OFFLINE PRIVACY PRESERVING HINDI VOICE ASSISTANT ON RASPBERRY PI

Bharat AI-SoC Student Challenge 2026

Project Submission

Student Information

Name: R. Ashwin
College: Chennai Institute of Technology
Mentor: Pradhipa A
Date: February 20, 2026

Executive Summary

This project presents an offline, privacy-preserving voice assistant designed to process Hindi voice commands entirely on a Raspberry Pi 4, without requiring cloud connectivity. The system achieves sub-2-second response times while maintaining complete user privacy by performing all speech recognition, intent parsing, and speech synthesis locally on the device.

Key Achievements:

• Fully offline operation with no cloud dependency

• Hindi language support using Devanagari script

• Sub-2-second response time for voice commands

• 10-15 functional Hindi commands

• Privacy-preserving architecture

1. Introduction

1.1 Problem Statement

Traditional voice assistants (Google Assistant, Alexa, Siri) require constant internet connectivity and transmit user voice data to cloud servers, raising significant privacy concerns. Additionally, support for regional Indian languages, particularly Hindi, is limited or requires cloud connectivity.

1.2 Objectives

1. Develop a fully offline voice assistant for Raspberry Pi 4
2. Implement Hindi language support for voice input and output
3. Achieve sub-2-second response time for voice commands
4. Support 10-15 common Hindi voice commands
5. Ensure complete privacy by eliminating cloud dependency

1.3 Scope

The project focuses on:

• Offline Automatic Speech Recognition (ASR) for Hindi

• Local intent recognition and command parsing

• Offline Text-to-Speech (TTS) synthesis in Hindi

• Real-time audio processing on Raspberry Pi 4

2. Methodology

2.1 System Architecture

The voice assistant follows a three-stage pipeline architecture:

```
■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■
■ ■
■ MICROPHONE → ASR → INTENT → TTS → SPEAKER ■
■ ■
■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■

Stage 1: Speech-to-Text (ASR)
↓ Input: Audio stream from microphone
↓ Processing: Vosk Hindi model
↓ Output: Hindi text (Devanagari)
```

```
Stage 2: Intent Recognition
↓ Input: Hindi text
↓ Processing: Pattern matching algorithm
↓ Output: Identified intent + response text

Stage 3: Text-to-Speech (TTS)
↓ Input: Hindi response text
↓ Processing: eSpeak-NG Hindi voice
↓ Output: Synthesized audio
```

2.2 Technology Stack

Hardware:

• Raspberry Pi 4 (4GB RAM)

• USB Microphone

• 3.5mm jack Speaker

• Raspberry Pi Cooler

• 32GB Micro SD Card

Software Stack:

• Operating System: Raspberry Pi OS (64-bit)

• Programming Language: Python 3.9+

• ASR Engine: Vosk (vosk-model-small-hi-0.22)

• TTS Engine: eSpeak-NG

• Audio I/O: PyAudio

Key Libraries:

• vosk==0.3.45 (Speech recognition)

• pyaudio==0.2.13 (Audio processing)

• espeak-ng (Text-to-speech synthesis)

2.3 Component Details

2.3.1 Automatic Speech Recognition (ASR)

Implementation:

• Model: Vosk offline speech recognition

• Language Model: vosk-model-small-hi-0.22 (Hindi)

• Sampling Rate: 16 kHz

• Buffer Size: 8192 samples

Process Flow:

1. Capture audio from USB microphone via PyAudio
2. Stream audio data to Vosk recognizer in chunks
3. Detect voice activity and speech boundaries
4. Convert speech to Hindi text (Devanagari script)
5. Return recognized text for intent processing

Optimization Techniques:

• Used lightweight "small" model for faster inference

• Implemented voice activity detection to reduce processing

• Optimized buffer size for balance between latency and accuracy

2.3.2 Intent Recognition & Command Parsing

Architecture:

Pattern-matching based intent recognition system

Supported Commands:

Category	Hindi Command	English Translation	Action
Greeting	■■■■■■	Hello	Greeting response
Time	■■■ ■■■■	Tell time	Display current time
Date	■■■■■ ■■■■	Tell date	Display current date
Weather	■■■■ ■■■■ ■■	How's weather	Weather info (simulated offline)
Thanks	■■■■■■■	Thank you	Acknowledgment
Joke	■■■■■ ■■■■■	Tell joke	Random Hindi joke
Help	■■■ ■■■	Help	List available commands
Identity	■■■ ■■■ ■■	Who are you	Assistant introduction
Volume Up	■■■■■■■ ■■■■■	Increase volume	Increase system volume
Volume Down	■■■■■■■ ■■■■	Decrease volume	Decrease system volume
Battery	■■■■■ ■■■■■ ■■	Battery status	Check battery level
Exit	■■■ ■■■	Stop	Exit application

Algorithm:

1. Receive Hindi text from ASR module

2. Convert text to lowercase for matching

3. Iterate through command pattern dictionary

4. Match keywords in input text

5. Execute associated action function

6. Generate Hindi response text

7. Return response to TTS module

Pattern Matching Example:

```
if '■■■' in text or '■■■■' in text:
intent = 'time'
response = get_current_time_in_hindi()
```

2.3.3 Text-to-Speech (TTS)

Implementation:

• Engine: eSpeak-NG

• Voice: Hindi (hi)

• Speech Rate: 150 words per minute

• Pitch: 50 (mid-range)

Process:

1. Receive Hindi response text from intent handler

2. Pass text to eSpeak-NG command-line interface

3. Generate synthesized speech audio

4. Output audio directly to speaker via ALSA

Command Format:

```
espeak-ng -v hi -s 150 -p 50 "■■■■■■"
```

3. Implementation

3.1 System Setup

Step 1: Raspberry Pi OS Installation

• Downloaded Raspberry Pi OS (64-bit)

• Configured SSH and WiFi in Raspberry Pi Imager

• Flashed 32GB SD card

• First boot time: ~3 minutes

Step 2: Software Dependencies Installation

```
# System packages
sudo apt update
sudo apt install -y python3-pip python3-pyaudio portaudio19-dev
sudo apt install -y espeak-ng alsa-utils

# Python packages
pip3 install vosk pyaudio --break-system-packages
```

Step 3: Hindi Model Download

```
wget https://alphacephei.com/vosk/models/vosk-model-small-hi-0.22.zip
unzip vosk-model-small-hi-0.22.zip
mv vosk-model-small-hi-0.22 vosk-model-hindi
```

3.2 Module Development

Module 1: asr_module.py

• Handles microphone input and speech recognition

• Implements Vosk recognizer with Hindi model

• Manages audio stream and voice activity detection

• Returns recognized Hindi text

Module 2: intent_handler.py

• Processes Hindi text from ASR

• Implements pattern matching for command recognition

• Executes appropriate action functions

• Generates Hindi response text

Module 3: tts_module.py

• Converts Hindi text to speech

• Manages eSpeak-NG interface

• Controls voice parameters (speed, pitch)

• Outputs audio to speaker

Module 4: main.py

• Coordinates all modules

• Implements main application loop

• Handles initialization and cleanup

• Tracks performance metrics

3.3 Code Structure

```
hindi-voice-assistant/
■■■ main.py # Main application entry point
■■■ asr_module.py # Speech recognition module
■■■ intent_handler.py # Command parsing module
■■■ tts_module.py # Speech synthesis module
■■■ requirements.txt # Python dependencies
■■■ README.md # Documentation
```

4. Results

4.1 Performance Metrics

Response Time Analysis:

Metric	Target	Achieved	Status
Average Response Time	< 2.0s	1.7s	■ Pass
Fastest Response	-	1.2s	-
Slowest Response	-	2.3s	-
Recognition Accuracy	> 90%	92%	■ Pass

Response Time Breakdown:

• ASR Processing: ~0.5s (29%)

• Intent Recognition: ~0.2s (12%)

• Action Execution: ~0.3s (18%)

• TTS Generation: ~0.7s (41%)

System Resource Utilization:

Resource	Usage	Available	Utilization
RAM	1.2 GB	4 GB	30%
CPU	65%	100%	65%

Storage	1.8 GB	32 GB	5.6%
CPU Temperature	52°C	85°C max	Safe

4.2 Functional Testing

Command Recognition Test Results:

Command	Tests	Success	Accuracy
■■■■■■ (Hello)	10	10	100%
■■■ ■■■■ (Time)	10	9	90%
■■■■■ ■■■■ (Date)	10	9	90%
■■■■ (Weather)	10	8	80%
■■■■■■■ (Thanks)	10	10	100%
■■■■■ ■■■■■ (Joke)	10	9	90%
■■■■■■■ ■■■■■ (Vol+)	10	9	90%
■■■ ■■■ (Help)	10	10	100%

Overall Accuracy: 92%

4.3 Offline Operation Verification

Test Scenario: Disconnected from internet and tested all commands

Results:

- ■ All commands functioned without internet
- ■ Response times unchanged
- ■ No degradation in recognition accuracy
- ■ Complete offline operation confirmed

5. Hardware Utilization

5.1 Raspberry Pi 4 Specifications

CPU: Broadcom BCM2711, Quad-core Cortex-A72 (ARM v8) 64-bit SoC @ 1.5GHz

RAM: 4GB LPDDR4-3200 SDRAM

Storage: 32GB microSD card

Connectivity: 2.4 GHz and 5.0 GHz IEEE 802.11ac wireless

5.2 Component Utilization

CPU Cores:

• Core 0: 75% (Main application + ASR)

• Core 1: 65% (Audio processing)

• Core 2: 55% (TTS synthesis)

• Core 3: 50% (System processes)

Memory Breakdown:

• System: 0.8 GB

• Vosk Model: 0.2 GB

• Application: 0.2 GB

• Free: 2.8 GB

Thermal Management:

• Idle Temperature: 42°C

• Under Load: 52°C

• Maximum Observed: 58°C

• Cooling: Passive heatsink + active fan

5.3 Power Consumption

Measurements:

• Idle: 2.7W

• Active Processing: 4.2W

• Peak: 5.1W

• Average: 3.8W

6. Optimization Techniques

6.1 Model Selection

Vosk Model Comparison:

Model	Size	Load Time	Accuracy	Selected
vosk-model-hi	2.3 GB	15s	96%	■ Too large
vosk-model-small-hi	45 MB	2s	92%	■ Optimal

Decision: Selected small model for:

• Faster loading (2s vs 15s)

• Lower memory footprint (45MB vs 2.3GB)

• Acceptable accuracy trade-off (92% vs 96%)

• Better response time

6.2 Audio Processing

Buffer Size Optimization:

• Tested: 2048, 4096, 8192, 16384 samples

• Selected: 8192 samples

• Reason: Best balance between latency and stability

Sampling Rate:

• Standard: 16 kHz (optimal for speech)

• Alternative 8 kHz: Lower quality, marginal speed improvement

• Alternative 44.1 kHz: Unnecessary overhead for speech

6.3 Intent Recognition

Pattern Matching vs Machine Learning:

• Pattern Matching: O(n) complexity, instant response

• ML-based: Would require training, higher latency

• Decision: Pattern matching for sub-2s requirement

Optimization:

• Early exit on first match

• Case-insensitive matching

• Keyword-based instead of full text parsing

6.4 System-Level Optimizations

1. Boot Optimization:

• Disabled GUI desktop (saved 200MB RAM)

• Disabled unnecessary services

• Boot time reduced from 45s to 25s

2. CPU Governor:

• Set to "performance" mode during active use

• Ensures consistent response times

3. Memory Management:

• Pre-loaded models at startup

• Avoided dynamic memory allocation in loops

• Reused audio buffers

7. Challenges and Solutions

7.1 Technical Challenges

Challenge 1: Hindi ASR Accuracy

• **Problem:** Initial accuracy was 75% due to accent variations

• **Solution:**

- Used fuzzy matching for intent recognition

- Added multiple pattern variations per command

- Improved microphone positioning guidelines

- **Result:** Accuracy increased to 92%

Challenge 2: TTS Voice Quality

• **Problem:** eSpeak-NG Hindi voice sounds robotic

• **Solution:**

- Adjusted speech rate to 150 WPM (from default 175)

- Optimized pitch to 50 (mid-range)

- Kept responses short and clear
- **Result:** Acceptable voice quality for offline system

Challenge 3: Response Latency

• **Problem:** Initial response time was 3.2 seconds
• **Solution:**
- Pre-loaded Vosk model at startup
- Optimized buffer sizes
- Used lightweight model variant
- Implemented efficient pattern matching
- **Result:** Reduced to 1.7s average (46% improvement)

Challenge 4: Resource Constraints

• **Problem:** 4GB RAM limitation with large models
• **Solution:**
- Selected small Vosk model (45MB vs 2.3GB)
- Disabled GUI desktop
- Optimized memory allocation
- **Result:** Peak RAM usage only 1.2GB (30% utilization)

7.2 Development Challenges

Challenge 5: WiFi Configuration

• **Problem:** Pi not connecting to network initially
• **Solution:** Verified SSID case-sensitivity in config
• **Lesson:** Always double-check network credentials

Challenge 6: SSH Connection Issues

• **Problem:** "Connection refused" error
• **Solution:** Ensured SSH was enabled in Raspberry Pi Imager
• **Lesson:** Enable SSH before flashing SD card

8. Future Enhancements

8.1 Short-term Improvements

1. Wake Word Detection

• Implement "OK Assistant" wake word

• Reduces false activations

• Improves user experience

2. Expanded Command Set

• Add calculator functionality

• Implement timer/alarm features

• Add music playback controls

3. Visual Feedback

• LED indicator for listening state

• Display current status

• Error indication

8.2 Medium-term Enhancements

1. Contextual Conversations

• Remember previous queries

• Support follow-up questions

• Multi-turn dialogues

2. User Preferences

• Customizable voice settings

• Personal command shortcuts

• User-specific responses

3. Multiple Language Support

• Add Tamil, Telugu, Bengali

• Language switching commands

• Multi-lingual responses

8.3 Long-term Vision

1. Advanced NLU

• Better intent recognition

• Entity extraction

• Sentiment analysis

2. Home Automation

• IoT device control

• Smart home integration

• Voice-controlled appliances

3. Offline Knowledge Base

• Local Wikipedia subset

• Common queries database

• Educational content

9. Conclusion

9.1 Project Outcomes

This project successfully demonstrates a fully functional offline Hindi voice assistant on Raspberry Pi 4, achieving all stated objectives:

Achievements:

■ Fully offline operation with zero cloud dependency

■ Hindi language support with 92% recognition accuracy

■ Sub-2-second response time (1.7s average)

■ 15 functional Hindi commands implemented

■ Complete privacy preservation

■ Efficient resource utilization (30% RAM, 65% CPU)

Key Contributions:

1. Demonstrated feasibility of offline voice assistants on low-cost hardware
2. Provided privacy-preserving alternative to cloud-based assistants
3. Advanced Hindi language support on embedded devices
4. Created modular, extensible architecture for future enhancements

9.2 Learning Outcomes

Technical Skills:

• Embedded systems programming on ARM architecture
• Speech recognition and synthesis implementation
• Real-time audio processing optimization
• Python development for resource-constrained devices

System Design:
• Modular architecture design
• Performance optimization techniques
• Resource-efficient algorithm selection
• Hardware-software co-optimization

Domain Knowledge:
• Natural Language Processing for Hindi
• Voice user interface design
• Offline AI model deployment
• Privacy-preserving system architecture

9.3 Impact and Applications

Privacy-Focused Computing:
• Demonstrates viability of local AI processing
• Eliminates data transmission to cloud servers
• Enables voice assistance in security-sensitive environments

Regional Language Support:
• Advances Hindi language technology
• Reduces dependency on English for voice interfaces
• Promotes digital inclusion for Hindi speakers

Educational Value:
• Serves as learning platform for voice AI
• Demonstrates embedded system capabilities
• Open-source contribution for community

10. References

10.1 Technologies Used

1. **Vosk Speech Recognition:** https://alphacephei.com/vosk/
2. **eSpeak-NG Text-to-Speech:** https://github.com/espeak-ng/espeak-ng
3. **PyAudio:** https://people.csail.mit.edu/hubert/pyaudio/
4. **Raspberry Pi Foundation:** https://www.raspberrypi.org/

10.2 Documentation

1. Vosk API Documentation: https://alphacephei.com/vosk/api
2. Raspberry Pi OS Guide: https://www.raspberrypi.com/documentation/
3. Python-docx Library: https://python-docx.readthedocs.io/
4. eSpeak-NG User Guide: https://github.com/espeak-ng/espeak-ng/blob/master/docs/guide.md

10.3 Research Papers

1. Povey, D., et al. "The Kaldi Speech Recognition Toolkit" (Vosk is based on Kaldi)
2. "Speech Recognition for Indian Languages" - Various research papers on Hindi ASR
3. "Privacy-Preserving Voice Interfaces" - Academic research on local voice processing

Appendices

Appendix A: Complete Command List

1. ■■■■■■ - Greeting
2. ■■■ ■■■■ - Current time
3. ■■■■■ ■■■■ - Current date
4. ■■■■ ■■■■ ■■ - Weather (simulated)
5. ■■■■■■■ - Thank you
6. ■■■■■ ■■■■■ - Tell joke
7. ■■■ ■■■ - List commands
8. ■■■ ■■■ ■■ - Identity
9. ■■■■■■■ ■■■■■ - Increase volume
10. ■■■■■■■ ■■■■ - Decrease volume
11. ■■■■■ ■■■■■ ■■ - Battery status

12. ■■■ ■■■ - Exit
13. ■■■■ ■■■ - Calculate (basic)
14. ■■■■■ ■■■ - Reboot (disabled for safety)
15. ■■■■ ■■■ ■■ - What time is it

Appendix B: System Specifications

Hardware:

• Device: Raspberry Pi 4 Model B

• RAM: 4GB LPDDR4

• Storage: SanDisk 32GB Class 10 microSD

• Microphone: Generic USB Microphone

• Speaker: 3.5mm jack compatible speaker

• Cooling: Aluminum heatsink + 5V fan

Software:

• OS: Raspberry Pi OS (64-bit) - Debian 11 (Bullseye)

• Kernel: Linux 6.1.0-rpi4-rpi-v8

• Python: 3.9.2

• Vosk: 0.3.45

• PyAudio: 0.2.13

• eSpeak-NG: 1.50

Appendix C: Installation Commands

```
# System setup
sudo apt update
sudo apt upgrade -y
sudo apt install -y python3-pip python3-pyaudio portaudio19-dev
sudo apt install -y git vim alsa-utils espeak-ng unzip wget

# Python packages
pip3 install vosk pyaudio --break-system-packages

# Download Hindi model
cd ~
wget https://alphacephei.com/vosk/models/vosk-model-small-hi-0.22.zip
unzip vosk-model-small-hi-0.22.zip
mv vosk-model-small-hi-0.22 vosk-model-hindi

# Test microphone
```

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

# Test TTS
espeak-ng -v hi "■■■■■■"

# Run application
python3 main.py
```

Acknowledgments

I would like to thank:

• **Arm India** and **IIT Madras** for organizing the Bharat AI-SoC Student Challenge

• **Faculty Mentor:** Pradhipa Mam for guidance and support

• **Vosk and eSpeak-NG communities** for excellent open-source tools

• **Raspberry Pi Foundation** for providing accessible computing platform

Project Repository: [\[GitHub URL\]](https://github.com/Ashwin-1209/hindi-voice-assistant-raspberry-pi/blob/main/README.md)

Demo Video: [\[YouTube/Drive URL\]](https://youtu.be/A5hqWm3FmXY)

Contact: rashwin.ece2024@citchennai.net