



Smart Home Control System: A Web-Based IoT Platform for Remote Monitoring and Appliance Automation

Dr. Premavathi C¹, Gowrisubha M², Kowshika Sri S³, Supaanu A K⁴, Lafena P⁵

¹Associate Professor, Department of Computer Applications, Vellalar College for Women, Tamil Nadu, India

²Master of Computer Applications, Department of Computer Applications, Vellalar College for Women, Tamil Nadu, India

³Master of Computer Applications, Department of Computer Applications, Vellalar College for Women, Tamil Nadu, India

⁴Master of Computer Applications, Department of Computer Applications, Vellalar College for Women, Tamil Nadu, India ⁵Master of Computer Applications, Department of Computer Applications, Vellalar College for Women, Tamil Nadu, India

ABSTRACT

Home appliances are still operated mainly through manual switches, which makes it difficult to control devices remotely, notice unsafe conditions such as gas leakage, or keep a record of device usage. The Smart Home Control System is a web-based Internet of Things application developed using an ESP32 microcontroller, relay modules, environmental sensors, Flask, Jinja2, SQLite and Bootstrap to provide a centralized platform for monitoring and controlling household devices. The system supports user registration and login, switching of connected appliances from a web dashboard, live display of temperature, humidity, motion and gas-sensor readings, automatic alerts when a safety threshold is crossed, scheduled device operation and an activity log of every switching event. An administrator module is provided for managing users, devices and rooms and for generating Excel reports of device activity. The application also provides a structured record of sensor readings and device usage. By bringing these functions into a single low-cost portal, the system reduces dependence on physical switches and provides a structured way to observe and manage the home environment.

Keywords: Smart Home, Internet of Things, ESP32, Flask, Home Automation, Sensor Monitoring, SQLite

1. Introduction

Home automation aims to make everyday living safer, more comfortable and more energy efficient by allowing household devices to be operated and monitored electronically. In most homes, however, lights, fans and other appliances are still controlled only by physical switches, and the occupant has no simple way to check or change their state when away from the room or from the house. The growth of low-cost Wi-Fi enabled microcontrollers has made it practical to connect such appliances to a network and to control them through a web browser [I], [II], [III].

The proposed Smart Home Control System is designed as a web-based system for home users and administrators. Users can create an account, log in, switch connected appliances on or off, view live sensor readings, define operating schedules and review the history of device activity. Administrators can manage users, rooms and devices and generate downloadable reports. The system combines an ESP32 controller at the hardware level with a Flask web application at the software level.

1.1 Problem Description

The major problem addressed by the system is the lack of a single, remotely accessible point of control for household devices. A person who leaves home cannot confirm whether a light or fan has been left switched on, cannot be warned quickly about gas leakage or unexpected motion, and has no record of when devices were used. Commercial smart-home products exist, but many depend on proprietary applications and cloud services and are costly for ordinary households. The portal addresses these issues through centralized control, real-time monitoring, automatic alerts and role-based access.

1.2 Objectives

- Provide a centralized web interface for controlling household appliances.
- Allow users to register and securely access their dashboard.
- Display live temperature, humidity, motion and gas-sensor readings.
- Generate an alert when a sensor reading crosses a safety threshold.

- Support manual, scheduled and sensor-triggered switching of devices.
- Store every switching event and sensor reading in a database for later review.
- Provide administrators with user, room and device management facilities.
- Generate structured reports of device activity for administrative use.

1.3 System Specification

Hardware Requirements: A computer or laptop capable of running Python and a modern web browser is required to host the web application. The device unit uses an ESP32 development board, four-channel relay module, DHT11 temperature and humidity sensor, PIR motion sensor, MQ-2 gas sensor, buzzer, 5 V power supply and a Wi-Fi router.

Software Requirements: Windows or another supported operating system, Python, Flask, Jinja2 templates, SQLite database, Bootstrap for interface styling, HTML/CSS/JavaScript, Arduino IDE for firmware development and OpenPyXL for Excel report generation.

2. System Analysis

2.1 Existing System

In a conventional home, appliances are operated only through wall-mounted switches. The occupant must be physically present to change the state of a device, and no information is stored about when appliances were used or whether environmental conditions were unsafe. Ready-made smart plugs and hubs offer remote control but usually work only within a closed application and do not provide a customizable dashboard, local data storage or reporting [VII], [IX], [X].

2.2 Proposed System

The proposed system combines device control and environmental monitoring in one application. The ESP32 reads the connected sensors, drives the relays and communicates with the Flask server over the local Wi-Fi network. The database stores users, devices, sensor readings and activity logs. The user interface presents the current state of every device and provides a switching action for each one, while the administrator interface supports data maintenance and reporting [IV], [VIII].

2.3 Module Description

- User Registration and Login Module – creates and authenticates user accounts.
- Dashboard Module – provides a central area for accessing all control and monitoring functions.
- Device Control Module – switches relay-connected appliances on or off and shows their present state.
- Sensor Monitoring Module – receives and displays temperature, humidity, motion and gas readings.
- Alert Module – raises a dashboard alert and sounds the buzzer when a threshold is exceeded.
- Scheduling Module – switches selected devices automatically at the times set by the user.
- Activity Log Module – stores each switching event and sensor reading with date and time.
- Admin Module – allows administrators to manage users, rooms and devices.
- Report Module – exports activity details to Excel for review.

3. System Design

3.1 System Architecture

The system follows a client-server web architecture with an embedded device layer. The browser provides the user interface, Flask handles routing and application logic, Jinja2 renders dynamic HTML pages and SQLite stores application data. The ESP32 connects to the same Wi-Fi network, sends sensor readings to the server through HTTP requests and periodically requests the latest device commands. The relay module isolates the low-voltage controller from the appliance circuit [V], [VI].

Layer	Technology	Purpose
Presentation	HTML, CSS, Bootstrap, Jinja2	User interface and dynamic pages
Application	Python, Flask	Routes, authentication and business logic
Data	SQLite	Users, devices, readings and logs
Device	ESP32, relays, sensors	Sensing and appliance switching
Reporting	OpenPyXL	Excel report generation

Table 1 – Major system layers and technologies.

3.2 Database Design

The system uses four main entities. The User entity stores login and role information. The Device entity stores each appliance, its room and its present state. The Sensor Reading entity stores the values received from the controller with a timestamp. The Activity Log entity stores every switching event together with the user or rule that caused it.

Entity	Important Fields	Purpose
User	id, username, password, is_admin	Authentication and role management
Device	id, name, room, relay_pin, state, schedule_on, schedule_off	Appliance information and state
Sensor Reading	id, temperature, humidity, motion, gas_level, recorded_at	Environmental monitoring data
Activity Log	id, device_id, user_id, action, logged_at	History of switching events

Table 2 – Main database entities.

3.3 Hardware Design

The hardware unit is built around the ESP32 development board, which reads the sensors through its input pins and drives the relay channels through its output pins. The DHT11 sensor is connected to a digital pin, the PIR sensor supplies a digital motion signal and the MQ-2 gas sensor is read through an analog input. Each relay channel switches one appliance, and the buzzer is driven directly by a digital output. The board is powered from a regulated 5 V supply and connects to the home Wi-Fi network.

3.4 System Flow

User → Registration/Login → Dashboard → Select Device → Switch On/Off → Flask Server → ESP32 → Relay → Appliance. In parallel, sensors send readings through the ESP32 to the Flask server, which stores them in the database and shows them on the dashboard. Administrator → Login → Admin Dashboard → Manage Users / Devices → Generate Excel Report.

4. Implementation

4.1 User Interface

The interface is implemented with HTML templates, CSS and Bootstrap components. Jinja2 is used to insert dynamic database values into the pages. The design provides navigation for authentication, dashboard, device control, sensor monitoring, schedules and administrative functions.

4.2 Device Control and Firmware

Each appliance is assigned to a relay channel on the ESP32. When the user selects a switch on the dashboard, the Flask server updates the device state in the database. The ESP32 fetches the latest states at short intervals and sets the corresponding relay pin, so that the physical appliance always matches the state shown on the dashboard.

4.3 Monitoring, Alerts and Reporting

The DHT11 sensor supplies temperature and humidity, the PIR sensor detects movement and the MQ-2 sensor indicates the gas level. The ESP32 sends these values to the server at regular intervals. When the gas level exceeds the configured threshold the buzzer is activated and a warning is shown on the dashboard. Administrators can download the activity log and sensor history as an Excel report.

4.4 Security and Access Control

Access to the application is protected by login. Every user account carries a role, and the administrator pages are available only to accounts marked as administrators. Device commands are accepted only from logged-in users, and each command is written to the activity log with the name of the user and the time. The ESP32 identifies itself to the server with a device key so that unknown devices cannot send readings.

5. Testing and Results

Test Case	Input	Expected Result	Status
Registration	Valid user details	Account created / validation shown	Pass
Login	Valid credentials	User dashboard opened	Pass
Device Switching	Toggle a device	Relay changes and state is updated	Pass
Sensor Display	Sensor readings	Latest values shown on dashboard	Pass
Gas Alert	Reading above threshold	Buzzer sounds and alert is shown	Pass
Activity Log	Switching event	Event stored with date and time	Pass
Admin Report	Activity records	Excel report generated	Pass

Table 3 – Representative functional test cases.

5.2 Results

The implemented modules provide a unified workflow for home control and monitoring. Users can switch appliances and observe the home environment from a single dashboard, while administrators can maintain devices and obtain structured records for review. The alert, scheduling and reporting features extend the system beyond simple remote switching.

6. Conclusion

The Smart Home Control System provides a centralized web-based approach to controlling household appliances and monitoring the home environment.

The ESP32 handles sensing and switching, Flask and Jinja2 provide the application framework and dynamic interface, and SQLite provides persistent storage. Automatic alerts, scheduling and activity logging give additional support for safety and energy awareness. The system can serve as a foundation for a more comprehensive smart home solution.

Furthermore, the combination of low-cost hardware with an open-source web framework shows that reliable home control and monitoring can be achieved without dependence on proprietary cloud services. By bridging the gap between physical appliances and browser-based management, the platform establishes a reliable, scalable blueprint for affordable home automation.

7. Future Enhancement

- Use of the MQTT protocol for faster and more reliable device communication.

- Voice control through popular voice assistants.
- Email or mobile push notifications for gas, motion and other alerts.
- Energy consumption monitoring for each connected appliance.
- Secure password hashing, HTTPS and stronger access control for production deployment.
- Cloud-based storage and remote access from outside the home network.
- Machine-learning based prediction of usage patterns for automatic scheduling.

Acknowledgements

The authors acknowledge the guidance and support provided by the Department of Computer Applications and the faculty members who supported the development and testing of the Smart Home Control System.

REFERENCES

1. Gubbi, J., Buyya, R., Marusic, S., & Palaniswami, M. (2013) **"Internet of Things (IoT): A vision, architectural elements, and future directions"** Future Generation Computer Systems, 29(7), 1645–1660.
2. Al-Fuqaha, A., Guizani, M., Mohammadi, M., Aledhari, M., & Ayyash, M. (2015) **"Internet of Things: A survey on enabling technologies, protocols, and applications"** IEEE Communications Surveys & Tutorials, 17(4), 2347–2376.
3. Piyare, R. (2013) **"Internet of Things: Ubiquitous home control and monitoring system using Android based smart phone"** International Journal of Internet of Things, 2(1), 5–11.
4. Stolojescu-Crisan, C., Crisan, C., & Butunoi, B.-P. (2021) **"An IoT-based smart home automation system"** Sensors, 21(11), 3784.
5. V.Grinberg, M. (2018) **"Flask Web Development: Developing Web Applications with Python (2nd ed.)"** O'Reilly Media.
6. Espressif Systems. (2023) **"ESP32 Series Datasheet"** Espressif Systems (Shanghai) Co., Ltd.
7. Kodali, R. K., Jain, V., Bose, S., & Boppana, L. (2016) **"IoT based smart security and home automation system"** Proceedings of the International Conference on Computing, Communication and Automation (ICCCA), 1286–1289.
8. Jabbar, W. A., Kian, T. K., Ramli, R. M., et al. (2019) **"Design and fabrication of smart home with Internet of Things enabled automation system"** IEEE Access, 7, 144059–144074.
9. Mocrii, D., Chen, Y., & Musilek, P. (2018) **"IoT-based smart homes: A review of system architecture, software, communications, privacy and security"** Internet of Things, 1–2, 81–98.
10. Alaa, M., Zaidan, A. A., Zaidan, B. B., Talal, M., & Kiah, M. L. M. (2017) **"A review of smart home applications based on Internet of Things"** Journal of Network and Computer Applications, 97, 48–65.