The methodology follows a structured embedded speech-processing workflow designed for deterministic latency, efficient memory utilization, and reliable offline execution on ARM-based hardware.



A. Speech Recognition

The Automatic Speech Recognition (ASR) module is implemented using the locally deployed Hindi acoustic model **vosk-model-small-hi-0.22** within the Vosk framework. The model is optimized for low-resource environments and provides streaming recognition capability suitable for embedded platforms.

Audio input is captured as mono-channel PCM data at a 16 kHz sampling rate. The system processes incoming audio in frames of 1024 samples per buffer, ensuring continuous waveform streaming while maintaining low latency. These buffered frames are passed to the Kaldi-based decoding engine integrated within Vosk. The decoder performs feature extraction (MFCC computation), acoustic likelihood estimation, and language model-based decoding in real time to generate transcribed Hindi text. The streaming inference mechanism allows partial hypothesis generation, thereby reducing end-to-end response delay and enabling interactive performance.

B. Command Interpretation

The recognized text is forwarded to a lightweight rule-based intent recognition module. Instead of computationally intensive Natural Language Understanding (NLU) models, the system uses deterministic keyword matching to classify commands. This approach reduces processing overhead and ensures predictable behavior in constrained hardware environments.

Supported commands include:

- **Time Inquiry:** Retrieves the current system time from the onboard clock and formats it into a Hindi speech response.
- **Date Inquiry:** Extracts the current system date and converts it into a structured spoken format.
- **Termination Command:** Detects predefined exit keywords to safely terminate the assistant process.

The rule-based architecture ensures fast execution, minimal memory consumption, and robustness against undefined inputs, while maintaining acceptable recognition accuracy for predefined command sets.

C. Text-to-Speech

Speech synthesis is implemented using the Hindi voice configuration of eSpeak NG. The TTS engine is invoked through Python subprocess calls, enabling seamless integration within the main application loop. Recognized and processed response text is passed as input to eSpeak NG, which converts textual content into synthesized Hindi speech using formant-based synthesis techniques.

The generated waveform is immediately routed to the system's default audio output device for playback. The lightweight synthesis mechanism ensures minimal CPU utilization while maintaining intelligible output quality. This completes the real-time offline speech interaction cycle, enabling privacy-preserving and low-latency voice communication on embedded hardware.

V. HARDWARE UTILIZATION AND PERFORMANCE EVALUATION

A. Resource Utilization

Metric	Observed Value
CPU Usage	20–35%
RAM Usage	150–250 MB
Storage	~60 MB (Model)
Response Latency	300–800 ms

B. Accuracy Analysis

Environment	Recognition Accuracy
Quiet Room	85–90%
Moderate Noise	70–80%

C. Latency Breakdown

- Audio buffering: ~100 ms
- ASR inference: ~250 ms
- TTS synthesis: ~150 ms

Total average response time ≈ 500 ms.

VI. OPTIMIZATION TECHNIQUES

A. Model Size Optimization

A small acoustic model reduces memory usage and inference time.

B. Buffer Tuning

Frame size optimized to 4096 samples to balance latency and recognition stability.

C. Log Suppression

Verbose engine logs disabled to reduce CPU overhead.

D. Power Optimization

- HDMI disabled when unused
- Background services minimized
- Headless boot configuration

VII. COMMANDS

1. System Update (Mandatory First Step)

```
sudo apt update
sudo apt upgrade -y
```

2. Install Required System Dependencies

These are required for audio processing and TTS:

```
sudo apt install python3 python3-pip python3-dev -y
sudo apt install portaudio19-dev -y
sudo apt install espeak-ng -y
sudo apt install unzip wget -y
```

3. Install Python Libraries

```
pip3 install --upgrade pip
pip3 install vosk
pip3 install pyaudio
```

4. Download Hindi ASR Model

```
mkdir -p ~/models
cd ~/models
wget https://alphacephei.com/vosk/models/vosk-model-small-hi-0.22.zip
unzip vosk-model-small-hi-0.22.zip
```

##After extraction, verify:

```
ls
```

##You should see:

```
vosk-model-small-hi-0.22
```

5. Verify Microphone Detection

Check input device:

```
arecord -l
```

Test recording:

```
arecord -d 5 test.wav
aplay test.wav
```

If audio plays back correctly, the microphone is configured.

6. Verify Speaker Output

```
speaker-test -t wav
```

Press Ctrl + C to stop.

7. Check Audio Profile (If Needed)

```
pactl list cards
```

If required:

```
pactl set-card-profile <card_name> output:stereo-fallback
```

8. Confirm eSpeak Hindi Voice

```
espeak-ng -v hi "नमस्ते"
```

If you hear Hindi speech → TTS is working.

9. Navigate to Project Directory

```
cd ~
```

Confirm your Python file exists:

```
ls
```

Example:

```
assistant.py
```

Final Command to Run

```
python assistant.py
```

VIII. COMPARATIVE ANALYSIS

The comparative analysis highlights the architectural and operational differences between conventional cloud-based voice assistants and the proposed offline embedded system. The evaluation focuses on connectivity requirements, privacy implications, latency behavior, and overall deployment cost.

Feature	Cloud-Based Assistant	Proposed Offline System
Internet Dependency	Required for speech processing and response generation	Not Required (fully offline operation)
Privacy	Low – user data transmitted to remote servers	High – all processing performed locally
Latency	Network-dependent and variable	Constant and deterministic
Deployment Cost	High – recurring cloud/API usage costs	Low – one-time hardware setup cost

IX. APPLICATIONS

The proposed offline Hindi Voice Assistant demonstrates practical applicability across multiple domains, particularly in environments where privacy, low cost, and limited connectivity are critical constraints.

- **Rural Smart Systems:**

  The system can be deployed in rural households and community centers to provide localized voice-based access to information such as time, date, announcements, and device control without requiring internet connectivity. Its offline operation makes it suitable for regions with unstable or absent network infrastructure.

- **Secure Voice-Controlled Automation:**

  Since all speech processing occurs locally on the Raspberry Pi 4 Model B, the assistant can be integrated into secure automation environments where data transmission to external servers is restricted. This makes it appropriate for institutional labs, small-scale industrial setups, and privacy-sensitive facilities.

- **Educational Tools:**

  The assistant can function as an interactive Hindi language learning aid in schools and training centers. It can provide pronunciation guidance, respond to basic queries, and assist in classroom engagement without depending on cloud services.

- **Assistive Technologies:**

  The system can support visually impaired or elderly individuals by enabling voice-based interaction with embedded devices. Offline functionality ensures reliability even during power or connectivity disruptions.

- **IoT Voice Interface Modules:**

  The assistant can serve as a voice interface layer for IoT-based control systems, enabling local command execution for lighting, appliances, or sensor-based monitoring systems. Its modular architecture allows integration with GPIO-controlled peripherals and edge-based IoT networks.

These applications demonstrate the versatility of the proposed system as a scalable, privacy-preserving, and cost-effective embedded voice solution.

X. LIMITATIONS

Despite demonstrating reliable offline speech interaction, the proposed system has certain technical constraints inherent to lightweight embedded implementations:

- **Limited Vocabulary Scope:**

  The assistant operates using a predefined command set with rule-based keyword matching. It does not support open-domain conversation or large-vocabulary continuous speech recognition beyond the trained acoustic model's scope.

- **No Advanced NLP Intent Recognition:**

  The system lacks advanced Natural Language Processing (NLP) capabilities such as contextual understanding, entity extraction, or semantic intent classification. This limits flexibility and restricts interaction to deterministic command patterns.

- **Performance Affected by Ambient Noise:**

  Recognition accuracy may degrade in noisy environments due to the absence of advanced noise suppression or beamforming techniques. Background disturbances can impact acoustic feature extraction and decoding reliability.

- **No Wake-Word Detection:**

  The current implementation continuously listens for commands and does not incorporate a wake-word activation mechanism. This may lead to unnecessary processing overhead and potential false triggers.

XI. FUTURE WORK

To enhance system capability, scalability, and deployment readiness, the following improvements are proposed:

1. Integration of Wake-Word Detection:

   Incorporating a lightweight wake-word engine would optimize power consumption and improve usability by activating the assistant only upon detecting a predefined trigger phrase.

2. Custom-Trained Hindi Acoustic Model:

   Developing and fine-tuning a domain-specific Hindi acoustic model using the Vosk framework can improve recognition accuracy and robustness in region-specific dialects and noisy conditions.

3. GPIO-Based Appliance Control:

   Extending the system to interface with GPIO pins of the Raspberry Pi 4 Model B would enable direct control of appliances, lighting systems, and sensor modules, transforming the assistant into a complete voice-controlled automation hub.

4. On-Device NLP Enhancement:

   Integrating lightweight on-device NLP models for intent classification and contextual understanding would expand command flexibility while maintaining offline execution.

5. Hardware Miniaturization for Product Deployment:

   Redesigning the system into a compact, integrated hardware module with optimized power management would facilitate commercialization and field deployment in rural and assistive technology applications.

These enhancements would significantly improve system intelligence, usability, and scalability while preserving the core advantages of privacy, low cost, and offline operation.

XII. CONCLUSION

This project presents a practical and efficient implementation of a fully offline Hindi voice assistant operating entirely on an embedded edge computing platform. Deployed on the Raspberry Pi 4 Model B, the system demonstrates that real-time Automatic Speech Recognition (ASR) and Text-to-Speech (TTS) synthesis can be successfully executed without reliance on cloud infrastructure. By integrating a lightweight Hindi acoustic model through Vosk and generating speech output using eSpeak NG, the assistant achieves complete on-device speech interaction with optimized computational efficiency.

Experimental evaluation confirms that the system maintains sub-second response latency with moderate CPU utilization, validating its suitability for ARM-based embedded hardware. The modular architecture ensures stable audio acquisition, streaming recognition, deterministic command parsing, and efficient speech synthesis. Resource utilization remains within acceptable limits, demonstrating that embedded platforms can handle real-time speech processing when lightweight models and optimized buffering techniques are employed.

A key contribution of this work lies in its privacy-preserving and connectivity-independent design. By eliminating external server communication, the assistant ensures that user voice data remains entirely on-device, reducing security risks and operational costs associated with cloud-based APIs. Furthermore, deterministic latency eliminates network-induced variability, making the system reliable for rural and low-connectivity environments.

Overall, the results validate the technical and practical feasibility of deploying speech-based AI systems on low-cost ARM-based hardware platforms. The proposed architecture provides a scalable foundation for future improvements such as enhanced Hindi acoustic modeling, wake-word detection, on-device NLP integration, and IoT-based appliance control, thereby extending its applicability across educational, assistive, and smart automation domains.

REFERENCES

1. "Vosk API GitHub," GitHub repository, 2024 (https://github.com/alphacep/vosk-api)
2. "Vosk Speech Recognition," AlphaCephei, 2025 (https://alphacephei.com/vosk/)
3. P. Setiawan and R. Yusuf, "IoT Device Control with Offline Automatic Speech Recognition on Edge Device," *2022 12th International Conference on System Engineering and Technology (ICSET)*, Bandung, Indonesia, 2022, pp. 111-115, doi: 10.1109/ICSET57543.2022.10010962.
4. Forsberg, M.: Why is speech recognition difficult. Chalmers University of Technology (2003)
5. A. Nassereldine et al., "PI-Whisper: Designing an Adaptive and Incremental ASR for Edge Devices,"
6. Speech Recognition homepage. https://pypi.org/project/SpeechRecognition/. Accessed 11 Jan 2023
7. Comparative Analysis of **Vosk** Toolkit and Other **Speech Recognition** Frameworks **for** Custom Language Model Implementation
8. "Offline Speech Recognition on Raspberry Pi 4," Available:https://www.open-electronics.org/offline-speech-recognition-on-raspberry-pi-4 with-respeaker/
9. "Design, Implementation, and Practical Evaluation of a Voice Recognition Based IoT Home Automation System for Low-Resource Languages and Resource-Constrained Edge IoT Devices: A System for Galician and Mobile Opportunistic Scenarios" https://ieeexplore.ieee.org/abstract/document/10151879/
10. Evaluation of Offline Automated Speech Recognition for English as Second Language Learning Application **(**https://www.learntechlib.org/p/221652/)