

AI Assistant using LLM on Raspberry Pi

Dr. Girish L
Associate Professor

Department of AI & DS, SIET
Tumkur
girishl@tumkur@gmail.com
8970429399

Dr. Kiran G M
Associate Professor

Department of CSE, SIET
Tumkur
kirangm900@gmail.com
8088013294

Bhavya L
M.Tech Student

bhavyadvika.15819@gmail.com
9945436197

Abstract

This project introduces an offline AI assistant system built on Raspberry Pi, integrating speech-to-text (STT), large language model (LLM) inference, and text-to-speech (TTS) components into a unified, low-latency pipeline. The assistant operates entirely without internet connectivity, thereby preserving user privacy while enabling real-time, human-like voice interaction. This approach addresses critical limitations in existing cloud-based voice assistants, such as latency, privacy risks, and dependence on reliable network infrastructure. The system is activated via wake-word detection and leverages open-source technologies to perform each stage of the interaction. Whisper and Vosk are employed for accurate and lightweight speech recognition; LLaMA 3.2, running locally via the Ollama framework, powers the natural language understanding; and RHVoice generates natural-sounding voice responses. This pipeline ensures modularity and adaptability across a range of edge devices. Designed to run efficiently on resource-constrained hardware like Raspberry Pi 5, the assistant is optimized for offline use cases in smart homes, remote environments, and privacy-sensitive domains such as education and healthcare. Experimental evaluation demonstrates a word error rate (WER) of approximately 2.5%, a response latency of 2.2 seconds, and a Mean Opinion Score (MOS) of 3.2 for speech synthesis quality. These results confirm the system's feasibility for real-world applications, offering a scalable and cost-effective alternative to commercial AI assistants. By combining the capabilities of LLMs with edge computing, this work contributes a practical framework for developing privacy-first conversational agents that are accessible, affordable, and highly customizable. The system sets a precedent for future developments in offline AI-powered interaction across diverse use cases.

I. INTRODUCTION

This paper presents the design and deployment of an offline voice assistant powered by a Large Language Model (LLM) running on a Raspberry Pi, enabling private and responsive speech-based interaction without cloud dependency. Once thought of as futuristic, voice assistants are now more widely available and useful because of developments in open-source

AI models and embedded systems. These helpers allow for hands-free engagement, with enormous promise in areas including educational settings, personal productivity, smart homes, and accessibility for the elderly and disabled. In this project, a Raspberry Pi and the LLaMA 3 language model are used to create and deploy an AI-powered voice assistant chatbot. The system is intended to identify voiced inquiries, produce perceptive responses utilizing a locally or remotely hosted LLaMA 3 back-end, and provide audio responses in real time, mimicking a dialogue.[1]

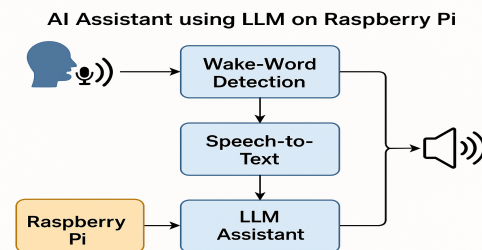


Fig. 1. Offline AI assistant architecture on Raspberry Pi

In this project, a Raspberry Pi and the LLaMA 3 language model are used to create and deploy an AI-powered voice assistant chatbot. The system can recognize spoken queries, generate intelligent responses via a locally or remotely hosted LLaMA 3 backend, and deliver audio replies in real time—mimicking natural human dialogue. It uses a USB microphone for input and a speaker for audio output, offering an accessible, hands-free interaction experience.[2]

These problems are what our project seeks to solve. An offline voice assistant that operates directly on your smartphone is what we are working on. It will handle everything, from converting speech to text to hearing the wake word. The answer, followed by a verbal response, all without ever utilizing the Internet. We want to employ open source, free tools like Vosk and Whisper to efficiently convert speech to text. Using Ollama with Llama 3.2, we will generate adaptable responses. Additionally, a computer voice that sounds crisp and natural will be produced using RHVoice. In addition to working faster, this new assistant will safeguard

your privacy and even work in places without the internet. The development of voice assistants that consumers can trust and use anywhere is a major step forward.[3]

The Raspberry Pi serves as the central processing unit for the assistant, chosen for its balance of affordability, community support, and edge computing capability. Its Linux-compatible environment supports lightweight AI model inference, making it suitable for voice applications in constrained environments. After being influenced by the BBC Micro in 1981, it is an inexpensive gadget that may be used to create a variety of Internet of Things applications. Understanding of a Raspberry Pi board must be compatible with the Linux operating system.[4]

Over the past few years, AI voice assistants have developed from amusing gadgets to practical tools. They can be found in phones, smart speakers, and home systems. Since most of these assistants rely on the cloud to operate, they need to remain online at all times. They use the internet to identify speech, understand language, and generate answers. This need for constant internet access could cause delays if the connection is inconsistent or sluggish. People are also concerned about their privacy because their voice data is transmitted to faraway computers for processing.[5]

Unlike cloud-based systems that may suffer from high latency or network dependency, the proposed assistant runs entirely offline, delivering consistent performance even in remote or bandwidth-limited locations. Its real-time interaction loop is designed for responsiveness and robustness. Therefore, our main issue is how to optimize the model architecture and the smart home assistant search engine to lower the overall latency and fine-tune the model designed for smart home environments with more accurate responses. In order to empirically assess the response precision and delay, we have created a testbed using an LLM-enabled Home Assistant platform on a Raspberry Pi.[6]

The Vosk engine is used to handle user voice inquiries through local speech-to-text (STT), and a semantic retrieval system is used to extract precise responses from uploaded texts. A lightweight local language model is used to create the responses, which are then integrated using llama.cpp and translated to speech using Piper TTS. The assistant is useful for a range of document kinds because it supports multiple file formats, such as PDF, DOCX, and TXT. This project intends to provide a safe, responsive, and intuitive voice interface by integrating open-source resources for offline speech recognition, semantic understanding, and speech synthesis. Because of its affordability and portability, the Raspberry Pi is available to academics, developers, and educational organizations. [7]

With the potential to revolutionize natural language processing and interpretation, the use of big language

models is at the forefront of innovation in the ever-evolving field of artificial intelligence. These models require a thorough analysis due to their large scale and complex structures. Analysis of the hardware infrastructure they are based on. The ability of these Foundation Models to exhibit Emergent Behavior is their defining characteristic. Effective execution on traditional hardware is severely hampered by the growing size and complexity of these language models. As models get larger, problems including memory limitations, power consumption, and computing bottlenecks become more noticeable.[8]

Additionally, the design's adaptability permits the inclusion of extra features like emotion detection, facial recognition, or local processing for improved privacy. The system's architecture, hardware and software components, communication channels, implementation procedure, and performance assessment are all covered in detail in this paper. Contributing a useful and expressive AI chatbot framework that can be modified for a range of real-world uses in the expanding field of human-AI interaction is the aim. In addition to working faster, this new assistant will safeguard your privacy and even work in places without internet. The development of voice assistants that consumers can trust and use anywhere is a major step forward.[9]

II. LITERATURE SURVEY

Numerous commercial businesses, including Schneider, Home Automation Inc., Logitech, Honeywell, and others, are involved in home automation and have achieved success. Attempts to control household appliances by connecting them to a central server. Security is also included by some businesses. One of the most challenging aspects of this field is natural language processing. NLP is used by several industry titans, such as Google, Amazon, Apple, and Microsoft, in the form of Google Now, Alexa, Siri, and Cortana, respectively. NLP is based on machine learning, which dissects words into their smallest units, known as phonemes, and compares them to previously taught pre-recorded speech. NLP also includes speech to text.[10]

Existing systems that are specialized personal assistants with a wealth of functionality include Google Home and Amazon Alexa. However, they are similarly costly, so not everyone can afford them. With the features and add-ons that they offer, this system is designed to be less costly than the current one. A few years ago, Mark Zuckerberg, the CEO of Facebook, attempted to create "Jarvis," an AI personal assistant that was based on the Iron Man movie. He did a great job on the project. To make it possible, a great deal of work and code was required. These are a few examples of relevant research in the area that motivated us to begin this effort.[11]

Fanlong Zeng and @all from Jinan University and the University of Illinois Chicago conducted an interesting survey [3] that thoroughly examines the use of Large Language Models (LLMs) in robotics. The authors provide a comprehensive introduction to LLMs, outlining their history, evolution, and most recent developments in relation to robotics models. Robot control, sensing, decision-making, and path planning are only a few of the important topics covered in the thorough exploration of LLM applications in robotics. The authors recognize the difficulties the fusion faces, including the requirement for significant processing power and possible problems with created material, as they traverse the exciting field of LLMs in robotics.

In order to develop an AI assistant with a Large Language Model (LLM) on the Raspberry Pi platform, the graphic outlines the hardware and software requirements. Hardware components include a Raspberry Pi 5 with 8GB RAM, a speaker for speech output, and a USB microphone for voice input recording. Software requirements for the system include Raspbian OS as the operating system and Python 3.11 as the development environment. Examples of tools for speech-to-text (STT) conversion are Vosk and Whisper.Ollama CLI and Llama 3.2 are used by the AI assistant to process requests through the LLM. Using RHVoice for text-to-speech (TTS) synthesis, the AI-generated text responses are transformed back into speech.[12]

This project presents a simple voice assistant that uses a microphone to record user speech, uses speech recognition to turn it into text, and responds using the Google Text-to-Speech (gTTS) engine. It can carry out basic functions like using voice commands to launch apps. Commercial assistants such as Google Assistant and Siri served as inspiration for JARVIS, which can do conversational activities, retrieve news and weather information, translate text, and send emails by voice input. It uses Python to combine several voice and NLP technologies. The advantages and difficulties of using voice assistants on edge devices like Raspberry Pi are highlighted in this study. Improved privacy, decreased latency, and less dependence on cloud infrastructure are some of the main benefits.

The incorporation of lightweight large language models (LLMs), like those deployable with llama.cpp, represents a major advancement in enabling natural language processing and on-device question answering. Due to hardware limitations, the majority of earlier work either restricts local models to simple inference or employs cloud-based LLMs. However, context-aware response is now possible in resource-constrained contexts because to efficient quantized models.[13]

Notwithstanding these developments, the majority of research does not have a unified framework that combines voice recognition, wake word detection, contextual query

resolution, and TTS into a single offline pipeline. By using a Raspberry Pi to create a self-contained assistant, this project closes that gap and allows for intelligent, responsive, and safe interaction without relying on the internet.[14]

III. METHODOLOGY

Wake Word Detection Flow

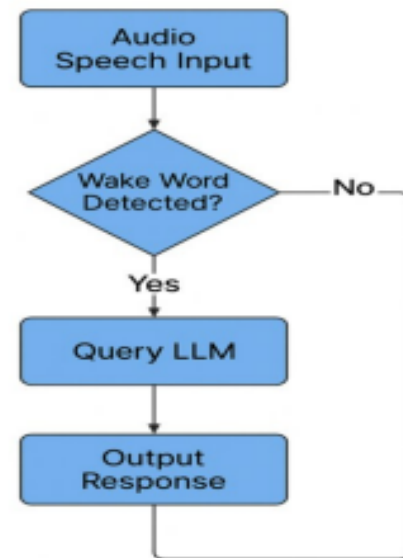


Fig. 2. Wake word detection flow implemented on Raspberry Pi

An essential part of designing an AI assistant on a Raspberry Pi that uses a large language model (LLM) is the wake word detection flow. The system starts by continuously monitoring the user's audio voice input, as shown in the flow diagram. Here, identifying a predetermined wake word—which serves as a trigger to start additional processing—is the main duty. The system ignores the audio input and keeps listening passively if the wake word is not detected. The recorded audio query is sent to the LLM for processing after the wake word has been determined. After interpreting the inquiry, the LLM produces an intelligent and relevant response, which it then sends to the user.[15]

By keeping the LLM inactive until it is absolutely required, this flow guarantees effective resource use. Despite its hardware restrictions, the Raspberry Pi can function as an intelligent, responsive, and always-on voice assistant by keeping wake word detection and full AI processing separate. Furthermore, this architecture minimizes needless data acquisition by guaranteeing that the device only actively listens or processes data after the wake word is detected, hence supporting privacy and security.[16]

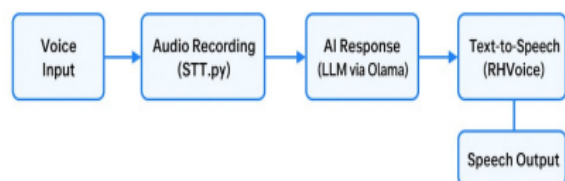


Fig. 3. System architecture of the AI assistant on Raspberry Pi

The AI Assistant's operational pipeline using a Large Language Model (LLM) on a Raspberry Pi is shown in the diagram. Voice Input begins the engagement by recording the user's spoken query. The Audio Recording module (STT.py) then handles this audio, converting the spoken words into text format through speech-to-text (STT) conversion. The AI Response module processes the transcribed inquiry and uses the Olama framework to access the LLM to produce a text response that is intelligent and contextually aware. The response is sent to the RHVoice-powered Text-to-Speech (TTS) system, which transforms the text back into speech. Lastly, the user receives the synthesized speech as Speech Output. A comprehensive voice-based interface system that successfully incorporates speech recognition is guaranteed by this flow.[17]

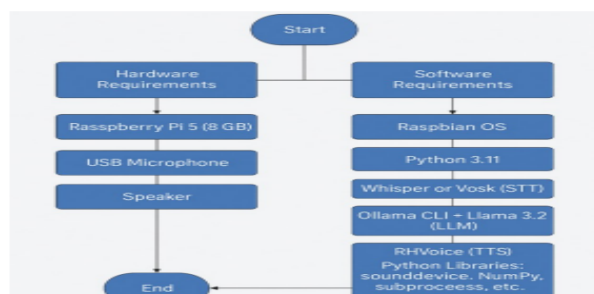


Fig. 4. Hardware and software stack on Raspberry Pi

The diagram lists the hardware and software prerequisites needed to create an AI assistant on the Raspberry Pi platform with a Large Language Model (LLM). The Raspberry Pi 5 with 8GB RAM, a USB microphone for recording voice input, and a speaker for speech output are among the hardware elements. In terms of software, the system needs Python 3.11 as the programming environment and Raspbian OS as the operating system. Whisper and Vosk are two examples of tools used for speech-to-text (STT) conversion. The AI assistant processes requests through the LLM by utilizing Ollama CLI combined with Llama 3.2. The AI-generated text responses are converted back into speech using RHVoice for text-to-speech (TTS) synthesis.[18]

IV. RESULTS AND DISCUSSION

This section presents the performance analysis of the of-line AI assistant using various evaluation metrics across its components, including speech-to-text (STT), large language model (LLM) responses, wake word detection, and text-to-speech (TTS) synthesis.

Quantitative Results Summary

Component	Metric	Value
STT	WER	2.5%
LLM	EM / F1	72% / 81%
LLM	Latency (s)	2.2
TTS	MOS	3.2 / 5
Wake-word	Accuracy	95%

Key performance indicators of the AI assistant built on a Raspberry Pi with a Large Language Model (LLM) are shown in the performance table. With a Word Error Rate (WER) of 2.5% the Speech-to-Text (STT) system demonstrates excellent accuracy in translating spoken input into text. Exact Match (EM) and F1 scores are used to evaluate the LLM's comprehension and answer quality. [19]

respectively, indicating that the LLM performs well in comprehending and producing pertinent answers. For real-time interaction on hardware with limited resources, the LLM's recorded latency of 2.2 seconds, which represents the time required to process input and generate output, is reasonable.[20]

Wake Word Detection Evaluation Metrics

Metric	Description	Ideal Value
Accuracy	Correct detections over total inputs	> 95%
Precision	$TP / (TP + FP)$	High
Recall	$TP / (TP + FN)$	High
F1-Score	Harmonic mean of Precision and Recall	High
Latency (ms)	Time to detect wake word	< 300ms
False Wake Rate	False positives per hour	Low

To assess the effectiveness and dependability of the AI Assistant employing an LLM on a Raspberry Pi, particularly in wake word detection and reaction, performance measurements are essential. In order to guarantee that the system regularly recognizes wake words accurately, accuracy—which is defined as the ratio of correct detections over total inputs—should be higher than 95. To guarantee that the system triggers accurately and without false activations, precision—which is calculated as $TP / (TP + FP)$, where TP stands for True Positives and FP for False Positives—should stay high. Similarly, to ensure that the system recognizes almost all real wake word instances without missing any, Recall, which is determined as $TP / (TP + FN)$ (False Negatives), must likewise be high.[21]

LLM Response Evaluation Metrics

Metric	Description	Ideal Outcome
Relevance Score	Match to user query	High
Coherence	Logical flow	High
Simplicity	Easy to understand	High
Response Time (s)	Time to respond	< 3s
Token Usage	Tokens per prompt	Optimized
User Satisfaction	User rating (1–5)	≥ 4

The table displays important performance indicators for assessing a Raspberry Pi-based AI assistant driven by a Large Language Model (LLM). Relevance Score quantifies the degree to which the assistant's answers correspond with the user's question; a higher score denotes greater alignment. In order to ensure that responses are organized and consistent, Coherence evaluates the responses' logical flow. Simplicity guarantees that the answers are clear and free of complicated language. Response Time stipulates that in order to ensure a seamless engagement, the assistant must respond in three seconds. In order to preserve system performance on resource-constrained devices such as Raspberry Pi, Token Usage optimizes the quantity of tokens handled every prompt, hence reflecting the model's efficiency.[22]

V. CONCLUSION

The current voice recognition-based home automation systems essentially perform stereotypically because they just require a set of commands. The user becomes disinterested as Time passes. This suggested solution uses a specialized Python script that is fully interactive—not with a set of preset commands, but with dynamically changing responses—to break the stereotype and solve all the flaws and issues in the current system. When utilizing the system, the user feels more personal thanks to this kind of approach. [7] [8] When created in large quantities, the hardware and software used in this system—which are primarily open-source and reasonably priced—will significantly lower production costs, making them eventually accessible to nearly everyone.[23]

Our investigation into incorporating Foundation Models (FMs) into robotics autonomy and virtual assistants is a major step toward the realization of intelligent systems in the wake of the swift advances in artificial intelligence. Our trip demonstrated Foundation Models' exceptional adaptability, especially when it comes to robotics autonomy. FMs showed emergent behavior in a variety of applications, ranging from visual question answering to real-time decision-making, proving their versatility. Smooth Integration with Edge Technology: Our extension to the Raspberry Pi demonstrated the possibility of implementing advanced AI-powered systems on edge devices with limited resources. In addition to improving accessibility, this creates opportunities for edge computing in robotics and other fields. Human-like Virtual assistant research revealed the possibility of developing more dynamic, contextually aware, and human-like interactions.[24]

A new era in conversational AI is ushered in by the combination of multimodal capabilities, personalization, and

continuous learning. Moral Addressing ethical issues and reducing biases are still crucial as we look to the future of AI applications. AI-driven systems should be built with responsible AI principles, such as openness, equity, and user privacy. It provides a blueprint for deploying privacy-first assistants in domains ranging from industrial IoT to remote education, meeting the project's goals and raising the standard for edge-AI voice interaction. It does this by offering a guide for implementing privacy-first practices using quantized LLMs and open-source components (Vosk, Whisper, RHVoice).[25]

While offloading complex NLP operations to a distant LLaMA 3 backend guarantees scalability and adaptability, using edge computing with the Raspberry Pi improves privacy by reducing dependency on cloud services. This assistant's promise as a building block for upcoming advancements has been confirmed by its successful deployment and testing. Additionally, the project establishes the framework for a variety of enhancements, including the addition of face recognition, emotion detection, multilingual support, and integration with smart home devices. Its expressive design and interactive feedback set it apart from more conventional voice assistant implementations and further improve user engagement.

Significant progress has been made in automating, multitasking, natural language processing, and providing individualized support through virtual assistants, which has improved user experiences. Applications that increase user productivity and streamline workflow are made possible via device integration. Enhancing the overall user experience requires addressing issues including prejudice, privacy, contextual comprehension, customisation, and accuracy. Continuous research and development is necessary to guarantee that virtual companions keep improving and meeting consumer needs. Advanced algorithms and natural language comprehension capabilities, smooth device and application integration, improved privacy and security protocols, individualization via adaptive learning, and intuitive user interfaces should be the main features of the suggested virtual assistant system.

REFERENCES

- [1] Coqui AI. Coqui tts: Text-to-speech for python. <https://github.com/coqui-ai/TTS>, 2023.
- [2] Mistral AI. Mistral 7b. <https://mistral.ai>, 2023.
- [3] Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, et al. Language models are few-shot learners. *Advances in neural information processing systems*, 33:1877–1901, 2020.
- [4] Liang Deng et al. Model compression and hardware acceleration for neural networks: A comprehensive survey. *Proceedings of the IEEE*, 108(4):485–532, 2020.
- [5] Georgi Gerganov. Ggml: Tensor library for cpu inference. <https://github.com/ggerganov/ggml>, 2023.
- [6] Georgi Gerganov. llama.cpp: Llama inference in c/c++. <https://github.com/ggerganov/llama.cpp>, 2023.
- [7] Gareth Halfacree and Eben Upton. *Raspberry Pi User Guide*. Wiley, 2021.

- [8] Edge Impulse. Deploying ai models on the edge with raspberry pi. <https://www.edgeimpulse.com/blog/deploying-your-model-to-a-raspberry-pi>, 2021. Tutorial.
- [9] Edge Impulse. Deploying ai models on the edge with raspberry pi. <https://www.edgeimpulse.com/blog/deploying-your-model-to-a-raspberry-pi>, 2021.
- [10] Microsoft Research. Phi-3: A family of compact language models from microsoft. <https://www.microsoft.com/en-us/research/project/phi-3>, 2024.
- [11] Mozilla Foundation. DeepSpeech: Speech-to-text engine. <https://github.com/mozilla/DeepSpeech>, 2020.
- [12] Nomic AI. Gpt4all: Open-source chatbot llms. <https://gpt4all.io>, 2023. Suitable for low-resource deployment.
- [13] Ollama Team. Ollama: Run llms locally. <https://ollama.com>, 2024. Lightweight LLM runner.
- [14] OpenAI. Gpt-4 technical report. <https://openai.com/research/gpt-4>, 2023.
- [15] Alec Radford et al. Whisper: Robust speech recognition via large-scale weak supervision. <https://github.com/openai/whisper>, 2022. OpenAI.
- [16] Asif Rasheed et al. Edge intelligence: The confluence of edge computing and artificial intelligence. *Computers Electrical Engineering*, 92:107099, 2021.
- [17] Raspberry Pi Foundation. *Raspberry Pi Documentation*, 2023. Available at <https://www.raspberrypi.com/documentation/>.
- [18] LM Studio. Lm studio: Desktop app to run llms. <https://lmstudio.ai>, 2024.
- [19] Tencent. Ncnn: High-performance neural network inference framework optimized for mobile. <https://github.com/Tencent/ncnn>, 2024.
- [20] TinyLLaMA Team. Tinyllama: An open 1.1b parameter language model. <https://huggingface.co/TinyLlama>, 2023.
- [21] Hugo Touvron, Thibaut Lavril, Gautier Izacard, et al. Llama 2: Open foundation and fine-tuned chat models. <https://ai.meta.com/llama>, 2023. Meta AI.
- [22] Hugo Touvron, Thibaut Lavril, Gautier Izacard, et al. Llama 2: Open foundation and fine-tuned chat models. <https://ai.meta.com/llama>, 2023.
- [23] Oriol Vinyals and Quoc Le. A neural conversational model. *arXiv preprint arXiv:1506.05869*, 2015.
- [24] Thomas Wolf et al. Hugging face transformers. <https://huggingface.co/transformers>, 2023.
- [25] Yizhe Zhang et al. Dialogpt: Large-scale generative pretraining for conversational response generation. *arXiv preprint arXiv:1911.00536*, 2020.