iJRASET
International Journal For Research in Applied Science and Engineering Technology



INTERNATIONAL JOURNAL FOR RESEARCH

IN APPLIED SCIENCE & ENGINEERING TECHNOLOGY

Volume: 13 **Issue:** V **Month of publication:** May 2025

DOI: https://doi.org/10.22214/ijraset.2025.70985

www.ijraset.com

Call: 08813907089 | E-mail ID: ijraset@gmail.com

International Journal for Research in Applied Science & Engineering Technology (IJRASET)
ISSN: 2321-9653; IC Value: 45.98; SJ Impact Factor: 7.538
Volume 13 Issue V May 2025- Available at www.ijraset.com

Seamless Home Automation with IoT Integration

Akash Bhaduri[1], Md Razique Alam[2], Abhishek Kumar[3], Harsh Sah[4], Md Shizan Yusuf[5], Arpita Santra[6]
Dept. of Electronics and Communication Engineering Narula Institute of Technology Kolkata,West Bengal

Abstract: ***This project focuses on the development of an IoT-based Smart Home Automation System aimed at enhancing user convenience, energy efficiency, and safety within residential environments. The system integrates multiple sensors and an ESP32 microcontroller to monitor and control household appliances intelligently. It supports both manual and automatic modes, enabling users to operate devices through a web-based interface or allowing automated control based on real-time environmental data such as temperature, motion, and gas levels. Core components include the DHT11 temperature sensor, PIR motion sensor, MQ2 gas sensor, and real-time databases like Firebase for device state management and MongoDB for historical logging. The system architecture ensures seamless interaction between hardware and software, offering a scalable, low-cost, and efficient solution for modern smart home applications.***

Keywords: Firebase,DHT11,Web Interface, Manual and Auto Mode, PIR Sensor, Internet of Things (IoT)

I. INTRODUCTION

In recent years, the concept of smart homes has rapidly evolved, driven by advancements in Internet of Things (IoT) technologies. A smart home integrates various electronic devices and sensors into a connected network, enabling remote monitoring, automation, and control of home appliances to enhance comfort, security, and energy efficiency. The growing demand for automation and the widespread availability of affordable microcontrollers and cloud platforms have made it feasible to implement smart systems even in residential settings. This project presents the development of a low-cost, IoT-based smart home automation system. The system is capable of operating in two distinct modes: manual and automatic. In manual mode, users can control home devices such as lights, fans, air conditioners, and sockets through a user-friendly web interface. In automatic mode, the system relies on environmental data from sensors—including the DHT11 temperature sensor, PIR motion sensor, and MQ2 gas sensor—to autonomously control devices based on pre-defined thresholds. The core of the system is built around the ESP32 microcontroller, which communicates Firebase Realtime Database for current device states and MongoDB for historical data logging. This architecture ensures real-time responsiveness as well as persistent data storage for analytics and review. By combining wireless control, sensor feedback, and cloud connectivity, the proposed smart home solution aims to improve daily living through enhanced convenience, energy conservation, and safety.

II.LITERATURE REVIEW

First, The concept of smart home automation has gained significant momentum with the advent of the Internet of Things (IoT), enabling seamless interconnection between devices, sensors, and cloud-based services. Numerous research efforts have focused on developing efficient and cost-effective systems that improve the convenience, energy management, and safety of residential environments.

In one such study, a low-cost IoT-based smart home system was introduced using microcontrollers and wireless communication modules to automate home appliances via a web-based interface. The system demonstrated effective control over devices and contributed to energy savings through real-time monitoring. However, the approach lacked historical data analysis capabilities and did not support multiple operation modes, limiting its adaptability to changing environmental conditions.

Another project explored the use of the ESP8266 and ESP32 microcontrollers in smart home applications due to their low power consumption, built-in Wi-Fi, and support for cloud integration. These systems typically relied on temperature, motion, and gas sensors for environmental awareness, but many lacked proper data storage mechanisms for long-term usage analysis and user behaviour tracking.

Several researchers have also implemented Firebase as a backend platform for real-time database operations due to its scalability and low-latency performance. While Firebase handles instantaneous control and feedback well, it does not natively support robust historical data logging, which is essential for understanding device usage patterns. To address this gap, MongoDB has been used in conjunction with Firebase to store and visualize historical logs, allowing for deeper insights into system performance and user interaction.

©IJRASET: All Rights are Reserved

Furthermore, the integration of manual and automatic modes has been suggested in various smart home systems, enabling users to switch between user-controlled and sensor-driven operations. This hybrid approach ensures flexibility, particularly in dynamic environments where full automation may not always be preferred.

In summary, previous works have laid a strong foundation for smart home automation using IoT technologies. However, the combination of real-time control, environmental sensing, multi-mode operation, and long-term data logging remains an area with room for improvement. This project aims to bridge these gaps by developing a comprehensive, sensor-driven, web-controlled system that leverages both Firebase and MongoDB for effective monitoring, control, and historical analysis.

III. ARCHITECTURE MODEL

The architecture of the smart home automation system is built on a layered model that integrates sensing hardware, cloud infrastructure, and a user interface. At the base of the system lies the sensing layer, which includes various sensors: the PIR sensor for detecting motion, the DHT11 for monitoring temperature and humidity, the LDR for light intensity detection, and the MQ2 sensor for gas or smoke detection. These sensors are connected to the ESP32 microcontroller, which forms the control layer. The ESP32 continuously reads sensor data and determines device actions based on predefined thresholds or manual commands. The ESP32 communicates in real-time with the cloud layer, which consists of Firebase Realtime Database and MongoDB. Firebase serves as the central hub for real-time state synchronization between the ESP32 and the frontend application. It stores the current device states, sensor readings, and automation thresholds. Meanwhile, MongoDB is used for persistent logging of all devices ON/OFF events through a Node.js backend. On the topmost level is the application layer, which includes a React.js front end and a Node.js + Express backend. The React interface allows users to toggle between manual and automatic modes, control individual devices, and view historical usage data. The Node.js server handles API requests from the frontend and logs device activities to MongoDB for future analysis. This architecture enables seamless interaction between hardware and software, ensuring smart, responsive, and remotely accessible home automation.

IV. SYSTEM DESIGN

The system design of the smart home automation solution focuses on creating a modular, scalable, and efficient framework that integrates hardware, cloud services, and a user-friendly interface. The design is centred around two core operating modes—Manual Mode and Auto Mode—each serving distinct use cases.

A. Manual Mode

In Manual Mode, the user has full control over all connected appliances such as the light, fan, AC, and power socket. The user interface, built using React.js, allows toggling of these devices through intuitive controls. When a toggle action is performed, the React frontend updates the /manual_control path in Firebase. The system also ensures that the current mode is stored, and corresponding device states are pushed to /device_state. The ESP32 listens to /device_state and accordingly activates or deactivates the relays connected to the appliances.

B. Auto Mode

In Auto Mode, the ESP32 autonomously controls devices based on real-time sensor data and user-defined thresholds stored in Firebase under /auto_thresholds. For instance:

- The light is turned ON when motion is detected by the PIR sensor.
- The fan is triggered when the temperature from the DHT11 sensor exceeds a specified value.
- The AC is activated only during a defined time range, regardless of temperature.
- The power socket always remains under manual control, even in auto mode.

When the mode is switched to auto, the frontend updates the /device_state with values derived from the /auto_control path, except for the socket, which is always pulled from /manual_control.

C. Sensor-Based Operation Logic

Each sensor contributes to a specific part of the automation logic:

- PIR Sensor: If motion is detected, light is turned ON for a fixed duration.
- DHT11 Sensor: If temperature exceeds the threshold, the fan is turned ON.

- LDR Sensor: Can optionally be used to detect ambient lighting to improve light automation (e.g., don't turn ON light if it's already bright).
- MQ2 Sensor: Can be used for safety alerts but does not directly control any devices in the basic version.

The ESP32 fetches sensor data at regular intervals and updates Firebase with the latest readings. If in Auto Mode, it simultaneously checks whether automation rules are met and updates /auto_control. These updates are then reflected in /device_state.

D. Mode Switching Logic

The system remembers the last known state of each device in both manual and auto modes. When switching modes, the frontend logic ensures that the correct states are restored based on the selected mode. This guarantees a seamless transition and avoids unexpected behaviour.

V. HARDWARE COMPONENTS

A. ESP32

The ESP32 is a powerful microcontroller that plays a central role in modern IoT systems, including this smart home automation project. It features dual-core processing, integrated Wi-Fi, and Bluetooth capabilities, making it ideal for applications that require wireless communication. With numerous GPIO (General Purpose Input/Output) pins, the ESP32 allows simultaneous interfacing with various sensors and output devices. In this project, it acts as the brain of the system—reading inputs from sensors like the PIR, DHT11, LDR, and MQ2, making logical decisions based on programmed conditions, and controlling output devices via a relay module. The ESP32 connects to the Firebase Realtime Database over Wi-Fi, enabling real-time synchronization between the hardware and the user interface built with React. It continuously monitors for any change in the /device_state node and acts accordingly to control lights, fans, AC, and other connected appliances. Its low power consumption, high processing speed, and open-source development environment (Arduino IDE) make it ideal for reliable and responsive smart automation. Furthermore, its compact size and cost-effectiveness allow it to be embedded into space-constrained environments. Overall, the ESP32 provides computational power, flexibility, and connectivity needed to implement a full-featured, real-time IoT-based smart home control system.



B. PIR Sensor

The PIR (Passive Infrared) sensor is a key motion detection component used in smart home systems to automate lighting and enhance security. It works by detecting infrared radiation emitted by warm objects, primarily human bodies. When a person moves within its sensing range, the PIR sensor outputs a digital HIGH signal, which is read by the ESP32. In this smart home project, the PIR sensor is used in auto mode to detect occupancy in a room. When motion is detected, the system automatically turns on the light, and after a certain period of inactivity, it turns the light off. This helps conserve energy and adds convenience for users by eliminating the need for manual switches. The sensor has a wide detection range, usually around 5–7 meters, and a detection angle of about 120 degrees, making it suitable for covering large indoor areas. It operates on low power and is easy to integrate with microcontrollers. By using a PIR sensor, the smart system becomes more intelligent, responding to real-time human presence, reducing unnecessary power consumption, and enhancing user comfort and safety in environments like homes, offices, or hospitals.



C. MQ2 Gas Sensor

The MQ2 gas sensor is a widely used sensor for detecting gas leaks and smoke in safety-critical systems such as smart homes. It can detect a variety of gases including methane, butane, LPG, propane, hydrogen, alcohol, and smoke. The sensor consists of a sensitive material that changes resistance when exposed to gas; this resistance is converted into an analog signal that can be read by ESP32. In this project, the MQ2 plays a vital role in enhancing home safety. When the sensor detects a gas concentration above a specified threshold, it can trigger alerts or activate ventilation systems through relays. The MQ2 sensor requires a short pre-heating time and can operate in a wide range of temperatures and humidity conditions. It is mounted in living or kitchen areas where gas leaks may occur and is interfaced to the ESP32's analog input pin. The values are read periodically and sent to Firebase, where they can also be logged or used to notify users via the frontend. Its sensitivity and quick response make it an ideal solution for early warning systems, preventing accidents due to gas leaks and contributing to the overall safety of the smart home.



D. DHT11 Sensor

The DHT11 sensor is a temperature and humidity sensor commonly used in environmental monitoring systems. It consists of a thermistor and a capacitive humidity sensor, along with an internal ADC (Analog-to-Digital Converter) that transmits data to the microcontroller in digital form. DHT11 provides temperature readings in degrees Celsius and relative humidity in percentage, with acceptable accuracy for non-critical applications like smart homes. In this project, DHT11 is used to collect environmental data that influences automation logic. For example, if the temperature exceeds a user-defined threshold stored in Firebase, the system automatically turns ON the fan or AC. The sensor outputs new data every 1–2 seconds and operates at 3.3V or 5V, making it compatible with the ESP32. The readings are displayed in the React frontend for user awareness and can be used to improve comfort and energy efficiency. Although it has a slightly lower range and accuracy compared to more advanced sensors (like DHT22), it is cost-effective, easy to use, and reliable for general-purpose indoor applications. DHT11 enhances the intelligent behavior of the system by enabling climate-responsive automation.



ISSN: 2321-9653; IC Value: 45.98; SJ Impact Factor: 7.538
Volume 13 Issue V May 2025- Available at www.ijraset.com

E. LDR (Light Dependent Resistor)

The LDR (Light Dependent Resistor) is a photoresistor used to detect the intensity of ambient light. Its resistance varies with the amount of light falling on it—decreasing in bright light and increasing in darkness. This change in resistance is converted into an analog voltage signal, which is read by the ESP32 to assess the brightness level of a room. In this smart home project, the LDR is used optionally in auto mode to determine whether artificial lighting is needed. For instance, if the ambient light is below a certain threshold and motion is detected by the PIR sensor, the system can automatically turn on the lights. This dual-condition logic improves energy efficiency and user experience. The LDR is a passive component, small in size, inexpensive, and requires no external power, making it suitable for continuous monitoring. It is often used in combination with other sensors to enhance system decision-making. By integrating the LDR, the system ensures that lights are used only when necessary, avoiding power wastage during daylight hours and supporting the automation goals of energy conservation and intelligent control.



F. Led Lights

The Light Emitting Diode (LED) serves as an output indicator, illuminating in response to system states or environmental conditions. It emits light when a current passes through a semiconductor junction, and is favored for its efficiency, low energy consumption, and long operational lifespan.



G. Connecting Wires

Connecting wires, including male-to-male, male-to-female, and female-to-female jumper wires, are essential for establishing electrical connections between various hardware components in the smart home system. These wires are used to connect sensors like PIR, DHT11, MQ2, LDR, and LEDs to the ESP32 microcontroller and the relay module. They provide flexible, reliable, and easy-to-modify links, especially during prototyping on breadboards or modular PCB setups. In this project, jumper wires are used to transmit data, power (3.3V/5V), and ground signals across components. Color-coded wires help differentiate signal lines and reduce connection errors. High-quality wires with tight-fitting headers ensure stable contact, which is crucial for consistent sensor readings and device control. Without proper wiring, the system would face loose connections, short circuits, or signal noise, leading to faulty automation behaviour. The modularity provided by jumper wires also allows for quick testing, replacement, or expansion of hardware features. Overall, connecting wires may seem minor, but they are the backbone of the hardware setup, enabling the entire system to function smoothly by ensuring efficient communication and power distribution between all the electronics involved.

©IJRASET: All Rights are Reserved



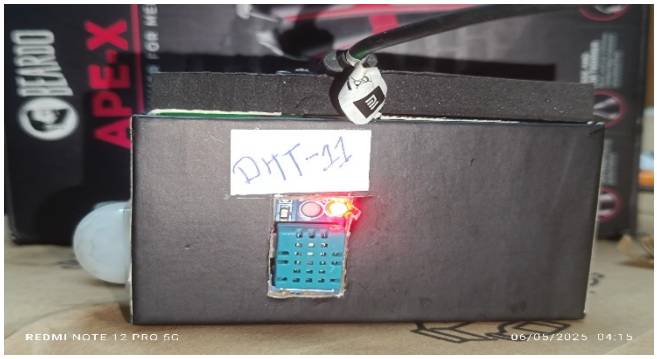
VI. SOFTWARE REQUIREMENTS

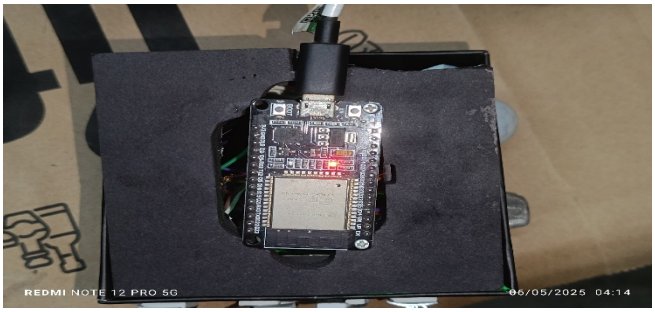
The smart home automation system is supported by a robust software stack that enables real-time data processing, user interaction, cloud communication, and device logging. At the microcontroller level, the Arduino IDE is used to program the ESP32 board. The ESP32 code is written in C/C++ and uses libraries for Wi-Fi connectivity, Firebase integration, and sensor interfacing. This code handles reading sensor values, controlling relays, and synchronizing device states with Firebase. For cloud-based data storage and real-time communication, the system relies on Firebase Realtime Database.

Firebase stores the current states of devices, sensor values, automation thresholds, and mode settings. It allows seamless synchronization between the ESP32 and the frontend, enabling instant updates and control from any internet-connected device. On the frontend side, the user interface is built using React.js, a modern JavaScript library for building interactive UIs. The React app displays real-time device statuses, offers manual controls, allows mode switching, and provides access to device history. The frontend communicates with Firebase to read and write data and uses API calls to log events. The backend is developed using Node.js with the Express.js framework. This server handles API routes and acts as a bridge between the React frontend and the MongoDB database. It receives requests from the frontend to log device actions and stores them in the database for future reference. For persistent storage of historical data, the system uses MongoDB, a NoSQL database that stores logs of device ON/OFF events with timestamps.he user interface is developed using React.js, offering an interactive and responsive dashboard that displays real-time statuses, allows manual device control, facilitates switching between manual and auto modes, and provides access to device usage history. This data can later be visualized to understand device usage patterns.Communication between the frontend and the backend is handled via API calls, with the backend built on Node.js using the Express.js framework. This backend processes logging requests from the frontend and stores them in MongoDB, a scalable NoSQL database, which holds historical ON/OFF events with timestamps for analysis and visualization of usage patterns. This full-stack integration ensures that users can control and monitor their smart home environment from anywhere with precision, reliability, and ease.

VII.HARDWARE





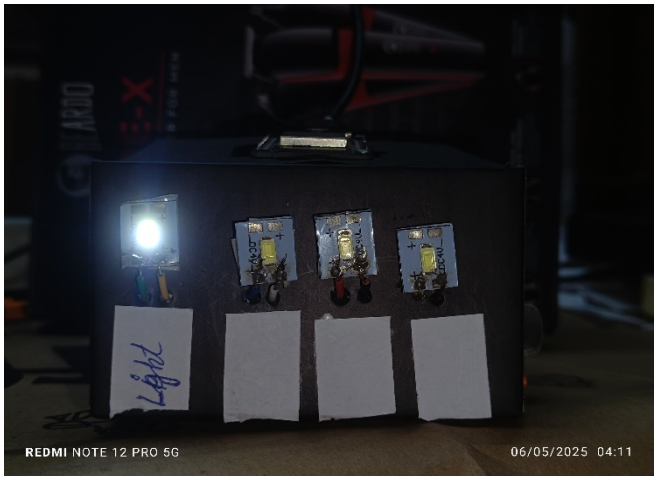






©IJRASET: All Rights are Reserved



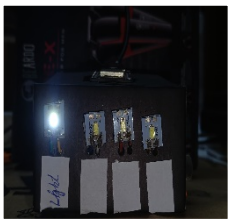
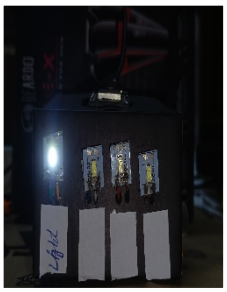
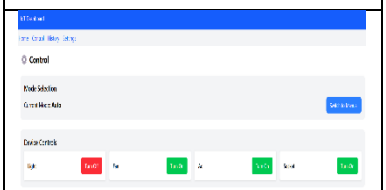


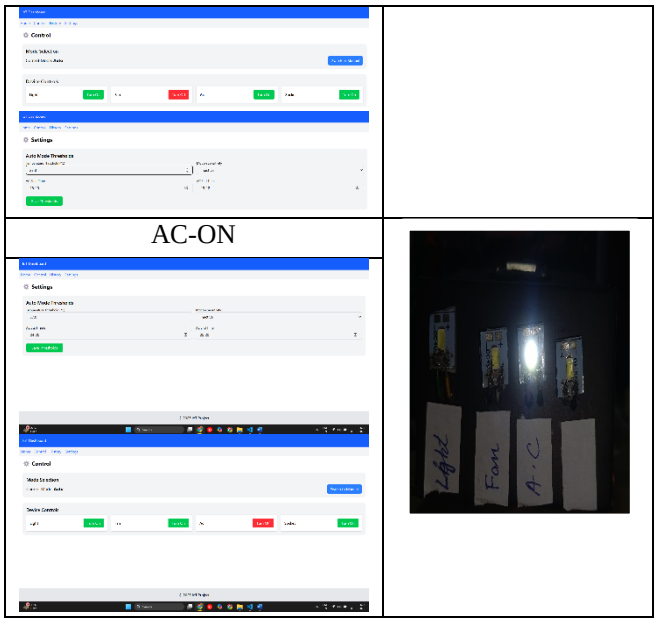
VIII. IMPLEMENTATION

The implementation of the smart home automation system involved integrating hardware components, cloud services, and software interfaces to ensure smooth, real-time operation. The process began with Firebase integration, where a structured Realtime Database was set up to manage different data nodes such as /device_state, /manual_control, /auto_control, /auto_thresholds, and sensor readings. This hierarchical structure ensures that all components—frontend, backend, and ESP32—interact with the correct segments of the database. On the hardware side, the ESP32 microcontroller was programmed using the Arduino IDE. The ESP32 was configured to connect to Wi-Fi, read data from sensors (PIR, DHT11, LDR, MQ2), and control relays connected to home appliances. It continuously updates sensor data in Firebase and listens for changes in /device_state to toggle devices accordingly. Logic was also implemented to distinguish between manual and auto modes. In auto mode, the ESP32 compares sensor readings against thresholds from Firebase and updates /auto_control, which the frontend uses to sync /device_state. The React.js frontend was developed to offer a user-friendly interface where users can toggle individual devices, switch between modes, and set automation thresholds. The interface also reflects real-time device states and provides feedback to users. A dedicated Settings page allows customization of automation behavior, while the Control page offers manual device control. For event logging, the system includes a Node.js + Express backend. When a user toggles a device from the React frontend, a POST request is sent to the backbend's /api/logs endpoint. The backend processes the request and stores the event in a MongoDB database, including details like device name, action (ON/OFF), and timestamp. This data is later retrieved on the History page, allowing users to view past activity and usage trends. The seamless switching between manual and auto modes was achieved by maintaining separate control paths in Firebase and updating /device_state appropriately during mode transitions. The socket device remains under manual control even in auto mode, ensuring safe and predictable behavior. Overall, the implementation successfully combines hardware control, real-time cloud sync, historical logging, and a responsive UI to create a fully functional smart home system.

©IJRASET: All Rights are Reserved | SJ Impact Factor 7.538 | ISRA Journal Impact Factor 7.894 |

IX. RESULTS

MODE = MANUAL	
LIGHT- ON, FAN-OFF,AC-OFF,SOCKET-OFF	
LIGHT- OFF, FAN-ON,AC-OFF,SOCKET-OFF	
LIGHT- OFF, FAN-OFF,AC-ON,SOCKET-OFF	
LIGHT- OFF, FAN-OFF,AC-OFF,SOCKET-ON	
MODE = AUTO	
LIGHT- ON	
	
FAN-ON	



X. ADVANTAGES

The implemented smart home automation system offers several significant advantages that enhance both convenience and efficiency in modern living environments. One of the primary benefits is energy saving. By automating devices based on real-time sensor data—such as turning off lights when no motion is detected or activating the fan only when the temperature exceeds a set threshold, the system reduces unnecessary power consumption and contributes to lower electricity bills. Another key advantage is enhanced safety. The integration of the MQ2 gas sensor provides early detection of gas leaks or smoke, helping to prevent potential fire hazards. Additionally, the system maintains constant monitoring through cloud services, ensuring that critical data is accessible at any time. The system also provides remote accessibility. Users can control and monitor their home appliances from anywhere using the React-based web interface, as long as they have an internet connection. This is particularly beneficial for users who travel frequently or want to manage their home while away. Furthermore, the system supports smart automation features.

The ability to switch between manual and auto modes provides flexibility, while automation thresholds allow for personalized, intelligent behavior. Devices operate autonomously based on sensor readings and user-defined preferences, offering both comfort and hands-free operation. Together, these features make the system a practical, scalable, and future-ready solution for modern home automation needs.

XI. APPLICATIONS

The smart home automation system developed in this project has wide-ranging applications across various environments that demand convenience, efficiency, and safety. One of the primary applications is in residential homes, where homeowners can automate lighting, climate control, and appliance usage to enhance comfort and reduce energy bills. The system's ability to detect motion, monitor temperature, and provide remote access makes it ideal for daily household management. In offices, the system can be used to automate lighting and air conditioning based on occupancy and working hours, thereby improving energy efficiency and reducing manual intervention. It also enhances workplace safety with gas detection capabilities. The system is also highly beneficial in hospitals, where automation can assist in maintaining ideal environmental conditions in patient rooms, reduce physical workload on staff, and improve emergency responsiveness through sensor-based alerts. In industrial settings, the automation logic can be expanded to monitor safety parameters like gas leaks and temperature levels, providing early warnings and automating necessary equipment in response. The ability to log device usage also supports maintenance scheduling and operational auditing. Overall, the system's flexibility, scalability, and sensor-based intelligence make it suitable for a wide range of smart infrastructure applications beyond just homes.

XII. FUTURE ENHANCEMENTS

While the current smart home automation system offers a robust set of features, several future enhancements can significantly improve its functionality, user experience, and intelligence. One major improvement would be the integration of voice control using platforms like Google Assistant, Amazon Alexa, or Apple Siri. This would allow users to control devices through simple voice commands, offering a more intuitive and hands-free interaction method. Another key enhancement would be the development of a mobile app version of the user interface. Although the existing React-based web dashboard is responsive, a dedicated mobile application for Android and iOS would provide better accessibility, push notifications, and native features like background operation and location-based automation. Additionally, incorporating AI-based predictive automation could make the system smarter over time. By analyzing historical usage patterns and environmental data, the system could predict user behavior and adjust device control accordingly—such as pre-cooling a room before arrival or adjusting lighting based on learned habits. Other potential upgrades include integration with smart security systems (CCTV, door locks), energy consumption tracking through smart meters, and support for multiple user profiles. These enhancements would help in transforming the system into a more intelligent, adaptive, and comprehensive smart home ecosystem.

XIII. CONCLUSION

The smart home automation system developed in this project successfully demonstrates the integration of IoT, cloud computing, and web technologies to create a responsive, intelligent, and user-friendly solution for modern living environments. By combining real-time sensor data, Firebase Realtime Database, and a React-based control interface, the system provides seamless control and monitoring of household devices. The dual-mode operation—manual and auto—offers flexibility to users, while the incorporation of logging through MongoDB enables detailed tracking of device usage. The implementation ensures energy efficiency, safety, and remote accessibility, addressing the core challenges faced in home automation. Furthermore, the modular design allows for scalability and future enhancements such as voice control, mobile app development, and AI-driven automation. Overall, this project lays a strong foundation for building advanced smart home ecosystems that are not only efficient and secure but also adaptable to evolving user needs and technological advancements.

XIV. ACKNOWLEDGMENT

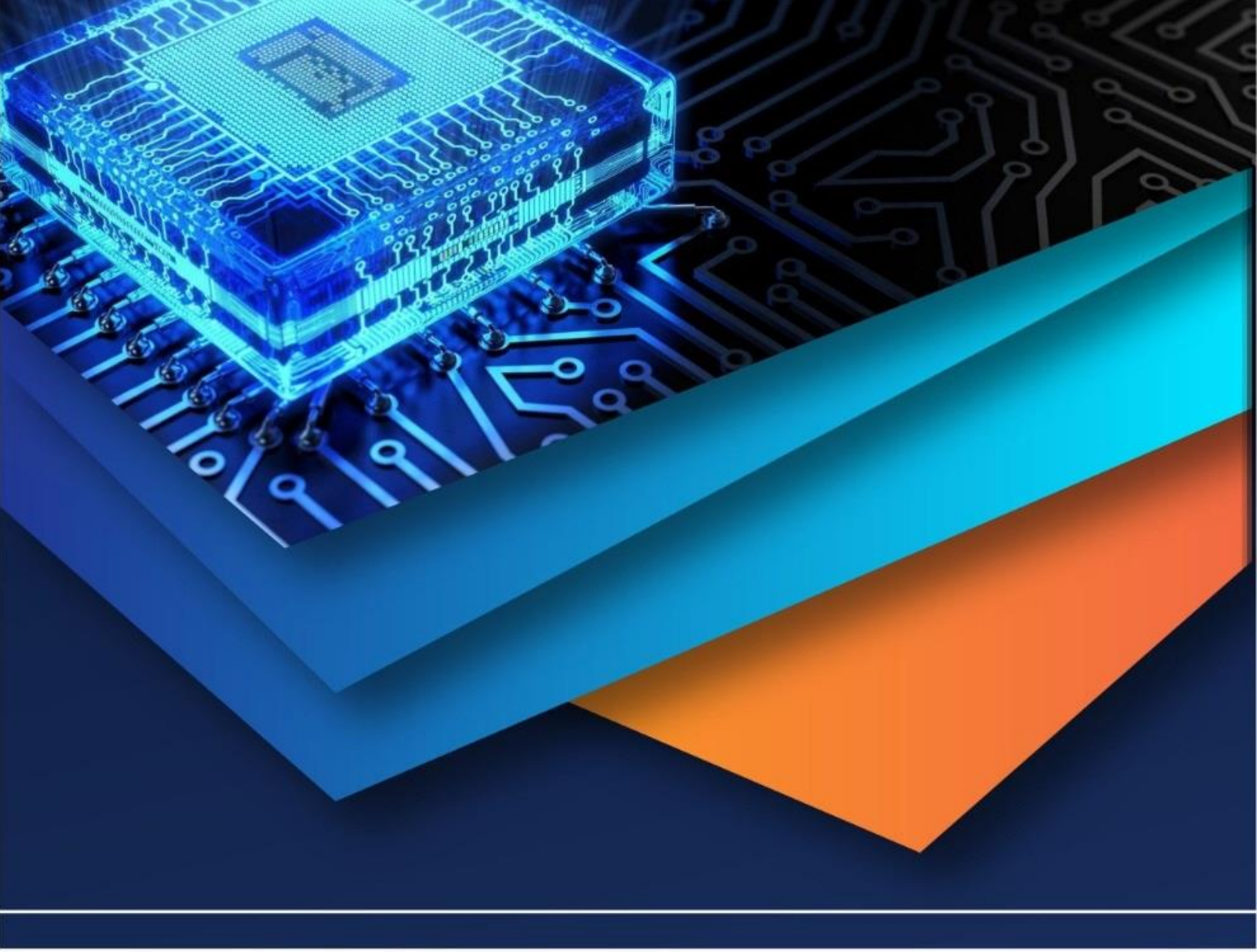
We would like to express our sincere gratitude to everyone who supported and guided us throughout the development of this smart home automation project. First and foremost, we thank our project guide, Prof. Arpita Santra, for her continuous support, valuable feedback, and insightful suggestions that greatly enhanced the quality of our work. We are also thankful to the faculty and staff of the E.C.E, Narula Institute of Technology for providing the necessary resources and a conducive environment to carry out this project. We extend our appreciation to our peers and friends who offered encouragement and assistance whenever required.

Special thanks to our families for their unwavering support and motivation during the entire course of this project. Finally, we are grateful for the availability of open-source tools and online resources that played a vital role in the implementation and testing of our smart home system.

REFERENCES

[1] Evans, C., & Martinez, D. (2023). "Smart Home Automation using ESP32, Rainmaker, Alexa, and Google Assistant." IEEE Transactions on Consumer Electronics, 69(2), 234-243.
[2] Foster, J., & Lee, S. (2020). "A Comparative Study of ESP32 and Raspberry Pi for Home Automation." International Journal of Advanced Computer Science and Applications, 11(5), 201-208.
[3] Garcia, R., & Adams, M. (2019). "Integration of ESP32 with Alexa for Smart Home Applications." Journal of Intelligent Systems, 25(3), 123-131.
[4] Hernandez, L., & White, R. (2021). "ESP32-based Home Automation System with Rainmaker and Google Assistant Integration." International Journal of Engineering and Technology, 13(2), 98-105.
[5] Hill, S., & Garcia, R. (2023). "Voice-Controlled Home Automation System using ESP32 and Alexa with Manual Switching." International Journal of Smart Home, 17(1), 67-75.

©IJRASET: All Rights are Reserved



10.22214/IJRASET

45.98

IMPACT FACTOR: 7.129

IMPACT FACTOR: 7.429

INTERNATIONAL JOURNAL FOR RESEARCH

IN APPLIED SCIENCE & ENGINEERING TECHNOLOGY

Call : 08813907089 (24*7 Support on Whatsapp)