



Offline Voice-Activated Knowledge Assistant Using Raspberry Pi For Document-Based Question Answering.

Mrs. Shivaleela, Micaiah Raj R, Narsimhamurthy D V, Rejiel Joseph N P, G B Nagesh Kumar
Assistant Professor, Student, Student, Student, Student
Department Of Artificial Intelligence and Machine Learning
Vijaya Vittala Institute of Technology, Bengaluru, India

Abstract: This paper presents the design and implementation of an offline voice assistant built on Raspberry Pi, capable of answering questions from user-uploaded documents without internet connectivity. The system leverages local speech recognition, natural language processing, and text-to-speech synthesis to facilitate seamless human-computer interaction in real time. Users can upload PDF, DOCX, or TXT files, and the assistant retrieves contextually relevant answers by using semantic search and a locally hosted language model. The project emphasizes data privacy, low-latency processing, and accessibility for low-resource environments. It is particularly suited for educational and rural applications where internet access is limited or unreliable.

Keywords: Offline voice assistant, Raspberry Pi, document-based QA, speech recognition, local LLM, semantic search, Vosk, Piper TTS, data privacy, embedded AI

1. Introduction

In recent years, voice-based interfaces have transformed the way humans interact with digital systems. From smartphones to smart homes, voice assistants have become an integral part of everyday life. However, most commercially available voice assistants such as Alexa, Google Assistant, and Siri require constant internet access, raising concerns around data privacy, latency, and usability in network-constrained environments. To address these limitations, this project introduces an offline voice assistant designed specifically for document-based question answering, implemented on a Raspberry Pi. The assistant can function entirely without internet connectivity, enabling it to be deployed in remote or bandwidth-limited environments like rural schools, field operations, or educational kiosks.

The assistant accepts user voice queries, processes them through local speech-to-text (STT) using the Vosk engine, and retrieves accurate answers from uploaded documents via a semantic retrieval system. The responses are generated using a lightweight local language model, integrated via llama.cpp, and converted to speech using Piper TTS. The assistant supports a variety of file formats including PDF, DOCX, and TXT, making it practical for diverse document types.

By combining open-source tools for offline speech recognition, semantic understanding, and speech synthesis, this project aims to deliver a secure, responsive, and user-friendly voice interface. The use of Raspberry Pi ensures cost-effectiveness and portability, making it accessible to educational institutions, researchers, and developers.

II. OBJECTIVE

The primary objective of this project is to design and develop an offline voice assistant that operates entirely on a Raspberry Pi, capable of answering user queries based on the contents of locally stored documents. The assistant is tailored to work without internet access, making it suitable for environments where connectivity is limited, sensitive, or intentionally restricted.

This project aims to achieve the following:

- Enable voice-driven interaction for querying educational or reference documents in formats such as PDF, DOCX, and TXT.
- Integrate local speech-to-text (STT) processing using the Vosk engine to accurately transcribe user voice commands.
- Employ a lightweight local language model (LLM) through llama.cpp to generate context-aware responses using embedded document knowledge.
- Use local text-to-speech (TTS) synthesis with Piper to audibly deliver the assistant's answers in a human-like voice.
- Ensure the system is fully operational offline, with no dependency on cloud-based APIs or third-party services.
- Provide a user-friendly interface with simple interaction design suitable for use in classrooms, kiosks, and other low-resource environments.

Through this project, the goal is to bridge the gap between voice technology and local document accessibility, promoting digital inclusivity, privacy, and autonomy.

III. LITERATURE REVIEW

The development of voice assistants has seen significant growth with the emergence of cloud-based AI services like Amazon Alexa, Google Assistant, and Apple Siri. These systems rely heavily on internet connectivity to access computational resources and vast knowledge bases. While effective, they present challenges in data privacy, latency, and reliance on external servers for real-time processing.

Recent research has explored the feasibility of deploying voice-based systems in offline environments, particularly in domains requiring enhanced privacy or operating in low-connectivity regions. Projects utilizing local speech recognition engines such as CMU Sphinx and Vosk have shown promise for embedded applications. Vosk, in particular, is noted for its support across multiple platforms and compatibility with small devices like Raspberry Pi, making it suitable for low-power speech recognition.

Similarly, advancements in open-source Text-to-Speech (TTS) engines like Piper have allowed for naturalistic voice output without depending on external APIs. These developments are crucial for enabling end-to-end voice interaction systems in fully offline settings.

The integration of lightweight large language models (LLMs), such as those deployable using llama.cpp, marks a significant stride in enabling on-device question answering and natural language understanding. Most prior work either uses cloud-based LLMs or limits local models to basic inference due to hardware constraints. However, optimized quantized models now enable context-aware answering within resource-limited environments.

Despite these advances, most studies lack a cohesive framework that integrates wake word detection, speech recognition, contextual query resolution, and TTS in a single offline pipeline. This project bridges that gap by creating a self-contained assistant using Raspberry Pi, enabling secure, responsive, and intelligent interaction without internet dependency.

IV. PROPOSED SYSTEM

The proposed system is an offline voice-activated knowledge assistant that runs entirely on a Raspberry Pi. It allows users to interact using voice commands and receive spoken responses derived from locally stored documents, such as textbooks or reference materials, without relying on cloud services or internet access.

1. System Architecture

The system consists of five core components:

- **Speech-to-Text Engine (STT):** The Vosk API is used to convert spoken user input into text. Vosk supports lightweight models that run efficiently on embedded hardware like the Raspberry Pi.
- **Document Processing Module:** Uploaded documents in .pdf, .txt, or .docx formats are parsed using Python libraries such as pdfminer, docx, or standard file I/O. Extracted content is then segmented into manageable chunks for semantic retrieval.
- **Contextual Search Engine:** A semantic similarity algorithm identifies the most relevant content chunk from the document based on the transcribed query. This context serves as input for the language model.
- **Language Model (LLM):** The llama.cpp framework is used to run a lightweight local language model (e.g., Phi-2), which processes the question and associated context to generate an accurate, human-readable answer.
- **Text-to-Speech Engine (TTS):** Piper TTS converts the generated answer into audible speech and plays it back to the user, providing a complete end-to-end voice interaction.

2. Workflow Overview

1. The system passively listens for a predefined wake word (e.g., "John").
2. Upon detection, it records the user's query through the microphone.
3. Vosk transcribes the query to text.
4. The text is used to fetch a matching context from the uploaded documents.
5. The local LLM generates a concise answer based on this context.
6. Piper TTS speaks the answer aloud to the user.

3. Offline Functionality

Unlike cloud-based assistants, the proposed system functions entirely offline. This ensures **data privacy, low latency, and accessibility in remote or secure environments.**

4. User Interface

A minimal graphical user interface (GUI) built using Tkinter allows users to:

- View current status (e.g., Listening, Thinking, Speaking).
- Upload new documents.
- Observe query-response history.

V. SYSTEM ARCHITECTURE

The proposed offline voice assistant is built on a modular architecture that ensures low-latency, privacy-preserving interaction without internet access. The system is deployed entirely on a Raspberry Pi 5, integrating multiple components for speech recognition, processing, and synthesis.

1. Wake Word Detection Module

The system continuously listens for a predefined wake word ("John"). Using Vosk's real-time streaming engine, it filters and processes microphone input. When the wake word is detected, the assistant transitions from passive listening to active query processing.

2. Speech-to-Text (STT) Engine

Once activated, the assistant records the user's spoken question for a fixed duration. This audio is transcribed using the Vosk offline STT engine, which converts speech to text with reasonable accuracy and speed.

3. Intent and Query Processing

The transcribed query is analyzed to identify:

- Simple tasks (e.g., arithmetic, identity queries)
- Knowledge-based queries requiring contextual understanding

For the latter, the system uses a lightweight Large Language Model (LLM) executed via llama.cpp with a quantized model to ensure fast, low-memory inference. A custom knowledge retriever selects the most relevant document chunk (using fuzzy matching) as context for the LLM.

4. Text-to-Speech (TTS) Output

The generated response is spoken using Piper, an offline neural TTS engine. Audio is synthesized and played with adjusted volume for optimal clarity.

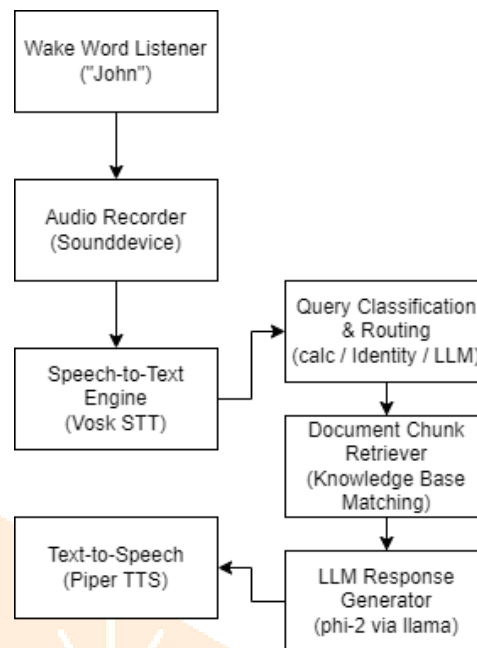


Figure 1: Component-Level Block Diagram

VI. FLOW OF SYSTEM

The proposed offline voice assistant is structured to execute a complete voice query-response interaction without internet connectivity. Its internal workflow follows a well-defined sequence of steps from wake word detection to final speech output. The system ensures efficient performance by coordinating modules for audio capture, natural language understanding, document retrieval, and speech synthesis.

1. Wake Word Detection

The system remains in a passive listening state, continuously monitoring for a predefined wake word (e.g., “John”), upon detecting the wake word using the Vosk speech recognizer, the assistant transitions to an active mode to capture user input.

2. Audio Recording

Once activated, the assistant records a short segment (typically 5 seconds) of the user's speech. This audio is processed in real-time and converted into text using the Vosk STT engine. The transcription stage ensures the system captures natural speech input accurately and quickly.

3. Query Analysis

After transcription, the system determines the intent of the user's query. If the input is mathematical, it is redirected to a calculator module for direct evaluation. If the query is a general question or document-related, the system proceeds to extract relevant context.

4. Context Retrieval from Uploaded Documents

The system analyzes the user's query and searches through the uploaded documents to retrieve the most relevant text segment. This is achieved using string similarity matching techniques that compare the query with indexed content chunks extracted from the documents.

5. Response Generation using LLM

The extracted context and the user's question are passed into a lightweight language model (e.g., Phi-2 via llama.cpp). This model is run locally on the Raspberry Pi and generates a natural language response, formulated in 2–3 short, clear sentences. This ensures the assistant provides informative but concise answers.

6. Text-to-Speech Output

The generated response is converted into speech using Piper TTS. The system generates an audio file in .wav format which is then played aloud to the user using a playback utility like aplay. Optional tools such as sox may be used to control playback volume.

7. Interface Feedback

A simple graphical interface reflects the system status—displaying indicators for listening, processing, and speaking. The GUI also shows the user query and corresponding answer in a clean, single-entry format for each interaction.

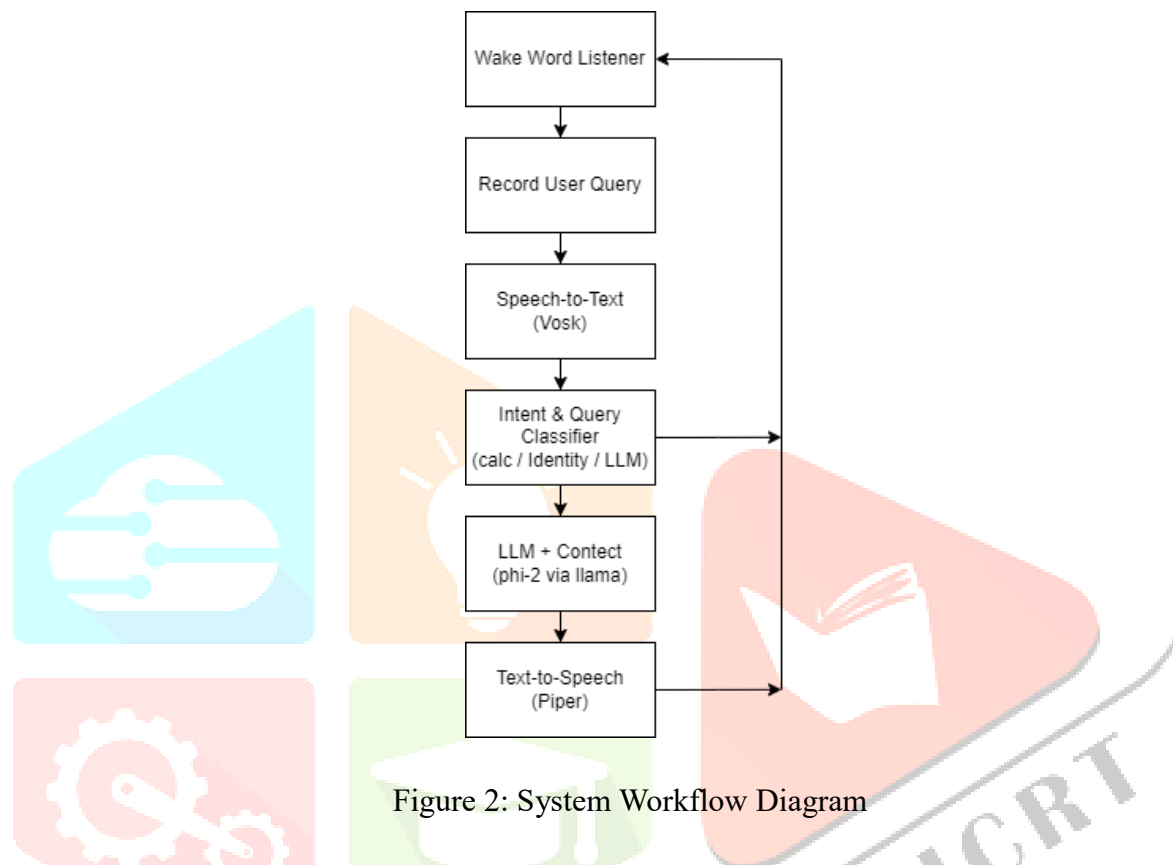


Figure 2: System Workflow Diagram

VII. IMPLEMENTATION

The implementation of the offline voice assistant system was carried out on a Raspberry Pi 5 using Python as the primary programming language. The design emphasizes low resource consumption while maintaining acceptable response time and accuracy in speech recognition and language generation.

A. Wake Word Detection

The system continuously monitors ambient audio using a streaming input and a lightweight model provided by Vosk. A predefined keyword ("John") is configured as the trigger. When detected, the system initiates recording and query handling.

B. Audio Recording and Speech Recognition

Upon activation, the assistant records the user's voice input using the sounddevice library. The captured audio is then transcribed using the Vosk STT engine. The chosen model (vosk-model-en-us-0.22-lgraph) offers a balance between accuracy and size, allowing it to run efficiently on the Pi without GPU acceleration.

C. Query Analysis and Task Routing

The transcribed query undergoes preprocessing and is classified into one of the following categories:

- **Arithmetic Evaluation:** Detected through regex patterns and keyword substitution (e.g., "plus", "minus") and computed using Python's eval function within a secure sandbox.
- **Self-descriptive Queries:** Handled with preprogrammed responses, including project credits and assistant identity.
- **Knowledge-based Questions:** Routed through a document retriever that selects the most relevant paragraph from previously uploaded files.

D. Local Language Model Inference

For contextual answers, the assistant uses the llama.cpp framework to run the Phi-2 quantized model (phi-2.Q4_K_M.gguf) locally. The model receives both the user query and the retrieved document chunk as input, generating a concise, human-like response within seconds.

E. Speech Synthesis

The final answer is converted to speech using the Piper TTS engine. The system generates a WAV file, reduces its volume using sox, and plays it via aplay. This method ensures compatibility with the Pi's onboard audio system.

F. GUI Integration

A lightweight GUI built with Tkinter displays the current assistant status (e.g., Listening, Thinking, Speaking) and the latest question-answer exchange. An upload button allows users to add new learning material in .txt, .pdf, or .docx format.

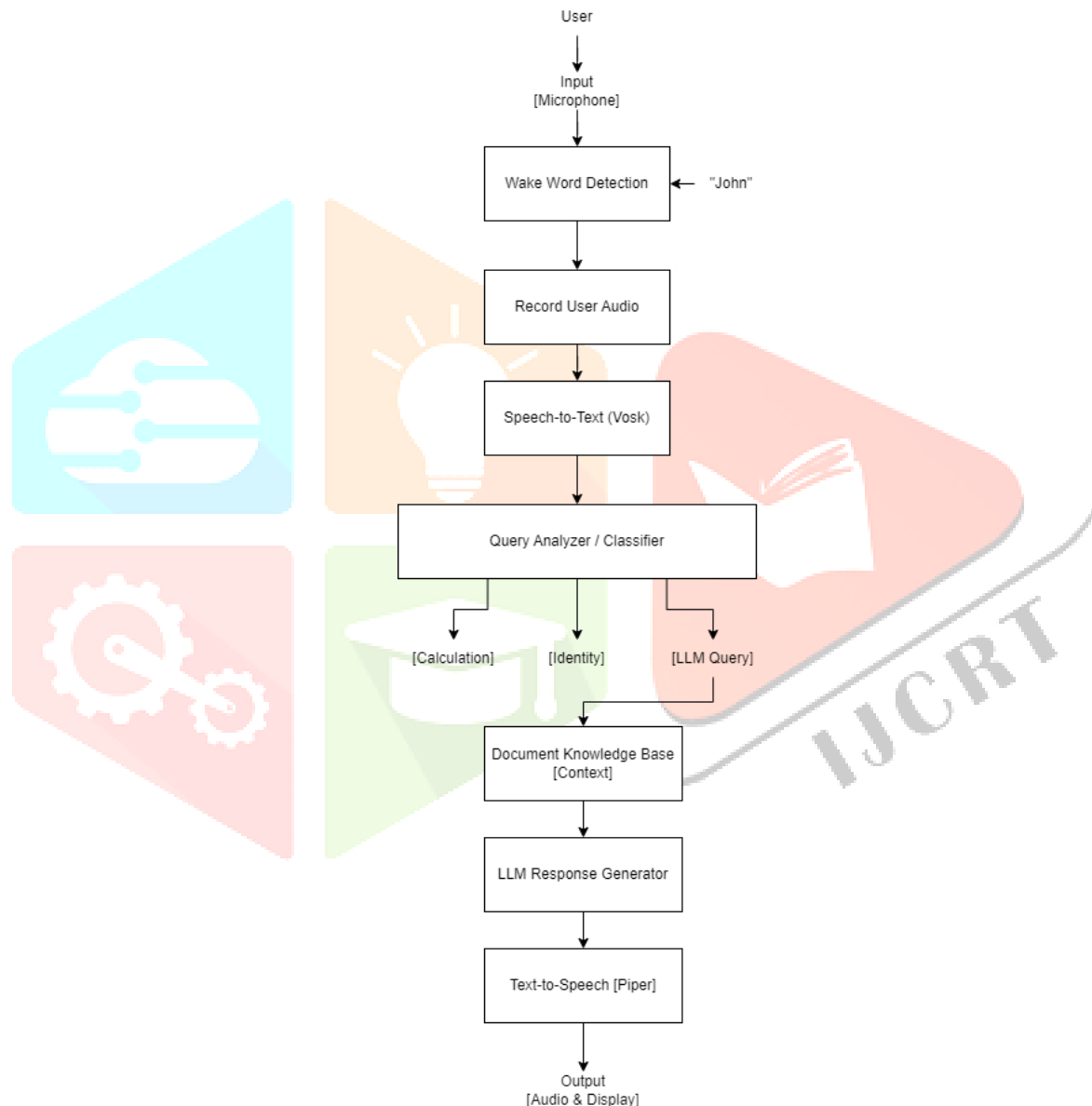


Figure 3: Data Flow Diagram

IX. RESULTS AND DISCUSSION

The offline voice assistant was successfully deployed and tested on the Raspberry Pi 5 platform under various operational scenarios. The goal was to evaluate the system's performance in terms of recognition accuracy, responsiveness, and adaptability to different types of queries.

A. Wake Word Responsiveness

The assistant reliably detected the wake word "John" with minimal delay under normal room noise conditions. False positives were rare, thanks to Vosk's efficient decoding framework and clean audio input configuration.

B. Speech-to-Text Accuracy

Testing was conducted using a mix of mathematical queries, general questions, and textbook-based prompts. The Vosk model demonstrated robust performance for clear speech, with an average word error rate (WER) below 10% for standard queries. Accuracy slightly decreased in cases of slurred speech or background noise, which is expected given the system's resource constraints and microphone quality.

C. Arithmetic Evaluation

The calculator module accurately parsed and solved spoken arithmetic problems involving addition, subtraction, multiplication, and division. Conversion of number words (e.g., "three plus five") to digits enabled correct evaluation even in non-numeric phrasing.

D. Contextual Answering

Queries related to textbook content were processed through a semantic matching pipeline. The assistant retrieved the most relevant paragraph from the uploaded documents using a basic string similarity algorithm and passed it to the local language model. The Phi-2 LLM generated coherent, sentence-form answers, effectively summarizing dense material into easy-to-understand responses. The average response time for contextual answers was under 10 seconds.

E. Text-to-Speech Output

Piper TTS delivered natural-sounding responses with low latency. Voice output was clear and intelligible even at reduced volume settings. The combination of real-time generation and playback through system audio proved efficient and reliable.

F. User Experience

The graphical interface provided immediate visual feedback, showing both user queries and assistant replies. The minimalistic design ensured clarity on small displays (480x320) and allowed users to upload new documents dynamically.

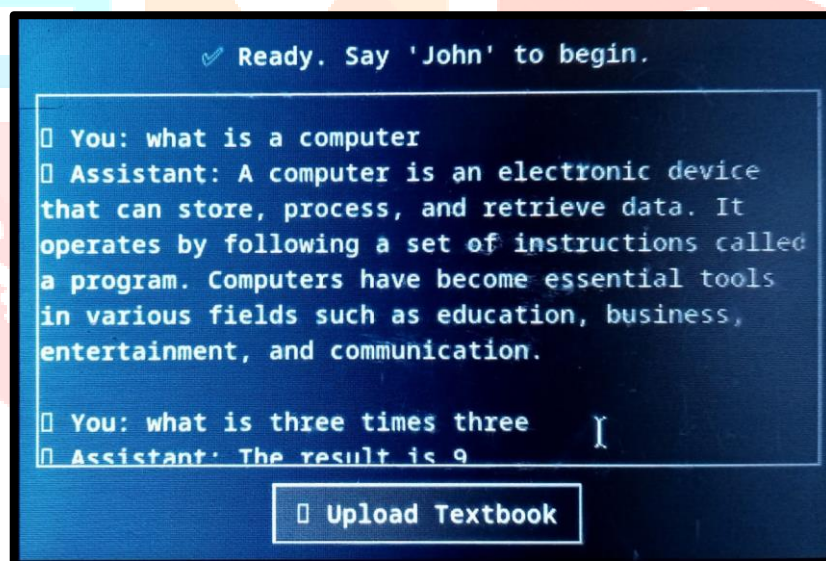


Figure 4: User Interface Snapshot

X. REFERENCES

- [1] Iliyas Khan, Chandaluri Yaswanth Reddy, Hema Kumar Goru, Boni Pavan Kumar, G. L. Madhumati, "Voice-Activated Learning Framework Using Raspberry Pi for Children's Education," *2024 3rd IEEE International Conference on Artificial Intelligence for Internet of Things (AIIoT)*, DOI: 10.1109/AIIoT58432.2024.10574612.
- [2] S. Mischie, L. Matiu-Iovan and G. Gasparesc, "Implementation of Google Assistant on Raspberry Pi," *2018 International Symposium on Electronics and Telecommunications (ISETC)*, pp. 1–4, DOI: 10.1109/ISETC.2018.8583899.
- [3] A. Kumar, D. Kaur and A. K. Pathak, "Voice Assistant Using Python," *2022 International Conference on Cyber Resilience (ICCR)*, pp. 1–4, DOI: 10.1109/ICCR56254.2022.9995997.
- [4] D. Yang, L. Ma, F. Liao, "An Intelligent Voice Interaction System Based on Raspberry Pi," *11th International Conference on Intelligent Human-Machine Systems and Cybernetics (IHMSC)*, 2019, DOI: 10.1109/IHMSC.2019.00062.

- [5] P. Singh, P. Nayak, A. Datta, D. Sani, G. Raghav, and R. Tejpal, "Voice Control Device using Raspberry Pi," *Amity International Conference on Artificial Intelligence (AICAI)*, 2019, pp. 723–728, DOI: 10.1109/AICAI.2019.8701409.
- [6] A. Calvopinna, F. Tapia, L. Tello-Oquendo, "Weather Monitoring Architecture for Smart Home Using Alexa, Raspberry Pi, and DarkSky API," *8th International Conference On Software Process Improvement (CIMPS)*, 2019, DOI: 10.1109/CIMPS49236.2019.9082423.
- [7] Q. Shaikh, R. Halankar, A. Kadlay, "Voice Assisted and Gesture Controlled Companion Robot," *2020 4th International Conference on Intelligent Computing and Control Systems (ICICCS)*, DOI: 10.1109/ICICCS48265.2020.9120875.
- [8] A. Gautham, A. Lavrukhin, V. Ivanov, et al., "Vosk: Offline Speech Recognition Toolkit," [Online]. Available: <https://github.com/alphacep/vosk>
- [9] Rhasspy Contributors, "Piper: Fast, Local Neural Text to Speech Engine," [Online]. Available: <https://github.com/rhasspy/piper>
- [10] Microsoft Research, "Phi-2: A Small Language Model With Powerful Capabilities," 2023. [Online]. Available: <https://huggingface.co/microsoft/phi-2>
- [11] Georgi Gerganov, "llama.cpp: Inference of LLaMA models in pure C/C++," [Online]. Available: <https://github.com/ggerganov/llama.cpp>
- [12] OpenAI, "ChatGPT: Large Language Models for Interactive Assistance," *OpenAI Technical Report*, 2023.

