Improving energy efficiency in buildings with an IoT-based smart monitoring system

energies

MDPI

Article

Improving Energy Efficiency in Buildings with an IoT-Based Smart Monitoring System

Fateme Dinmohammadi [1,*], Anaah M. Farook [1] and Mahmood Shafiee [2,*]

[1] School of Computing and Engineering, University of West London, London W5 5RF, UK
[2] Energy Resilience Centre, School of Engineering, University of Surrey, Guildford GU2 7XH, UK
* Correspondence: fateme.dinmohammadi@uwl.ac.uk (F.D.); m.shafiee@surrey.ac.uk (M.S.)

Abstract: With greenhouse gas emissions and climate change continuing to be major global concerns, researchers are increasingly focusing on reducing energy consumption as a key strategy to address these challenges. In recent years, various devices and technologies have been developed for residential buildings to implement energy-saving strategies and enhance energy efficiency. This paper presents a real-time IoT-based smart monitoring system designed to optimize energy consumption and enhance residents' safety through efficient monitoring of home conditions and appliance usage. The system is built on a Raspberry Pi Model 4B as its core platform, integrating various IoT sensors, including the DS18B20 for temperature monitoring, the BH1750 for measuring light intensity, a passive infrared (PIR) sensor for motion detection, and the MQ7 sensor for carbon monoxide detection. The Adafruit IO platform is used for both data storage and the design of a graphical user interface (GUI), enabling residents to remotely control their home environment. Our solution significantly enhances energy efficiency by monitoring the status of lighting and heating systems and notifying users when these systems are active in unoccupied areas. Additionally, safety is improved through IFTTT notifications, which alert users if the temperature exceeds a set limit or if carbon monoxide is detected. The smart home monitoring device is tested in a university residential building, demonstrating its reliability, accuracy, and efficiency in detecting and monitoring various home conditions.

Keywords: energy efficiency; residential buildings; smart monitoring; internet of things (IoT)

Academic Editor: Marcin Kaminski

Received: 20 January 2025
Revised: 10 February 2025
Accepted: 18 February 2025
Published: 5 March 2025

Citation: Dinmohammadi, F.; Farook, A.M.; Shafiee, M. Improving Energy Efficiency in Buildings with an IoT-Based Smart Monitoring System. *Energies* **2025**, *18*, 1269. https://doi.org/10.3390/en18051269

1. Introduction

As the global population expands, the demand for energy continues to rise, placing increasing pressure on existing energy systems. Much of the existing energy infrastructure was developed decades ago, at a time when global population and energy demands were considerably lower. While these systems have undergone continuous upgrades to improve efficiency and capacity, their ability to meet the growing future energy demands remains uncertain. Moreover, the heavy reliance on fossil fuel sources, which still dominate global energy production, further complicates the situation. Fossil fuels, such as coal, oil, and natural gas, are not only finite resources but also major contributors to harmful greenhouse gas emissions [1]. These emissions accelerate climate change and lead to severe social and environmental impacts, including rising global temperatures, extreme weather events, and loss of biodiversity [2]. The environmental degradation caused by fossil fuel consumption also leads to air and water pollution, which can harm public health, damage ecosystems, and disrupt natural processes.

Residential buildings are among the largest energy consumers globally, accounting for a substantial share of total energy consumption and contributing significantly to greenhouse gas emissions [3]. They consume substantial energy for heating and cooling, lighting, powering appliances, and other household activities such as cooking or cleaning. Therefore, enhancing energy efficiency in residential buildings is a highly effective strategy to reduce energy consumption, lower greenhouse gas emissions, and alleviate pressure on existing energy infrastructure. By implementing energy-saving technologies like smart thermostats, LED lighting, energy-efficient HVAC systems, and advanced insulation, buildings can operate more sustainably and reduce energy consumption [4]. Ongoing research and innovation in these technologies are crucial for further improving building energy performance and driving sustainability efforts forward.

Smart home technologies represent a new generation of innovations aimed at enhancing convenience, improving energy efficiency, and increasing safety through automation and real-time home environment monitoring [5]. These technologies often integrate a diverse range of smart devices and sensors that utilize wireless communication and computing to create a heterogeneous network, commonly referred to as the Internet of Things (IoT) [6]. A key benefit of smart home technologies is their ability to optimize energy use by enabling homeowners to remotely monitor and assess their energy consumption in real time via a smartphone or computer [7]. Additionally, smart home systems offer enhanced security features compared to traditional setups, providing greater protection by monitoring carbon monoxide levels, detecting signs of fire, and overseeing all areas of the home [8]. These features have made smart home technologies increasingly popular across a wide range of households and industries. Nevertheless, recent advancements in automatic control and artificial intelligence (AI) have revolutionized smart home monitoring systems, introducing extraordinary capabilities such as predictive analytics, automated decision-making, and advanced pattern recognition [9,10].

Smart home monitoring technology typically consists of sensors, actuators, control centers, communication networks, and other components, all designed to monitor and control various aspects of the home environment [11]. These systems collect data from multiple sources, process it using built-in computational resources, and enable residents to automatically adjust home settings to optimize comfort, energy efficiency, and security. Nowadays, with the widespread availability of smartphones and internet-connected computers, users can connect to their home network and manage their appliances from virtually anywhere in the world—whether at work, traveling, or simply away from home. For example, users can adjust home thermostat settings, turn off lights, or check the status of their washing machine through user-friendly mobile apps or web interfaces. This level of control not only enhances convenience but also improves energy efficiency, potentially leading to cost savings on utility bills. Additionally, remote monitoring capabilities contribute to home security by enabling users to monitor security cameras, receive alerts about unusual activity, and manage smart locks from a distance.

Despite the aforementioned advantages, commercially available smart home monitoring technologies often offer limited customization options and are not easily expandable or modifiable to meet individual user needs. Many of these devices adopt a one-size-fits-all approach, which can be restrictive for users seeking tailored solutions that align with their specific preferences and requirements. Additionally, there are significant concerns regarding the privacy and security of smart home monitoring devices [12]. The data collected from these devices are often stored by the manufacturers in cloud servers, leading to concerns among users about potential risks such as data breaches, unauthorized access, and the misuse of personal information [13]. The centralized storage of data makes it vulnerable to hacking and other cyber threats, potentially compromising the confidentiality and integrity

of user information. As smart home monitoring devices become increasingly integrated into daily life, addressing these privacy and security challenges is essential for ensuring user trust and promoting the responsible use of technology.

The primary aim of this paper is to design, develop, and test a secure IoT-based smart home monitoring system that enables homeowners to remotely monitor real-time home conditions, such as lighting and heating usage. The system is built on a Raspberry Pi Model 4B as its core platform, integrating various sensors to monitor the home environment, including the DS18B20 for temperature monitoring, the BH1750 for light intensity measurement, a PIR sensor for motion detection, and the MQ7 for carbon monoxide detection. The system is prototyped using a combination of hardware and software components, including microcontrollers, data storage, and processing units, as well as a graphical user interface (GUI) for data visualization and user interaction. Intelligent algorithms are implemented to process sensor data, detect active heating or lighting in unoccupied areas, identify extreme temperatures, and sense the presence of harmful gases. The device alerts users to energy waste caused by unnecessary heating or lighting and enhances safety by notifying them of potential environmental hazards. This will lead to improved energy efficiency, cost savings, and a safer home environment.

The rest of the paper is organized as follows. Section 2 provides a comprehensive overview of the current knowledge on smart home monitoring systems, with a specific focus on the use of IoT sensors. Section 3 outlines the research methodology, detailing the step-by-step process involved in developing the proposed smart home monitoring system. Section 4 discusses the design and prototyping of the system, covering its architecture, hardware components, software platforms, GUI integration, and notification mechanisms. Section 5 presents the results from the system testing, including performance assessments and user feedback analysis. Finally, Section 6 concludes the paper and explores potential directions for future research.

2. Literature Review

This section provides a comprehensive overview of existing knowledge on smart home monitoring systems, highlighting research related to their design and implementation using various technologies, sensors, and devices. It also examines case studies on emerging IoT technologies that can be integrated into smart home monitoring systems to improve energy efficiency and enhance residential security. Furthermore, it identifies gaps in the literature and suggests potential areas for improvement.

2.1. State-of-the-Art Smart Home Monitoring Systems Utilizing IoT Technologies

In recent years, IoT-based home monitoring technologies have gained significant attention as a means for residents to remotely monitor and manage their homes while automating various tasks [14,15]. Among these tasks, energy consumption monitoring stands out as one of the most critical applications, which uses advanced technologies such as IoT sensors, smart meters, and data analytics to track and optimize energy usage [16]. In the following paragraphs, we review past research on these technologies, focusing on their applications, benefits, and limitations in home energy management.

Pan et al. [17] developed a proof-of-concept experimental testbed to prototype an IoT system using a smartphone app and cloud-computing technologies to monitor the energy efficiency of office buildings. The system collected one year of energy usage data to identify the building's consumption patterns and explore methods for improving energy efficiency. Hadwan and Reddy [18] developed a smart home system using a Raspberry Pi 2 Model B, Arduino Uno, DHT11 sensor, PIR sensor, wireless transceiver, and relay modules to control and monitor various home conditions, such as lighting, room tempera-

ture, and security alarms. The system also included smart switches and an access point, enabling a remote control via an Android smartphone app. Al-Ali et al. [19] prototyped an energy management system using a microcontroller, an RFID reader, and temperature and humidity sensors to monitor and control energy consumption in heating, ventilation, and air conditioning (HVAC) units. The system collected energy usage data from various smart home devices and transmitted it to a server for further processing and analysis. The prototype was subsequently tested in a lab environment to validate its effectiveness.

Agyeman et al. [20] designed a smart meter system to monitor energy use in residential buildings. The system's control unit was built with a Raspberry Pi 3 Model B, while the client unit, which featured a current sensor, relays, and an LCD display, was developed with an Arduino. A mobile app was also developed using Java, allowing users to monitor the real-time energy usage of home appliances and receive alerts for excessive consumption. Hamdan et al. [21] designed a smart home automation system to monitor and control appliances via a mobile interface and through text or voice commands using natural language processing (NLP). The system featured an ARM-based single-board room controller, a cloud server, and a Raspberry Pi 3B+ home server, which acted as a bridge between the room controller and the cloud. Key functions included real-time command translation, appliance control, secure data reception, and data decryption to ensure communication privacy and integrity. Jabbar et al. [22] designed a smart home device called IoT@HoMe for elderly individuals to remotely control their home appliances. The device integrated various sensors, including an LDR for light intensity, a DHT11 for temperature and humidity, an MQ2 for gas leakage, a PIR for motion detection, and an HC-SR04 ultrasonic sensor for object distance measurement. A node microcontroller unit (NodeMCU) was used as a Wi-Fi-based gateway to connect various sensors and upload their data to the Adafruit IO cloud server, enabling notifications in response to unusual events.

Khattar et al. [23] developed a virtual home assistant named "Olivia" with enhanced features for individuals with visual impairments. Olivia was built on a Raspberry Pi 3B+, integrating services like weather updates, stock prices, and music playback. Microphones and speakers were installed in each room for voice commands and responses. The system was equipped with a PIR sensor and a webcam to detect visitors at the door, utilizing the eigenface method for face recognition. If the owner was recognized, Olivia unlocked the door; otherwise, it asked for the visitor's name and sent the details, along with a photo, to the owner via email or text. The main disadvantage of Olivia is that if the system incorrectly identifies a stranger as the owner, it could pose a security risk. Rashid et al. [24] developed an energy management system for residential buildings in Malaysia, integrating real-time analytics, machine learning, IoT sensors, and embedded devices. A Raspberry Pi 3A+ with a non-invasive current sensor collected data from various devices, while Matplotlib visualized real-time energy consumption. A Long Short-Term Memory (LSTM) model predicted energy costs based on historical data and user preferences, with Google Colab utilized for data set training. Alsaleem et al. [25] developed an IoT-based wearable device that enabled individuals to control their personal thermal comfort in residential buildings. Various supervised learning algorithms were tested to determine the ideal room temperature for each occupant in the building. The personalized comfort model was then used to simulate an intelligent controller, which applied the particle swarm optimization (PSO) method to identify the optimal thermal comfort parameters.

Khan et al. [26] designed an IoT-based monitoring system to reduce energy consumption in buildings by measuring parameters such as voltage, current, and active power. The system comprised an Arduino UNO, ESP8266 Wi-Fi module, ACS712 current sensors, voltage sensing circuits, and an LCD display for real-time monitoring. Pujari et al. [27] developed a smart home monitoring system using a NodeMCU ESP32 and sensors, in-

cluding PIR, DHT22, MQ-6, and LDR. The system enabled elderly individuals and those with disabilities to remotely control household appliances via a mobile app. Majumder and Izaguirre [28] developed an IoT system to detect security threats in smart homes. The system used a Raspberry Pi 2 as the central unit, equipped with a NoIR Pi camera module for video recording and image capture, and a PIR sensor for motion detection. Mudaliar and Sivakumar [29] discussed the implementation of an IoT-based real-time monitoring system to control energy consumption in an industrial building. The system collected data from energy meters using a Raspberry Pi 3B+ module and stored it for access via a laptop or mobile phone through Grafana. When connected to the internet, the system provided a graphical display of various electrical parameters monitored by the energy meters.

Sooraj et al. [30] designed an IoT-based smart home assistant for elderly and disabled individuals, enabling them to control household appliances through speech recognition. A Raspberry Pi 3B+ was used as the server, equipped with a camera and PIR sensor to capture images upon motion detection, which were then emailed to users. An Arduino Uno, connected to a relay, controlled appliances via an HC-05 Bluetooth module for remote and voice control. The system also included an intuitive GUI for energy consumption monitoring. Valov and Valova [31] introduced a home automation system that allows users to remotely monitor and control household devices via the internet. The system integrated sensors, such as the BH1750 digital light sensor and the BME280 digital barometric pressure sensor, connected to a Raspberry Pi 3B+ through an I2C interface to collect data on light, temperature, pressure, and humidity. A web-based software platform was also developed using the MariaDB database management system to visualize the data. Ramelan et al. [32] developed a building energy monitoring system using LoRa modulation and MQTT protocols to monitor and control energy consumption in real time. The system featured an Arduino Uno as the microcontroller, along with an ACS712 current sensor, ZMPT101B voltage sensor, and a Dragino LoRa gateway to connect the measured parameters to an IoT cloud server.

Taiwo and Ezugwu [33] designed a cloud-based intelligent home automation system to control household appliances and monitor environmental conditions such as temperature and humidity. The system consisted of an ESP8266 board, a 5 V four-channel relay module, and various sensors, including a DHT11 sensor, an LDR sensor, and an HC-SR501 PIR sensor. It also incorporated an ESP32 camera for home security, detecting motion and capturing images. To minimize false alarms, a support vector machine (SVM) algorithm classified the images, identifying whether the subject was a regular home occupant or an intruder. Tukymbekov et al. [34] developed a solar-powered autonomous device to monitor energy consumption for street lighting systems. They implemented an LSTM algorithm to predict solar panel output, using data from light sensors detecting sunset and nightfall, along with motion sensors tracking vehicle activity. The device's wireless connection was established using LoRa technology with SX1278 chips. Data from the lamp post, sensor readings, weather forecasts, and solar radiation predictions were sent to a cloud server for storage and to control lamp brightness, automating nighttime illumination. Remote monitoring of battery levels, temperature, humidity, and other sensor readings was made possible through a mobile phone or computer connected to the internet.

Desnanjaya et al. [35] developed a classroom monitoring system in an Indonesian school using an ESP-12E microcontroller. The system monitored temperature, humidity, and light intensity through DHT22 and BH1750 sensors, respectively, with data displayed on a 16 $\times$ 2 I2C LCD screen. The device also included a dual-driver SSR for controlling the classroom lights. Saleem et al. [36] discussed the design, deployment, and implementation of an IoT-based smart energy management system for monitoring load profiles in electricity grids. The system was built using a MiddleWare platform and structured across three

layers: the application layer, the communication layer, and the energy control layer. It provided valuable information, such as peak load and base load data, to both customers and suppliers, allowing suppliers to adjust incentives and encourage customers to reduce energy consumption. Arunkumar et al. [37] developed a smart home system using wireless protocols like Z-Wave, Bluetooth, and GSM to monitor kitchen appliances. The system was built with an Arduino UNO microcontroller, integrating various sensors, including a DHT11 for humidity and temperature, an MQ135 for air quality, a PIR sensor for motion detection, and a camera module. The code was executed on the NodeMCU using the Adafruit IDE. For safety, IFTTT sent immediate notifications upon detecting unknown objects. Users could remotely control fans, lights, and appliances via the Blynk app, with voice commands enabled through Google Assistant.

García-Monge et al. [38] designed and implemented a smart home monitoring system to enhance energy efficiency in university buildings by optimizing HVAC performance. The system utilized an Aranet4 Pro sensor to monitor CO_2, relative humidity, temperature, and atmospheric pressure; a SenseCAP EU868 sensor for CO_2 monitoring; a QPA1004 sensor for air quality monitoring; and an LHT65 sensor for temperature and humidity monitoring. It continuously monitored temperature to prevent overheating and regulated real-time CO_2 levels in rooms with varying occupancy. Umair et al. [39] implemented and deployed a real-world IoT testbed in a smart home environment to efficiently reduce energy consumption while ensuring human comfort. The system used data analytics and machine learning algorithms to predict home occupancy, which was then integrated into a comfort-aware energy-saving mechanism named the prediction- and feedback-based proactive energy conservation algorithm to forecast energy consumption.

2.2. Gaps of the Reviewed Literature

As highlighted in the literature review, the use of IoT technologies and devices for smart home monitoring has been explored extensively in various research papers. Most studies have focused on monitoring and automating household appliances via mobile and web-based applications, enabling users to control their homes remotely. Some research has emphasized enhancing home security through the integration of cameras, facial recognition technology, and alarm systems. Additionally, other studies have aimed to develop smart home devices that conserve energy while assisting the elderly and disabled in their daily routines. The reviewed literature indicates that these systems were built using a variety of microcontrollers, smart sensors, mobile applications, and cloud storage platforms.

According to previous studies, Raspberry Pi is the most used microcontroller for developing smart home monitoring devices. Its popularity is due to its high computing power and versatility, allowing it to manage multiple tasks simultaneously, making it ideal for complex smart home applications. PIR sensors were widely used for motion detection due to their high sensitivity and reliability. For light detection, researchers either used LDR sensors or preferred the more accurate BH1750 digital light sensor. Temperature and humidity were monitored using DHT11 and DHT22 sensors, while air quality was assessed using MQ2, MQ6, and MQ135 sensors. Cloud storage services such as Adafruit IO, Firebase Database, and MariaDB were commonly used for data storage and processing, while IFTTT and MQTT protocols were used for sending notifications and alarms.

While many studies focus on the technical aspects of smart home devices—such as sensor integration, microcontrollers, and communication protocols—there is a notable lack of detailed analysis of how these systems perform under real-world conditions, such as network instability, power outages, or sensor malfunctions. Additionally, few studies comprehensively evaluate the energy consumption of these devices, a key factor in assessing the sustainability and cost-effectiveness of smart home systems. Performance metrics like re-

sponse time, data processing speed, and long-term durability are often overlooked, leaving gaps in understanding how to optimize these systems for larger-scale deployments. Furthermore, many studies fail to address the security vulnerabilities inherent in IoT devices, which could expose smart homes to cyberattacks or unauthorized access. The robustness of communication protocols and encryption methods is another area where the literature falls short, with limited evaluation of how these systems protect sensitive user data and prevent security breaches. To address these research gaps, we propose the development of a secure IoT-based smart home monitoring system with the following functionalities:

- Our smart home monitoring system is built around a microcontroller with high processing power, supported by a range of highly precise sensors that provide accurate temperature and light intensity readings.
- The proposed system offers several functions: it automatically adjusts room temperature based on outdoor conditions, identifies lights or heating systems left on without occupancy and sends notifications to the user, and detects carbon monoxide gas, alerting the user to evacuate the home due to its extreme toxicity.
- The device ensures user trust and promotes the responsible use of technology by addressing concerns about potential risks, such as data breaches, unauthorized access, and the misuse of personal information.
- The technology is tested in a university residential building, demonstrating its reliability, accuracy, and efficiency in detecting and monitoring various home conditions.

3. The Methodology

This section outlines the methodology for designing and implementing a smart home system using IoT sensors and microcontrollers to collect environmental data and monitor the status of home appliances. The process begins with defining the hardware and software requirements, followed by the development of a low-fidelity prototype. After the prototype phase, the complete smart home system is designed, incorporating all necessary hardware, software, and GUI components. The system's code is then tested to ensure proper functionality and compliance with the requirements. Any issues identified during unit testing are addressed, and this process is repeated until all problems are resolved. Finally, the full smart home system is built. These steps are outlined in detail in the following subsections.

3.1. Hardware and Software Requirements of the System

Table 1 outlines the specific hardware and software components chosen for prototyping the proposed smart home monitoring system.

Table 1. The hardware and software components of the smart home monitoring system.

Hardware components	Raspberry Pi 4B, MicroSD, a laptop or PC with internet connectivity, a mobile phone with internet connectivity, USB-C power supply, BH1750 digital light sensor, DS18B20 temperature sensor, PIR motion sensor, MQ7 gas sensor, 10 k ohm resistor, jumper wires, breadboard.
Software components	Raspbian Linux Operating System, Adafruit IO, IFTTT (If This Then That), OpenWeather, Wi-Fi connection, Python programming language.

3.2. Low-Fidelity Prototype of the System

A prototype of the proposed smart home monitoring system is illustrated in Figure 1, showcasing various hardware components such as sensors, a MicroSD memory card, resistors, jumper wires, and a breadboard, all connected to a Raspberry Pi. It also highlights

the integration of Adafruit IO cloud storage and IFTTT mobile notifications to enhance system functionality. The combination of Adafruit IO with IFTTT automates numerous operations through pre-built applets. Adafruit IO is highly regarded not only as a cloud storage platform but also for its user-friendly interface, making it ideal for smart home applications. The prototype further demonstrates how users can remotely access sensor data via the Adafruit IO GUI, illustrating the interaction between devices over a Wi-Fi connection. This prototype serves as an initial model for the final design.

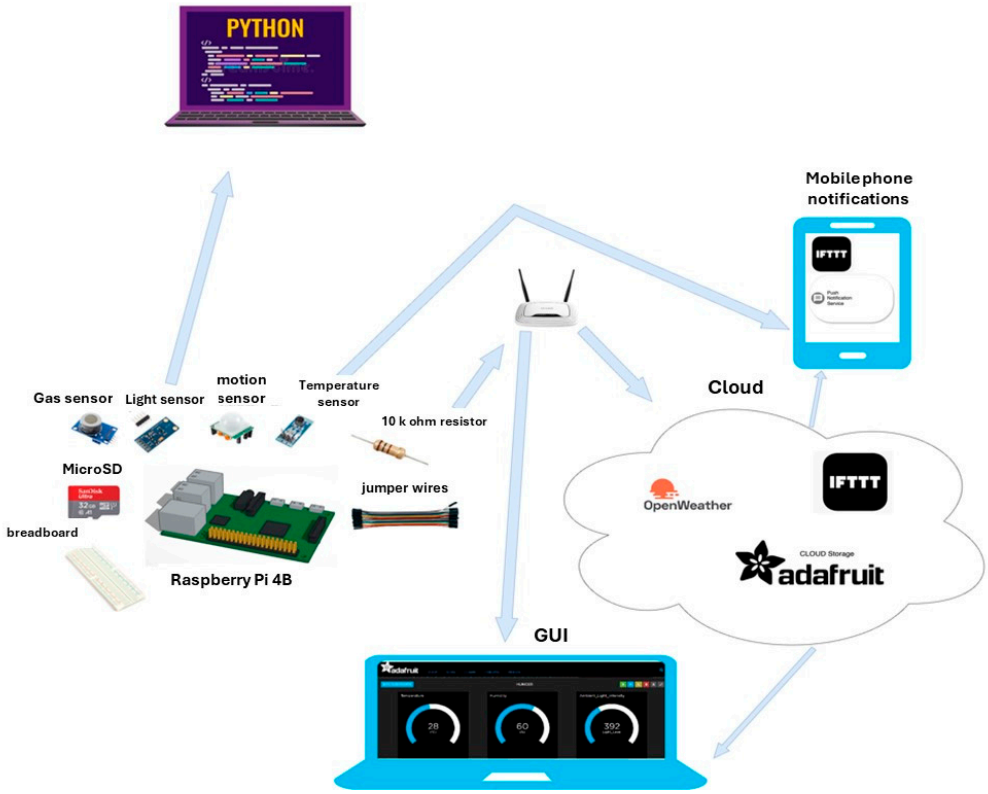


Figure 1. A prototype of the proposed smart home monitoring system.

3.3. System Development

Data collection is a critical step in system development. In the proposed smart home system, sensors play a key role in this process. They are connected to the Raspberry Pi to monitor ambient temperature, light intensity, human occupancy, and the presence of hazardous gases within the home. The collected data are stored in Adafruit IO cloud storage, enabling easy access and real-time analysis. This setup ensures that users are continuously informed about home conditions, including the status of lighting and heating systems, in real time. Users can remotely access the data, which is displayed in graphs and tables for easy interpretation. To protect occupant privacy, the data collection process is anonymized, with additional security measures like password protection to prevent unauthorized access and misuse. The system also employs statistical analysis, which is divided into two main types: descriptive and inferential statistics. Descriptive statistics help in drawing conclusions from the overall data, while inferential statistics highlight significant differences between different data sets. Inferential statistics further enable comparative and relationship analysis. Testing hypotheses through data analysis is crucial for better understanding variables in a smart home. For example, our system analyzes the correlation between the home's temperature and the heating on/off status to assess user comfort. Similarly, it examines the correlation between light intensity and the light

on/off status. Patterns of usage behaviors are identified through graphical representations, enabling informed decisions and optimizing the smart home system.

After data are gathered, it is analyzed to support decision-making processes. In the proposed smart home monitoring system, decisions are driven by motion, lighting, and heating data. For example, if the system detects that lighting or heating is active without any motion being detected, it flags this as a potential energy waste. The system utilizes statistical analysis, which is categorized into two main types: descriptive and inferential statistics. Descriptive statistics provide insights into overall data trends, while inferential statistics identify significant differences between data sets, facilitating comparative and relationship analysis. Hypothesis testing is a crucial aspect of data analysis, as it helps understand the relationships between variables in a smart home environment. For instance, the system analyzes the correlation between the home's temperature and the heating on/off status to assess user comfort. Similarly, it examines the relationship between light intensity and the light on/off status. Usage patterns are visually represented through graphs, enabling informed decision-making and optimization of the smart home system.

3.4. System Testing

Extensive testing is essential to ensure the system's functionalities meet all user requirements. The system undergoes rigorous testing by collecting data on key aspects such as sensor performance, system response time, and data accuracy. For example, the temperature sensor's performance and reaction time are verified by opening windows and observing the corresponding drop in room temperature. Unit testing plays a central role in this process, focusing on the independent assessment of each module, including sensors and code functions, to ensure seamless integration and correct output. By evaluating each component in isolation, unit testing guarantees functionality, reliability, and usability before the full system is integrated. In addition to unit testing, controlled experiments are conducted to assess system performance regarding reaction time, data accuracy, and stability. Computer code is developed to simulate the system in a variety of real-world scenarios. For instance, the smart home system is tested in rooms with different light levels to evaluate its light detection capabilities. Adjustments to the code are made based on light intensity readings to ensure the system is robust and adaptable to diverse environmental conditions.

4. System Design and Implementation

This section provides a comprehensive overview of the design and implementation of the IoT-based smart home monitoring system. It details the step-by-step process of hardware integration and the selection of appropriate software platforms. Pseudo-code is included to demonstrate how hardware and software are seamlessly integrated, resulting in a fully functional smart home system with a GUI. Additionally, this section presents evidence from the tests conducted to verify the system's performance, along with a discussion of the challenges encountered at each stage of development.

4.1. Design of the System

4.1.1. System Architecture

The proposed smart home monitoring system is built on a Raspberry Pi microcontroller that monitors home conditions through sensors, providing real-time information about heating and lighting. Additionally, the system detects the presence of hazardous gases and temperatures exceeding 50 °C, alerting the residents to take appropriate safety measures. The primary objective of our system is to conserve energy, which is achieved through automated IFTTT notifications when no movement is detected while appliances remain on.

The smart home system utilizes Adafruit IO, offering a user-friendly interface for remote monitoring and cloud storage.

4.1.2. Hardware Components

Raspberry Pi—The Raspberry Pi 4 Model B serves as the core of the proposed smart home system. With 4 GB of RAM and a quad-core Cortex-A72 64-bit processor, it delivers high performance and powerful computing capabilities. Figure 2 illustrates the components and connectors of the Raspberry Pi model.

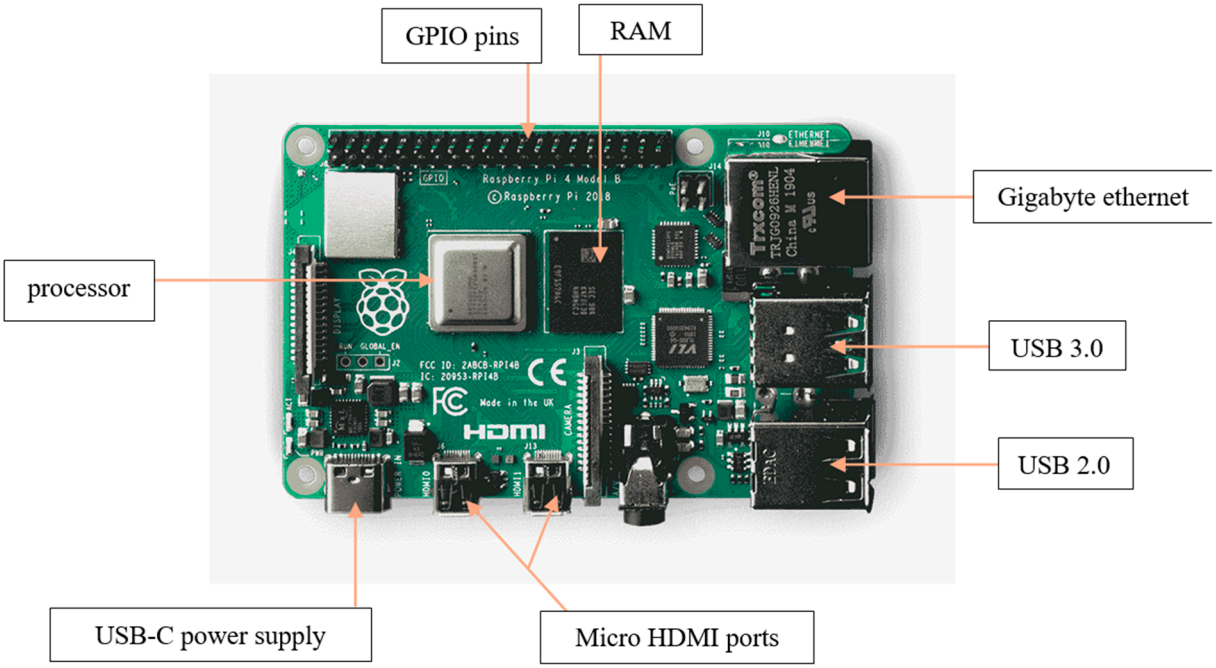


Figure 2. The Raspberry Pi 4 model used in the smart home monitoring system.

- 2.4 GHz and 5.0 GHz wireless LAN: It supports dual-band Wi-Fi, allowing connection to networks without the need for additional hardware.
- Bluetooth 5.0: It provides easy connectivity to wireless networks and devices.
- Gigabit ethernet: It enables high-speed connection to local area networks.
- USB ports: They are equipped with two USB 2.0 and two USB 3.0 ports, providing flexibility to connect various devices.
- Micro HDMI ports: They allow for dual display output in 4K resolution, an upgrade from earlier versions that only supported resolutions up to 1080p.
- Mobile Industry Processor Interface (MIPI) ports: They include a 2-lane MIPI display port and a 2-lane MIPI camera port, facilitating video and display interfaces.
- Audio/video ports: A 4-pole stereo audio and composite video port supports audio and video functions.
- MicroSD card slot: It is used for importing the operating system and storing data.
- Power options: The device can be powered through a 5 V DC USB-C connector, a 5 V DC GPIO header, or Power over Ethernet (PoE).

Figure 3 illustrates the 40-pin GPIO header, which contains groups of power, ground, and GPIO pins that support I2C, SPI, UART, PWM, and GPIO protocols, with voltage levels of 3.3 V and 5 V. In summary, this model offers greater processing power, increased memory, multiple connectivity options, enhanced I/O and multimedia capabilities, and compatibility with the latest Raspberry Pi OS and third-party software, making it significantly more versatile than previous versions.

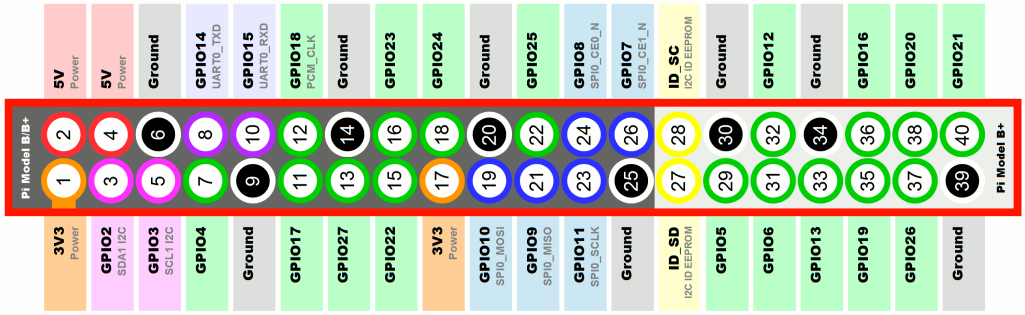
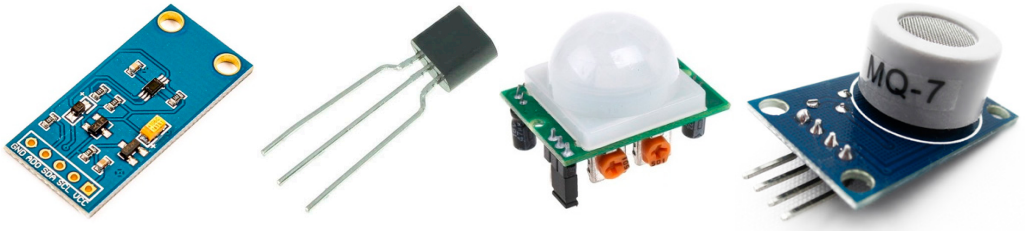


Figure 3. The layout of Raspberry Pi GPIO pin.

BH1750 digital light sensor—A BH1750 digital light sensor, as shown in Figure 4a, was selected for the smart home monitoring system to measure ambient light intensity. This digital sensor can measure light intensity across a broad range of environments and light sources, including infrared rays, sunlight, incandescent lamps, fluorescent lamps, and LEDs, with a detection range from 1 lux to 65,535 lux. The sensor has five pins: Vcc (providing power), GND (connected to the ground of the circuit), SCL (providing clock pulses for I2C communication), SDA (transmitting data via I2C communication), and ADD (selecting the sensor's I2C address). The sensor uses a standard I2C communication protocol to interact with the central controller. The controller sends control commands and reads light measurements from the BH1750 via the I2C interface. This sensor is favored due to its quicker response time, higher accuracy, and immunity to environmental factors that can affect other light sensors, such as LDRs (light-dependent resistors). Thus, the BH1750 sensor proves to be a more reliable choice for precise light measurement.



(**a**) (**b**) (**c**) (**d**)

Figure 4. (**a**) BH1750 light sensor, (**b**) DS18B20 temperature sensor, (**c**) PIR motion sensor, (**d**) MQ7 gas sensor.

DS18B20 temperature sensor—Figure 4b shows the DS18B20 temperature sensor used in the proposed system to monitor ambient room temperature. This sensor is capable of measuring temperatures ranging from −55 °C to 125 °C with an accuracy of ±0.5 °C, making it ideal for smart home applications. The sensor communicates using a one-wire protocol and features a unique 64-bit bus, allowing multiple sensors to be connected simultaneously on the same one-wire bus. The sensor has three pins: Vcc (providing power), the data pin (sending temperature readings to the controller), and the GND pin (connecting the sensor to the ground). This simple and reliable sensor design makes the DS18B20 a robust choice for monitoring temperature in smart home environments.

PIR motion sensor—Figure 4c shows the HC-SR501 PIR sensor used in the proposed smart home monitoring system for motion detection. This sensor detects changes in infrared radiation emitted by living beings within a range of 3 to 10 m. It offers adjustable sensitivity and detection range, enabling it to detect movement without requiring physical contact or visible light, making it particularly suitable for low-light or dark environments. The PIR sensor's pin configuration includes Vcc (connected to a +5 V power supply), the GND pin (connected to the ground), and the OUT pin (sending digital pulses to the system

upon detecting motion). With its non-intrusive nature and ability to operate in various lighting conditions, the HC-SR501 sensor is ideal for enhancing security and automation in smart home systems.

MQ7 gas sensor—Carbon monoxide (CO) is a poisonous gas emitted by household appliances such as gas stoves, furnaces, boilers, and water heaters. Even at low levels, CO poses serious health risks, and at higher concentrations, it can be lethal. The MQ7 sensor, shown in Figure 4d, is integrated into the smart home system to monitor CO levels. The MQ7 sensor is composed of a tin dioxide (SnO_2) sensing layer on a small circuit board, along with additional electronic components. It detects CO levels ranging from 20 to 2000 ppm by measuring changes in electrical resistance caused by CO molecules reacting with the SnO_2 layer. The MQ7 sensor has four pins: Vcc (providing power to the sensor), GND (connected to the ground), DO (Digital Out) pin (producing digital signals when a threshold level is reached), and AO (Analog Out) pin (providing an analog output for more precise detection of CO). The MQ7 sensor ensures continuous monitoring of CO levels in the home, alerting users in case of dangerous concentrations.

4.1.3. Software Components

Adafruit IO—Adafruit IO is the IoT platform used in the proposed smart home monitoring system, serving as the primary cloud service for data storage and remote management. It enables the storage and retrieval of data, connecting devices over the internet, and managing them remotely by integrating various sensors, microcontrollers, and other electronic components. The key features of Adafruit IO include feeds (storing data from sensors and displaying it in graphs and tables for easy analysis), dashboards (allowing users to visualize data through graphs, charts, and buttons), real-time communication (enabling instant data transmission and notifications when specific events occur), and integration (enabling automation and enhanced functionality across multiple services). This platform is ideal for remote monitoring and controlling devices, making it highly suitable for smart home applications.

IFTTT—IFTTT, short for "If This Then That", is a web service and mobile app that notifies users when specific events occur through applets. Applets consist of two parts: a condition or event, known as the "trigger", which prompts the applet to run, and the corresponding "action", which is initiated by IFTTT. For instance, a trigger could detect no motion while the lights are still on, and the action would notify the user with a message such as "You have left the lights on". IFTTT can seamlessly integrate with a wide range of apps and services, including Adafruit IO, to automate various smart home tasks.

4.1.4. Flow Chart of the System Operation

A flow chart representing the system operation is depicted in Appendix A.

4.1.5. Pseudo-Code

A pseudo-code has been developed to monitor home conditions in real time, including temperature, lighting, gas emissions, and motion. It automates notifications for excessive heat, unattended lighting, carbon monoxide detection, and potential fire hazards. The code is provided in Appendix B.

4.2. Implementation of the System

4.2.1. Setting up Raspberry Pi

The proposed smart home monitoring system is built on a Raspberry Pi microcontroller that monitors home conditions. The Raspberry Pi Imager software tool, available on the official Raspberry Pi website (https://www.raspberrypi.com/software/ accessed on 6 January 2025), is used to install the operating system. After downloading the tool, users

can select "Operating System" and choose "RASPBERRY PI OS (32 Bit)". This OS includes a graphical desktop and essential software like a web browser, office suite, media player, and programming tools, making it ideal for smart home monitoring. The next step is to insert a 32 GB MicroSD card into a card reader. Under "Storage" we select the inserted SD card, click the "Settings" icon to enable SSH (Secure Shell), set a hostname, and configure the username, password, Wi-Fi credentials, and country code. After these configurations, we click "Write" to install the Raspberry Pi OS onto the SD card. SSH is a protocol that allows remote access to the Raspberry Pi via the terminal from a laptop. Once the SD card is inserted into the Raspberry Pi and powered via USB-C, it automatically connects to Wi-Fi, with an LED indicating the connection status. The Pi can then be accessed remotely using the terminal with the command "ssh pi@ipaddress", authenticated by password.

To access the graphical desktop interface of the Raspberry Pi remotely, the VNC (Virtual Network Connection) viewer is installed. First, the IP address of the Raspberry Pi is entered, and the viewer then prompts for the Raspberry Pi's username and password for authentication. Once authenticated, the desktop of the Raspberry Pi can be accessed and controlled remotely. The Raspberry Pi runs a Linux-based OS, and Python was chosen as the programming language for the smart home system due to its simplicity and versatility. Python simplifies the integration of sensors with various APIs and services, automating tasks in a smart home environment. The Raspberry Pi OS comes with Python preinstalled, along with the Thonny Python Integrated Development Environment (IDE), which offers a user-friendly interface for coding, debugging, and testing.

4.2.2. Installation of Sensors

Temperature sensor—The necessary components for installing the temperature sensor include a DS18B20 sensor, a 10 kΩ resistor, and jumper wires. The DS18B20 sensor is connected to GND, data, and Vcc on a breadboard, and a 10 kΩ resistor is placed between the data and Vcc pins on the breadboard. Using jumper wires, the sensor's data pin is connected to GPIO4 (pin 7), the GND pin is connected to the ground (pin 6), and the Vcc pin is connected to 3.3 V (pin 1) on the Raspberry Pi board. Once the hardware components are connected, the terminal on Mac OS is opened, and the command "sudo nano/boot/config.txt" is entered to edit the configuration file. At the bottom of the file, the line "dtoverlay = w1-gpio" is added, then the file is saved and exited. The Raspberry Pi is rebooted using the command "sudo reboot", and after logging back in, the commands "sudo modprobe w1-gpio" and "sudo modprobe w1-therm" are executed to load the required modules. Next, the directory is changed using the command "cd/sys/bus/w1/devices", followed by "ls" to display the sensor's address. Once the sensor address is identified, navigating to the directory where the temperature readings are stored is performed using "cd 28-xxxx" (where "xxxx" is the sensor's unique address). Finally, the command "cat w1_slave" is used to display the temperature readings in degrees Celsius.

Motion sensor—The PIR sensor is connected using jumper wires to detect motion. The GND pin of the sensor is connected to the ground (pin 39), the OUT pin is connected to GPIO17 (pin 11), and the Vcc pin is connected to 5 V (pin 2) on the Raspberry Pi board. Once the PIR sensor is properly connected, a Python code is implemented to detect motion. The sensor outputs a high signal when motion is detected, indicating high sensitivity. If the sensitivity is too high, it can be adjusted by turning the sensor's potentiometer anti-clockwise to reduce it.

Light sensor—The BH1750 light sensor is connected to the breadboard and then to the Raspberry Pi using jumper wires. The GND pin of the sensor is connected to the Raspberry Pi's ground (pin 25), the ADD pin is connected to the sensor's GND pin, the SDA pin is connected to GPIO2 (pin 3), the SCL pin is connected to GPIO3 (pin 5), and

the Vcc pin is connected to the 3.3 V power supply (pin 17) on the Raspberry Pi board. The sensor communicates using the I2C interface, which needs to be enabled by running the command sudo raspi-config, selecting "Interface Options" and enabling I2C. Next, the required Python package is installed with "sudo apt-get install -y python-smbus i2c-tools", followed by the command "sudo i2cdetect -y 1" to detect the BH1750 sensor. A Python script is then written to read data from the BH1750 digital light sensor. Upon successfully running the code, the output prints the lux value of the detected light intensity.

Gas sensor—The MQ7 sensor is connected to the Raspberry Pi using jumper wires. The Vcc pin is connected to the power supply (pin 4), the GND pin is connected to the ground (pin 34), the DOUT pin is connected to GPIO16 (pin 36), and the AOUT pin is connected to GPIO20 (pin 38) on the Raspberry Pi board. After making the connections, a Python script is written to detect the presence of CO gas in the environment, enabling the system to monitor air quality and alert users when hazardous gas levels are detected. Figure 5 illustrates the connections between the Raspberry Pi and multiple sensors: the DS18B20 temperature sensor, BH1750 light sensor, PIR motion sensor, and MQ7 gas sensor.

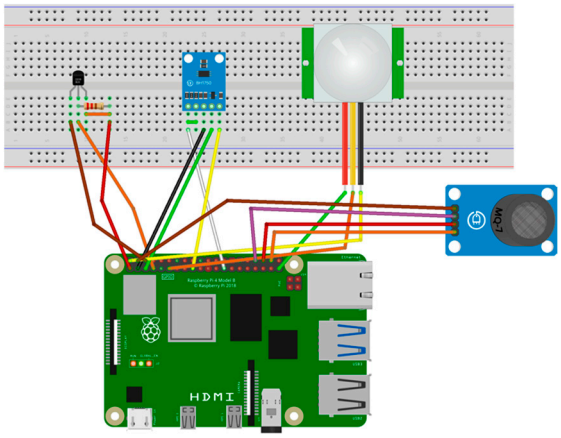


Figure 5. Wiring diagram.

The DS18B20 communicates with the Raspberry Pi via the 1-wire interface, connected to GPIO4, to measure ambient temperature. The BH1750 light sensor uses the I2C protocol, with the SDA and SCL data lines connected, to measure room light intensity in lux units. The PIR sensor is connected to GPIO17, detecting motion by sensing infrared radiation emitted by objects. The MQ7 sensor, responsible for detecting carbon monoxide (CO), is connected via its analog output pin to the Raspberry Pi.

4.2.3. Integration of Adafruit

To enable cloud storage for the collected data, Adafruit IO cloud services are integrated into the smart home monitoring system. This is achieved by modifying the Python code to include Adafruit IO libraries, which facilitate communication between the smart home system and the cloud. First, an account is created on the Adafruit IO website (https://io.adafruit.com). Once the account is set up, new feeds—such as temperature, light, motion, and gas—are created to store data from the sensors. The Python code communicates with Adafruit IO using the username and key credentials from the account, allowing authentication and secure data transfer to the Adafruit cloud platform. Next, the Adafruit IO Python library is installed by running the command "pip install Adafruit-io" in the terminal. This library allows the Raspberry Pi to transfer data to Adafruit IO via the REST API. The Python code is then updated by importing the necessary libraries and configuring the Adafruit IO client and feeds. When sensor data are read, it is sent to the corresponding

feed on Adafruit IO, allowing for real-time monitoring and remote access to the data through the cloud platform.

4.2.4. Detection of Appliances

The temperature and light intensity readings are analyzed to determine the on and off status of heating and lighting within the smart home system. Initially, the system is placed in a dark room with the lights turned off to establish a baseline reading of zero lux for light intensity. Then, the lights are turned on, and multiple readings are taken from different positions within the room to account for variations in lighting levels. This method helps establish an accurate threshold range for determining the lighting status. Since natural sunlight also affects lighting conditions, the system is tested at different times of day, under both natural and artificial light, to ensure it can accurately detect the lighting status in various scenarios. The Python code is modified to account for sunlight intensity during the day and darkness at night, enabling reliable detection of light status throughout the day. To determine the heating status, a threshold temperature is set to detect whether the heating system is on or off. However, external temperature fluctuations can affect the internal temperature and impact heating decisions. To address this, the OpenWeather API is integrated into the Python code to provide real-time external temperature data. This allows the system to adjust the internal heating threshold more accurately, considering both internal and external temperatures. Various combinations of internal and external temperature thresholds are tested to fine-tune the heating status detection. The Python code is further optimized to send data to Adafruit IO cloud services under different conditions. For instance, the system sends data to the Adafruit IO "lighting" feed when the lights are on, off, or when they are left on while the room is unoccupied. Similarly, data are sent to the "heating" feed when the heating system is on, off, or left on without anyone in the room. This allows for continuous tracking of both lighting and heating statuses, offering insights into energy consumption patterns. For example, data showing that the heating or lighting is on while the room is unoccupied helps identify instances of unnecessary energy use, enabling more efficient energy management.

4.2.5. Integration of User Interface

The smart home user interface is managed through the Adafruit IO dashboard, allowing users to interact with the smart home system in real time. The dashboard provides a live display of various home conditions, including temperature, light intensity, CO levels, motion detection, and the status of the heating and lighting systems. Additionally, the dashboard presents graphs that track key metrics over time, such as light intensity and lighting status, temperature and heating status, as well as correlations between lighting and motion, and heating and motion. The Python code running on the Raspberry Pi continuously publishes sensor data to the appropriate Adafruit IO feeds, which the dashboard then visualizes. Temperature, light intensity, motion detection, and CO levels are presented using gauges for real-time monitoring, while simple indicators display the on/off status of the heating and lighting systems. This setup gives users a comprehensive, intuitive interface for monitoring home conditions and managing energy usage effectively, ensuring both comfort and efficiency.

4.2.6. Notification Automation

The proposed smart home monitoring system includes an automated notification feature to alert users when specific events occur. This is accomplished by integrating the IFTTT platform with the Adafruit IO feeds used in the system. Custom applets are created to monitor key conditions and trigger alerts. For example, notifications are sent when the heating or lighting is left on while the room is unoccupied, the temperature exceeds 50 °C,

or carbon monoxide is detected. To enable these notifications, the IFTTT mobile app is downloaded, and the user signs in with the same account used to create the applets. When any of the specified conditions are met, the system sends an instant notification to the user's phone, allowing them to take action—such as remotely turning off the heating if it has been left on. This setup ensures real-time alerts, enhancing safety and energy efficiency in the smart home by allowing users to respond quickly to important changes.

5. Results and Analysis

This section outlines the validation process for the proposed smart home monitoring system, highlighting the successful integration and operation of both hardware and software components. The system's performance is evaluated, with particular emphasis on the accuracy and reliability of data displayed in the Adafruit IO feeds and the seamless functioning of the IFTTT applets for automated notifications. Additionally, a user-friendly dashboard is developed to visualize real-time monitoring of key metrics, including temperature, light intensity, motion detection, and CO levels, ensuring an intuitive interface for efficient home management.

5.1. Demonstration of the System

Figure 6 illustrates the proposed smart home monitoring system. The components labeled in the figure include the following: (1) Raspberry Pi, (2) BH1750 light sensor, (3) MQ7 gas sensor, (4) HC-SR501 PIR sensor, and (5) DS18B20 temperature sensor. Each component is connected to the Raspberry Pi to form an integrated system for real-time home monitoring.

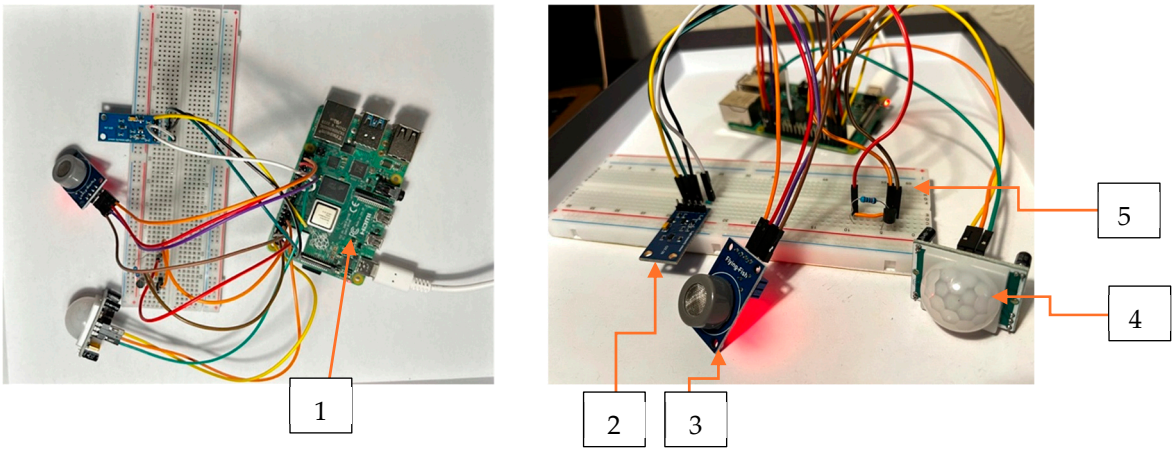


Figure 6. The demonstration of the smart home monitoring system.

The primary objective of the proposed system is to monitor home conditions and manage heating and lighting usage by collecting environmental data. The Python code processes these data and makes decisions regarding the lighting and heating status based on the collected information. Both raw and processed data are sent to the Adafruit IO feed for real-time storage and analysis. The results for the lighting and heating status are presented below:

- When the light is on, the system sends a "1" to the "lighting" Adafruit IO feed.
- When the light is on but there is no movement detected in the room, the system sends a "2" to the "lighting" feed.
- When the system detects that the light is off, the "lighting" feed receives a "0".
- When the heating is on, the system sends a "1" to the "heating" Adafruit IO feed.

- When the heating is on but there is no movement in the room, the system sends a "2" to the "heating" feed.
- When the system detects that the heating is off, the "heating" feed receives a "0".

The integration of IFTTT with Adafruit IO facilitates the energy conservation goals through multiple applets, which notify the user via push notifications during uncertain events. The smart home monitoring system updates the Adafruit IO feed every 30 min, and the IFTTT applets continuously monitor these data to trigger alerts. When specific conditions are met, the system sends notifications to the user.

- If the light is left on without any movement in the room, the system notifies the user: "You have left the lights on". This occurs when the "lighting" feed receives a value greater than 1.
- If the heating is left on without any movement in the room, the system notifies the user: "You have left the heating on". This occurs when the "heating" feed receives a value greater than 1.
- If the home temperature exceeds 50 °C, the system alerts the user: "There is a fire hazard in your home". This happens when the "temperature" feed receives a value greater than 50.
- If the system detects the presence of carbon monoxide gas, the user receives the notification: "Poisonous gas detected in your home". This is triggered when the "gas" feed receives a value greater than 0.

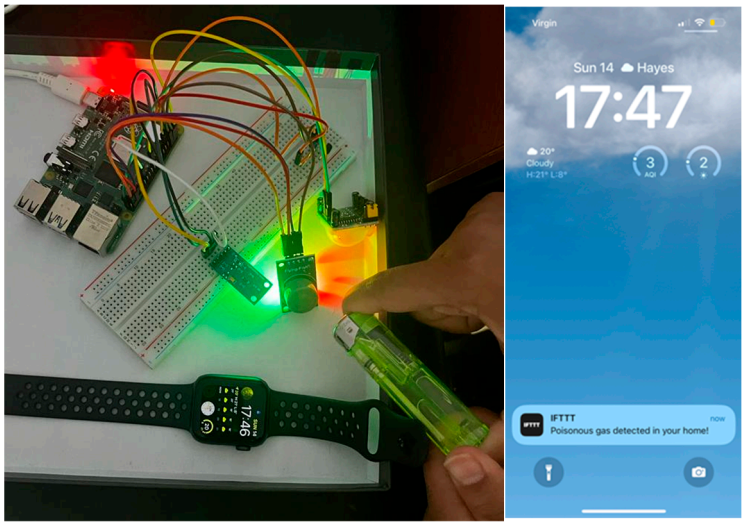
5.2. Results of the System Validation

This section presents the results validating the functionalities of the proposed smart home monitoring system. Our solution effectively monitors home conditions, including temperature, gas levels, light intensity, motion, and the status of the heating and lighting systems. These parameters are monitored through Adafruit IO, with automated notifications triggered via IFTTT. The Raspberry Pi is connected to the Wi-Fi network, allowing for real-time monitoring through the Adafruit IO dashboard and feeds. The Adafruit IO feed has been successfully integrated into the smart home system to store the data collected from various sensors. A dedicated Adafruit IO feed menu page is created, listing feeds such as gas, heating, lighting, motion, and temperature. The system stores sensor data and heating/lighting status updates in real time to their corresponding feeds. The menu page displays essential information, including the feed name, key, the last recorded value, and how long ago that value was updated. By clicking on a specific feed, users can access the feed details page. This page displays the feed name and a graph plotting feed data on the *Y*-axis against time on the *X*-axis. Additionally, users can view and download the data as JSON or CSV files. Options are also available to add or modify data directly within the feed. Extensive testing was performed on the smart home sensors to ensure their functionality and compatibility with the programmed code. Real-time monitoring confirmed that the collected data were being transmitted to the appropriate Adafruit IO feeds promptly, verifying the system's accuracy. The smart home system consistently ensures that the Adafruit IO feed remains up to date, accurately reflecting the current state of the sensors in real time.

5.2.1. Detection of CO Gas

The MQ7 gas sensor was tested for its ability to detect CO levels, demonstrating a fast response time and high sensitivity, capable of identifying even trace amounts of CO in the air. Previous readings consistently showed "0", indicating no CO detection and confirming that the ambient air was free from harmful gases. In Figure 7a, a lighter was used to test the sensor's sensitivity. Upon detecting CO gas, a green LED lights up, confirming successful detection. Additionally, a watch was placed next to the sensor to assess how quickly data

are transmitted, ensuring CO detection is recorded instantaneously. The system has an additional feature that alerts users when CO gas is detected via IFTTT push notifications. Figure 7b confirms that the system sends an alert to the user within a minute of detecting poisonous gas by comparing the timestamp of detection and the timestamp of the alert.



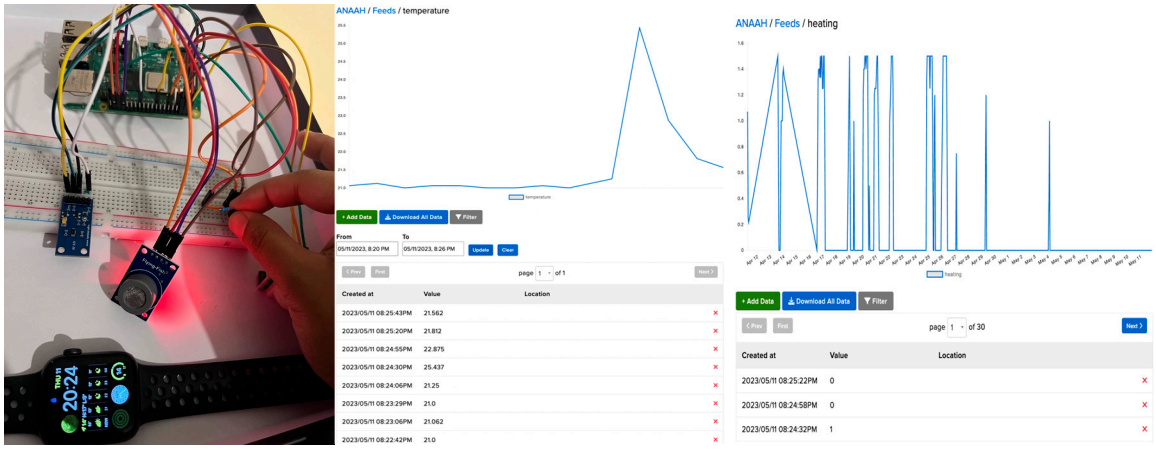
(**a**) (**b**)

Figure 7. Testing the carbon monoxide gas detection, (**a**) Testing the carbon monoxide gas detection, (**b**) IFTTT notification to the user's mobile phone.

5.2.2. Detection of Temperature, Heating, and Fire Hazard

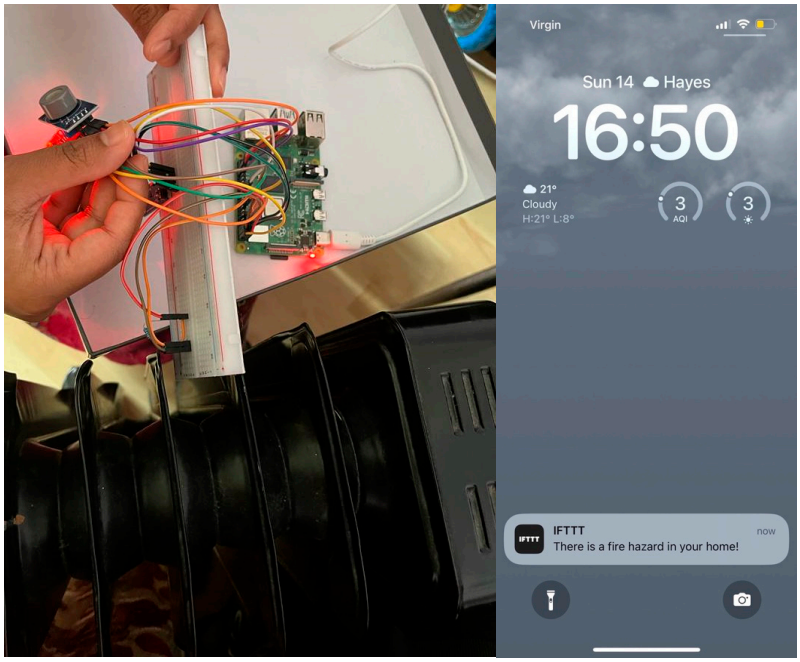
The DS18B20 temperature sensor was tested to ensure proper functionality and accurate readings, which are crucial for determining the heating status in the smart home system. To verify the sensor's effectiveness, it was briefly touched, causing the temperature to rise to 25.437 °C. Figure 8a shows a time-stamped feed at 20:24, corresponding to the highlighted temperature data, confirming that the sensor readings are accurate and that data transmission occurs in real time. Figure 8b illustrates the temperature readings over a brief period. Initially, the readings were stable, fluctuating between 21 °C and 21.6 °C. Upon touching the sensor, the temperature spiked to 25.437 °C, indicated by a sharp upward movement on the graph. After the heat source was removed, the temperature gradually returned to room temperature within a minute, demonstrating the sensor's responsiveness to ambient changes. At the same timestamp, the heating status data indicated a value of "1", meaning the heating was on. The outdoor temperature was 14 °C, while the indoor temperature was 25.75 °C. We configured the system so that if the external temperature is between 12 °C and 17 °C and the indoor temperature exceeds 23.50 °C, the heating system is automatically switched on. Figure 8c confirms that the system correctly activated the heating based on these conditions, demonstrating its ability to make accurate decisions based on real-time sensor data.

The system's ability to detect fire hazards was also tested. For this purpose, the DS18B20 temperature sensor was placed in contact with an electric heater, as shown in Figure 9a, and the temperature readings were recorded in real time. The results demonstrated a steady rise in temperature, reaching 50.05 °C. Once the temperature exceeds 50 °C, the system promptly sends an alert to the user's mobile phone, as shown in Figure 9b, indicating a potential fire hazard.



(**a**) (**b**) (**c**)

Figure 8. Testing the temperature and heating detection, (**a**) Testing temperature and heating detection, (**b**) the temperature readings, and (**c**) the activated heating based on temperature conditions.

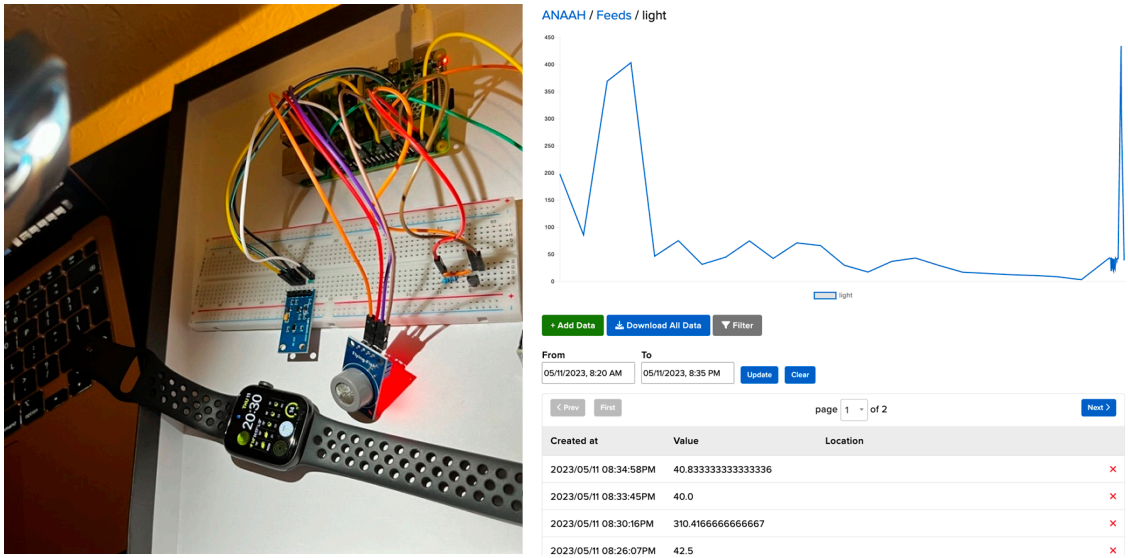


(**a**) (**b**)

Figure 9. Testing the fire hazard detection, (**a**) Testing the fire hazard detection, (**b**) IFTTT notification to the user's mobile phone.

5.2.3. Detection of Light Intensity

The BH1750 light sensor was tested by flashing a mobile phone torch on it, as shown in Figure 10a, to confirm that the sensor accurately responds to changes in light intensity. A reading of 310.30 lux was recorded in the light feed, indicating a significant increase in light intensity due to the torch. Figure 10b confirms this, showing a sharp spike in the graph when the torch was shined, followed by a drop as the light source was removed. The previous data represented the room's ambient light intensity, showing only minor fluctuations in lux. Additionally, the real-time data transmission was verified by placing a watch next to the sensor, ensuring that the light detection and the recorded data in the light feed occurred simultaneously. Overall, the system demonstrated excellent light detection performance, with high accuracy and a fast response time.

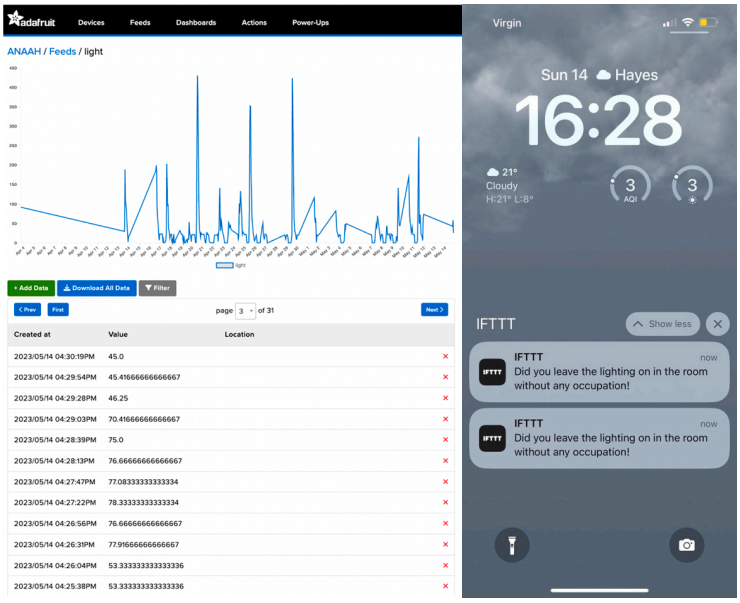


(**a**) (**b**)

Figure 10. Testing the light intensity, (**a**) Testing the light density, (**b**) light intensity readings.

5.2.4. Detection of Lighting Status

The lighting status was tested by leaving the lights on during the daytime to verify whether the system accurately detects when the lights are on, off, or left on without motion detected in the room. A Python script was developed to account for varying light intensities within the room, using a threshold range of 20 to 30 lux depending on different areas. The lights were turned on without any motion to ensure the smart home monitoring system could correctly identify all scenarios. As shown in Figure 11a, the home light intensity reached 53.333 lux at 4:25 p.m. At 4:26 p.m., the lights were turned on, resulting in a light intensity of 77.916 lux, an increase of 24.6 lux, confirming the lighting was turned on. Additionally, at 4:28 p.m., the system detected that the lights were left on without any occupancy, triggering an automated IFTTT push notification to the user's mobile phone, alerting them of the situation (see Figure 11b).



(**a**) (**b**)

Figure 11. Testing of lighting status, (**a**) Testing of lighting status, (**b**) IFTTT notifications to the user's mobile phone.

5.2.5. Detection of Motion

Motion detection serves as a key security measure. When the home is expected to be empty, any detected motion could indicate potential unauthorized access. To test this, motion was simulated by moving hands in front of the PIR sensor while noting the time. The motion was accurately reflected on the Adafruit IO motion feed, where a value of "1" was recorded, indicating motion detection. The timestamp associated with this detection matched the exact time when the motion occurred. This verifies that the system has an accurate motion detection feature, contributing to both energy saving by identifying unoccupied spaces and promoting home security by detecting unauthorized access.

5.3. GUI Demonstration

The Adafruit IO dashboard served as the GUI for managing and visualizing data. This customizable GUI enables users to personalize the interface according to their preferences. Figure 12 shows the GUI of the system, where users can view an up-to-date and accurate representation of the home's current state through various gauges. The temperature gauge displays the current room temperature in degrees Celsius. The motion gauge indicates whether motion has been detected, with "0" meaning no motion detected and "1" indicating motion. The CO gauge works similarly, with "0" indicating no CO detected and "1" showing CO detection. Lastly, the light intensity gauge displays the room's current light intensity. The GUI also includes indicators for heating and lighting status. When the heating is on, the indicator turns green; when the lighting is on, it turns blue; otherwise, the indicator remains red.

Figure 12. The designed GUI for the smart home monitoring system.

The dashboard is customized with several graphs to provide detailed insights into energy usage. The first two graphs display "lighting" vs. "motion" and "heating" vs. "motion", with the *x*-axis representing the timeline of the last 30 days and the y-axes illustrating the correlation between heating and motion, and lighting and motion, respectively. These graphs show motion detection alongside the corresponding heating or lighting status. Users can easily identify instances of energy wastage, which are marked by scenarios where the heating or lighting status is "2" or "1" while motion is "0", indicating that energy is being consumed without occupancy. This feature enables users to detect patterns of energy waste and optimize their energy usage accordingly. The second pair of graphs compares "lighting" vs. "light intensity" and "heating" vs. "temperature". Again, the *x*-axis covers the last 30 days, while the y-axes represent the correlation between lighting and light intensity, and heating and temperature. The "lighting vs. light intensity" graph helps users spot cases where artificial lighting is left on despite sufficient natural light in the room, as indicated by high light intensity values. Similarly, the "heating vs. temperature" graph reveals situations where the heating system is unnecessarily active, even when the room temperature is already high, allowing users to identify and reduce heating inefficiencies. Additionally, the dashboard offers flexible timeline options such as the past 2, 4, 8, and 24 h, or 2, 7, and 30 days, giving users the ability to zoom in and out of specific periods. This functionality aids in closely tracking energy waste trends and adjusting the smart home system to improve energy efficiency.

6. Discussion

This section discusses the results obtained through the testing and analysis of the smart home monitoring system. The outputs of the research are presented, highlighting the system's effectiveness in energy conservation, security, and safety. The key findings include the system's ability to provide real-time monitoring, send automated notifications, and optimize energy usage based on collected data. The system demonstrated its capacity to reduce energy waste, detect potential hazards, and enhance home security. However, the limitations of the research are also acknowledged, including challenges related to scalability, integration with other smart devices, and the need for further testing in more diverse environments to ensure robustness and adaptability across various settings.

6.1. Main Outputs

This research study produced several key outputs in the following areas:

- A comprehensive review was conducted, analyzing existing studies on smart home systems. This review helped identify research gaps and expand the knowledge base, guiding the development of more advanced and efficient solutions.
- Smart home monitoring system design: During the design phase, critical components such as microcontrollers, sensors, cloud services, and notification platforms were carefully selected. This process ensured the implementation of effective and efficient monitoring strategies to optimize home energy usage and safety.
- Smart home monitoring system implementation: The research had two main outputs in this area. The first was hardware implementation, where a Raspberry Pi was connected to various sensors using jumper wires, ensuring proper integration and functionality of each sensor. The second output was software implementation, where a Python code was developed to monitor home conditions, manage data storage and retrieval from cloud services, and make informed decisions regarding heating and lighting based on the collected data. This combination of hardware and software contributed significantly to effective energy monitoring and safety detection within the system.

- Testing and validation: Rigorous testing was carried out to verify the system's functionality, reliability, and performance. Each component was analyzed, and the results confirmed that the smart monitoring system effectively reduced energy consumption and accurately detected safety hazards.
- Optimization: Several optimizations were made throughout the implementation process to improve the system's overall performance. For instance, integration with the OpenWeather API enhanced the decision-making process for heating by using real-time external weather data for more precise control.

6.2. Findings

The developed system focuses on both energy saving and safety by providing real-time monitoring of home conditions and automating notifications. It utilized an intelligent IoT system that incorporates sensors and automation to continuously monitor and control home conditions. The system employs Python code to process data from various sensors, enabling it to make intelligent decisions about lighting and heating based on environmental factors such as light intensity, temperature, and the presence of hazardous gases like carbon monoxide. Using IoT technology, the system ensures efficient energy management and enhanced safety, automatically alerting the user when necessary. This intelligent IoT solution helps reduce unnecessary energy consumption while swiftly detecting potential safety hazards like extreme temperatures or gas leaks. To achieve optimal performance, the most efficient hardware and software components were selected for the system. Previous studies primarily focused on the development of smart homes using microcontrollers like NodeMCU, Arduino, ESP8266, and earlier versions of the Raspberry Pi (up to the 3B+), with goals such as maintaining home comfort, supporting elderly care, improving surveillance, and conserving energy. This research addressed gaps in the literature by employing the more powerful Raspberry Pi 4B microcontroller and integrating more precise sensors, including the BH1750 digital light sensor, DS18B20 temperature sensor, MQ7 gas sensor, and PIR motion sensor. Compared to earlier studies that often used less accurate sensors, this research advanced smart home technology with superior sensor accuracy, resulting in improved functionality and precision.

It was observed that certain sensors, such as the LDR, exhibited reduced precision at higher temperatures, making it difficult for them to detect small changes in light intensity. Similarly, the DHT11 and DHT22 sensors had limitations in accurately detecting a wide range of temperatures. To overcome these issues, the BH1750 digital light sensor was employed for detecting light intensity. This sensor was highly effective, capable of measuring both small and large changes in light levels, and able to distinguish between different light sources, such as sunlight and fluorescent lights. As a result, the system accurately identified unnecessary lighting usage, even when natural light was present. Furthermore, the DS18B20 sensor proved effective in detecting even minor temperature changes, making the system efficient at monitoring the home's heating conditions. The choice of Adafruit IO cloud services facilitated both cloud storage and an intuitive GUI. The dashboard provided a user-friendly interface that enabled real-time monitoring of home conditions, along with visual confirmation of heating and lighting status through color-coded indicators. Additionally, the system's graphical data presentation allowed users to easily identify energy waste patterns and their underlying causes. Energy conservation and safety were further supported through IFTTT push notifications, which alerted users to take immediate action when necessary. By automating these notifications, the system helped prevent unnecessary energy waste, while the detailed graphs gave users deeper insights into how and when energy was being wasted. This combination of advanced sensors,

intelligent automation, and user-friendly interfaces allowed the smart home monitoring system to effectively optimize energy usage while also enhancing the safety of the home.

6.3. Limitations of the System

The developed smart home monitoring system has proven to be highly efficient and effective, contributing to both energy conservation and enhanced safety within a single room. However, there are some limitations that should be acknowledged. The key limitations of the system are as follows:

- Scalability and integration: The current implementation is limited to monitoring only one room at a time. While the system efficiently handles the monitoring of a single room, scaling it to accommodate larger homes or integrating it with other smart devices may present challenges. Additional development and testing will be necessary to ensure that the system remains robust and functional across multiple rooms and devices.
- Inflexibility to sudden weather changes: The system's decision-making regarding heating is based on threshold values tied to external temperature. While this method works well for gradual weather changes, it is less accurate in responding to sudden temperature drops or spikes. As a result, the heating decisions may not be timely or efficient in such cases.
- Machine learning for predictive efficiency: The system currently relies on static threshold values for decision-making, which limits its ability to adapt to evolving energy usage patterns. Integrating a machine learning algorithm could enhance the system's ability to predict energy usage behaviors and make more accurate decisions regarding heating and lighting over time.
- Environmental testing: The system was tested under specific environmental conditions. Further testing in diverse settings and climates is necessary to ensure that the system performs consistently and adapts well to various home environments.

Addressing these limitations would enhance the system's effectiveness in energy conservation and home safety, making it more adaptable, scalable, and responsive to dynamic conditions.

7. Conclusions and Future Work

This research aimed to design and develop an IoT-based technology to conserve energy by monitoring home conditions and appliances. Extensive tests were conducted to evaluate the system's performance and functionality in terms of gas detection, temperature detection, light intensity, heating and lighting status detection, and the automation of notifications. The system has proven to be effective in saving energy by continuously monitoring home environments and alerting users to take immediate actions, such as turning off lights and heating when not in use. This not only enhances energy efficiency but also helps reduce unnecessary energy consumption, leading to lower energy bills. The user-friendly interface, real-time monitoring, and automated alerts provide users with valuable, actionable insights to optimize their home energy usage.

Beyond energy conservation, the smart home system offers several safety features. It actively monitors hazardous gases, such as carbon monoxide, and detects high temperatures, providing protection against fire hazards. This ensures the safety of both occupants and property from potential accidents. The system also improves security by detecting motion when the home is supposed to be empty, alerting users to potential intrusions, and adding an extra layer of protection. The developed IoT system is secure, featuring password protection integrated into the Raspberry Pi. Access to platforms such as Adafruit IO, IFTTT, and the OpenWeather API requires a username and password, providing a robust

level of authentication that restricts unauthorized access. Additionally, the system is fully customizable, allowing it to be tailored to the specific environment in which it operates. For example, light intensity thresholds can be adjusted to match the lighting conditions in different rooms, ensuring the system adapts to the unique characteristics and preferences of each environment for optimal performance and ease of use.

The proposed smart home monitoring system can be expanded in several areas to enhance energy conservation. One potential improvement is the addition of more sensors, such as a non-invasive current sensor and the integration of a smart meter. Non-invasive current sensors can measure current flow without direct contact with electrical connections. By placing them on the cables of individual appliances with high energy consumption, the system can collect real-time data on energy use. Adding smart meters would provide comprehensive data on energy consumption for the entire house. By comparing the data from the smart meter with individual appliance consumption, users can identify which appliances contribute significantly to overall energy use.

Author Contributions: Conceptualization, F.D. and A.M.F.; methodology, A.M.F.; software, A.M.F.; validation, F.D., A.M.F., and M.S.; formal analysis, A.M.F.; investigation, A.M.F.; resources, F.D.; data curation, F.D. and A.M.F.; writing—original draft preparation, F.D. and M.S.; writing—review and editing, F.D. and M.S.; visualization, A.M.F.; supervision, F.D.; project administration, F.D.; funding acquisition, F.D. and M.S. All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Data Availability Statement: The data presented in this study are available on request from the corresponding author.

Conflicts of Interest: The authors declare no conflicts of interest.

Appendix A

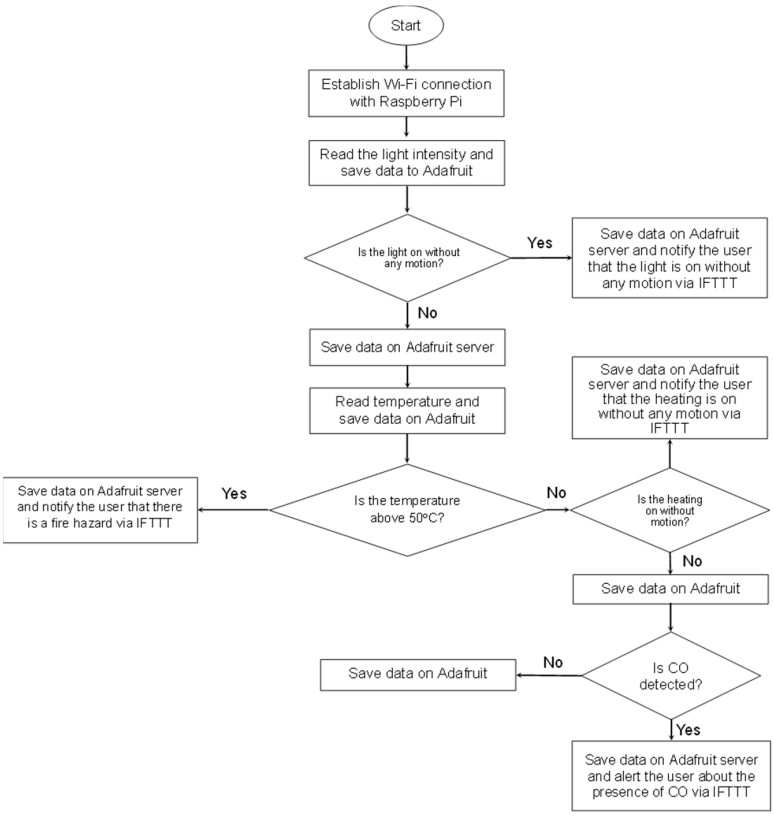


Figure A1. Flow chart of the system operation.

Appendix B

The pseudo-code for the proposed smart home monitoring system.

Setup Adafruit IO, Open weather
Setup RaspberryPi.GPIO
Setup Adafruit IO feeds (temperature, light, motion, lighting, heating and gas) //to upload data in Adafruit database
Setup RaspberryPi.GPIO for sensors //to collect sensor data
L Light intensity //via BH1750 digital light sensor
T Temperature //via DS18B20 sensor
G Gas //MQ7 sensor
M Motion //HC-SR501 PIR motion sensor
Initialise light _intensity to zero
CL Current Light intensity
Raspberry Pi gets sensor data
//Main loop
While true
Set threshold Sensor values
L_Min Minimum light intensity threshold
L_Max Maximum light intensity threshold
T_In_Min Minimum Indoor Temperature threshold
T_In_Max Maximum Indoor Temperature threshold
T_Out_Min Minimum outdoor Temperature threshold
T_Out_Max Maximum outdoor Temperature threshold
Read T and send data to Adafruit feed
Read L and send data to Adafruit feed
Read M and send data to Adafruit feed
//TASK 1
//detect light ON/OFF status
If (L $\geq$ CL + L_Min AND L $\leq$ CL + L_Max) then
    Then lights are ON and send data to Adafruit feed
    If no M detected
        Then light is ON without occupation and send data to Adafruit feed
        (Notify the user light is ON without any occupation via IFTTT)
Elif (light _intensity $\geq$ L_Min AND light _intensity $\leq$ L_Max $\geq$ OR L (light _intensity + L_Min) and L $\leq$ (light _intensity + L_Max))
    Then lights are ON and send data to Adafruit feed
    If no M detected
        Then light is ON without occupation
        (Notify the user light is ON without any occupation via IFTTT)
Else light is OFF and send data to Adafruit IO
TASK 2
Get outdoor temperature from Open weather map API
T_Out outdoor temperature
T_In indoor temperature
Continue
//detect heating ON/OFF status
If T_Out $\leq$ 6 AND T_In < 18

Then heating is OFF And send data to Adafruit feed
(Notify the user heating is ON without any occupation via IFTTT)
elif T_Out $\leq$ 6 AND T_In $\geq$ 18
Then heating is ON And send data to Adafruit feed
If no M detected
Then heating is ON without occupation and send data to Adafruit feed
(Notify the user heating is ON without any occupation via IFTT)
elif T_Out $\geq$ 6 AND T_Out < 8 AND T_In $\geq$ 20.50
Then heating is ON And send data to Adafruit feed
If no M detected
Then heating is ON without occupation and send data to Adafruit feed
(Notify the user heating is ON without any occupation via IFTT)
elif T_Out $\geq$ 8 AND T_Out < 12 AND T_In $\geq$ 22.50
Then heating is ON And send data to Adafruit feed
If no M detected
Then heating is ON without occupation and send data to Adafruit feed
(Notify the user heating is ON without any occupation via IFTT)
elif T_Out $\geq$ 12 AND T_Out < 17 AND T_In $\geq$ 23.50
Then heating is ON And send data to Adafruit feed
If no M detected
Then heating is ON without occupation and send data to Adafruit feed
(Notify the user heating is ON without any occupation via IFTT)
elif T_Out $\geq$ 17 AND T_Out < 20 AND T_In $\geq$ 23.75
Then heating is ON And send data to Adafruit feed
If no M detected
Then heating is ON without occupation and send data to Adafruit feed
(Notify the user heating is ON without any occupation via IFTT)
elif T_Out $\geq$ 19 AND T_Out < 22 AND T_In $\geq$ 24 AND T_In $\leq$ 26
Then heating is OFF And send data to Adafruit feed
elif T_Out < 21 AND T_In > 26
Then heating is ON And send data to Adafruit feed
If no M detected
Then heating is ON without occupation and send data to Adafruit feed
(Notify the user heating is ON without any occupation via IFTT)
Elif T_In > 50
Then fire hazard and send data to Adafruit feed
(Notify the user fire hazard via IFTT)

81. **Else** Then heating is OFF And send data to Adafruit feed
82. TASK 3
83. //detect carbon monoxide gas
84. **Get** the G sensor value
85. **If** value==1 **then** CO gas detected and send data to Adafruit feed
86. (Notify the user poisonous gas detected via IFTT)
87. **Else** CO gas NOT detected and send data to Adafruit feed
88. **REPEAT** the loop every 1800 s
89. **Adafruit IO** displays all the real-time data In Dashboard and feed
90. **IFTT** sends notification to user during abnormal conditions

References

1. Lund, P.D.; Mikkola, J.; Ypyä, J. Smart energy system design for large clean power schemes in urban areas. *J. Clean. Prod.* **2015**, *103*, 437–445. [CrossRef]
2. Cevik, S. Climate Change and Energy Security: The dilemma or Opportunity of the Century? International Monetary Fund, Working Paper No. 2022/174. Published 22 September 2022. Available online: https://papers.ssrn.com/sol3/papers.cfm?abstract_id=4224062 (accessed on 6 January 2025).
3. Dinmohammadi, F.; Han, Y.; Shafiee, M. Predicting energy consumption in residential buildings using advanced machine learning algorithms. *Energies* **2023**, *16*, 3748. [CrossRef]
4. Filho, G.P.R.; Villas, L.A.; Gonçalves, V.P.; Pessin, G.; Loureiro, A.A.F.; Ueyama, J. Energy-efficient smart home systems: Infrastructure and decision-making process. *Internet Things* **2019**, *5*, 153–167. [CrossRef]
5. Büyük, M.; Avşar, E.; İnci, M. Overview of smart home concepts through energy management systems, numerical research, and future perspective. *Energy Sources Part A Recovery Util. Environ. Eff.* **2022**, 1–26. [CrossRef]
6. Stojkoska, B.L.R.; Trivodaliev, K.V. A review of Internet of Things for smart home: Challenges and solutions. *J. Clean. Prod.* **2017**, *140*, 1454–1464. [CrossRef]
7. Rock, L.Y.; Tajudeen, F.P.; Chung, Y.W. Usage and impact of the internet-of-things-based smart home technology: A quality-of-life perspective. *Univers. Access Inf. Soc.* **2024**, *23*, 345–364. [CrossRef] [PubMed]
8. Albany, M.; Alsahafi, E.; Alruwili, I.; Elkhediri, S. A review: Secure Internet of thing system for smart houses. *Procedia Comput. Sci.* **2022**, *201*, 437–444. [CrossRef]
9. Chakraborty, A.; Islam, M.; Shahriyar, F.; Islam, S.; Zaman, H.U.; Hasan, M. Smart home system: A comprehensive review. *J. Electr. Comput. Eng.* **2023**, *2023*, 30. [CrossRef]
10. Ding, C.; Ke, J.; Levine, M.; Zhou, N. Potential of artificial intelligence in reducing energy and carbon emissions of commercial buildings at scale. *Nat. Commun.* **2024**, *15*, 5916. [CrossRef]
11. Sovacool, B.K.; Furszyfer Del Rio, D.D. Smart home technologies in Europe: A critical review of concepts, benefits, risks and policies. *Renew. Sustain. Energy Rev.* **2020**, *120*, 109663. [CrossRef]
12. Aldahmani, A.; Ouni, B.; Lestable, T.; Debbah, M. Cyber-security of embedded IoTs in smart homes: Challenges, requirements, countermeasures, and trends. *IEEE Open J. Veh. Technol.* **2023**, *4*, 281–292. [CrossRef]
13. Marikyan, D.; Papagiannidis, S.; Alamanos, E. A systematic review of the smart home literature: A user perspective. *Technol. Forecast. Soc. Chang.* **2019**, *138*, 139–154. [CrossRef]
14. Badar, A.Q.H.; Anvari-Moghaddam, A. Smart home energy management system—A review. *Adv. Build. Energy Res.* **2022**, *16*, 118–143. [CrossRef]
15. Cano-Suñén, E.; Martínez, I.; Fernández, Á.; Zalba, B.; Casas, R. Internet of Things (IoT) in buildings: A learning factory. *Sustainability* **2023**, *15*, 12219. [CrossRef]
16. Al-Obaidi, K.M.; Hossain, M.; Alduais, N.A.M.; Al-Duais, H.S.; Omrany, H.; Ghaffarianhoseini, A. A review of using IoT for energy efficient buildings and cities: A built environment perspective. *Energies* **2022**, *15*, 5991. [CrossRef]
17. Pan, J.; Jain, R.; Paul, S.; Vu, T.; Saifullah, A.; Sha, M. An Internet of Things framework for smart energy in buildings: Designs, prototype, and experiments. *IEEE Internet Things J.* **2015**, *2*, 527–537. [CrossRef]
18. Hadwan, H.H.; Reddy, Y.P. Smart home control by using Raspberry Pi & Arduino UNO. *Int. J. Adv. Res. Comput. Commun. Eng.* **2016**, *5*, 283–288.
19. Al-Ali, A.R.; Zualkernan, I.A.; Rashid, M.; Gupta, R.; Alikarar, M. A smart home energy management system using IoT and big data analytics approach. *IEEE Trans. Consum. Electron.* **2017**, *63*, 426–434. [CrossRef]
20. Agyeman, M.O.; Al-Waisi, Z.; Hoxha, I. Design and implementation of an IoT-based energy monitoring system for managing smart homes. In Proceedings of the 4th International Conference on Fog and Mobile Edge Computing, Rome, Italy, 10–13 June 2019. [CrossRef]

21. Hamdan, O.; Shanableh, H.; Zaki, I.; Al-Ali, A.R.; Shanableh, T. IoT-based interactive dual mode smart home automation. In Proceedings of the IEEE International Conference on Consumer Electronics, Las Vegas, NV, USA, 11–13 January 2019. [CrossRef]
22. Jabbar, W.A.; Kian, T.K.; Ramli, R.M.; Zubir, S.N.; Zamrizaman, N.S.M.; Balfaqih, M.; Shepelev, V.; Alharbi, S. Design and fabrication of smart home with Internet of Things enabled automation system. *IEEE Access* **2019**, *7*, 144059–144074. [CrossRef]
23. Khattar, S.; Sachdeva, A.; Kumar, R.; Gupta, R. Smart home with virtual assistant using Raspberry Pi. In Proceedings of the 9th International Conference on Cloud Computing, Data Science & Engineering, Noida, India, 10–11 January 2019. [CrossRef]
24. Rashid, R.A.; Chin, L.; Sarijari, M.A.; Sudirman, R.; Ide, T. Machine learning for smart energy monitoring of home appliances using IoT. In Proceedings of the 11th International Conference on Ubiquitous and Future Networks, Zagreb, Croatia, 2–5 July 2019. [CrossRef]
25. Alsaleem, F.; Tesfay, M.K.; Rafaie, M.; Sinkar, K.; Besarla, D.; Arunasalam, P. An IoT framework for modeling and controlling thermal comfort in buildings. *Front. Built Environ.* **2020**, *6*, 87. [CrossRef]
26. Khan, F.; Siddiqui, M.A.B.; Rehman, A.U.; Khan, J.; Asad, M.T.S.A.; Asad, A. IoT based power monitoring system for smart grid applications. In Proceedings of the International Conference on Engineering and Emerging Technologies, Lahore, Pakistan, 22–23 February 2020. [CrossRef]
27. Pujari, U.; Patilb, P.; Bahadurec NManvita, M. Internet of Things based integrated smart home automation system. In Proceedings of the 2nd International Conference on Communication and Information Processing, Pune, India, 26–27 June 2020. [CrossRef]
28. Majumder, A.J.; Izaguirre, J.A. A smart IoT security system for smart-home using motion detection and facial recognition. In Proceedings of the 44th Annual Computers, Software, and Applications Conference, Madrid, Spain, 13–17 July 2020. [CrossRef]
29. Mudaliar, M.D.; Sivakumar, N. IoT based real time energy monitoring system using Raspberry Pi. *Internet Things* **2020**, *12*, 100292. [CrossRef]
30. Sooraj, S.K.; Sundaravel, E.; Shreesh BSireesha, K. IoT smart home assistant for physically challenged and elderly people. In Proceedings of the International Conference on Smart Electronics and Communication, Trichy, India, 10–12 September 2020. [CrossRef]
31. Valov, N.; Valova, I. Home automation system with Raspberry Pi. In Proceedings of the 7th International Conference on Energy Efficiency and Agricultural Engineering, Ruse, Bulgaria, 12–14 November 2020. [CrossRef]
32. Ramelan, A.; Adriyanto, F.; Hermanu, B.A.C.; Ibrahim, M.H.; Saputro, J.S.; Setiawan, O. IoT based building energy monitoring and controlling system using LoRa modulation and MQTT protocol. *IOP Conf. Ser. Mater. Sci. Eng.* **2021**, *1096*, 012069. [CrossRef]
33. Taiwo, O.; Ezugwu, A.E. Internet of Things-based intelligent smart home control system. *Secur. Commun. Netw.* **2021**, *2021*, 17. [CrossRef]
34. Tukymbekov, D.; Saymbetov, A.; Nurgaliyev, M.; Kuttybay, N.; Dosymbetova, G.; Svanbayev, Y. Intelligent autonomous street lighting system based on weather forecast using LSTM. *Energy* **2021**, *231*, 120902. [CrossRef]
35. Desnanjaya, I.G.M.N.; Ariana, A.A.G.B.; Nugraha, I.M.A.; Wiguna, I.K.A.G.; Sumaharja, I.M.U. Room monitoring uses ESP-12E based DHT22 and BH1750 sensors. *J. Robot. Control* **2022**, *3*, 205–211. [CrossRef]
36. Saleem, M.U.; Usman, M.R.; Usman, M.A.; Politis, C. Design, deployment and performance evaluation of an IoT based smart energy management system for demand side management in smart grid. *IEEE Access* **2022**, *10*, 15261–15278. [CrossRef]
37. Arunkumar, M.; Archana, V.; Devi, S.A.; Sri, S.N. Advanced smart home office automation using, I.o.T. In Proceedings of the 5th International Conference on Smart Systems and Inventive Technology, Tirunelveli, India, 23–25 January 2023. [CrossRef]
38. García-Monge, M.; Zalba, B.; Casas, R.; Cano, E.; Guillén-Lambea, S.; López-Mesa, B.; Martínez, I. Is IoT monitoring key to improve building energy efficiency? Case study of a smart campus in Spain. *Energy Build.* **2023**, *285*, 112882. [CrossRef]
39. Umair, M.; Cheema, M.A.; Afzal, B.; Shah, G. Energy management of smart homes over fog-based IoT architecture. *Sustain. Comput. Inform. Syst.* **2023**, *39*, 100898. [CrossRef]