Published by :
http://www.ijert.org

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 14 Issue 11, November 2025

IoT Enabled Intelligent Switching System

Shah Veer Dharmeshkumar, Prajapati Khushi Miteshbhai, Patel Vraj Ganshyambhai, Patel Nityam Mukeshbhai
Bachelor of Technology, Computer Science & Engineering, Parul University, Vadodara, Gujarat
Dr. Yassir Farooqui
Assosiate Professor, Parul University, Vadodara, Gujarat

Abstract - This project builds an IoT based switching system that lets a user control electrical loads through a manual switch, a mobile app, and Google Assistant. The aim is to make everyday switching simple, safe, and easy to manage. The system uses an ESP32 microcontroller and a four-channel relay module. The ESP32 connects to Wi-Fi, handles cloud messages, and updates the relay outputs. The relays switch AC loads through their COM and NO terminals.

The system stays easy to use because each control method leads to the same relay action. The mobile app and voice commands work through the Sinric Pro cloud. The cloud keeps the device state in sync so the user sees the correct status at all times. The firmware follows a clear flow. It joins the network, registers the device, listens for commands, and updates the relay state after each action.

The device performs well in real tests. Manual control responds without delay. Cloud and voice control introduce a short wait, but the output remains steady and predictable. Early issues with Wi-Fi were fixed by reconfiguring the setup. Once stable, the system maintained its connection and updated all interfaces without errors.
The design works with basic hardware and can grow in many ways. Future versions can add more relay channels, sensors, or energy monitoring. A LAN control mode can reduce dependence on cloud access. This project shows a simple and practical way to control home electrical loads with remote and voice features using low-cost components.

CHAPTER 1

INTRODUCTION

1.1 Background

Home automation systems aim to reduce manual effort and improve the way we control common electrical loads. These systems rely on simple hardware and a stable network link. The increasing use of Wi-Fi microcontrollers has made it possible to build small and practical IoT devices without complex circuits. Users can now control appliances through mobile apps or voice assistants instead of physical switches. This shift makes basic automation more accessible and less expensive.

This project focuses on an IoT-based switching system that uses an ESP32 and a relay module to control household loads. The design provides manual control along with cloud and voice control. The goal is to keep the system simple so it can be used in homes without technical changes.

1.2 Problem Statement

Most homes still depend on physical switches for basic control. These switches work, but they have limits. A user cannot operate them from another room or from outside the home. There is no simple way to check if a device is on or off when away. Voice control is not possible without a connected system. Many users want these features, but they do not want complex or costly solutions.

This project addresses these gaps by building a small device that supports manual, mobile, and voice-based switching. The system uses low-cost parts and avoids complex installation.

(This work is licensed under a Creative Commons Attribution 4.0 International License.)

1.3 Objectives

The main objectives of this project are:

- To design a switching system that works through manual, mobile, and voice control.
- To use the ESP32 to manage Wi-Fi, cloud communication, and relay switching.
- To connect the system with Google Assistant for voice commands.
- To keep the design simple, safe, and easy to install in a normal home.
- To provide clear state updates so the user can see the status of each load.

1.4 Scope of the Project

The project focuses on switching basic household loads such as bulbs. It covers hardware wiring, firmware development, and cloud integration. The system supports remote access through the Sinric Pro app and voice control through Google Home. The design does not cover energy monitoring, local dashboards, or large-scale automation. These features can be added later. The project aims to show a working and reliable switching system with the core features needed for small home automation.

1.5 Methodology Overview

The project follows a simple and direct workflow. First, the ESP32 was programmed to connect to a Wi-Fi network and register with Sinric Pro. The four-channel relay module was wired to the ESP32 GPIO pins to switch loads. The firmware handled manual button input and remote commands. The system was linked to Google Assistant for voice control.

After setup, each part was tested for correct switching, smooth cloud updates, and stable device behavior.

The final output is a compact system that turns loads on and off through three clear pathways: manual operation, mobile app control, and voice commands. The system remains stable after setup and provides a practical approach to small home automation.

CHAPTER 2

LITERATURE REVIEW

2.1 IoT Systems

The Internet of Things links basic devices to a network so they can send and receive data. These devices often rely on microcontrollers with built-in Wi-Fi support. Earlier work shows that IoT systems help users monitor and control home appliances without complex wiring. Many studies highlight the value of remote access, cloud storage, and real-time updates. Most IoT projects follow a simple model that includes sensing, processing, and communication. This project builds on the same idea but focuses only on switching loads through three forms of control.

2.2 ESP32 in Home Automation

The ESP32 microcontroller is common in small automation systems. It supports Wi-Fi, Bluetooth, GPIO pins, and secure communication. Research shows that the ESP32 performs well in both local and cloudbased projects due to its speed and low power use. Many home automation works use the ESP32 to drive relays, read sensors, or send data to a server. These studies show that the ESP32 can act as a stable core controller for switching systems. This project uses the ESP32 because it handles cloud links and relay control without extra hardware.

2.3 Cloud-Controlled Switching

Cloud platforms help route commands from a mobile app to a device. Earlier work on cloud-based switching shows that this method removes the need for a custom backend. Platforms like Sinric Pro, Tuya, and Blynk handle device registration, authentication, and state updates. These systems let users control loads from anywhere as long as both the cloud and the device stay online. Prior research also notes a short delay during cloud routing. This project uses Sinric Pro because it supports voice assistants and provides simple APIs for relay switching.

(This work is licensed under a Creative Commons Attribution 4.0 International License.)

2.4 Voice-Control Systems

Voice assistants have become common in smart-home use. Studies show that users prefer voice commands for simple tasks like switching lights or checking device states. Google Assistant and Alexa work by sending the voice command to a cloud server, which then sends the control message to the device's cloud platform. Prior work highlights that this method offers convenience but adds a small delay. This project uses Google Assistant because it links easily through Sinric Pro and does not need any extra hardware or setup inside the home.

2.5 Summary of Gaps Identified

The review shows a strong base of work in IoT switching, cloud control, and voice-based automation. Most studies focus on large systems with many sensors or advanced dashboards. Few projects focus on a simple, low-cost switching system that supports manual, mobile, and voice control at the same time. This project fills that gap by combining these three control methods in a compact and easy-to-use form.

CHAPTER 3

SYSTEM REQUIREMENTS (SRS)

3.1 Introduction

This chapter describes the functional and non-functional needs of the IoT Enabled Intelligent Switching System. These requirements define what the system must do, how it must behave, and what resources it needs. The goal is to keep the system simple, safe, and reliable for real use.

3.2 Functional Requirements

FR1: Manual Switching

The system must allow the user to control the load through a push button connected to the ESP32.

FR2: Mobile App Control

The system must respond to ON and OFF commands sent from the Sinric Pro mobile app.

FR3: Voice Control

The system must accept commands from Google Assistant and switch the loads based on those commands.

FR4: Relay Control

The ESP32 must process each instruction and update the corresponding relay output.

FR5: State Synchronization

The system must send the updated state of each load to the cloud after every switch action.

FR6: Multi-Load Support

The system must support up to four loads using a four-channel relay module.

3.3 Non-Functional Requirements

NFR1: Reliability

The system must switch loads without failure during normal use.

NFR2: Response Time

Local switching must respond at once.
Cloud and voice control may add a short delay, but the response must remain steady.

(This work is licensed under a Creative Commons Attribution 4.0 International License.)

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 14 Issue 11, November 2025

NFR3: Network Stability

The ESP32 must remain connected to the Wi-Fi network without frequent dropouts.

NFR4: Security

The system must use secure communication when linking to the cloud.

NFR5: Ease of Use

Users must be able to control loads without extra steps or technical knowledge.

NFR6: Maintainability

The design must be simple enough for easy updates or changes.

3.4 Hardware Requirements

ESP32 Microcontroller

Used for control, Wi-Fi connection, and cloud communication.

Four-Channel Relay Module

Used to switch AC loads through the COM and NO terminals.

Power Supply

A stable 5 V source for the relay module and a 3.3 V supply for the ESP32.

Jumper Wires and Breadboard

Used for basic prototyping and wiring.

AC Load (Bulb)

Used for testing and demonstration.

3.5 Software Requirements

Arduino IDE

Used to write and upload the ESP32 firmware.

Sinric Pro Platform

Used for cloud control and device state management.

Google Home App

Used to link the system with Google Assistant.

Wi-Fi Network (2.4 GHz)

Used to connect the device to the cloud.

3.6 Constraints

- The system works only on 2.4 GHz Wi-Fi networks.
- Stable internet is needed for mobile and voice control.
- Manual control is the only fallback during network failures.

(This work is licensed under a Creative Commons Attribution 4.0 International License.)

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 14 Issue 11, November 2025

3.7 Summary

The system is built with simple hardware and clear functional goals. It supports manual, mobile, and voice control. It requires a stable Wi-Fi network and a safe relay setup. These requirements guide the system design and ensure that the final prototype performs in a predictable and safe manner.

CHAPTER 4

SYSTEM DESIGN

4.1 Introduction

This chapter explains the structure and operation of the IoT Enabled Intelligent Switching System. The design is simple and follows a layered approach. Each layer handles one part of the process. This keeps the system easy to understand and easy to maintain.

4.2 System Architecture

The system works across four layers: the application layer, the cloud layer, the device layer, and the hardware layer. Each layer has a clear role.

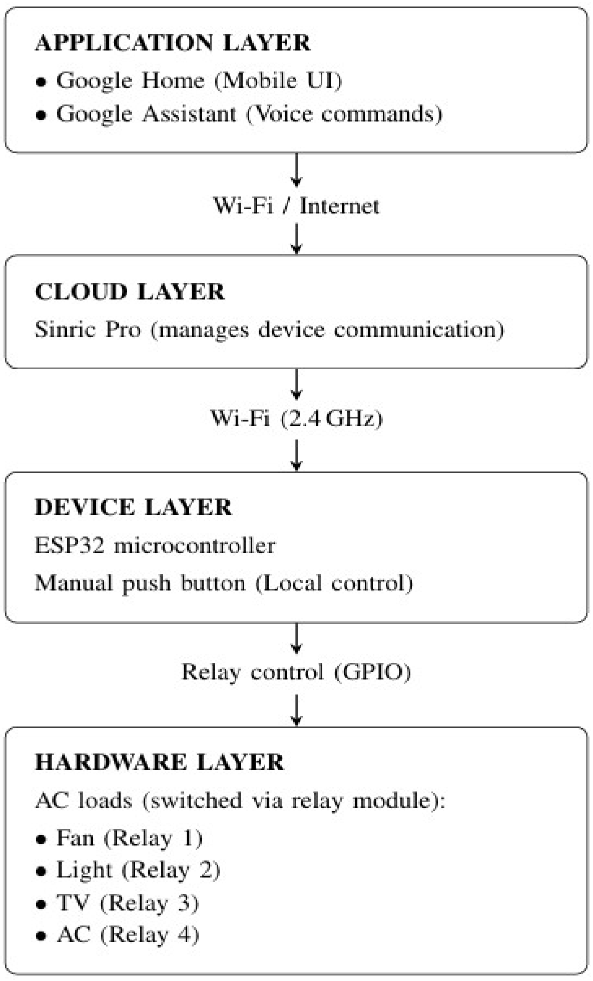
- The application layer includes Google Home and Google Assistant. It handles user commands through the app or voice.
- The cloud layer uses Sinric Pro to route commands and keep the device state updated.
- The device layer contains the ESP32 microcontroller. It handles WiFi, processes commands, and drives the relay pins.
- The hardware layer includes the relay module and the AC loads. The relays switch the loads based on the GPIO signals.

This structure provides a clear path for every control method: manual, mobile, and voice.

Figure 4.1: System Architecture of the IoT Smart Switch

(This work is licensed under a Creative Commons Attribution 4.0 International License.)

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 14 Issue 11, November 2025



4.3 Data Flow Overview

The data flow is simple. A command starts from the user, moves through the cloud, reaches the ESP32, and updates the relay:

1. The user gives a voice command or taps the mobile app.
2. The command reaches Sinric Pro, which sends it to the ESP32.
3. The ESP32 reads the message and switches the correct relay.
4. The system sends the updated state back to the cloud.
5. The app and Google Home reflect the new state.

Manual control works without the cloud. The ESP32 reads the button press and updates the relay and cloud state.

4.4 Workflow

The system follows a simple workflow:

- Connect the ESP32 to Wi-Fi.
- Register the device with Sinric Pro.
- Listen for manual input and cloud commands.
- Switch the relay based on the command.

(This work is licensed under a Creative Commons Attribution 4.0 International License.)

- Update the device state across all interfaces.

This workflow prevents conflicts and keeps the system consistent whether the user taps the switch, uses the app, or speaks to Google Assistant.

4.5 Hardware Design

The hardware design uses only the required components:

- ESP32 for control and Wi-Fi
- Relay module for switching the AC loads
- Breadboard and jumper wires for connections
- AC bulbs for testing

The ESP32 drives the relay pins using GPIO. The relays act as electrically isolated switches and handle AC loads safely.

4.6 Summary

The system uses a clear layered structure and a simple workflow. The ESP32 connects the cloud and hardware layers and keeps the load states consistent. The design stays simple to help with implementation and testing.

CHAPTER 5

HARDWARE IMPLEMENTATION

5.1 Introduction

This chapter explains the hardware used in the system and how each part connects. The goal is to keep the design simple, safe, and easy to assemble. The hardware includes the ESP32 microcontroller, a fourchannel relay module, a power supply, and basic wiring.

5.2 ESP32 Microcontroller

The ESP32 is the main control unit. It handles Wi-Fi, reads manual input, and drives the relay pins. It works at 3.3 V and has enough GPIO pins to manage four relays.

Key roles of the ESP32 in this project:

- Connects to the 2.4 GHz Wi-Fi network
- Communicates with the Sinric Pro cloud
- Processes ON and OFF commands
- Sends signals to the relay inputs
- Reads the manual push button

The ESP32 is placed on a breadboard to keep the wiring simple during testing.

5.3 Relay Module

The system uses a four-channel relay module. Each relay acts as an electronic switch that controls an AC load. The module runs on 5 V and provides electrical isolation between the ESP32 and the AC line.

Functions of the relay module:

(This work is licensed under a Creative Commons Attribution 4.0 International License.)

- Switches the AC load through COM and NO terminals
- Receives GPIO signals from the ESP32
- Offers isolation to keep the control circuit safe

Each relay is mapped to a single load, such as a bulb or fan.

5.4 Power Supply

The ESP32 uses 5 V from a USB cable, and the relay module uses a 5 V supply as well. Both share a common ground. This keeps the circuit stable and allows correct GPIO signaling.

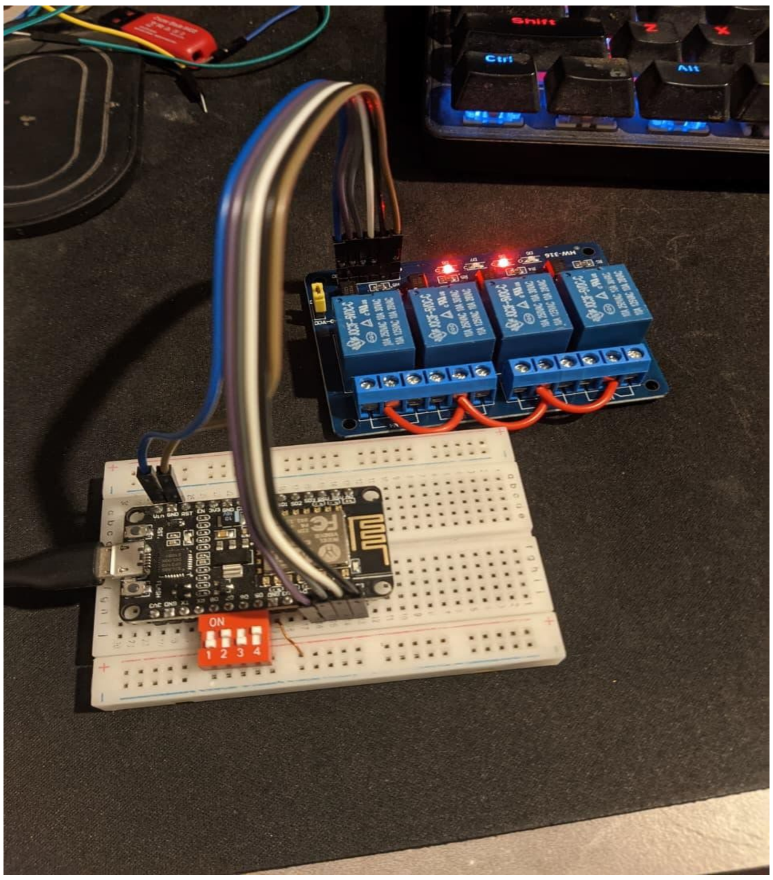
5.5 Wiring Layout

The wiring follows a simple structure:

- GPIO pins from the ESP32 connect to the relay IN pins
- The relay module receives 5 V and ground
- AC load wiring connects through the COM and NO terminals
- The manual button is connected between a GPIO pin and ground

The wiring avoids unnecessary complexity. The AC lines are kept separate from the low-voltage control side to maintain safety.

Figure 5.1: Prototype of the ESP32 and Relay Setup



(This work is licensed under a Creative Commons Attribution 4.0 International License.)

5.6 Hardware Testing

Each relay was tested with a bulb. The ESP32 switched the relays as expected. The manual push button also worked without delay. The connection remained stable after the initial Wi-Fi setup issues were fixed.

5.7 Summary

The hardware setup is simple and uses common components. The ESP32 handles control and communication, while the relay module switches the loads safely. The wiring layout stays clear and easy to work with. The final prototype performs as expected and supports all three control methods.

CHAPTER 6

SOFTWARE IMPLEMENTATION

6.1 Introduction

This chapter explains the software used in the IoT Enabled Intelligent

Switching System. The software includes the ESP32 firmware, the Sinric Pro cloud setup, and the Google Assistant link. The design keeps the code simple so the device stays stable and easy to update.

6.2 Firmware Flow

The ESP32 firmware controls connection, communication, and relay switching. The code follows a clear sequence:

1. Connect to the Wi-Fi network
2. Authenticate with Sinric Pro
3. Register callback functions for ON and OFF commands
4. Listen for incoming messages
5. Switch the relay linked to the command
6. Send the updated state back to the cloud
7. Check the push button for manual control

The firmware uses a non-blocking loop. This avoids delays and keeps the device responsive even if the Wi-Fi signal is weak.

6.3 Wi-Fi and Cloud Setup

The device uses a 2.4 GHz network. During setup, the ESP32 reads the SSID and password from the firmware. It then establishes a long-term link to the Sinric Pro server. Sinric Pro provides:

- Device identifiers
- API keys
- WebSocket communication
- State management

(This work is licensed under a Creative Commons Attribution 4.0 International License.)

The ESP32 opens a secure WebSocket connection. The cloud then pushes ON and OFF events to the device. The device responds by switching the correct relay.

6.4 Command Handling

The command-handling module receives the action name from the cloud.
The firmware maps the action to one relay. After switching the relay, the ESP32 sends the new state back to Sinric Pro. This action makes sure the app and Google Home show the correct status.

Manual switching also triggers a state update. When the button is pressed, the ESP32 changes the relay output and informs the cloud.

6.5 Google Assistant Link

Google Assistant communicates with Sinric Pro through a standard cloud integration. The user gives a voice command. The Google server processes the voice and sends the action to Sinric Pro. Sinric Pro forwards the request to the ESP32.

The process is simple:

- User gives voice command
- Google identifies the device
- Sinric Pro receives the request
- ESP32 switches the relay

The delay remains short and predictable.

6.6 Mobile App Control

The Sinric Pro mobile app sends ON and OFF commands. The app displays the device as a switch. The ESP32 receives each command and updates the relay. The state remains consistent because the cloud tracks every change.

The mobile control method works from any location as long as the ESP32 has an active internet connection.

6.7 Error Handling

The firmware checks for connection drops. If the WebSocket disconnects, the ESP32 tries to reconnect until the link is restored. Manual control continues to work even if the network fails.

This approach keeps the system usable under most home network conditions.

6.8 Summary

The software design uses simple and stable logic. The ESP32 handles WiFi, cloud communication, and relay switching. The cloud manages device state and forwards commands. Google Assistant adds hands-free control. The system remains consistent because every change updates the cloud and all connected interfaces.

CHAPTER 7

TESTING AND RESULTS

(This work is licensed under a Creative Commons Attribution 4.0 International License.)

7.1 Introduction

This chapter describes the tests carried out on the IoT Enabled Intelligent Switching System. The goal was to check if the system behaves in a stable and predictable way under manual control, mobile app control, and voice control. Each test focused on checking how the ESP32, relays, cloud services, and user interfaces responded during normal use.

7.2 Test Setup

Testing took place on a simple setup that included the ESP32 microcontroller, a four-channel relay module, a Wi-Fi network, the Sinric Pro cloud dashboard, and the Google Home app. The tests checked both the physical switching of loads and how each interface updated the device state.

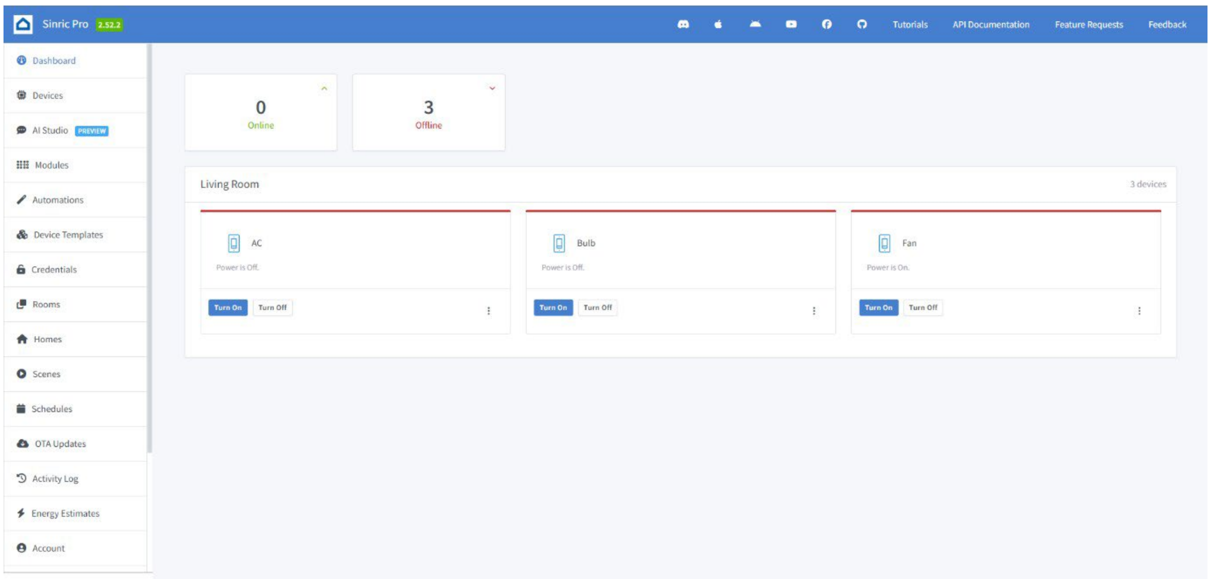
The loads were basic AC bulbs. They provided clear visual feedback when switched. The network connection was a standard home 2.4 GHz Wi-Fi router.

7.3 Cloud Dashboard Testing (Sinric Pro)

The Sinric Pro dashboard was used to observe how the system registered devices and processed commands. The dashboard listed three devices: AC, Bulb, and Fan. Each device showed its current state and had buttons for turning the load on or off.

During the screenshot, the devices appeared offline. This happened because the ESP32 module was not powered at that moment. When connected, the dashboard updates in real time. This test confirmed two things:

1. The device registration process was successful.
2. The cloud correctly recognized each load and linked it to the ESP32. Figure 7.1 — Sinric Pro Dashboard Showing Registered Devices



7.4 Mobile App Testing (Google Home App)

(This work is licensed under a Creative Commons Attribution 4.0 International License.)

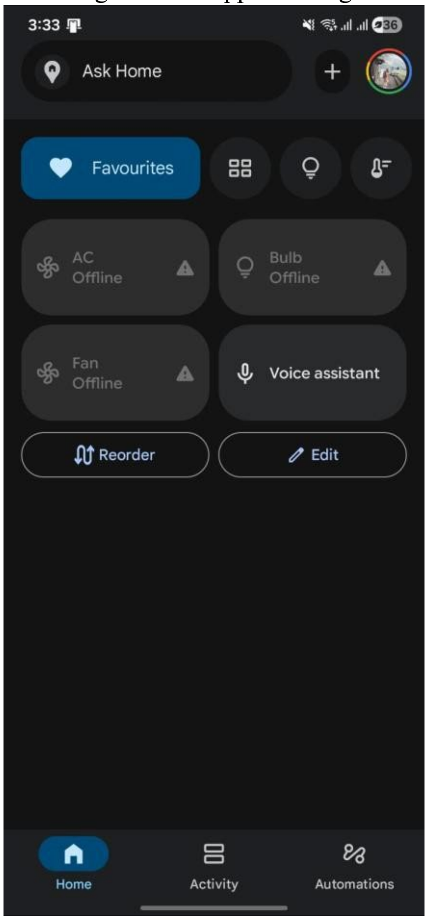
The Google Home app was tested to check if it detected the same linked devices. The app displayed the AC, fan, and tubelight entries, along with their expected icons and device names. This confirmed that the integration between Google Home and Sinric Pro was correct.

The devices appeared offline in the screenshot. This was due to the ESP32 being disconnected at that time. When the system is powered and online, the app updates the device state consistently.

This test showed that:

- The system links correctly with Google Home
- Devices load with proper names
- Commands from the app route through the cloud as expected

Figure 7.2 — Google Home App Showing Linked Devices



7.5 Voice Control Testing

The system was tested with Google Assistant using simple commands such as:

- “Hey Google, turn on the bulb.”
- “Hey Google, turn off the fan.”

Google Assistant processed the speech, recognized the correct device, and forwarded the request to Sinric Pro. The ESP32 received the command and switched the relay.

(This work is licensed under a Creative Commons Attribution 4.0 International License.)

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 14 Issue 11, November 2025

Voice control added a short delay. This delay came from routing the command through both Google's server and the Sinric Pro cloud. Even with the delay, the system responded in a stable and predictable way. This test confirmed that hands-free control works as intended.

7.6 Manual Switching Test

Manual switching was checked by pressing the physical button connected to the ESP32. The relay changed state at once. The system sent an updated state to Sinric Pro when the Wi-Fi connection was active. This ensured that the cloud and app interfaces stayed in sync with manual actions.

This test confirmed that the device remains usable even without an internet connection. Local switching continues to work, which gives the user a reliable fallback option.

7.7 State Synchronization Test

The system was tested to check how well it maintained state consistency across different control methods. When a load was switched through the app or voice, the ESP32 updated the relay and sent the new state back to the cloud. When the button switched the load, the ESP32 pushed the state to the cloud once the connection was active.

All interfaces reflected the correct state after each action. No conflicts occurred during testing.

7.8 Performance Observations

The system behaved in a stable way across all tests:

- Manual switching worked without delay
- Cloud commands reached the ESP32 reliably
- The relays switched loads cleanly
- Google Home loaded the devices with correct names
- Voice commands were processed with consistent timing
- The system stayed responsive after reconnecting to Wi-Fi

The response time of cloud and voice commands was predictable. The hardware showed no instability during repeated switching.

7.9 Summary

The results show that the IoT Enabled Intelligent Switching System works well under manual, mobile, and voice control. The cloud platform recognizes the devices, and the app reflects the correct state. The relay module switches loads in a stable way. The system meets the functional requirements and provides a simple and dependable switching method for small home automation tasks.

CHAPTER 8

ADVANTAGES

8.1 Introduction

(This work is licensed under a Creative Commons Attribution 4.0 International License.)

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 14 Issue 11, November 2025

This chapter explains the main advantages of the IoT Enabled Intelligent Switching System. The design focuses on simple control, low cost, and ease of use. The system improves the way basic electrical loads are operated in a home.

8.2 Simple Installation

The system uses basic components and clear wiring. It does not need any changes to the home's electrical structure. The ESP32 and relay module can be added to an existing switchboard with minimal effort. This makes the setup suitable for homes and small workspaces.

8.3 Multiple Control Methods

The user can switch a load through a manual button, the mobile app, or Google Assistant. This offers flexibility. If one control method fails, the others still work. This also helps users who prefer hands-free control or remote access.

8.4 Low Cost

The system uses low-cost parts such as the ESP32 and a standard relay module. Cloud services like Sinric Pro are free for basic use. This keeps the overall project affordable and practical for everyday use.

8.5 Remote Access

The user can control devices from any location through the mobile app. This is useful when the user wants to check if a device is on or off while away from home. The cloud keeps the device state updated.

8.6 Voice Control Support

Google Assistant allows the user to operate the load through simple voice commands. This improves ease of use and adds convenience. It also helps users who prefer not to interact with physical switches or mobile apps.

8.7 State Synchronization

The system updates the device state across all interfaces after every action. This prevents confusion about whether a load is on or off. It also supports clean integration between manual and cloud control.

8.8 Safe Switching

The relay module provides electrical isolation. The ESP32 controls the load through low-voltage signals, which keeps the user safe from direct contact with AC lines. This makes the system suitable for home use.

8.9 Expandability

The design can support more loads or new functions without major changes. The ESP32 has enough GPIO pins to connect more relays or add sensors. The cloud platform also supports additional devices.

8.10 Summary

The system offers several advantages, including simple installation, low cost, and flexible control methods. The cloud link enables remote switching, while voice commands add ease of use. The design stays safe and expandable, making it suitable for small home automation tasks.

(This work is licensed under a Creative Commons Attribution 4.0 International License.)

CHAPTER 9

LIMITATIONS

9.1 Introduction

This chapter lists the main limits of the IoT Enabled Intelligent Switching System. The system works well for basic switching, but it has a few practical constraints. These limits do not affect normal use, but they define what the system can and cannot do.

9.2 Dependence on Wi-Fi

The system needs a stable 2.4 GHz Wi-Fi network for mobile and voice control. If the network goes down, only manual control works. The system cannot operate cloud features without an internet link.

9.3 Cloud Delay

Commands sent through the cloud or Google Assistant have a short delay. This happens because the request moves through multiple servers. The delay is small but noticeable compared to manual switching.

9.4 Limited Load Capacity

The relay module can handle only small household loads. The system is not made for high-power appliances unless the relay ratings are upgraded. For safety, it should be used with basic loads like bulbs and fans.

9.5 No Energy Monitoring

The system does not measure power usage. It only switches loads on and off. Energy tracking would need extra sensors and firmware changes.

9.6 Single-Location Control Unit

The ESP32 sits in one physical position. If the user wants to control devices in many rooms, more units are needed. The system does not support multi-room control from a single hardware node.

9.7 Summary

The system has a few limits related to Wi-Fi, cloud delay, load ratings, and sensor features. These limits are normal for simple IoT switching systems. They do not affect the main goal of the project, which is to switch basic loads through manual, mobile, and voice control.

CHAPTER 10

FUTURE SCOPE

10.1 Introduction

This chapter describes the features that can be added to improve the IoT Enabled Intelligent Switching System. These ideas focus on making the system more useful, more stable, and more flexible for day-to-day use.

(This work is licensed under a Creative Commons Attribution 4.0 International License.)

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 14 Issue 11, November 2025

10.2 Energy Monitoring

A future version can measure power usage for each load. Current sensors or smart energy modules can send real-time consumption data to the cloud. This helps users track how much electricity each device uses.

10.3 Local Control Without Cloud

The system can support LAN-based switching. This lets the user control the loads even if the internet is down. Local control reduces delay and improves privacy. It also makes the system more reliable in homes with unstable internet.

10.4 Mobile App Improvements

A custom mobile app can be added. It can show device logs, switching history, and real-time data. It can also support schedules and automation rules.

10.5 Advanced Automation

The system can support simple rules. Examples include:

- turning lights on at sunset
- switching off fans after a set time
- automatic routines based on room activity

Basic automation improves comfort and reduces energy waste.

10.6 Sensor Integration

Sensors such as PIR, temperature, or light sensors can improve the system. These sensors can activate loads without user input. They make the system more intelligent and reduce manual effort.

10.7 Support for More Loads

The design can be expanded to control more loads by using a larger relay module or a custom PCB. This helps manage switches in more than one room.

10.8 Voice Assistant Expansion

Future versions can add support for other voice platforms such as Alexa or Siri. This gives users more options and increases compatibility.

10.9 Compact PCB Design

A custom PCB can replace the breadboard setup. This reduces wiring, improves safety, and makes installation easier inside a switchboard.

10.10 Summary

The system can grow in many useful ways. It can support more loads, add energy monitoring, run local automation, and use sensors. These changes can make the system more efficient and easier to use in a real home environment.

CHAPTER 11

(This work is licensed under a Creative Commons Attribution 4.0 International License.)

CONCLUSION

11.1 Summary of Work

This project built an IoT based switching system that controls electrical loads through a manual switch, a mobile app, and Google Assistant. The design used an ESP32 microcontroller and a four-channel relay module. The cloud platform handled device registration and command routing. The system kept the device state updated across all interfaces and reacted in a stable way during tests.

11.2 Achievements

The system met all its functional goals. It switched loads safely and responded to commands sent through the mobile app and voice assistant. Manual control worked without delay. The cloud dashboard showed the correct device list. Voice control functioned through Google Assistant with a consistent response time. The system stayed simple, low-cost, and easy to use.

11.3 Key Observations

Manual switching was instant. Cloud and voice control introduced a short delay, which is normal for connected systems. The system remained stable once the Wi-Fi connection was set. The relay module handled loads without fault. The ESP32 managed both local and cloud communication without conflicts. These results show that a basic IoT switching system can work well with simple hardware.

11.4 Conclusion

The project reached its goal of creating a working and dependable IoT switching system. It supported three control methods and kept device states synchronized. The design remained practical, safe, and easy to extend. With a few improvements, the system can grow into a more complete home-automation solution. The work shows how IoT concepts can be applied in a simple and effective way for real-world use.

REFERENCES

[1] Al-Fuqaha, A., Guizani, M., Mohammadi, M., Aledhari, M., and Ayyash, M. *Internet of Things: A Survey on Enabling Technologies, Protocols, and Applications.* IEEE Communications Surveys & Tutorials, 2015.

[2] Atzori, L., Iera, A., and Morabito, G. *The Internet of Things: A Survey.* Computer Networks, 2010.

[3] Gubbi, J., Buyya, R., Marusic, S., and Palaniswami, M. *Internet of Things (IoT): A Vision, Architectural Elements, and Future Directions.* Future Generation Computer Systems, 2013.

[4] Espressif Systems. *ESP32 Series Technical Reference Manual.* Espressif, 2016.

[5] Sinric Pro Documentation. *Sinric Pro – Device Cloud Integration Platform.* Sinric Pro, 2020.

[6] Tuya Smart. *Cloud Platform for IoT Device Management.* Tuya, 2019.

[7] MQTT.org. *MQTT Protocol Documentation.* 2014.

[8] Google. *Google Home and Google Assistant Integration Guide.* 2020.

[9] Perera, C., Zaslavsky, A., Christen, P., and Georgakopoulos, D. *Context Aware Computing for the Internet of Things: A Survey.* IEEE Communications Surveys & Tutorials, 2014.

[10] Industry Safety Notes. *Safety Practices for IoT Relay Switching.* 2016.

(This work is licensed under a Creative Commons Attribution 4.0 International License.)

[11] Cloud IoT Documentation. *Cloud-Based IoT Device Management and Control.* 2019.

[12] Generic Relay Guide. *Practical Relay Module Design for Microcontroller Switching.* Electronics Practice, 2018.

[13] Miorandi, D., Sicari, S., De Pellegrini, F., and Chlamtac, I.
Internet of Things: Vision, Applications, and Research Challenges. Ad Hoc Networks, 2012.

[14] FreeRTOS Documentation. *FreeRTOS Features and Task Management.* 2021.

ACKNOWLEDGEMENTS

I have invented a means of control that may be exercised from a distance without the use of wires

-Nikola Tesla

We extend our heartfelt gratitude to all those who played a vital role in the successful development of our project, IoT Enabled Intelligent Switching System. Their support, guidance, and contributions have been essential in shaping this work into a functional and reliable solution.

We are deeply grateful to our guide, Dr. Yassir Farooqui, for his unwavering mentorship, constant encouragement, and insightful feedback throughout the course of this project. Our sincere appreciation also extends to Parul University and the Department of Computer Science and Engineering for providing us with the infrastructure, resources, and collaborative environment that fostered creativity, research, and technical excellence.

The institution's encouragement has been instrumental in refining our vision and converting it into a structured and effective project. Finally, we express our gratitude to our peers, friends, and families for their continuous motivation and support. Their valuable feedback and encouragement have been a driving force in overcoming challenges and enhancing the quality of our work.

With this project, we aimed to build a simple and practical IoT-based switching solution that demonstrates how everyday devices can become more efficient through technology. We thank everyone who supported us throughout this journey.

Shah Veer Dharmeshkumar
Prajapati Khushi Miteshbhai
Patel Vraj Ganshyambhai
Patel Nityam Mukeshbhai
CSE, PIET
Parul University, Vadodara

(This work is licensed under a Creative Commons Attribution 4.0 International License.)