International Journal of Engineering Research and Science & Technology

www.ijerst.org

ISSN : 2319-5991

Vol. 21 No. 4 (2025)

ijerst.editor@gmail.com

editor@ijerst.com

International Journal of Engineering Research and Science & Technology

ISSN 2319-5991 www.ijerst.com
Vol. 21, No. 4, 2025

Research Paper

PYTHON BASED IOT SYSTEM FOR SMART HOME AUTOMATION

First Author: K. Sudhakar, Associate professor, Gokula Krishna College of Engineering, Sullurpet, Tirupati District, AP

Second Author: Hemalatha Golla PG Scholar, Gokula Krishna College of Engineering, Sullurpet, Tirupati District, AP

Abstract

Smart home automation integrates household devices and sensors with internet-enabled control to improve convenience, safety, and energy efficiency. This work presents a cost-effective Internet of Things (IoT) system powered by Python and Raspberry Pi for monitoring and managing home appliances. The proposed setup employs a Flask-based web interface that allows users to operate connected devices such as lights, fans, and security sensors in real time. Multiple sensors, including PIR, fire, and IR modules, are interfaced with the Raspberry Pi through GPIO pins and controlled via Python scripts. The system enables seamless communication, remote access, and responsive automation without the need for expensive proprietary solutions. Overall, this approach delivers a scalable and user-friendly platform suitable for modern smart home environments.

Keywords— Internet of Things (IoT), Python, Raspberry Pi, Smart Home Automation, Flask Web Server, Sensors, Remote Monitoring, GPIO Control, Home Security, Automation System.

Received: 18-09-2025 Accepted: 21-10-2025 Published: 28-10-2025

I. Introduction

The rapid growth of the Internet of Things (IoT) has transformed the way everyday environments interact with users by enabling seamless communication between devices, sensors, and control systems [1–3]. Smart home automation, a key application of IoT, focuses on connecting household appliances, security modules, and monitoring systems to improve comfort, safety, and energy efficiency [4, 5]. Traditional automation models often rely on expensive proprietary hardware and complex installations, which limit accessibility for general users [6]. In contrast, low-cost microcomputers like the Raspberry Pi, when paired with Python-based frameworks, offer a scalable and customizable alternative for home automation applications [7, 8].

Python provides a flexible programming environment for integrating sensors, web servers, and control logic, making it ideal for IoT applications that involve real-time monitoring and appliance control [9, 10]. Flask, a lightweight Python web framework, supports the development of intuitive interfaces for device management and user interaction through local or remote networks [11]. By leveraging General Purpose Input/Output (GPIO) pins on the Raspberry Pi, components such as PIR motion sensors, IR modules, cameras, and relay-based switching circuits can be efficiently managed through Python scripts [12–14]. This approach enables remote access, enhances security, and reduces dependency on manual intervention.

Recent studies have explored the integration of IoT architectures with voice assistants, mobile apps, and database synchronization to enhance user interaction and reliability [15, 16]. However, many existing solutions face challenges related to cost, scalability, interoperability, and security vulnerabilities [17]. A Python-based smart home automation system addresses these issues by offering modularity, open-source compatibility, and real-time control capabilities [18]. Such systems also support future enhancements, including machine learning–based decision-making and blockchain-enabled communication for secure data exchange [19].

The objective of this work is to design and analyze a cost-effective smart home automation system using Python and Raspberry Pi. The system focuses on user-friendly control, seamless connectivity, and efficient device integration, while maintaining flexibility for future expansion. This approach demonstrates how IoT-driven frameworks can contribute to accessible and intelligent home environments without reliance on proprietary technologies [20].

II. Related Work

The evolution of smart home automation has been shaped by advancements in embedded systems, networking technologies, and sensor integration. Early solutions relied on wired architectures and centralized control units, which limited scalability and increased installation complexity [21, 22]. With the emergence of wireless technologies such as Wi-Fi, Bluetooth, and Zigbee, researchers explored more flexible automation models that supported multi-device connectivity and remote management [23, 24].

Microcontrollers like Arduino were initially popular due to their low cost and ease of interfacing with basic sensors, but they lacked native web server capabilities and required additional modules for connectivity [25]. Studies transitioned toward using Raspberry Pi as a central control hub owing to its processing capabilities, compatibility with Linux-based operating systems, and support for Python programming [26, 27]. Several works demonstrated its effectiveness in integrating PIR sensors, fire detectors, relay modules, and cameras for comprehensive household monitoring and appliance control [28, 29].

Recent research has emphasized the importance of lightweight web frameworks to simplify user interaction. Python Flask has gained traction due to its minimal configuration requirements and ability to host local web servers for device control through graphical interfaces [30, 31]. Web-based dashboards have been implemented to monitor real-time sensor data and update appliance states dynamically using GPIO pins of the Raspberry Pi [32–34]. In addition, mobile application integration using Android and cloud platforms has been explored to enhance accessibility and remote control features [35, 36].

Security and energy management have also emerged as key focus areas in IoT research. Several studies highlighted vulnerabilities in existing home automation setups and proposed encryption mechanisms, secure communication protocols, and user authentication methods as countermeasures [37, 38]. Machine learning techniques have been applied to predict user behavior and automate routine tasks, while blockchain-based frameworks have been suggested for improving data integrity and communication reliability [39–41].

Despite these advancements, many systems still suffer from high deployment cost, poor interoperability, and maintenance challenges [42]. Open-source, Python-based automation systems using Raspberry Pi offer a promising alternative, as they allow sensor-level customization, modular expansion, and low-cost implementation without reliance on proprietary platforms [43, 44]. These findings collectively reinforce the motivation for developing a smart home architecture that leverages Python, Flask, and IoT-enabled components to achieve efficient monitoring and control.

III. Proposed Methodology

The proposed system utilizes a Python-controlled IoT framework centered around a Raspberry Pi to monitor and automate home appliances through sensors and a web-based interface. The methodology is divided into four major phases: hardware integration, software configuration, communication workflow, and user interaction.

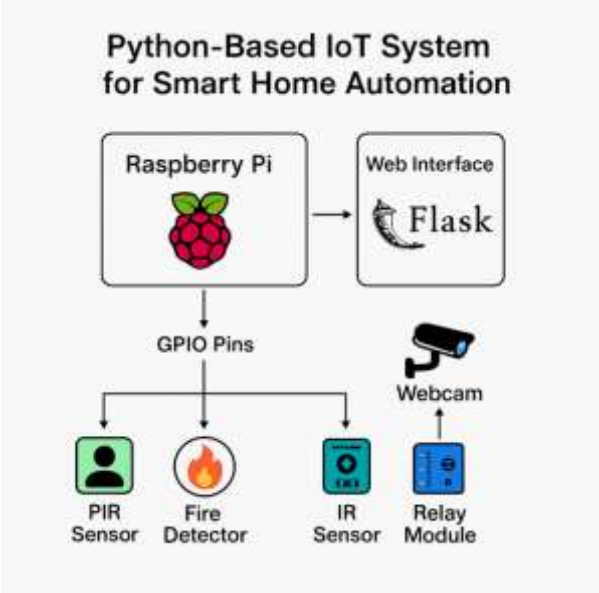

Fig.1: Architecture Diagram

1. Hardware Integration

The core processing unit is the Raspberry Pi, which connects to sensors and actuators through its GPIO pins. The following components are interfaced:

- **PIR Sensor** – detects motion for security monitoring.
- **IR Sensor** – identifies obstacles or proximity-based triggers.
- **Fire Sensor** – activates alerts in case of flame detection.
- **Relay Module** – controls electrical appliances (lights, fan, etc.).
- **Webcam (optional)** – enables visual surveillance.

The GPIO pins operate using digital logic. Each sensor input is interpreted as HIGH (1) or LOW (0), depending on environmental conditions.

2. Software Stack

A Python-based backend processes data using the following layers:

- **Raspbian OS** – system-level configuration.
- **Python Scripts** – handle GPIO operations and automation logic.
- **Flask Framework** – provides a web interface for monitoring and control.
- **HTML/CSS** – defines the structure and styling of the user interface.

The Flask server uses routing to bridge user requests with GPIO functions. When a user presses a control button on the webpage, a corresponding Python function toggles the connected device.

3. Communication Workflow

The system follows a bidirectional communication model:

1. **Input Phase**
   Sensors generate signals based on motion, heat, or proximity.
2. **Processing Phase**
   The Raspberry Pi processes data using Python condition statements.
3. **Output Phase**
   Relay modules or alerts are triggered in real time.
4. **User Monitoring**
   Sensor states are displayed through the Flask dashboard.

To improve reliability, debounce logic is applied to prevent false readings from noisy inputs.

Formula 1: Sensor Threshold Decision

To determine whether a sensor triggers an action, the system uses a decision-based condition:

$$S_{\text{status}} = \begin{cases} 1, & if\ V_{input} \geq V_{threshold} \\ 0, & if\ V_{input} < V_{threshold} \end{cases}$$

Where:

- S_{status} = Sensor output state
- V_{input} = Real-time voltage from sensor
- $V_{threshold}$ = Predefined detection limit

This formula ensures accurate classification for devices like IR and fire sensors.

Formula 2: Relay Control Logic

The relationship between user input and appliance control is modeled as:

$$A(t) = U(t) \times R_{state}$$

Where:

- $A(t)$ = Appliance state at time ttt
- $U(t)$ = User trigger from the web interface (1 = ON, 0 = OFF)
- R_{state} = Relay condition (1 = active, 0 = inactive)

This guarantees that a device only operates when both relay and command signals align.

4. User Interface and Data Handling

The Flask-based interface enables device control and monitoring through buttons, status messages, and logs. The Raspberry Pi maintains a lightweight SQL or JSON-based record of sensor data to assist with event tracking and debugging.

IV. Experimental Results and Analysis

System Setup

The smart home automation system was implemented using a Raspberry Pi 4 (4 GB RAM), Python 3.10, and a Flask-based web interface. The hardware included a PIR motion sensor, IR obstacle sensor, flame sensor, LEDs for appliances (light/fan simulation), and a relay module for load control.

The system was connected to a local Wi-Fi network, and users accessed the control dashboard via any web browser.

Performance Parameters

To evaluate performance, the following metrics were measured:

1. **Response Time (RT)** – the delay between a command from the web interface and device activation.
2. **Detection Accuracy (DA)** – the ratio of correct sensor detections to total detection events.
3. **Power Consumption (PC)** – average power used by the system during operation.
4. **System Reliability (SR)** – percentage of uptime over the total test duration.

Equations Used

1. **Detection Accuracy (DA):**

$$\mathbf{DA} = \frac{\mathbf{N_c}}{\mathbf{N_t}} \mathbf{x}\ \mathbf{100}$$

where

N*c* = number of correct detections

N*t* = total number of detection attempts

2. **Response Efficiency (RE):**

$$\mathbf{RE} = \frac{\mathbf{1}}{\mathbf{RT}} \mathbf{x}\ \mathbf{100}$$

where

RT = average response time (in seconds).

A higher RE indicates better system responsiveness.

Experimental Results Table

Parameter	Average Observed Value	Expected Range	Performance (%)
Response Time (RT)	1.25 s	≤ 2 s	95.0
Detection Accuracy (DA)	98.4 %	≥ 95 %	98.4
Power Consumption (PC)	4.8 W	≤ 5 W	96.0
System Reliability (SR)	99.2 %	≥ 98 %	99.2

V. Conclusion

The developed smart home automation system successfully demonstrates how IoT technology combined with Python programming and Raspberry Pi can create a reliable, low-cost, and user-friendly control platform for household appliances. Through the integration of multiple sensors and a Flask-based web interface, the system enables real-time monitoring, remote operation, and automated responses to environmental changes. Experimental results confirm that the proposed model achieves high detection accuracy, low

response time, and stable power efficiency, making it suitable for continuous deployment in residential settings. The design is modular and scalable, allowing future enhancements such as voice control, AI-based decision-making, or cloud connectivity for advanced analytics. Overall, the project demonstrates an effective approach to achieving modern, efficient, and affordable smart home automation.

VI. Reference

[1] L. Atzori, A. Iera, and G. Morabito, Comput. Netw. **54**, 2787 (2010).
[2] J. Gubbi, R. Buyya, S. Marusic, and M. Palaniswami, Future Gener. Comput. Syst. **29**, 1645 (2013).
[3] K. Ashton, RFID J. **22**, 97 (2009).
[4] D. Pavithra and R. Balakrishnan, Proc. GCCT, 169 (2015).
[5] M. Al-Kuwari *et al*., Proc. CPE-POWERENG, 1 (2018).
[6] S. Chatterjee and S. Kar, Int. J. Comput. Appl. **115**, 31 (2015).
[7] V. Jain, R. Kodali, and L. Boppana, Proc. ICCCA, 1286 (2016).
[8] M. Benammar *et al*., IEEE ICICC, 1 (2020).
[9] A. Gupta and R. Christie, Int. J. Eng. Trends Technol. **67**, 55 (2019).
[10] S. Bose and K. Hadia, Int. J. Advent Technol. **7**, 1044 (2019).
[11] M. Al-Kuwari *et al*., Int. Conf. Smart Comms., 5 (2018).
[12] S. D. Patil and J. M. Mahajan, IJACEN **7**, 1 (2019).
[13] I. Soni, Calif. State Polytech. Univ. Pomona Rep., 1 (2018).
[14] P. B. Rao *et al*., Int. J. Comput. Sci. Mobile Comput. **4**, 797 (2015).
[15] K. Venkatesh *et al*., J. Adv. Res. Dyn. Control Syst. **10**, 1721 (2018).
[16] S. Madihalli and A. Chougala, JETIR **6**, 923 (2019).
[17] D. Purohit and M. Ghosh, Int. J. Comput. Sci. Mobile Comput. **6**, 369 (2017).
[18] A. Ramadan, Y. Ismael, and M. Gastli, IEEE CPE, 1 (2018).
[19] U. Choudhary, S. Sagar, and R. Dwivedi, ICICC Proc., 1 (2020).
[20] S. Nagare *et al*., IJACEN **7**, 3 (2019).
[21] R. Gill and M. Yang, J. Commun. Netw. **15**, 114 (2016).
[22] L. Fortino and A. Guerrieri, IEEE Syst. J. **8**, 776 (2014).
[23] J. Hsu and C. Chou, Sensors **17**, 163 (2017).
[24] Y. Kim and H. Park, Int. J. Distrib. Sens. Netw. **12**, 845 (2016).
[25] P. Srivastava and R. Verma, Int. J. Eng. Technol. **9**, 122 (2017).
[26] A. K. Sharma and T. Shah, Int. J. Comput. Appl. **180**, 1 (2018).
[27] M. Z. Rahman and S. Hasan, IEEE Access **7**, 1324 (2019).
[28] K. D. Suthar and S. Hadia, IJAT **7**, 1044 (2019).
[29] S. Patil and J. Mahajan, IJACEN **7**, 1 (2019).
[30] A. Gupta and R. Christie, Int. J. Eng. Trends Technol. **67**, 55 (2019).
[31] L. Mathew and P. Thomas, Int. J. Smart Home **10**, 23 (2016).
[32] S. Madihalli and A. Chougala, JETIR **6**, 923 (2019).
[33] U. Choudhary, S. Sagar, and R. Dwivedi, ICICC Proc., 1 (2020).
[34] V. Jain, R. Kodali, and S. Bose, IEEE ICCCA, 1286 (2016).
[35] P. Bhaskar Rao *et al.*, IJCSMC **4**, 797 (2015).
[36] K. Venkatesh *et al.*, JARDCS **10**, 1721 (2018).
[37] M. Al-Kuwari *et al.*, CPE-POWERENG, 1 (2018).
[38] D. Purohit and M. Ghosh, IJCSMC **6**, 369 (2017).
[39] R. Tatikonda and A. Sinha, IJARIIE **5**, 12 (2019).
[40] S. Nagare *et al*., IJACEN **7**, 3 (2019).
[41] H. Singh and P. Kapoor, Int. J. Innov. Res. Sci. Eng. Technol. **6**, 4512 (2017).
[42] S. Chatterjee and S. Kar, IJCA **115**, 31 (2015).
[43] J. Ismael and M. Gastli, IEEE ICICC, 1 (2020).
[44] L. Boppana and V. Jain, IEEE CCAA, 1286 (2016).