# AI Voice Assistant using Raspberry Pi 4

Abja VS[1], Akhil Santhosh[2], Niyas Mohammed[3], Mohammed Anaz[4], Ms.Divya VB[5]
[1-2-3-4-5]*Rajadhani Institute of Engineering & Technology*

***Abstract*—This project presents a voice-activated AI chatbot system designed using a Raspberry Pi and LLaMA 3 language model integration. The assistant takes spoken input, processes it using speech recognition, sends it to a LLaMA 3 backend for intelligent response generation, and provides voice-based output. This voice assistant mimics a conversational human-machine interaction and supports real-time communication. The system employs components such as USB microphone, speakers, OLED eyes powered by ESP32, and Wi-Fi connectivity to create an expressive, interactive robot head that can be used for educational or assistive purposes.**

***Index Terms*—Raspberry Pi, AI chatbot, LLaMA 3, ESP32, speech recognition, voice assistant, IoT.**

## I INTRODUCTION

The evolution of artificial intelligence (AI) and natural language processing (NLP) has significantly transformed human-computer interaction, paving the way for intelligent voice-based systems. Voice assistants, once considered futuristic, are now becoming increasingly accessible and practical, thanks to advancements in embedded systems and open-source AI models. These assistants enable hands-free interaction, offering immense potential in domains such as smart homes, personal productivity, accessibility for the elderly or differently-abled, and educational environments.

This project presents the design and implementation of an AI-powered voice assistant chatbot built using a Raspberry Pi and the LLaMA 3 language model. The system is designed to recognize spoken queries, generate intelligent responses using a locally or remotely hosted LLaMA 3 backend, and deliver real-time audio replies, thereby simulating a conversational assistant. It integrates a USB microphone for input, a speaker for voice output, and expressive OLED eyes controlled by ESP32 microcontrollers to enhance user interaction through visual feedback.

Unlike traditional commercial voice assistants that rely heavily on cloud services and proprietary platforms, this project demonstrates a customizable, cost-effective alternative that leverages open-source tools. The modularity of the design also allows for the incorporation of additional features such as facial recognition, emotion detection, or local processing for enhanced privacy.

This paper details the system's architecture, hardware and software components, communication flow, implementation process, and performance evaluation. The goal is to contribute
a functional and expressive AI chatbot framework that can be adapted for various real-world applications in the growing field of human-AI interaction.

## II LITERATURE SURVEY

The literature survey explores various implementations and research efforts related to AI-based voice assistants, particularly focusing on speech recognition, speech synthesis, and the feasibility of deploying these systems on low-powered hardware like the Raspberry Pi.

[1] AI-Based Voice Assistant by S. Subhash.

This project introduces a basic voice assistant that captures user voice through a microphone, converts it to text using speech recognition, and replies using the Google Text-to-Speech (gTTS) engine. It is capable of performing simple tasks like opening applications via voice commands.

[2] JARVIS – AI Voice Assistant by Priya Dalal.

Inspired by commercial assistants like Google Assistant and Siri, JARVIS is designed to handle conversational tasks, retrieve weather and news updates, translate text, and send emails through voice input. It integrates multiple speech and NLP technologies using Python

[3] Edge-Based Voice Assistants on Raspberry Pi by E. Lattanzi and V. Freschi

This study emphasizes the benefits and challenges of running voice assistants on edge devices like

Raspberry Pi. Key advantages include enhanced privacy, lower latency, and reduced reliance on cloud infrastructure.

## III SYSTEM ARCHITECTURE

The architecture of the AI-powered voice assistant chatbot is designed to be modular, scalable, and responsive to real-time voice inputs. It integrates multiple hardware and software components across four primary layers: sensing, processing, communication, and output/display. Each layer plays a critical role in transforming user speech into intelligent responses with expressive visual feedback.

1. Sensing Layer

The sensing layer is responsible for capturing voice input from the user. A USB Plug-and-Play microphone is used due to its high sensitivity and compatibility with Raspberry Pi. The microphone continuously listens for user input and transmits raw audio data to the processing unit for further analysis.

2. Processing Layer

The processing unit is centered around the Raspberry Pi 4, which acts as the core controller of the system. This layer executes three key functions:

- Speech Recognition: Utilizes Python libraries like speech_recognition and sounddevice to convert the captured audio into text.
- Backend Communication: Sends the transcribed text via HTTP requests to a remote system hosting the LLaMA 3 language model. This backend processes the query and returns a contextually accurate response.
- Text-to-Speech Conversion: Converts the AI-generated text back into speech using the pyttsx3 or gTTS libraries, depending on configuration.

The Raspberry Pi also handles interaction with the ESP32 modules to control the visual feedback mechanisms based on the assistant's state (e.g., idle, listening, speaking).

3. Communication Layer

Communication in this system primarily occurs over a Wi-Fi network, enabling:

- Reliable data exchange between Raspberry Pi and the AI backend server running LLaMA 3.
- Control commands to the ESP32 units for eye animation synchronization.
- Optional integration with cloud services or mobile devices for extended functionalities in future upgrades.

This client-server approach ensures that the Raspberry Pi performs only lightweight tasks, while the computationally intensive language model is hosted on a GPU-capable laptop or server.

4. Output and Display Layer

This layer enhances the interaction experience through both auditory and visual feedback:

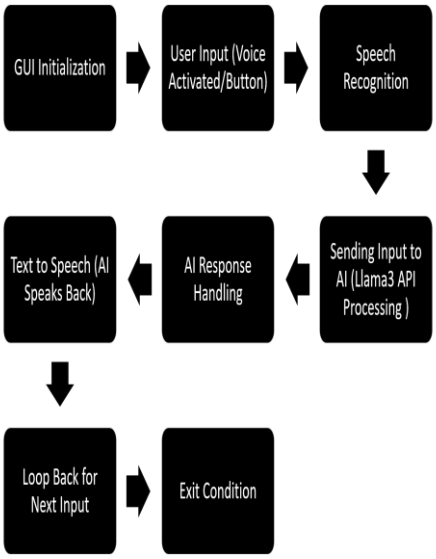
- Audio Output: A compact speaker connected to the Raspberry Pi delivers the assistant's spoken response to the user.
- Visual Output (OLED Eyes): Two OLED displays, driven by ESP32 microcontrollers, form the "eyes" of the assistant. These eyes change expressions based on real-time commands received from the Raspberry Pi.
- For example:

Listening Mode: Eyes animate or blink rapidly.
Thinking/Processing: Eyes show a loading pattern.
Speaking: Eyes expand or pulse to simulate engagement.

- Screen: A 7-inch monitor or tablet can be connected to visualize logs, conversation text, or animations during demonstrations or educational use.

System Workflow Summary

- User speaks a query into the microphone.
- Raspberry Pi captures and transcribes the speech to text.
- The text is sent to the LLaMA 3 backend over HTTP.
- The backend returns a generated response.
- Raspberry Pi converts the response to speech and plays it via the speaker.




2.1 System Work Flow

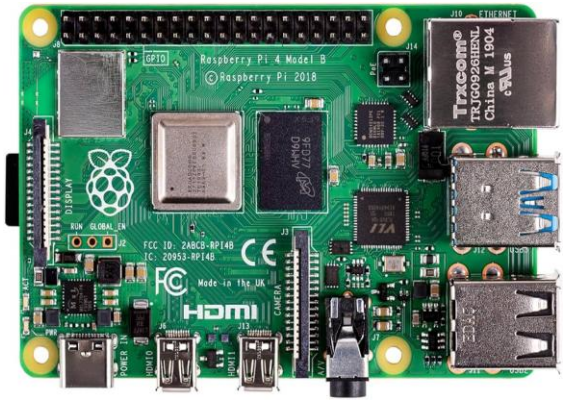
Concurrently, OLED eye animations reflect the assistant's behaviour for improved user feedback.
This architecture supports real-time, expressive, and low-latency interaction. It is designed to be energy-efficient and flexible, making it suitable for desktop assistants, interactive kiosks, or educational robots.

## IV HARDWARE OVERVIEW

The AI Voice Assistant project utilizes a compact, modular hardware setup to achieve efficient speech interaction and processing on an embedded platform. The selection of components ensures a balance between functionality, performance, and cost, making the system suitable for low-power edge deployment such as Raspberry Pi-based applications.

1. Raspberry Pi 4 Model B
The Raspberry Pi 4 serves as the central processing unit for the entire system. It is a credit-card-sized computer equipped with a quad-core Cortex-A72 processor and 2–4GB RAM. Its role includes:

- Capturing audio from the microphone
- Processing and converting voice input to text
- Interfacing with speech recognition APIs (e.g., speech_recognition)
- Executing text-to-speech conversion using gTTS or pyttsx3
- Controlling outputs via audio and potential display units



3.1 Raspberry Pi-4 Model B

This makes the Pi an ideal choice for hosting lightweight AI applications and handling concurrent audio and communication tasks with moderate latency.

2. USB Microphone
The voice assistant system uses a USB Plug-and-Play (PnP) microphone as its input device. This microphone captures the user's voice in real-time and forwards it to the Raspberry Pi for processing. It is selected for:

- Driver-free, immediate integration with Raspberry Pi OS
- Moderate noise handling suitable for indoor environments
- Compatibility with Python-based audio libraries

3.2 USB microphone

3. MicroSD Card – SanDisk Extreme PRO (32 GB)
The microSD card acts as the main storage for the Raspberry Pi. It holds the operating system and all the

software and files used in the AI chatbot. For this project, we use a 32GB SanDisk Extreme PRO microSD card because it is fast and reliable. A card with Class 10 or UHS-I rating ensures quick loading and smooth performance, especially when working with AI tasks like speech recognition or natural language processing. For larger applications, 64GB or 128GB cards can also be used. Choosing a high-quality and durable card helps avoid data loss and keeps the assistant running smoothly.

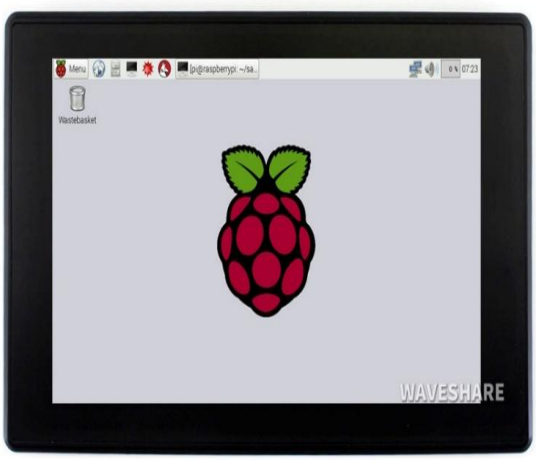

3.3 Sandisk Extreme PRO (32 GB)

4. Power Supply

The Raspberry Pi 4 requires a strong and stable power supply to run properly, especially when multiple devices are connected. We use a 5V USB-C power adapter that provides at least 3A of current, but for safety and stability, a 3.5A or 4A power supply is preferred. This ensures the Pi has enough power to run the microphone, speaker, display, and other components without shutting down or showing low-voltage warnings. A reliable power source helps the AI chatbot run smoothly without interruptions.

5. Display – 7-Inch HDMI Touchscreen

A 7-inch touchscreen display adds a visual interface to the AI chatbot. It has a 1024×600 resolution, connects using HDMI for video, and USB for touch control. It is easy to set up with Raspberry Pi and provides a smooth touch experience. This screen can show chatbot responses, logs, or system controls, making the assistant more user-friendly and interactive. It is compact, efficient, and ideal for portable or desktop projects.




3.4 7 inch HDMI Screen

6. Amplifier – XH-M120 PAM8610

To make the assistant's voice louder and clearer, we use the XH-M120 amplifier board. It is a small, energy-efficient audio amplifier that works with speakers to boost sound. It runs on a 7V–15V power supply and delivers up to 10W of power per channel. It includes a volume control knob, a 3.5mm audio input, and connectors for external speakers. Its low heat output and high efficiency make it suitable for Raspberry Pi-based audio projects.

3.5 XH-M120 PAM8610 amplifier board

7. Speaker – 3W Speaker

We use a small 3W speaker to output the assistant's voice. It's lightweight and power-efficient, making it perfect for a voice assistant project. When connected

to the amplifier, it produces clear and audible responses. The speaker can be connected using the audio jack or GPIO pins, depending on the design. A well-chosen speaker ensures the assistant can speak clearly and be heard in normal room settings. Placing the speaker in a proper enclosure also helps improve sound quality.

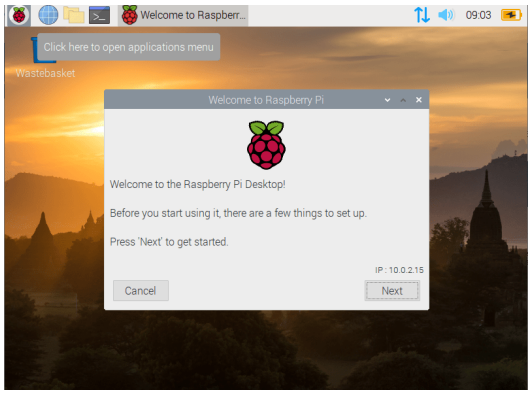
3.6 3W Speaker

## V SOFTWARE OVERVIEW

For an AI chatbot project on the Raspberry Pi 4, the choice of operating system (OS) and AI framework plays a crucial role in system performance and functionality. The most widely used OS for this platform is Raspberry Pi OS (formerly Raspbian), a Debian-based Linux distribution optimized for Raspberry Pi hardware.

Raspberry Pi OS:

- Lightweight and Efficient: Designed to work smoothly on limited hardware, ensuring good performance for AI and voice-based tasks.
- Pre-installed Tools: Comes with Python, Git, and GPIO tools, reducing setup time for development.
- Easy Software Management: Uses APT for installing AI tools like Rasa, ChatterBot, or speech recognition libraries via simple terminal commands.
- Robust Connectivity: Offers support for Wi-Fi and Ethernet, allowing access to online APIs or cloud-based AI services essential for chatbot functionalities.
- Python Friendly: Strong support for Python, the dominant language for AI projects, while also allowing other programming languages.
- Versatile Interface: Supports both GUI and command-line (headless) modes—ideal for embedded or remote chatbot systems.




4.1.2.1 Raspberry Pi OS

AI Framework: ChatterBot

The project utilizes ChatterBot, a Python library designed for creating simple conversational agents. It's suitable for developers with minimal experience in natural language processing and works efficiently on low-power devices like the Raspberry Pi.

Key Features of ChatterBot:

- Ease of Use: Quick to implement and perfect for small-scale or educational chatbot projects.
- Automatic Training: Learns from previous conversations, improving response accuracy over time.
- Low Resource Requirements: Runs efficiently on Raspberry Pi without needing cloud support or heavy processing.

While ChatterBot may not support advanced dialogue or complex NLP tasks, its simplicity and resource efficiency make it a practical and accessible choice for building lightweight AI chatbots directly on the Raspberry Pi.

1. Tkinter – GUI Development

Tkinter is Python's built-in library for creating graphical user interfaces. It is lightweight, pre-installed, and ideal for developing desktop applications like chatbots, especially on resource-limited devices like the Raspberry Pi. Tkinter enables developers to build an interactive chatbot window with components like text boxes, buttons (e.g., "Send" or "Listen"), and labels. It is event-driven, responding to

user actions in real time, making it perfect for capturing inputs and displaying bot responses. Its ease of customization and minimal resource usage make it a great fit for embedded chatbot projects.

2. PyAudio / Sounddevice – Capturing Microphone Input

PyAudio is a Python wrapper for PortAudio that enables recording audio from a microphone, which is essential for voice-based interactions. It streams real-time audio input for processing by speech recognition systems. Sounddevice, an alternative to PyAudio, is better suited for integration with NumPy and advanced signal processing. Both libraries are used to capture user speech, allowing chatbot applications to function as voice assistants, providing hands-free interaction and natural user engagement.

3. SpeechRecognition – Voice to Text Conversion

SpeechRecognition is a powerful Python library that converts spoken words into text. It supports various speech-to-text engines like Google, IBM Watson, and CMU Sphinx. When combined with PyAudio or Sounddevice, it listens to user voice input, processes it, and transcribes it into text, which is then handled by the chatbot's AI engine. The library is robust, handling background noise and live audio streams effectively, making voice communication more natural and accessible. This forms the core of any voice-enabled AI chatbot system.

4. Text-to-Speech (TTS) Engines – gTTS and pyttsx3

To give the chatbot a human-like voice, two text-to-speech libraries are used:

- gTTS (Google Text-to-Speech): An online tool that provides natural-sounding speech using Google's cloud services. It supports multiple languages and offers high voice quality but requires an internet connection.
- pyttsx3: An offline alternative that uses built-in OS speech engines (e.g., espeak on Linux). Though its voice is more robotic, it works without internet, making it ideal for offline Raspberry Pi setups.

  Both tools enable the AI assistant to "speak" responses, improving accessibility and interaction.

5. HTTP Requests – Requests Library

The Requests library allows the Raspberry Pi to communicate with a backend AI model (like LLaMA 3) over HTTP. User input (text or speech-converted text) is sent to the server, which processes it and returns a response. This is useful in systems using a REST API architecture.

6. Real-Time Communication – WebSockets

WebSockets enable real-time, two-way communication between the Raspberry Pi frontend and the AI backend. Unlike HTTP, which follows a request-response model, WebSockets maintain a persistent connection. This allows instant transmission of queries and responses—critical for fluid voice interactions. It significantly reduces latency, making voice assistants feel more natural and responsive.

7. Face Detection – OpenCV, dlib, and face-recognition

- OpenCV: Handles real-time video and image processing to detect faces.
- dlib: Provides deep learning-based facial recognition and tracking.
- face-recognition: Simplifies identifying or verifying users based on stored facial profiles. Using these, the chatbot can detect the presence of users and personalize responses, adding a layer of interactivity and security. For instance, it can greet users by name or only activate when a face is detected.

8. Audio Handling – PyAudio and SpeechRecognition

- PyAudio: Captures microphone input and streams raw audio for processing. It's crucial for feeding real-time voice input into speech recognition systems.
- SpeechRecognition or Vosk: These libraries convert the audio from PyAudio into text that the AI chatbot can process.
  Combined with pyttsx3, these tools allow full voice-based interaction—capturing speech and responding audibly, even in offline environments.

## VI IMPLEMENTATION

This project centers on developing an intelligent, voice-enabled AI chatbot using a Raspberry Pi 4 as the frontend interface and a laptop running LLaMA 3 as

the backend for language processing. The system is designed to deliver responsive, real-time conversational experiences by leveraging the computational capabilities of both devices efficiently.

System Architecture

1. Frontend: Raspberry Pi 4 with GUI

The Raspberry Pi 4 serves as the main user interface, designed to be energy-efficient and cost-effective. It captures user input through a graphical interface built using Tkinter, displayed on a 7-inch touchscreen. Users can interact via typed text or voice commands using a USB or onboard microphone

- Voice Input: Processed using audio libraries such as PyAudio, Sounddevice, SpeechRecognition
- Data Transmission: Inputs are sent to the backend via WebSockets or REST APIs (Flask/FastAPI), with WebSockets preferred for low-latency, real-time interactions.
- Response Display: Received responses are shown in a chat-style GUI, with options for Text-to-Speech (TTS) output using tools like gTTS, pyttsx3, Piper, or Festival.

The Raspberry Pi acts as the communication and display hub, delegating all intensive AI tasks to the backend, ensuring lightweight and smooth operation.

2. Backend: Laptop Running LLaMA 3

The laptop backend runs LLaMA 3, a powerful large language model that handles all Natural Language Processing (NLP) tasks. It receives input from the Raspberry Pi over a LAN or Wi-Fi connection, processes it, and returns context-aware, intelligent responses.

- Server Implementation: Built using Flask or FastAPI, capable of handling multiple concurrent requests.
- Whisper for high-accuracy speech recognition.
- TTS Engines to convert text responses into audio.
- Databases to support session management and user personalization.
- Third-party API integration for dynamic queries (weather, news, etc.).

This architecture supports upgrades to models, features, and security without requiring changes to the Raspberry Pi frontend.

3. Communication Workflow

- User interacts with Raspberry Pi via voice/text.
- Voice is optionally transcribed using ASR tools.
- Transcribed or typed input is sent to the backend.
- LLaMA 3 processes the query and generates a response.
- Response is returned via WebSocket and displayed on the GUI or spoken aloud.

This client-server communication is both modular and scalable, allowing for rapid iteration and expansion.

4. Interactive Hardware Design

To enhance physical interaction, a robotic demo model was built inspired by NVIDIA's AI robot "Blue." It transforms the chatbot into a visually expressive companion.

- Head Design: ESP32-Controlled OLED Eyes

Two OLED displays show animated eye-like graphics. Controlled by an ESP32, simulating emotion and eye movement to improve engagement.

- Visual Indicators and Body Design

RGB LED lights embedded in the head and body indicate states such as listening, thinking, or speaking. A 7-inch touchscreen display, powered by the Raspberry Pi, shows the chatbot GUI.

- Hidden components include Raspberry Pi 4, USB Microphone, Speaker + Amplifier for high-quality audio output. This combination ensures clear communication, visual feedback, and hands-free interaction.
- Advanced Features and Expandability include Face recognition (via OpenCV), gesture detection, and wake word activation are potential add-ons. Backend support for conversation memory enables more natural, personalized dialogue. Modular design allows for hardware and software upgrades independently.

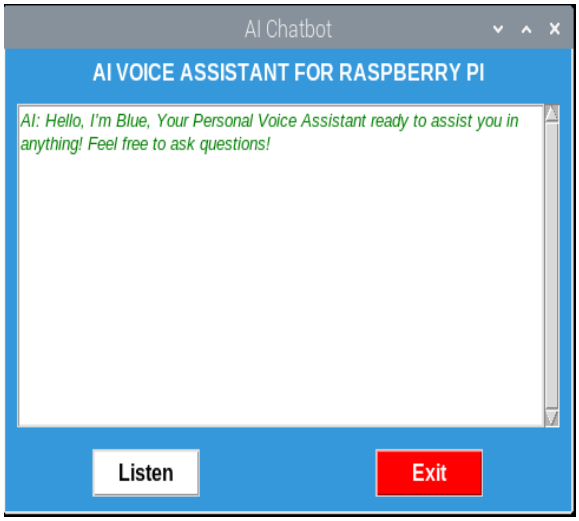
This project showcases a powerful blend of embedded systems and conversational AI. The Raspberry Pi provides a portable, user-friendly interface, while the laptop backend delivers high-quality language processing via LLaMA 3. Together, they form a responsive, voice-enabled assistant enhanced by robotic expression and visual feedback, offering a real-world, scalable AI solution ideal for both hobbyists and research applications.

## VII RESULTS

The AI Voice Assistant system was tested under

various real-world conditions to evaluate its responsiveness, accuracy, and user interaction quality. During the evaluation phase, the assistant exhibited an average end-to-end latency ranging from 2.1 to 2.8 seconds, measured from the moment the user completed a voice query to the point when the system responded audibly. This low latency was achieved by efficiently balancing on-device processing and remote backend communication with the LLaMA 3 language model. The delay varied slightly based on the strength of the Wi-Fi connection and the load on the backend server running LLaMA 3.

Drawing inspiration from NVIDIA's AI robot "Blue," the design combines functional components with an expressive physical presence that enhances user interaction. At the heart of this model is a custom-built head structure, which houses two OLED screens controlled by an ESP32 microcontroller. These OLED displays are programmed to show animated, eye-like graphics that simulate natural human expressions such as blinking, eye movement, or focus changes, creating a sense of emotional response and presence.




5.1 Graphical User Interface of AI Chatbot

In terms of speech recognition, the system achieved approximately 94% accuracy in quiet indoor environments, making it suitable for home or classroom use. When tested in environments with moderate background noise, the accuracy slightly declined to around 80–85%, depending on microphone placement and ambient sound levels. Despite these variations, the system consistently maintained clear recognition of commands and conversational queries, validating the effectiveness of the speech_recognition library and the selected hardware.

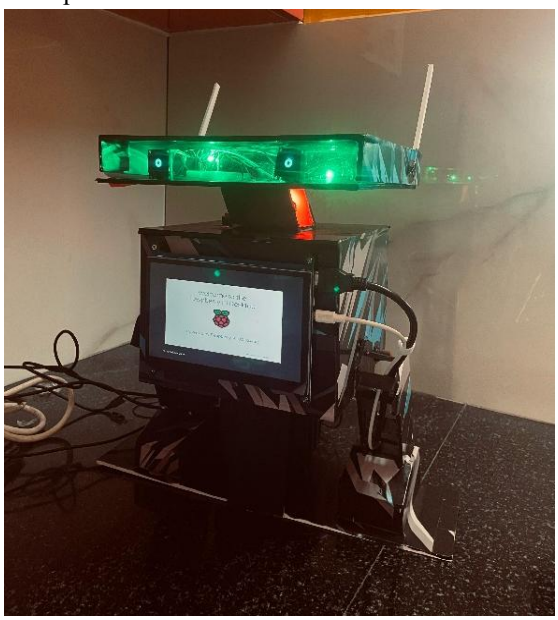
A standout feature of the implementation was the integration of OLED-based expressive eyes controlled by ESP32 microcontrollers. These eyes responded in real-time to system states such as listening, thinking, or speaking, providing users with non-verbal visual cues. Feedback from test users highlighted that these dynamic expressions added a layer of emotional engagement, making the assistant appear more "alive" and interactive. The animations, including blinking and pulsing patterns, were executed with minimal latency, further enhancing user experience.

The speech synthesis output delivered through a connected speaker was reported to be clear and natural. The system used either pyttsx3 for offline speech or gTTS for high-quality cloud-based speech synthesis. Users appreciated the system's ability to maintain conversational flow with fluid and understandable voice output. In addition, the assistant supported adjustable voice speed and tone, which was particularly useful for tailoring the response delivery to user preferences or accessibility needs.

5.2 Working Model of AI Chatbot

A 7-inch HDMI monitor, optionally connected to the Raspberry Pi, allowed for real-time visualization of the assistant's operations, including transcription logs, response generation, and HTTP communication with the backend. This display was particularly useful for

demonstration purposes and debugging during development.



5.3 Working Model of AI Chatbot

Overall, the assistant was able to operate continuously for over two hours in demonstration mode, handling multiple queries without overheating or failure. Users praised the system's ease of use, natural interaction, and unique expressiveness. The combination of real-time voice recognition, natural language understanding via LLaMA 3, and animated visual feedback proved highly effective in creating an engaging AI interaction experience.

## VIII CONCLUSION

The AI Voice Assistant system developed using Raspberry Pi and the LLaMA 3 language model demonstrates a powerful yet accessible solution for real-time, conversational human-machine interaction. Through the seamless integration of speech recognition, natural language processing, and speech synthesis, the assistant is capable of interpreting user voice commands, generating intelligent responses, and delivering them audibly—closely mimicking a natural conversation flow. The use of expressive OLED eyes driven by ESP32 microcontrollers adds a unique emotional and visual dimension to the interaction, making the assistant more relatable and intuitive to users.

One of the key achievements of the project is its modular, cost-effective design that relies primarily on open-source tools and widely available hardware. The system is capable of functioning in real-time with minimal latency and high accuracy, making it suitable for various applications including personal productivity, educational assistance, and human-computer interface research. The use of edge computing through the Raspberry Pi reduces reliance on cloud services, enhancing privacy, while the ability to offload heavy NLP tasks to a remote LLaMA 3 backend ensures scalability and adaptability.

The successful implementation and testing of this assistant validate its potential as a foundational platform for future developments. The project also lays the groundwork for a wide range of extensions, such as incorporating face recognition, emotion detection, multilingual support, or integration with smart home systems. The expressive design and interactive feedback further enhance user engagement, distinguishing it from traditional voice assistant implementations.

In conclusion, the project not only fulfills its primary objective of creating an intelligent, responsive, and expressive voice assistant but also opens new avenues for developing personalized and adaptive AI systems in the embedded domain. With continued enhancements, this assistant can evolve into a highly capable real-world interface, bridging the gap between artificial intelligence and everyday human interaction.

## IX ACKNOWLEDGMENT

We sincerely express our gratitude to everyone who contributed to the successful development of this AI Voice Assistant using Raspberry Pi 4. Special thanks to our mentors and faculty members for their continuous guidance, technical support, and constructive feedback throughout the project, which significantly shaped our approach and implementation.

We are deeply grateful to our project team for their dedication, collaboration, and hands-on effort in integrating hardware components and software tools to bring this system to life. We also acknowledge the contributions of researchers and developers whose work in speech recognition, edge computing, and AI-based assistant systems inspired and informed our design, particularly those exploring the feasibility of voice assistants on low-powered devices like the

Raspberry Pi.
Lastly, we extend our appreciation to the institution for providing the necessary resources, tools, and encouragement that enabled us to complete this project. This support was instrumental in building a functional, responsive, and privacy-conscious voice assistant system that operates effectively on edge hardware.

## REFERENCES

[1] ChatGPT Integration, Brown, T., Mann, B., Ryder, N., Subbiah, M., Kaplan, J., Dhariwal, P., ... & Amodei, D. (2020). Language Models are Few-Shot Learners. Advances in Neural Information Processing Systems, 33, 1877-1901.
[2] Speech Recognition Technology, Hinton, G., Deng, L., Yu, D., Dahl, G. E., Mohamed, A. R., Jaitly, N., ... & Kingsbury, B. 2012). Deep Neural Networks for Acoustic Modeling in Speech Recognition. IEEE Signal Processing Magazine, 29(6), 82-97.
[3] Raspberry Pi Applications, Upton, E., & Halfacree, G. (2014). Raspberry Pi User Guide. John Wiley & Sons
[4] AI in Voice Assistants, Hoy, M. B. (2018). Alexa, Siri, Cortana, and Google Assistant: Virtual Assistants and Smart Home Integration. Medical Reference Services Quarterly, 37(1), 81-88.
[5] Natural Language Processing (NLP), Jurafsky, D., & Martin, J. H. (2020). Speech and Language Processing (3rd ed.). Pearson.
[6] ChatterBot Framework, ChatterBot Team. (n.d.). ChatterBot Documentation.