A MINOR PROJECT REPORT

ON

Multi-Modal AI Voice Assistant with Integrated Home Automation

SUBMITTED IN PARTIAL FULFILLMENT FOR THE AWARD OF DEGREE OF

BACHELOR OF TECHNOLOGY

IN

ELECTRONICS AND COMMUNICATION ENGINEERING

Submitted by:

Arnav Srivastav (9922102030)

Vanisha Agarwal (9922102035)

Archit Jain (9922102006)

Under the Guidance of:

Dr. Vimal Kumar Mishra

DEPARTMENT OF ELECTRONICS AND COMMUNICATION ENGINEERING
JAYPEE INSTITUTE OF INFORMATION TECHNOLOGY, NOIDA (U.P.)

November, 2024

DECLARATION

We hereby declare that this written submission represents our own ideas in our own words and where others' ideas or words have been included, have been adequately cited and referenced the original sources. We also declare that we have adhered to all principles of academic honesty and integrity and have not misrepresented or fabricated or falsified any idea/data/fact/source in our submission.

Place: Jaypee Institute of Information Technology, Sec 128

Date: 21 Nov, 2024

Name: Arnav Srivastav
Enrollment: 9922102030

Name: Vanisha Agarwal
Enrollment: 9922102035

Name: Archit Jain
Enrollment: 9922102006

CERTIFICATE

This is to certify that the minor project report entitled, "**Multi-Modal AI Voice Assistant With Integrated Home Automation**" submitted by **Archit Jain, Arnav Srivastav, Vanisha Agarwal** in partial fulfillment of the requirements for the award of Bachelor of Technology Degree in **Electronics and Communication Engineering** of the Jaypee Institute of Information Technology, Noida is an authentic work carried out by them under my supervision and guidance. The matter embodied in this report is original and has not been submitted for the award of any other degree.

Signature of Supervisor:

Name of the Supervisor: Dr. Vimal Kumar Mishra

ECE Department,

JIIT, Sec-128,

Noida-201304

Dated: 21 Nov, 2024

ABSTRACT

The "Multi-Modal AI Voice Assistant with Integrated Home Automation" presents a seamless integration of artificial intelligence and IoT technologies, aimed at enhancing modern living experiences. This project leverages Python, advanced AI models, and APIs to develop an intelligent voice assistant capable of managing computer functions such as camera control, clipboard operations, and screen interactions, alongside enabling voice-driven home automation.

A custom-designed PCB, incorporating relay modules, forms the foundation of the hardware, ensuring efficient and reliable control of devices. This hardware seamlessly connects an ESP32 microcontroller, which communicates with the software to enable real-time control of smart home appliances. The system employs advanced speech recognition and text-to-speech technologies, enabling natural and human-like interactions. Additionally, APIs provide access to real-time data, including weather updates and news, enhancing its utility as a daily assistant.

With the completion of both hardware and software components, the project achieves a scalable and cost-effective solution, capable of expanding its functionality to include additional devices and features. This integration demonstrates the potential of AI-powered IoT systems in transforming homes into intelligent, connected environments, improving convenience, productivity, and user satisfaction.

The assistant's flexibility extends to its adaptability for different environments, making it equally effective in residential, commercial, and office settings. As a future-proof solution, it paves the way for incorporating emerging technologies such as machine learning-driven predictive analytics and integration with upcoming IoT standards, further solidifying its role as a cornerstone of smart living.

ACKNOWLEDGEMENTS

I would like to express my heartfelt gratitude to everyone who has contributed to the successful completion of this project.

First and foremost, I am deeply thankful to **Dr. Vimal Kumar Mishra**, Department of Electronics and Communication Engineering, **Jaypee Institute of Information Technology**, for their invaluable guidance, constant support, and constructive feedback throughout the project. Their insights and encouragement have been instrumental in shaping this work.

I also extend my gratitude to **Jaypee Institute of Information Technology** and the Department of Electronics and Communication Engineering for providing the necessary resources and a conducive environment for carrying out this project.

This project has been a remarkable learning experience, and I am humbled by the opportunity to apply theoretical knowledge to a practical application.

Archit Jain

Arnav Srivastav

Vanisha Agarwal

Table of Contents

List of Figures

CHAPTER 1

INTRODUCTION

Technology today plays a big role in making our lives easier and more efficient. Artificial Intelligence (AI) and the Internet of Things (IoT) are two areas that have revolutionized how we interact with devices, turning simple tools into smart, responsive systems. This project, "Multi-Modal AI Voice Assistant with Integrated Home Automation," focuses on combining these technologies to create an intelligent assistant that can manage tasks both on a computer and in a smart home.

The aim of this project is to develop a voice assistant that can perform various tasks like controlling appliances, managing desktop functions, and providing updates like weather or news. This system is built to be simple, cost-effective, and flexible, making it useful for homes, offices, and other spaces. By integrating a Python-based software backend with a custom-designed hardware setup using an ESP32 microcontroller, this project demonstrates how AI and IoT can work together to make life more convenient.

1.1 Why Do We Need Smart Systems?

In today's fast-paced world, automation has become essential. Whether it's turning off lights with a voice command or controlling devices from a phone, these systems save time and energy. Voice assistants like Alexa and Google Assistant have shown us how useful hands-free systems can be, but they often come with high costs and limited customization. This project is designed to fill these gaps, offering an affordable and customizable solution for personal and professional needs.

1.2 Key Features of the Project

The voice assistant developed in this project incorporates several key features that make it a versatile and efficient tool. At its core, the system integrates Artificial Intelligence (AI) using Python, enabling it to process voice commands and perform a variety of tasks seamlessly. A major highlight of the system is its home automation capability, allowing users to control devices like lights and fans using simple voice instructions. This functionality is designed to enhance convenience and bring smart living within reach.

The project also includes custom-designed hardware, featuring a specially crafted circuit board that ensures smooth and reliable operations. This custom PCB forms the foundation of the system's hardware, enabling seamless communication between components. Central to this setup is the ESP32 microcontroller, which acts as a bridge between the hardware and software. With its built-in Wi-Fi and Bluetooth capabilities, the ESP32 ensures real-time responses and efficient operation.

To make the user experience even more intuitive, the system employs advanced speech recognition technology, allowing it to accurately interpret voice commands. Complementing this is text-to-speech (TTS) functionality, which provides clear, spoken responses, making interactions feel natural and engaging. Furthermore, the system leverages APIs to fetch real-time updates, such as news and weather, adding to its practicality as a daily assistant. Together, these features create a robust and user-friendly voice assistant that is adaptable to various environments and needs.

1.3 Importance of IoT in Smart Systems

IoT allows devices to communicate and respond automatically, creating a network of connected systems. In this project, the ESP32 microcontroller is key to making this happen. Its Wi-Fi and Bluetooth features ensure all devices can connect and work smoothly. Plus, it's cost-effective and energy-efficient, making it a great choice for long-term use.

With IoT integration, the voice assistant can expand its features in the future to include things like motion sensors, energy monitoring, or even predictive actions based on user habits.

1.4 Making Life Simpler

This project is about more than just technology; it's about making life easier. Whether it's helping elderly users with voice commands, reducing energy consumption by automating lights, or offering a customizable tool for unique needs, this system is built for modern living. It's designed to adapt to different environments and provide solutions that are practical and user-friendly.

In summary, the "Multi-Modal AI Voice Assistant with Integrated Home Automation" combines AI and IoT to bring smarter living to everyday life. By providing a simple, efficient, and affordable solution, it opens up new possibilities for creating connected and intelligent spaces.

CHAPTER 2
LITERATURE REVIEW

The literature survey serves as a foundational step in understanding the existing knowledge, technologies, and methodologies relevant to the project, *"Multi-Modal AI Voice Assistant with Integrated Home Automation."* This survey explores advancements in artificial intelligence, voice recognition, natural language processing, and Internet of Things (IoT) technologies, which collectively form the backbone of this project.

Objectives of the Literature Survey

1. To review state-of-the-art voice assistant technologies and their applications in automation.
2. To explore the integration of AI and IoT for creating intelligent and interactive ecosystems.
3. To understand the role of microcontrollers, such as the ESP32, in achieving cost-effective and scalable automation.
4. To analyze challenges in voice recognition, data security, and hardware-software integration.

2.1 Artificial Intelligence Voice Assistant and Home Automation

S. Singh, S. Panwar, H. Dahiya, and Khushboo, "Artificial intelligence voice assistant and home automation," Int. J. Sci. Res. Arch., vol. 12, pp. 2006-2017, May 2024, doi: 10.30574/ijsra.2024.12.1.0954 [1]

The paper explores the integration of AI voice assistants with home automation systems to create intelligent ecosystems aimed at simplifying daily tasks, enhancing accessibility, and promoting energy efficiency. These systems leverage advanced natural language processing (NLP) to interpret user commands effectively. Machine learning algorithms further augment their adaptability, enabling the assistants to learn user preferences over time. Python serves as a pivotal development tool, offering flexibility and access to libraries designed for AI and IoT applications.

Integration with IoT technologies extends the functionality of these assistants, facilitating seamless voice control of various home appliances.

A review of research from 2014 to 2021 highlights significant advancements in voice assistant technology. Initial studies in 2014 focused on basic voice activation features but encountered issues such as limited adaptability and challenges in public settings. Research between 2018 and 2020 concentrated on enhancing human-machine interaction by incorporating gesture and image recognition, improved speech-to-text systems, and IoT integrations. By 2021, developments had progressed to hybrid systems combining AI, IoT, and advanced data analytics for more robust and versatile functionalities.

Despite these advancements, AI voice assistants face several challenges. Speech recognition often struggles in noisy environments or when interpreting ambiguous commands, which can lead to user dissatisfaction. Privacy and security concerns also persist, as the systems handle sensitive voice data that could be at risk of breaches or misuse. Overcoming these challenges is crucial for broader adoption and increased trust in these technologies.

The findings of this research closely align with the objectives of the project by emphasizing the use of Python for natural language processing and speech recognition, which serve as foundational elements of the software framework. Furthermore, the discussion on hardware control using microcontrollers, such as Arduino, parallels the project's implementation of the ESP32 and custom PCB design.

2.2 AI-Based Desktop Voice Assistant

P. Kunekar, A. Deshmukh, S. Gajalwad, A. Bichare, K. Gunjal and S. Hingade, "AI-based Desktop Voice Assistant," 2023 5th Biennial International Conference on Nascent Technologies in Engineering (ICNTE), Navi Mumbai, India, 2023, pp. 1-4, doi: 10.1109/ICNTE56631.2023.10146699. [2]

The paper discusses the development of a voice assistant utilizing Artificial Intelligence (AI) and Natural Language Processing (NLP) to enable human interaction through natural language, allowing users to perform tasks based on voice commands. The assistant employs advanced voice recognition and NLP capabilities by integrating Google's online speech recognition API, ensuring accurate interpretation of user commands for smooth interactions. Additionally, it features an audio visualization tool that converts user voice inputs into graphical representations, offering a unique and interactive experience. The system also includes text-to-speech (TTS) technology, which converts text into synthesized speech for clear communication. A chatbot module, powered by deep learning and NLP, enhances the assistant's conversational abilities by providing more context-aware responses. Furthermore, the assistant uses APIs to access real-time data, such as localized weather updates through the OpenWeatherMap API and curated news headlines via the NewsAPI, enhancing its utility as a practical tool.

The paper addresses common issues with existing voice assistants, such as inconsistent speech recognition, poor task execution, and limited functionality. The developed assistant seeks to improve these areas by optimizing speech-to-text accuracy and task execution on desktop platforms. Its use of flexible APIs further extends its capabilities, offering a more reliable and versatile user experience.

However, the system is currently limited to desktop applications and lacks IoT integration, preventing it from functioning as a fully comprehensive smart assistant. The paper suggests that future enhancements will incorporate machine learning to make interactions more adaptive and intelligent. Additionally, IoT integration will be added to enable control over nearby devices, creating an ecosystem similar to more advanced voice assistants like Alexa. These upgrades will greatly expand the assistant’s scope and usability in various environments.

The methodology presented in the paper is relevant to the current project, particularly in its modular approach and use of APIs for specific tasks such as weather and news updates. As the project aims to integrate a custom PCB with an ESP32 for hardware automation, the future integration of IoT functionalities aligns well with these goals, supporting the development of a more versatile smart assistant.

2.3 Desktop AI Assistant: J.A.R.V.I.S Just A Rather Very Intelligent System

A. C. Maharajpet, V. S. Jadhav, A. M. Panchamukhi, P. Adagatti, and V. S. Gondkar, "Desktop AI Assistant: J.A.R.V.I.S Just a Rather Very Intelligent System," in

Proceedings of the Journal of Emerging Technologies and Innovative Research (JETIR), vol. 11, no. 3, pp. 582-591, March 2024. [Online]. [3]

The JARVIS project aims to develop a Python-based personal assistant designed to enhance human-computer interaction and automate daily tasks. It supports functionalities such as voice command recognition, task execution, and integration with third-party services, making it a versatile and adaptable tool. The assistant leverages technologies like Natural Language Processing (NLP) and speech recognition to interpret voice commands and perform tasks. Python libraries, including speech_recognition, wikipedia, and Google's Speech Recognition API, play a crucial role in converting speech to text and retrieving information. Advanced AI techniques, such as neural networks and NLP, further refine the assistant's ability to understand user intent and provide contextually relevant responses.

The system uses algorithms like Hidden Markov Models (HMMs) and Convolutional Neural Networks (CNNs) to process audio inputs into text. Once the audio is transcribed, the text is analyzed using tokenization and entity recognition techniques to determine user intent and context. Tasks are executed through Python scripts, system commands, and API calls, ensuring a seamless interaction between the user and the assistant.

Despite these advancements, the system faces challenges in delivering real-time responses and reducing errors during task execution. Maintaining conversational context across multiple interactions is also difficult, as it requires advanced dialog management systems that are computationally intensive and challenging to implement on resource-limited devices.

The system architecture, which includes modules for speech recognition, natural language understanding, and task execution, aligns closely with the objectives of the current project. Its flexibility and the use of Python make it particularly suitable for integrating hardware components like the ESP32, providing a robust framework for further development.

2.4 IDAS: Intelligent Driving Assistance System Using RAG

L. -B. Hernandez-Salinas et al., "IDAS: Intelligent Driving Assistance System Using RAG," in IEEE Open Journal of Vehicular Technology, vol. 5, pp. 1139-1165, 2024, doi: 10.1109/OJVT.2024.3447449. [4]

The paper focuses on the development of an AI-driven Intelligent Driving Assistance System (IDAS) designed to provide real-time, context-aware support to drivers. Its primary goal is to enable efficient, multilingual voice command interactions for accessing vehicle-specific information from user manuals, enhancing driver safety, convenience, and overall user experience. Unlike traditional systems that rely on basic AI functionalities like object detection and predefined responses, IDAS employs dynamic and contextually accurate interactions using vector databases and multimodal integration. This approach significantly improves accessibility by offering real-time, tailored responses to diverse scenarios.

The system combines information retrieval with generative AI to deliver accurate responses. OpenAI Whisper ensures precise real-time transcription of voice commands, while GPT-based models process these inputs to generate relevant outputs. Chroma is utilized to manage and retrieve information from car manuals, ensuring efficient data handling. The system also incorporates multimodal communication through voice commands, text-to-speech capabilities (via ElevenLabs), and touch inputs, creating a user-friendly interface.

However, IDAS faces challenges related to the variability in the accuracy and comprehensiveness of vehicle manuals across different manufacturers and models. Additionally, the proprietary AI models used in the system require significant computational resources, posing difficulties for deployment in low-power environments like vehicles. These limitations emphasize the need for optimization to enhance real-world usability.

The methodologies employed in IDAS are relevant to the current project, particularly its use of lightweight hardware and optimized models for edge devices, which align with the integration of the ESP32 and a custom PCB. The modular RAG pipelines and vector databases offer a scalable architecture that can inspire the design of a voice assistant capable of handling hardware-specific tasks efficiently.

2.5 A Closer Look at Access Control in Multi-User Voice Systems

H. A. Shafei and C. C. Tan, "A Closer Look at Access Control in Multi-User Voice Systems," in IEEE Access, vol. 12, pp. 40933-40946, 2024, doi: 10.1109/ACCESS.2024.3379141. [5]

This study examines the challenges of privacy and access control in multi-user voice systems, particularly in smart home assistants like Amazon Alexa. Shared environments require robust mechanisms to prevent unauthorized access while maintaining ease of use. Multi-user voice systems face challenges due to shared devices, where multiple individuals may access the system, increasing the risk of unauthorized commands and exposure of sensitive data. Current solutions, such as PINs or voice authentication, attempt to mitigate these risks but often suffer from usability issues or errors, especially in real-world scenarios. Many systems rely on coarse-grained access controls, offering either full access or no access, with voice profiles providing only minimal differentiation between users. This can result in unauthorized actions, such as accessing purchase histories or making unapproved transactions, which compromise user privacy.

To address these vulnerabilities, the study introduces the TVOS Framework, a tool designed to test and identify gaps in user-specific protections within voice systems. The framework focuses on improving security by exposing areas where existing systems fail to differentiate users effectively. Particular attention is given to shopping applications, as features like cart management and purchase history are especially vulnerable to weak access protections, potentially leading to significant privacy breaches.

The study's insights are directly relevant to the development of the AI voice assistant, which prioritizes security by implementing granular access controls for hardware operations and data retrieval. It also addresses the complexities of multi-user environments by focusing on reliable user differentiation, particularly for executing sensitive commands. To ensure robust performance, the project incorporates plans to adapt a testing framework like TVOS to stress-test access controls and evaluate user interactions under diverse conditions, thereby enhancing security and reliability in shared-use scenarios.

2.6 Voice Controlled Smart Home

Y. Neelima, K. Prajwal, M. Asma, and R. Joseph, "Voice controlled smart home," *Industrial Engineering Journal*, vol. 53, no. 4, pp. 844–850, Apr. 2024. [6]

The project aims to provide a cost-effective, energy-efficient, and user-friendly home automation solution, addressing the needs of individuals with mobility challenges, including the elderly and disabled. It also enables remote monitoring and control of devices through Wi-Fi connectivity and cloud integration. Key technologies in smart home systems include versatile microcontrollers like the Arduino Nano and NodeMCU, which support efficient device control and connectivity. Integration with voice assistants such as Amazon Alexa and Google Assistant enhances usability by enabling seamless voice-based interactions. However, challenges such as the impact of background noise on voice recognition accuracy and security concerns related to IoT device vulnerabilities require attention, with robust encryption needed to protect user data and privacy.

The system uses the ESP32 microcontroller as its central processing unit, offering Wi-Fi connectivity for real-time control and data transmission. Physical components, such as relays, are integrated to manage the ON/OFF states of connected appliances effectively. Remote control is facilitated through mobile applications, providing a user-friendly interface. Programming and deployment are carried out using the Arduino IDE, ensuring efficient and reliable system performance.

Challenges include the economic feasibility of scaling the system for larger or more complex setups, which may increase costs, and user adoption, as effective operation requires a basic understanding of the technology.

The use of ESP32 aligns closely with the custom PCB design in the project for hardware automation, while energy-efficient relays ensure effective appliance control. Research on integrating NLP and voice recognition enhances command accuracy in AI-based assistants. Incorporating Wi-Fi modules and cloud services for remote control mirrors modern AI system advancements, while the emphasis on accessibility for individuals with special needs brings valuable societal impact to the project.

2.7 LLM-Powered Conversational Voice Assistants: Interaction Patterns, Opportunities, Challenges, and Design Guidelines [7]

A. Mahmood, J. Wang, B. Yao, D. Wang, and C.-M. Huang, "LLM-Powered Conversational Voice Assistants: Interaction Patterns, Opportunities, Challenges, and Design Guidelines," preprint, arXiv:2309.13879, Sep. 2023.

The research focuses on integrating large language models (LLMs) like ChatGPT into voice assistants (VAs) to enhance interaction patterns and capabilities. It addresses the limitations of traditional keyword-based systems, examines user interactions in scenarios such as medical self-diagnosis and creative planning, and improves conversational resilience by reducing intent recognition errors. LLM-powered VAs provides significant advancements in natural language understanding, offering fluid multi-turn interactions with better context retention compared to traditional systems. Applications span various domains, including self-diagnosis, creative planning, and opinion-based discussions, showcasing the versatility of these models. Additionally, LLMs manage interaction patterns effectively, handling conversational breakdowns through recovery strategies like clarifications and rephrasing to ensure a smoother user experience.
The system employs ChatGPT (GPT-3.5-turbo) as its foundation, leveraging a transformer architecture and extensive datasets to produce coherent and context-aware responses. It integrates LLMs with traditional VA platforms, such as Alexa Skills, using API-based mechanisms for efficient speech-to-text conversion and response generation. Features like small talk and filler responses are included to maintain conversational flow during processing delays, enhancing user experience.

Challenges include response delays for complex queries processed by LLMs and occasional inconsistencies or contradictions in generated responses. Repetitive or overly detailed replies can also affect user engagement, leading to potential mismatches between user expectations and system capabilities.

The project aligns with these advancements by incorporating the ESP32 microcontroller for automated hardware control, using an API-based structure to ensure seamless interaction between hardware and software in custom PCB designs. AI assistant capabilities are enhanced with LLMs for precise voice recognition and command processing. Error recovery mechanisms, including clarifications initiated by both the user and the assistant, improve conversational depth and user satisfaction. Furthermore, the project emphasizes accessibility and practical applications, addressing diverse needs in scenarios like medical diagnosis and trip planning, making the system adaptable to various user requirements.

2.8 Voice Assistant Using Python

N. Singh, D. Yagyasen, S. V. Singh, G. Kumar, and H. Agrawal, "Voice Assistant Using Python," *International Journal of Innovative Research in Technology*, vol. 8, no. 2, pp. 419-423, Jul. 2021. [8]

The research focuses on developing a Python-based desktop voice assistant designed to minimize reliance on traditional input devices like keyboards and mice. It enables users to perform tasks such as email access, web searches, and application execution using voice commands, emphasizing accessibility for the elderly and visually impaired. While its current scope is limited to basic desktop functionalities, the study envisions future enhancements, including IoT integration, to expand its utility.

Python's rich ecosystem of libraries and modules supports the development of advanced features for speech-to-text processing, command execution, and seamless system interaction. Speech Recognition APIs facilitate the conversion of voice inputs into actionable commands, allowing users to control applications efficiently. Machine learning techniques, such as supervised and unsupervised learning, improve adaptability and interaction capabilities. Future integration of IoT and AI introduces potential for controlling nearby devices, paving the way for a more versatile and comprehensive system.

However, the system faces limitations, such as restricted functionality focused on basic tasks like application execution and web searches. Speech accuracy is affected by factors like noise, accents, and speech variations, requiring robust algorithms and high-quality training data. Scalability remains a challenge, as the current implementation is desktop-based and lacks IoT integration for broader applications.

The project aligns closely with these principles by incorporating voice-driven automation to manage hardware tasks through the ESP32 microcontroller. Future development emphasizes custom PCB design to achieve seamless IoT integration with dedicated hardware. The focus on accessibility matches the objective of catering to diverse user needs through intuitive and inclusive interactions. By integrating AI technologies, including large language models (LLMs) and machine learning, the project enhances conversational accuracy and adaptability, making the system more efficient and user-friendly.

2.9 Internet of Things (IoT) Based Home Automation Using ESP-32

Mrs. K. Anusha, S. Aravind, N. Umamaheshwar Rao, and T. Ravisagar, "Internet of Things (IoT) Based Home Automation Using ESP-32," Int. J. Res. Appl. Sci. Eng. Technol., vol. 12, pp. 1672-1676, May 2024, doi: 10.22214/ijraset.2024.61912. [9]

The paper "Internet of Things (IoT) Based Home Automation Using ESP-32" examines the use of the ESP32 microcontroller in building efficient and secure home automation systems. It highlights features such as real-time monitoring, remote control through mobile apps, and energy optimization by integrating sensors and actuators to manage household devices seamlessly.

The system uses the ESP32 microcontroller as the central control hub, taking advantage of its built-in Wi-Fi capabilities for smooth communication. It integrates various sensors, such as motion, temperature, and humidity sensors, to collect real-time data. Actuators, like relays and servo motors, are used to control appliances and devices, enabling versatile automation. The system offers remote access via user-friendly apps like Blynk, allowing users to monitor and manage the system easily. Additional functionalities such as task scheduling and predefined automation routines improve convenience. Energy-efficient operations and strong security measures are key priorities, ensuring reliable performance and protecting user data.

Despite its strong features, the system faces challenges related to scalability when applied to larger or more complex environments. Security vulnerabilities in IoT devices and communication protocols are significant concerns, requiring regular updates and monitoring. Balancing advanced features with cost-effectiveness and energy efficiency remains an ongoing challenge. Additionally, ensuring consistent real-time performance across varying conditions complicates development. Accessibility for users with limited technical expertise also needs attention, requiring intuitive interfaces and robust user support.

This research aligns with our project by offering valuable insights into applying ESP32 for IoT automation, directly relating to the hardware framework of our AI voice assistant. The focus on sensor-actuator integration, energy optimization, and user-friendly design is essential for enabling seamless control of household devices. Moreover, addressing scalability and security concerns will be crucial for ensuring the reliability and usability of our system.

2.10 IoT-Based Home Automation Using Personal Assistant

V. Chayapathy, "IoT Based Home Automation by Using Personal Assistant," June 12, 2018 [10]

This paper focuses on developing a cost-effective personal assistant for home automation using IoT, implemented on a Raspberry Pi platform. The assistant enables interaction with household appliances through voice and gesture commands, providing a versatile and affordable solution for smart home environments.

The system is built on a core IoT framework that connects smart devices and sensors to gather data and execute commands. At its core, the Raspberry Pi serves as the central processing unit, interfacing with sensors and managing tasks based on voice and gesture inputs. Google's Speech Recognition API is employed for speech recognition, converting spoken commands into actionable text, ensuring accurate and efficient communication. The system further incorporates smart controls, including automated lighting that responds to room occupancy and smart alarms synchronized with email notifications, combining convenience and efficiency.

The paper critiques existing home automation solutions like Amazon Echo, highlighting their high costs and limitations in unstructured scenarios, such as when invalid commands are given. Additionally, the issue of compatibility and expense when integrating branded smart appliances is addressed.

The system faces challenges in gesture recognition, particularly in environments with clutter where distinguishing between the background and foreground can be difficult. The speech recognition component also needs improvement to accommodate a broader range of accents and ensure clearer detection of user inputs. Future improvements are planned to develop advanced gesture recognition algorithms for better performance in dynamic environments and to refine speech recognition for more inclusive and accurate interactions. These advancements aim to enhance the system's reliability and adaptability for a variety of use cases.

This paper’s insights into integrating IoT with affordable, modular design align closely with the objectives of this project. The use of ESP32 with custom PCBs for hardware automation directly reflects the principles discussed, and the strategies for cost-effective implementation and modular design via APIs can be applied to optimize the architecture of the AI voice assistant being developed.

CHAPTER 3

SYSTEM DESIGN AND ARCHITECTURE

Fig 3.1 shows the main project structure of our AI voice assistant. It starts with Voice Input, where the user speaks into a microphone. This audio input is then passed to the Voice Recognition module, which transcribes the speech into text. The text is sent to the Python Backend, the core of the system, which analyzes the command and decides on the next steps. Depending on the request, the backend can make API Calls to external services (like fetching weather data), execute a System Call to control local applications, or interact with Automation Devices for tasks like turning on lights.

Finally, the Python backend generates a text response that is sent to the Text-to-Speech (TTS) module, converting it back into spoken audio. The output is delivered to the user as audible feedback, completing the interaction cycle. This integrated approach allows for seamless user interaction with both digital services and physical automation devices.

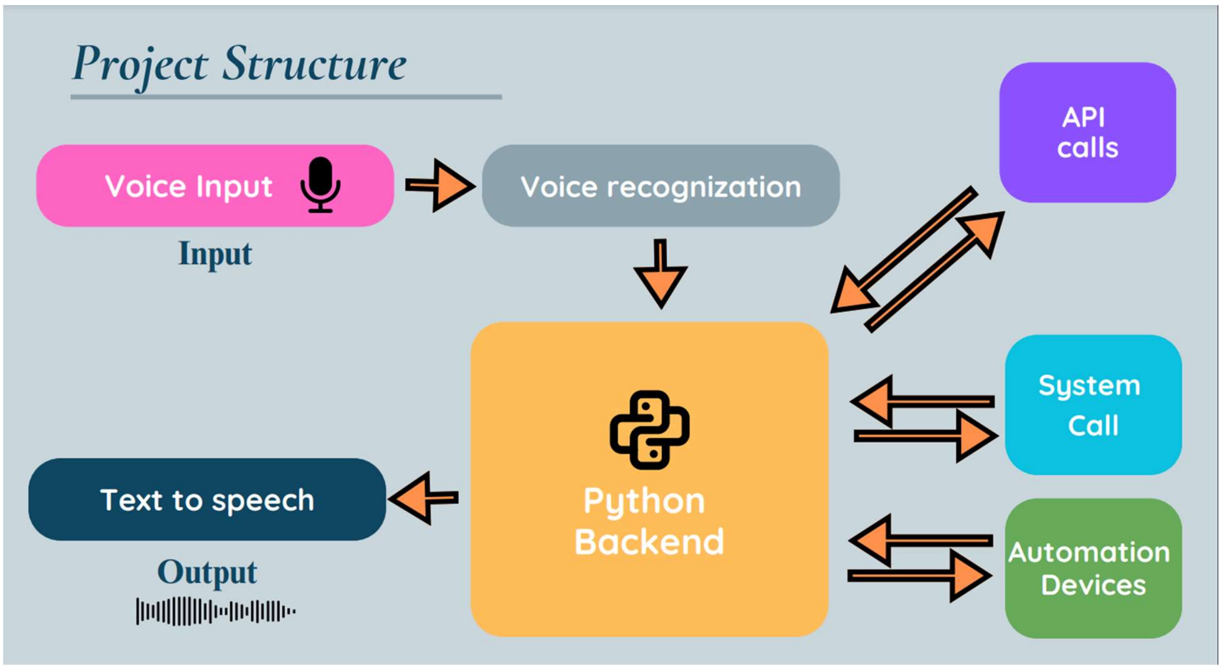


Fig. 3.1: Block Diagram of the System

3.1 HARDWARE DESIGN

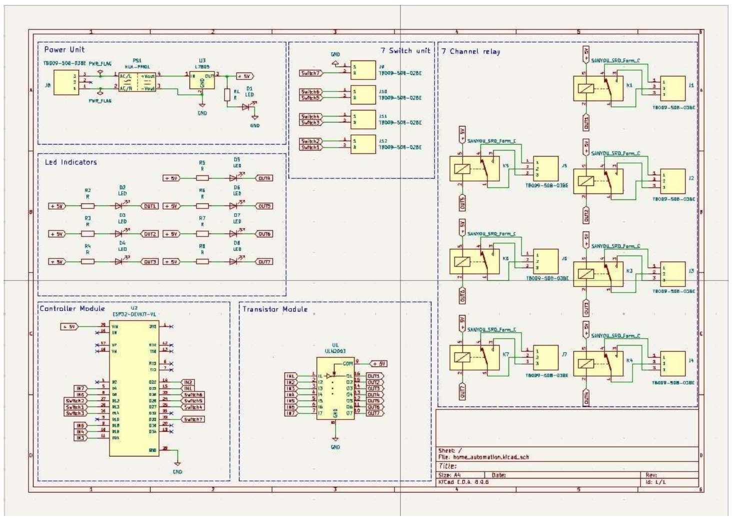


Fig. 3.2: Schematic Capture of the Circuit Diagram

Fig 3.2 illustrates a Home Automation System. The **Power Unit** generates a regulated 5V supply for the circuit. **LED Indicators** visually show the status of outputs. The **ESP32 Controller Module** interfaces with switches and controls the relays via the **Transistor Module (ULN2003)**, which boosts signals to drive the relays. The **7-Channel Relay Module** controls high-power devices like appliances, with relays toggled by the ESP32 through the ULN2003. Each block is interconnected to ensure seamless operation of the automation system.

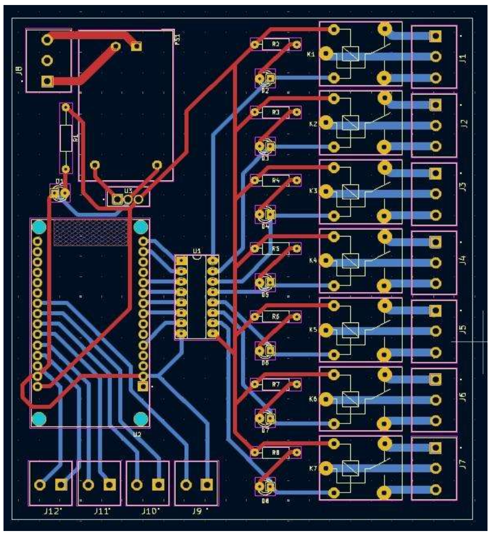


Fig. 3.3: PCB Layout

Fig 3.3 or the PCB layout represents a home automation system integrating a 7-channel relay module, an ESP32 microcontroller, and associated components. The left section houses the **ESP32 microcontroller socket** and power supply circuitry, providing regulated 5V power. The **ULN2003 transistor array** is centrally positioned to drive the relays effectively.

The right side contains the **relay module footprints** (K1 to K7) connected to output terminals for high-power loads. Traces are routed systematically: **red traces represent the top layer** (likely for signal lines and power distribution), and **blue traces represent the bottom layer** for return paths (ground and other connections). The design ensures minimal interference and clear separation between high-power and low-power circuits.

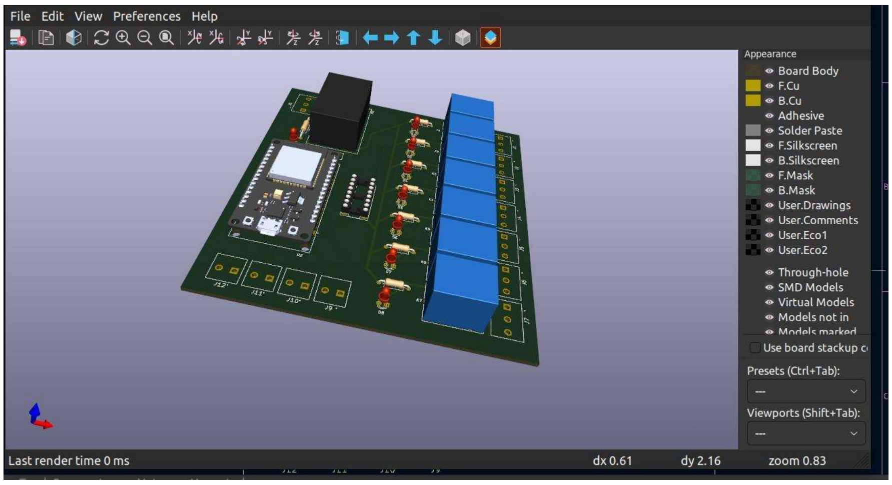


Fig. 3.4: 3D View of the PCB Design

Fig 3.4 is a 3D render of a PCB for a home automation system. It includes an ESP32 microcontroller, a ULN2003 driver IC, 7 relay modules (blue components) for controlling high-power devices, and LEDs for status indication.

3.1.1 Overview of the ESP32 Microcontroller

The ESP32 is a low-cost, low-power microcontroller with integrated Wi-Fi and Bluetooth capabilities, making it an ideal choice for IoT and smart home automation applications. It features a dual-core processor, extensive GPIO pins, and support for multiple peripherals, allowing flexible hardware integrations. Its compatibility with relay modules and various sensors enables seamless control of devices like lights, fans, and appliances.

3.1.2 Why ESP32 is Best Suited for our Project

The ESP32 (Fig 3.5) stands out as an excellent choice for home automation projects due to its built-in Wi-Fi and Bluetooth capabilities. These features enable seamless remote control and communication with smart home devices over a network, allowing for real-time execution of commands. This connectivity ensures that the system can function efficiently, even in dynamic environments.

Another significant advantage of the ESP32 is its cost-effectiveness. It offers a range of advanced features at a fraction of the cost of alternatives like the Raspberry Pi. This makes it an optimal choice for projects where budget constraints are a consideration, especially in systems requiring scalability.

The ESP32 also excels in energy efficiency, consuming minimal power during operation. This is particularly beneficial for home automation systems that need to run continuously, ensuring that the setup remains sustainable over the long term.

Moreover, the ESP32 offers excellent integration capabilities, being compatible with Python-based AI systems and APIs. This ensures smooth communication between the software and hardware components of the project, enhancing overall system functionality.

Lastly, the versatility and scalability of the ESP32 make it ideal for expanding automation tasks. Its ability to support multiple devices and functionalities allows for future upgrades and additional integrations, ensuring that the system can grow alongside evolving project requirements.

Fig. 3.5: ESP 32 microcontroller board

3.1.3 Power Supply Design and Other Components

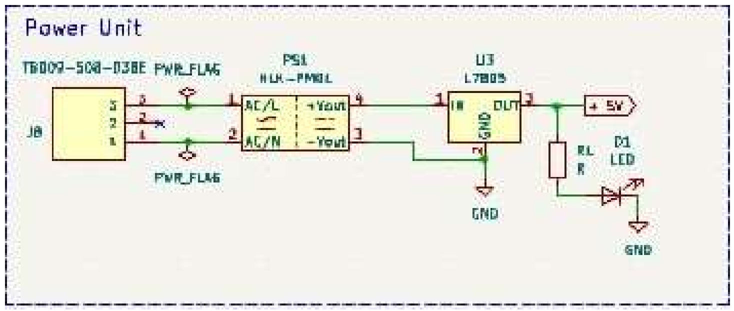


Fig. 3.6: Power unit

The Power Unit (shown in fig 3.6) serves as the foundation of the system by providing a regulated 5V power supply essential for the circuit's operation. It ensures all components receive stable and reliable power, which is crucial for seamless functionality. Complementing the system's functionality are the LED Indicators, which act as visual cues to display the status of various outputs, offering users an intuitive way to monitor system activity.

At the core of the system is the Controller Module, represented by the ESP32, a powerful microcontroller that handles the automation and communication tasks. It is supported by a Switch Unit, which enables manual control over the connected devices, providing flexibility alongside automation. For controlling high-power devices like lights or appliances, the system incorporates a 7-Channel Relay Module, a critical component. This module consists of individual relays (K1 to K7) that function as electrical switches. Each relay features input terminals (IN1 to IN7) to receive toggle signals from the ESP32 and output terminals that connect to the load, such as lights or fans.

To drive these relays effectively, the system uses a Transistor Module built around the ULN2003 Darlington transistor array. This module boosts the low-power signals from the ESP32 to levels suitable for activating the relays. The ULN2003 can handle multiple relays simultaneously, with its input pins (IN1 to IN7) receiving control signals. Together, these components form a cohesive and efficient system designed for automation and manual control of high-power electrical devices.

3.1.4 Physical Components Overview

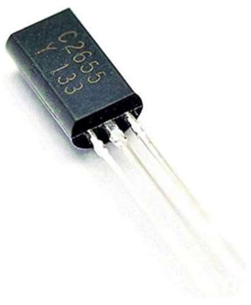


Fig. 3.7: Transistor C2655

2C2655 (Fig 3.7) is a TO-92L or TO-92MOD packaging BJT transistor with many intriguing features in a tiny package that makes it perfect for use in a wide range of applications. It is an NPN transistor with a maximum collector current of 2A, allowing it to drive a wide range of electrical components that require 2A current. In our project, we are using this as a switch in our hardware.

Fig. 3.8: AMS1117 5V Voltage Regulator

AMS1117-5.0V SOT-223 (Fig 3.8) Voltage Regulator IC is a series of low dropout three -terminal regulators with a dropout of 5.0V at 1A load current. AMS 1117 features a very low standby current of 2mA compared to 5mA of a competitor. Other than a fixed version, Vout = 1.2V, 1.5V, 1.8V, 2.5V, 3.3V, 5V, and 12V, AMS1117 has an adjustable version, which can provide an output voltage from 1.25 to 12V with only two external resistors.

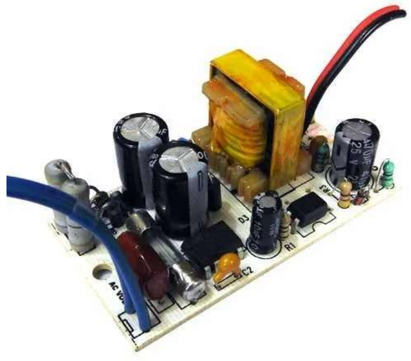


Fig 3.9: 12V 1Amp Adapter

12 Volt 1 Amp Power Adapter (shown in Fig 3.9) takes an AC INPUT of 100-240V and gives 12V 1A DC output. 12 Volt DC 1 Amp power supply is suitable for powering a wide range of applications including CCTV cameras, wireless routers, Robotics and DIY kits.

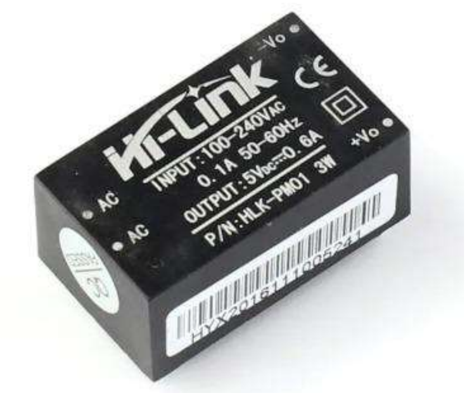


Fig 3.10: Hi-link 5V

HLK-PM01 5V/3W Switch Power Supply Module shown in Fig 3.10 is a plastic enclosed PCB mounted isolated switching step-down power supply module. It can supply 5V DC from 120V AC – 230V AC and has a power rating of 3 Watt. This makes it perfect for small projects that need a 5 volt supply from mains. There are many advantages for these modules, such as low-temperature rise, low power, high efficiency, high reliability, high-security isolation etc.

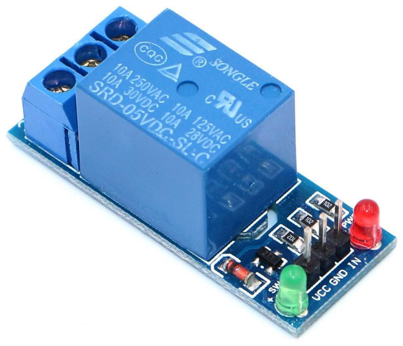


Fig 3.11: Relay 5V

The 1-channel relay module (Fig 3.11) features one normally closed (NC) contact and one normally open (NO) contact, making it versatile for controlling various devices and appliances. It is designed with a high-impedance controller pin and defaults to a high-level trigger for easy operation. The module includes a pull-down circuit to prevent malfunctions and is equipped with a power supply indicator lamp for status monitoring. It is capable of controlling appliances and other equipment with large current loads, using standard TTL level logic for seamless integration with microcontrollers. Additionally, the module has four fixed screw holes with a diameter of 3.1mm, ensuring convenient installation and secure fixation.

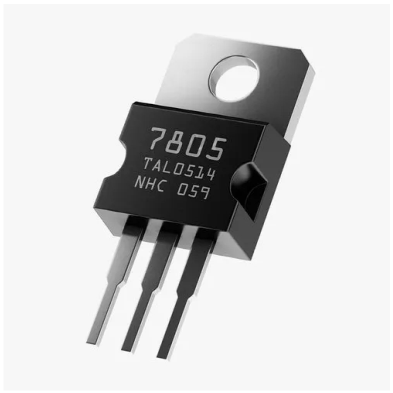


Fig. 3.12: L7805CV Voltage Regulator

The L7805CV regulator (shown in Fig 3.12) can provide local on-card regulation, eliminating the distribution problems associated with single point regulation. It employs internal current limiting, thermal shutdown and safe area protection, making it essentially indestructible. If adequate heat sinking is provided, it can deliver over 1A output current. Although designed primarily as a fixed voltage regulator, this device can be used with external components to obtain adjustable voltages and currents.

3.2 SOFTWARE DEVELOPMENT

3.2.1 Programming Softwares and Tools Used

- **VS Code (Visual Studio Code):** Used for developing and debugging Python code for the voice assistant and AI integrations, providing essential tools for writing and managing the complex functionalities of the project.
- **Arduino IDE:** Programmed the ESP32 microcontroller for home automation tasks, allowing for control of smart devices and integration with the voice assistant's system.
- **KiCad:** Designed, simulated and printed electronic circuits to ensure proper functionality and integration with the ESP32 before physical assembly, optimizing the hardware setup for the project.

3.2.2 Main Python Program Workflow

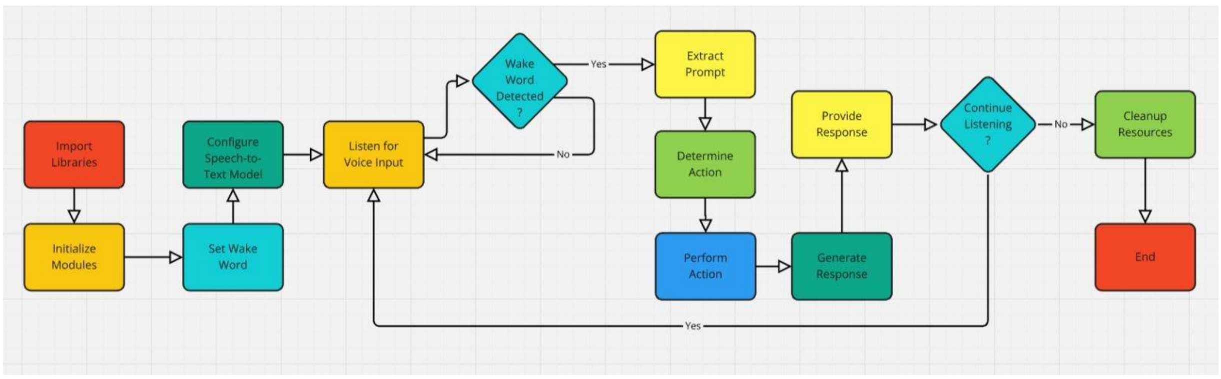


Fig. 3.13: Main code Flowchart

The flowchart as shown in Fig 3.13, illustrates the workflow of a voice-controlled assistant, showing how it processes user commands step by step. The process starts by setting up the necessary environment. Libraries and modules required for the assistant’s functionality are imported and initialized. This is followed by configuring a speech-to-text model that converts spoken words into text. At this stage, a "wake word" is also set—this is a specific word or phrase, such as "Hey Assistant," that signals the system to start listening actively for commands.

Once set up, the system begins listening for voice input. It continuously monitors for the wake word while ignoring other sounds or conversations. When the wake word is detected, the assistant moves into action. If the wake word is not detected, the system remains in its listening mode, waiting for the activation signal.

After activation, the system processes the user’s spoken input. It extracts the prompt (the content of the user’s command) and analyzes it to determine the appropriate action. This could involve retrieving information, performing a task like setting a reminder, or interacting with a connected device. Once the action is completed, the system generates a response, such as providing feedback or confirming that the task is done, and delivers it to the user.

Following the response, the system checks whether it should continue listening for additional commands. If the user wants to continue, the process loops back to the listening stage. Otherwise, the system ends the session by cleaning up resources and shutting down its operations. This structured workflow ensures smooth and efficient interaction between the user and the voice assistant.

3.2.3 ESP32 Program Workflow

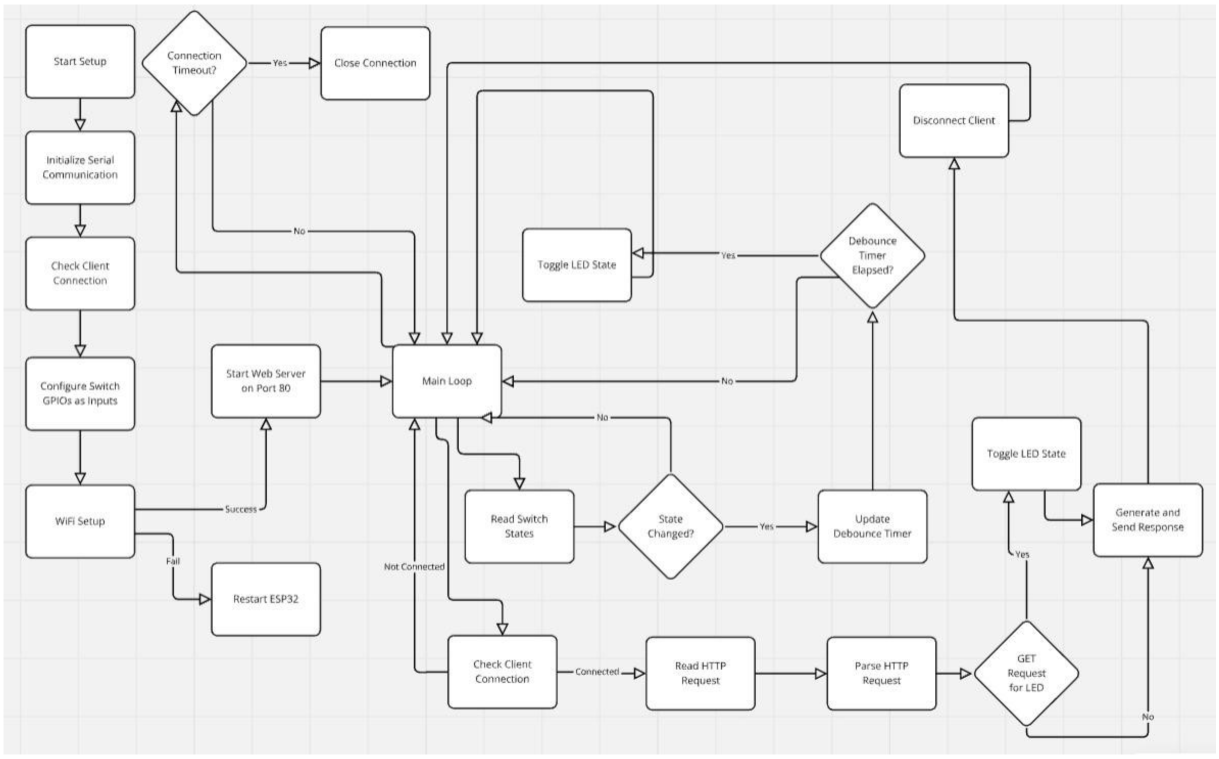


Fig. 3.14: Embedded C program flowchart

The flowchart shown in Fig 3.14 outlines the workflow of a system, likely based on an ESP32 microcontroller, managing LED control via a web server. Here's a simplified explanation:

The process starts with system setup, including initializing serial communication and configuring GPIO pins as inputs for switches. The system then attempts to connect to Wi-Fi. If the connection fails, the ESP32 is restarted to retry the setup process. Once the Wi-Fi connection is established, a web server is started on port 80, ready to handle client requests.

The main loop checks if a client is connected to the web server. If no client is connected, the system keeps monitoring. When a connection is established, it reads HTTP requests sent by the client. If the request involves toggling an LED (e.g., a specific GET request), the system updates the LED state and generates a response for the client. Additionally, the system checks the state of physical switches and uses a debounce timer to ensure accurate state changes, preventing false triggers. If a switch state change is detected, it updates the debounce timer and toggles the LED accordingly.

The workflow includes connection timeout handling; if a client remains inactive for too long, the connection is closed. Throughout the process, the system continuously monitors both the client connection and switch inputs, ensuring reliable operation and responsiveness.

CHAPTER 4

RESULT AND DISCUSSION

The development and integration of the "Multi-Modal AI Voice Assistant with Integrated Home Automation" yielded significant results, demonstrating the project's practical viability and performance efficiency. The outcomes are presented and analyzed in the following paragraphs, emphasizing both the hardware and software aspects.

4.1 System Performance

The voice assistant successfully integrated Python-based AI functionalities with IoT hardware for seamless operation. There is reliable interaction with the system, even in varied environments, though minor performance drops were observed in noisy conditions. The text-to-speech feature delivered clear responses, enhancing user engagement.

4.2 Hardware Functionality

The custom-designed PCB performed as expected, providing stable and efficient hardware control. The ESP32 microcontroller, serving as the system's core, exhibited robust connectivity and efficient task execution. The 7-channel relay module operated seamlessly, allowing precise control of high-power devices. The power supply unit ensured consistent voltage levels, safeguarding the system's reliability during extended usage.

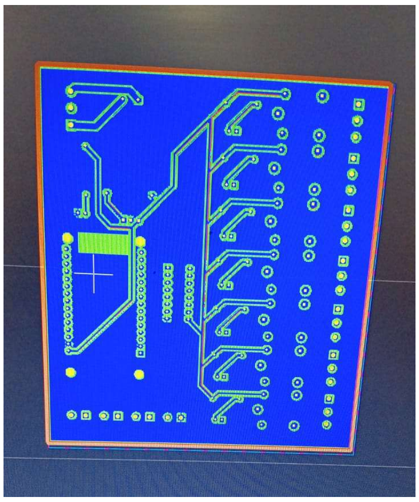


Fig. 4.1: Top View of PCB

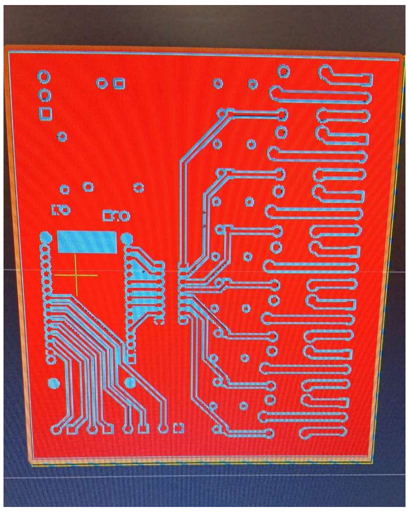


Fig. 4.2: Back View of PCB

4.3 Home Automation Integration

The home automation component showcased the project's potential to simplify daily tasks. Users could control devices such as lights and fans through voice commands, achieving instant responses. The integration of real-time API-driven functionalities, like weather updates and news retrieval, further enhanced the assistant's utility, making it a comprehensive tool for modern living.

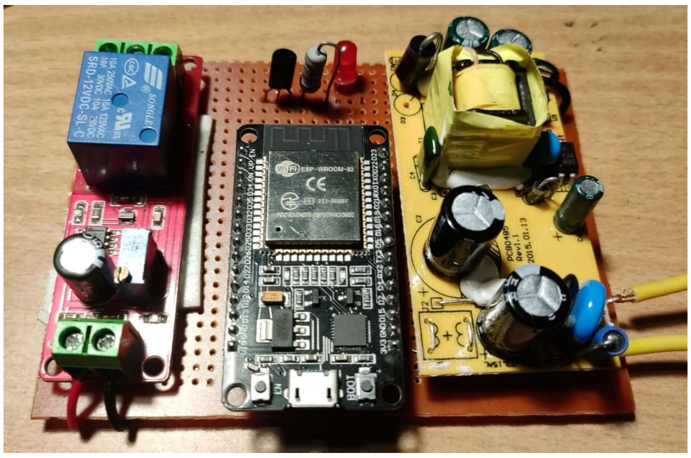


Fig. 4.3: Fallback Circuit Design

Fig 4.3 shows a designed and assembled hardware setup using transistors, relays, and other discrete components previously discussed in section 3.1.4. This interim configuration allowed us to replicate the functionality intended for the PCB, enabling us to proceed with testing and validation.

4.4 Challenges and Limitations

Despite the project's overall success, some challenges were encountered. Speech recognition struggled in environments with significant background noise, reducing accuracy and usability. Additionally, the system's reliance on stable internet connectivity for certain features, such as API calls, highlighted potential limitations in areas with poor network infrastructure.

4.5 User Experience

The assistant was evaluated for user-friendliness and adaptability. Its intuitive interface and responsive functionalities received positive feedback, particularly for its accessibility features, which are beneficial for elderly users or individuals with limited mobility. The modular design of both hardware and software components allowed for straightforward troubleshooting and potential upgrades, underscoring the system's scalability.

Compared to existing commercial solutions, the project demonstrated a cost-effective alternative without sacrificing essential features. The system's customization potential outperformed proprietary voice assistants, offering tailored solutions for specific user needs, such as custom device integration.

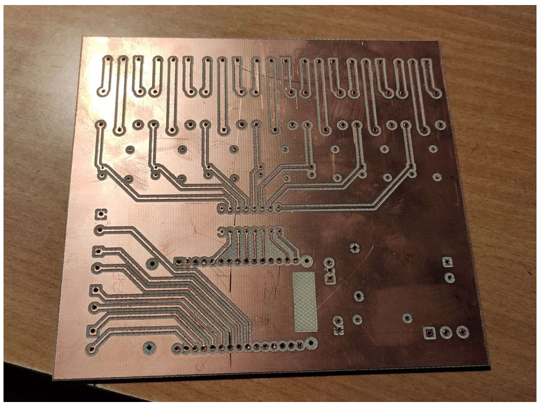


Fig. 4.4: Fabricated PCB from Fabrication Lab, JIIT 62

4.6 Discussion

These results validate the project's objectives, showcasing the successful integration of AI and IoT to create a practical, scalable, and affordable voice assistant for home automation. The challenges identified serve as opportunities for future improvement, particularly in noise reduction and offline functionality development. The project highlights the transformative impact of merging emerging technologies to enhance daily life, reinforcing the viability of AI-powered IoT systems as a cornerstone of smart living.

CHAPTER 5

CONCLUSION AND FUTURE SCOPE

The "Multi-Modal AI Voice Assistant with Integrated Home Automation" showcases the immense potential of combining artificial intelligence and IoT to simplify and enrich everyday life. This project integrates a Python-based voice assistant with a custom-designed PCB and an ESP32 microcontroller, creating a voice-controlled system capable of managing smart home devices and performing digital tasks efficiently.

In terms of future scope, the system can be significantly enhanced by incorporating Retrieval-Augmented Generation (RAG) for faster and more accurate responses. This capability would make the assistant ideal for specialized environments such as construction, healthcare, robotics, and software development by retrieving precise information from databases. Adding advanced database systems could further enhance personalization and efficiency by storing user preferences and task histories, enabling a tailored experience.

Expanding hardware capabilities by integrating sensors and actuators could make the system more interactive and intelligent. For instance, temperature, motion, and light sensors could enable real-time environmental monitoring, while actuators could perform physical tasks, making the system suitable for automation and industrial applications. Additionally, leveraging IoT and advanced technologies, such as sensor-driven automation, edge computing for real-time operations, and cross-platform compatibility with ecosystems like Alexa or Google Home, would enhance scalability and versatility, making it more efficient and adaptable to diverse domains.

This project also provided valuable learning experiences at every stage of development. Designing the custom PCB improved our understanding of hardware integration, while programming the ESP32 deepened our knowledge of microcontrollers and communication protocols. Developing the voice assistant allowed us to explore AI techniques such as speech recognition, text-to-speech, and natural language processing, as well as modular coding and efficient system design. Addressing challenges like optimizing performance and ensuring seamless hardware-software integration honed our problem-solving and debugging skills.

Overall, this cost-effective, user-friendly, and scalable system demonstrates the transformative power of AI and IoT in creating smarter, connected homes. It provides a solid foundation for further advancements, offering smarter automation, greater adaptability, and improved convenience across various domains, while serving as a platform to apply and expand technical expertise in real-world applications.

REFERENCES

1. S. Singh, S. Panwar, H. Dahiya, and Khushboo, "Artificial intelligence voice assistant and home automation," Int. J. Sci. Res. Arch., vol. 12, pp. 2006-2017, May 2024, doi: 10.30574/ijsra.2024.12.1.0954
2. P. Kunekar, A. Deshmukh, S. Gajalwad, A. Bichare, K. Gunjal and S. Hingade, "AI-based Desktop Voice Assistant," 2023 5th Biennial International Conference on Nascent Technologies in Engineering (ICNTE), Navi Mumbai, India, 2023, pp. 1-4, doi: 10.1109/ICNTE56631.2023.10146699.
3. A. C. Maharajpet, V. S. Jadhav, A. M. Panchamukhi, P. Adagatti, and V. S. Gondkar, "Desktop AI Assistant: J.A.R.V.I.S Just a Rather Very Intelligent System," in Proceedings of the Journal of Emerging Technologies and Innovative Research (JETIR), vol. 11, no. 3, pp. 582-591, March 2024. [Online].
4. L. -B. Hernandez-Salinas et al., "IDAS: Intelligent Driving Assistance System Using RAG," in IEEE Open Journal of Vehicular Technology, vol. 5, pp. 1139-1165, 2024, doi: 10.1109/OJVT.2024.3447449.
5. H. A. Shafei and C. C. Tan, "A Closer Look at Access Control in Multi-User Voice Systems," in IEEE Access, vol. 12, pp. 40933-40946, 2024, doi: 10.1109/ACCESS.2024.3379141.
6. Y. Neelima, K. Prajwal, M. Asma, and R. Joseph, “Voice controlled smart home,” *Industrial Engineering Journal*, vol. 53, no. 4, pp. 844–850, Apr. 2024.
7. A. Mahmood, J. Wang, B. Yao, D. Wang, and C.-M. Huang, "LLM-Powered Conversational Voice Assistants: Interaction Patterns, Opportunities, Challenges, and Design Guidelines," preprint, arXiv:2309.13879, Sep. 2023.
8. N. Singh, D. Yagyasen, S. V. Singh, G. Kumar, and H. Agrawal, "Voice Assistant Using Python," *International Journal of Innovative Research in Technology*, vol. 8, no. 2, pp. 419-423, Jul. 2021.
9. Mrs. K. Anusha, S. Aravind, N. Umamaheshwar Rao, and T. Ravisagar, "Internet of Things (IoT) Based Home Automation Using ESP-32," Int. J. Res. Appl. Sci. Eng. Technol., vol. 12, pp. 1672-1676, May 2024, doi: 10.22214/ijraset.2024.61912.
10. V. Chayapathy, "IoT Based Home Automation by Using Personal Assistant," June 12, 2018

APPENDIX

Schematics, Diagrams, and Code Specifications

System Architecture

Block Diagram of the System: Illustrates the flow from voice input to hardware response, including interaction with APIs and automation devices.

Customized PCB Design Schematic Capture: Shows the circuit layout detailing connections between ESP32, relays, and sensors. PCB Layout: Provides the top-down view of the physical PCB design. 3D View: Offers a visual representation of the assembled board.

ESP32 Integration

Power Supply Design: Details of the regulated 5V power supply, LED indicators, and switch modules. Relay Module Configuration: Explanation of the relay operation and connections to high-power devices.

Hardware Specifications

ESP32 Microcontroller: Description of the features utilized in this project, such as Wi-Fi, GPIO pins, and energy efficiency.

Libraries Used:

1. **requests:** Sends HTTP requests to control ESP32 pins.
2. **groq:** Interacts with the Groq API for chat responses.
3. **PIL (Pillow):** Captures and processes images/screenshots.
4. **faster_whisper:** Transcribes audio to text using Whisper.
5. **speech_recognition:** Captures audio input from the microphone.
6. **pyttsx3:** Provides text-to-speech functionality.
7. **cv2 (OpenCV):** Captures images from the webcam.

8. **pyperclip:** Accesses clipboard content.
9. **google.generativeai:** Analyzes image context using generative AI.
10. **os:** Retrieves environment variables for API keys.
11. **time:** Controls sleep intervals in loops.
12. **re:** Matches patterns in text using regular expressions.
13. **atexit:** Executes cleanup functions when the program exits.
14. **ctypes:** Suppress ALSA audio warnings.

Functions

1. **control_esp32:** Sends commands to control ESP32 GPIO states.
2. **groq_prompt**: Gets a conversational response from the Groq API.
3. **vision_prompt:** Generates image context analysis using generative AI.
4. **function_call:** Determines the user's requested action.
5. **take_screenshot:** Captures and saves a screenshot.
6. **web_cam_capture:** Captures and saves an image from the webcam.
7. **get_clipboard_text:** Retrieves text from the clipboard.
8. **speech:** Converts text to speech and plays it.
9. **wav_to_text:** Transcribes audio files to text using Whisper.
10. **extract_prompt**: Extracts the command from transcribed text.
11. **callback:** Processes the user's audio input and executes actions.
12. **start_listening:** Initiates continuous voice recognition.
13. **cleanup:** Releases webcam resources on program exit.

Libraries Used in ESP Program

1. **WiFi.h**: Establishes a connection to the Wi-Fi network.
2. **WiFiManager.h**: Manages Wi-Fi setup, including auto-connect and configuration portal.

Variables and Arrays in ESP Program

1. **WiFiServer server**: Initializes a web server on port 80.
2. **header**: Stores incoming HTTP request data.

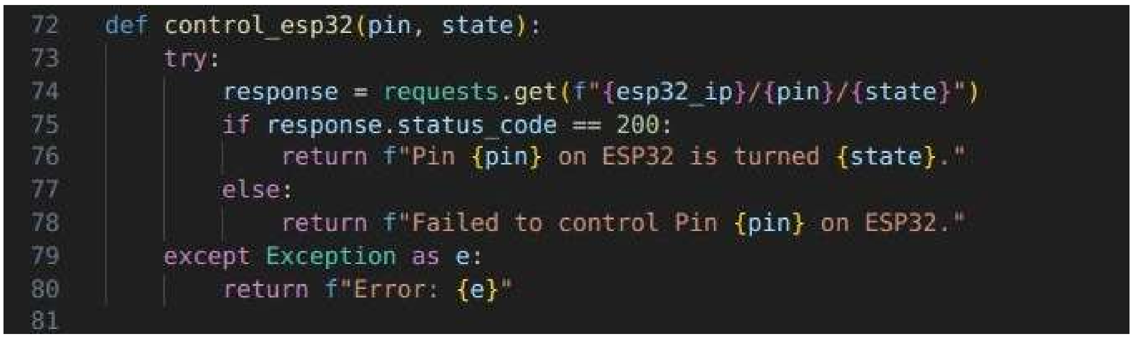
3. **ledPins[]**: Stores GPIO pins connected to LEDs.
4. **switchPins[]**: Stores GPIO pins connected to switches.
5. **pinCount**: Calculates the number of LED/switch pins.
6. **pinStates[]**: Stores the state ("on"/"off") of each LED pin.
7. **currentTime, previousTime**: Track timing for client connection timeout.
8. **timeoutTime**: Defines the maximum time for an HTTP connection.
9. **lastDebounceTimes[]**: Stores last debounce time for each switch.
10. **debounceDelay**: Sets debounce duration to 50 ms.
11. **lastSwitchStates[]**: Stores previous states of switches for debouncing.

Functions in ESP Program

1. **setup()**: Initializes serial communication, configures Wi-Fi using WiFiManager, sets GPIO pins for LEDs and switches, and starts the web server.
2. **loop()**: Main execution loop that handles HTTP client requests, updates the state of LEDs, and checks switch inputs for debouncing.
3. **server.begin()**: Starts the web server to listen for incoming client requests.
4. **WiFiClient client = server.available()**: Checks if a client is trying to connect to the web server.
5. **client.read()**: Reads incoming HTTP request data from the client.
6. **header.indexOf()**: Searches for specific "GET" requests in the HTTP header to identify pin control commands (turning LEDs on/off).
7. **digitalWrite()**: Sets the specified LED pin state (HIGH for ON, LOW for OFF).
8. **client.println()**: Sends HTTP response content (HTML) to the client, including control buttons for each LED.
9. **client.stop()**: Ends the current client connection after sending the response.
10. **digitalRead()**: Reads the current state of each switch pin to detect if it has been pressed.
11. **Debounce check in loop**: Prevents rapid or false triggering of switch inputs by introducing a delay (50 ms).
12. **WiFiManager.autoConnect()**: Attempts to connect to a saved Wi-Fi network or starts an access point for configuration if no network is found.
13. **ESP.restart()**: Restarts the ESP32 if the Wi-Fi connection fails after a timeout.

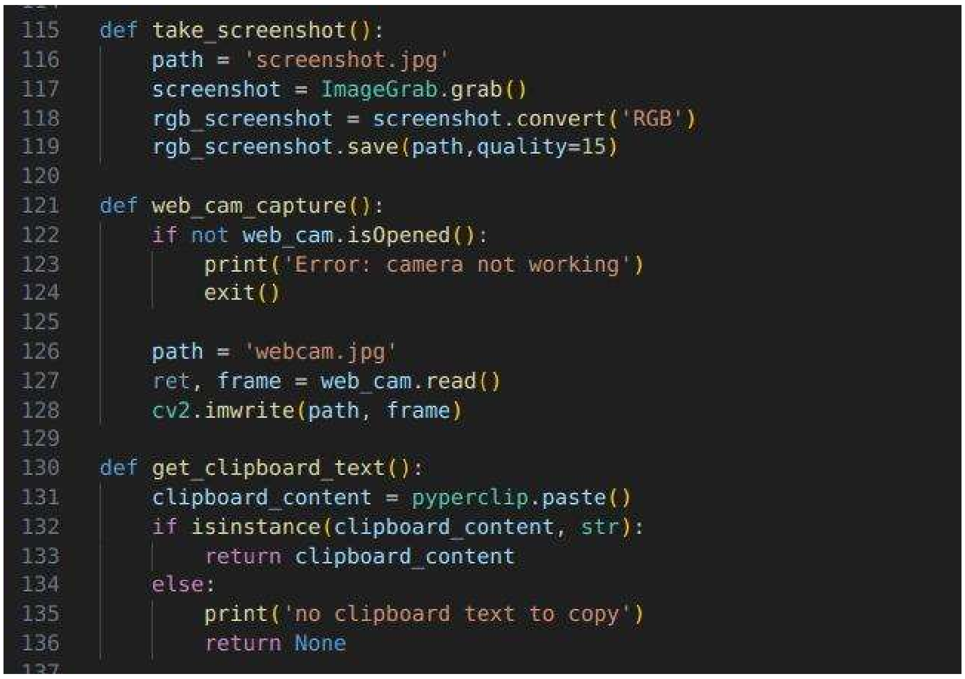
Code Snippets

- **ESP32 Control function:**



```python
def control_esp32(pin, state):
    try:
        response = requests.get(f"{esp32_ip}/{pin}/{state}")
        if response.status_code == 200:
            return f"Pin {pin} on ESP32 is turned {state}."
        else:
            return f"Failed to control Pin {pin} on ESP32."
    except Exception as e:
        return f"Error: {e}"
```

- **Some Functions for the AI Model**



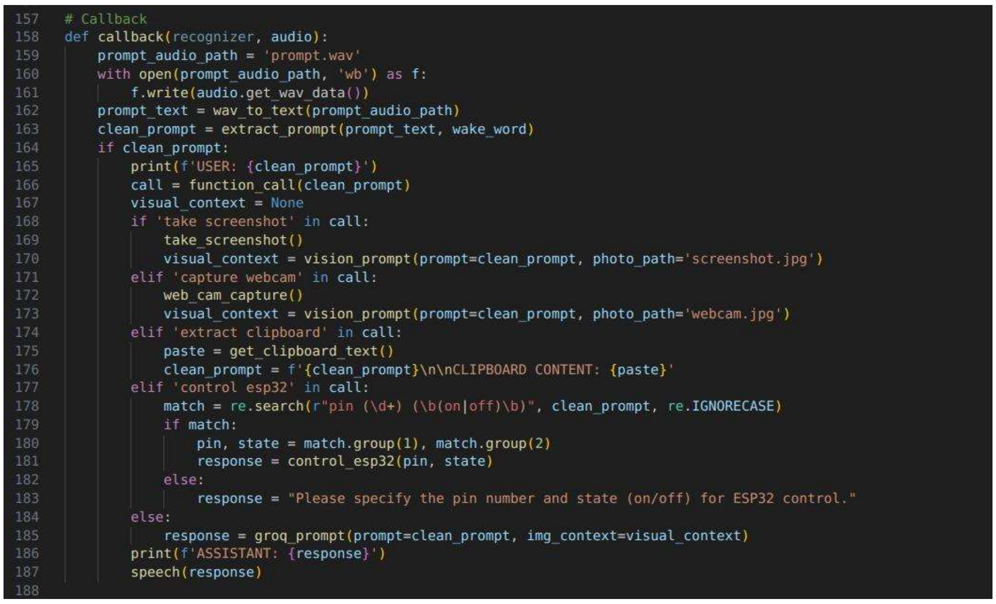
```python
def take_screenshot():
    path = 'screenshot.jpg'
    screenshot = ImageGrab.grab()
    rgb_screenshot = screenshot.convert('RGB')
    rgb_screenshot.save(path,quality=15)

def web_cam_capture():
    if not web_cam.isOpened():
        print('Error: camera not working')
        exit()

    path = 'webcam.jpg'
    ret, frame = web_cam.read()
    cv2.imwrite(path, frame)

def get_clipboard_text():
    clipboard_content = pyperclip.paste()
    if isinstance(clipboard_content, str):
        return clipboard_content
    else:
        print('no clipboard text to copy')
        return None
```

- **Callback Function:**



```python
# Callback
def callback(recognizer, audio):
    prompt_audio_path = 'prompt.wav'
    with open(prompt_audio_path, 'wb') as f:
        f.write(audio.get_wav_data())
    prompt_text = wav_to_text(prompt_audio_path)
    clean_prompt = extract_prompt(prompt_text, wake_word)
    if clean_prompt:
        print(f'USER: {clean_prompt}')
        call = function_call(clean_prompt)
        visual_context = None
        if 'take screenshot' in call:
            take_screenshot()
            visual_context = vision_prompt(prompt=clean_prompt, photo_path='screenshot.jpg')
        elif 'capture webcam' in call:
            web_cam_capture()
            visual_context = vision_prompt(prompt=clean_prompt, photo_path='webcam.jpg')
        elif 'extract clipboard' in call:
            paste = get_clipboard_text()
            clean_prompt = f'{clean_prompt}\n\nCLIPBOARD CONTENT: {paste}'
        elif 'control esp32' in call:
            match = re.search(r"pin (\d+) (\b(on|off)\b)", clean_prompt, re.IGNORECASE)
            if match:
                pin, state = match.group(1), match.group(2)
                response = control_esp32(pin, state)
            else:
                response = "Please specify the pin number and state (on/off) for ESP32 control."
        else:
            response = groq_prompt(prompt=clean_prompt, img_context=visual_context)
        print(f'ASSISTANT: {response}')
        speech(response)
```