

Research Article

Design of a Secure Wireless Home Automation System with an Open Home Automation Bus (OpenHAB 2) Framework

Robert A. Sowah , Dale E. Boahene, Dalton C. Owoh, Rexford Addo, Godfrey A. Mills, Wiafe Owusu-Banahene, Gifty Buah, and Baffour Sarkodie-Mensah

Department of Computer Engineering, University of Ghana, Accra, Ghana P.O. Box LG 77, Legon

Correspondence should be addressed to Robert A. Sowah; rasowah@ug.edu.gh

Received 28 June 2020; Revised 4 August 2020; Accepted 11 September 2020; Published 30 October 2020

Academic Editor: Rafael Morales

Copyright © 2020 Robert A. Sowah et al. This is an open access article distributed under the Creative Commons Attribution License, which permits unrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited.

There is rapid interest growing in the use of smart, connected devices. The developing world market for smart technology is evolving to adopt and adapt to the interconnected world of devices leading to the Internet of Things (IoT) everywhere. This research paper presents the design, development, and deployment of a prototype for the secure wireless home automation system with OpenHAB 2. We employed the use of two (2) high-performance microcontrollers, namely, the Arduino Mega 2560, interfaced with a 16-channel relay, and Raspberry Pi Model B, running the OpenHAB software. The Raspberry Pi functioned as the server to develop a prototype of an automated smart home that is remotely controllable from both a web application and an Android mobile app. In designing a wireless controlled switch for home appliances, two security procedures were implemented, namely, the token-based JSON Web Token (JWT) interface and Advanced Encryption Standard (AES) procedures for authentication and data encryption. Our system delivered a home automation system that leverages on the power of the latest version of OpenHAB to maximize productivity and overall home security while making it adaptable to the management of individual devices. When tested, both the developed hardware and software modules performed extremely well to meet the goal of a secured home automation system. Industry-standard penetration testing tools and frameworks, including Aircrack-ng, were utilized; wireless network audit began with a full sweep of the wireless frequencies with excellent results. It also ensures the efficient use of energy in the home as devices are intelligently controlled from both mobile and web applications. The results of the design and implementation of the additional layer for the security of the OpenHAB framework provide various theoretical and practical implications for home automation.

1. Introduction

Home automation is the adoption of a system to control lighting, atmospheric conditions, entertainment systems, surveillance systems, and home appliances. It allows for devices in the home to be connected to a remotely controllable network. This technology makes life easier for the user and saves energy by utilizing devices with the most priority and importance [1–5]. Controls can be made as easy as turning off lights with a remote or as complex as setting up a network of items [6] that can program a controller via smartphones from everywhere in the world. It allows for devices in the home to be connected to a remotely controllable network.

The idea of home automation has been around for decades [7], but only in recent times were actual architectural designs being implemented. The earliest form of home automation dates back to 1893 when the first television with a remote control system was patented. The popularity of home automation grew afterward in the early 2000s due to an increase in the demand for various technologies. The need for home automation has continued to increase as more affordable domestic technologies emerge in the market today.

Developing one's smart devices to work with all home automation systems would prove a difficult task as it would require the assistance of the smartphone companies to properly set up these devices to communicate with the home automation systems. This research paper uses OpenHAB 2—the

latest stable version of the software with the Raspberry Pi 3 Model B and Arduino Mega microcontroller to deliver a prototype of a secure wireless home automation system.

In recent times, there have been attempts at bridging the gap between compatibility with appliances of different vendors. A quick but rather inefficient fix for this problem would be to have separate applications for each kind of device [8–10]. OpenHAB serves to overcome most of the common integration issues associated with wireless home automation. It integrates several cross-platform home automation systems into a single solution, acts as a common communicative language among various devices, makes interdevice communication neutral, and makes device integration easy [9, 10].

With recent technological developments, there has been an ongoing debate on the need to switch from traditional home settings to secure automated homes. Most homeowners in developing countries would see this new initiative as an unwarranted luxury. They would point to the cost of installations and the lack of an enabling environment as a significant hindrance. It should, however, be noted that improvements in home automation systems would improve productivity and security in households at affordable costs.

The drawbacks of the traditional home systems are highlighted below: (1) the repetitive nature of most tasks creates room for inefficiency and time-wastage and thus makes the use of home automation essential; (2) the regular traditional homes cannot be easily monitored or controlled. Events going on at different parts of the home cannot be monitored and controlled effectively. The status of the home appliances cannot be checked without the homeowner being physically present; and (3) the cost of installing and maintaining security devices in traditional homes is high, and therefore, inexpensive options to home security should be harnessed [2, 10, 11].

Most homeowners in developing countries lack complete and total control over their homes. They are not able to access vital home automation features such as control and monitoring of home appliances, low-cost security, and efficient energy usage by implication. This paper seeks to explore a way to replace traditional home systems with secure wireless automated systems.

The OpenHAB 2 protocol is relevant to automating the home while interconnecting appliances based on its flexibility and capability for full customization. Therefore, the following objectives are pursued within the context of this research work, namely, (1) provision of an easy control mechanism for home appliances via a mobile or web application; (2) improvement of the security of connected home appliances through the system's inbuilt intrusion detection, alarm, and wireless communication data encryption standards; (3) meeting the essential energy management requirement of the household by providing a means to monitor and remotely turn off unnecessary active appliances to conserve energy and reduce electricity bills; (4) offering a wirelessly controlled switch for all home appliances; and (5) making a significant contribution to the existing body of knowledge on secured home automation.

There are key factors that drive sustainable automation using OpenHAB software. These are specified below:

- (1) *Security*. The OpenHAB software does not enforce any access control mechanism for its users and depends solely on the security of the wireless network. It can, however, be secured through user authentication and authorization.
- (2) *Cost-Effectiveness*. The initial cost of installation is relatively higher than in conventional homes. It has a high maintenance cost. However, inbuilt energy-saving mechanisms could save some running costs and enables optimal utilization of electrical utilities.
- (3) *Alignment to Existing Environmental Infrastructure*. It is not suitable in areas without 24-hour power supply. It becomes a high-security risk, and extra cost can be incurred if external power generating equipment is acquired.
- (4) *Flexibility in Application and Adaptation*. The OpenHAB software is open-source software that ensures flexibility in the adaptation of smart devices. It serves as a hub that brings together a diverse range of heterogeneous devices. Smart home technologies are flexible when it comes to the accommodation of new devices, appliances, and other technologies.
- (5) *OpenHAB Is Central to Smart Homes*. Like electric cars, smart grid, and other next-generation technologies, the smart home promises high market value in the near future. Presently, in most underdeveloped and developing communities, smart homes may be an imprudent acquisition as other supporting platforms are not already in place.
- (6) *Legal Factors*. In many areas, smart home technology is moving faster than the laws governing such technology. For instance, smart homes often come equipped with high-tech security features, such as audio and video recording devices. In many states and jurisdictions, prior consent is required before making an audio or video recording of someone, for instance.

Additionally, a summary of the countries who have adopted smart home automation technology is given in Table 1.

In Africa, however, smart home technology is still at a very infant stage. Nairobi in Kenya is regarded as the smartest city in Africa [13]. South Africa is already implementing smart home technologies, including other African countries like Nigeria, Ghana, Rwanda, and Ethiopia. The calendar specification of the trend in smart homes reveals an upward trend [14].

The invention of home appliances started during the first two decades of the 20th century. The middle of the 20th century saw technological advancement with the invention of the Echo IV and the kitchen computer in 1966. The Echo IV was the first smart device, although it was never

TABLE 1: Cross country summary on smart homes [12].

Rank (number of smart homes)	Country	Number of smart homes (millions)	Percentage of smart homes (% of total homes)
1	USA	40.3	32
2	China	19.3	4.9
3	Japan	7	15
4	Germany	6.1	15.7
5	UK	5.3	19.7
6	South Korea	4	20
7	India	2.2	<1
8	Australia	1.8	19.12
9	Brazil	1.2	1.9
10	Russia	0.9	1.7

commercially sold. It could compute shopping lists and turn appliances on and off. The early 1990s saw the innovation rendered by gerontechnology, which was the invention of technology to aid gerontology, e.g., medical alarms for the aged. In the 1998 and early 2000s, smart home technology took a boom for its creative innovations by different vendors. The home automation market was estimated at US\$5.77 billion in 2013 and is predicted to reach a market value of US\$12.81 billion by the year 2020 [15].

In-depth research was done on existing works and systems that had been implemented around the scope of the research work defined in this paper, namely, to design secure wireless home automation based on privacy by design. The focus of the literature review was to acknowledge the work that was done, establish their strengths and gaps, and access how different ecosystems influence outcomes. Among the many papers of interest, the under listed were considered of direct interest to this proposed research work [1–3, 7–25] and thus were reviewed. Their relative strengths and weaknesses are highlighted in the related work section.

The unique contributions of this paper are (1) design and development of cost-effective, secure home automation using the OpenHAB 2 framework with capability for device programming and customizations and (2) development of mobile and web applications for energy management and switching of connected home devices and interactive visual interface for home automation. It leverages the developed hardware and software modules to provide optimal energy management for the home. While there are studies that explored the application of OpenHAB in home automation, to the best of our knowledge, there is none that we found in the published literature that addresses the prevailing security challenges of the default OpenHAB server infrastructure, especially for connections over the Internet for smart home automation. In this paper, we implemented additional security layers of user authentication and authorization while keeping the overall cost of implementation low and maintaining the ease of deployment for everyday home use. Per our server configuration, the OpenHAB communication through the Internet is made through the JSON Web Token authentication procedure. This process makes it difficult for user identity to be hijacked by a malicious attacker. This approach proved to be more secure than the default Open-

HAB server configuration. Consequently, we leveraged on the open-source OpenHAB REST API to develop a mobile or web application that is flexible and easily adaptable for traditional home use.

Section 2 presents the literature review on smart home automation systems using different techniques and algorithms for energy management and control. It addresses the various home automation technologies with implemented web and mobile apps. It identifies the strengths and weaknesses of such systems. It proposes a novel method using the AES and JWT security protocols for enhanced security and ensures privacy by the design of home automation. The novelty of the system design architecture and methodology, including the concepts, algorithms, activity diagrams, and flowcharts, is presented in Section 3. This section highlights the various modules with their design and procedures for implementation. Section 4 focuses on the actual design implementations with circuit diagrams and simulations done in Proteus software, experimental verification, and corresponding integration of the developed modules. This section provides sufficient details on the testing and performance evaluation of the proposed secure home automation system with associated modules for efficient monitoring and control of home appliances. Finally, Section 5 provides conclusions on the system design and methodology and relevant recommendations for future enhancements.

2. Review of Related and Existing Systems

Jin et al. [1] implemented a remotely controlled home automation system based on the wireless sensor network and an embedded system with GPRS integration. The system was unique for good reasons; it allowed users to control the equipment in the home and collect data about appliances' status as well as weather conditions. Additionally, the system possessed powerful security features like the inbuilt fire detection and home intrusion detection and notification system. The system is tightly integrated with the Chinese instant mobile messaging service.

A significant drawback of this system is the absence of a mobile or web application interface to control the home appliances. The reliance of the SMS system alone for

communication meant that if the GSM device is misplaced, access control will no longer be valid.

The authors in [6] designed a system to give the homeowner complete control over their home using the Short Message Service (SMS) communicating over GSM. In this system, a GSM modem is connected to an RS232 interfaced with a MAX232 that is also connected to a PIC microcontroller, as the RS232 is not compatible with the microcontroller. When a user sends an SMS message, it is sent to the RS232, and the MAX232 converts the text messages to a TTL signal for the PIC microcontroller. The PIC controls the relay, which is the connection point for all the home appliances [6].

The gap in this system is that it is highly intrusive and requires a rewiring of electrical appliances for configuration purposes. The system is also limited in the use of only a GSM, and a SIM card is needed to control the home remotely, and if the GSM or SIM is misplaced, the system ceases to function. Additionally, no security standards were put in place to secure the system against intruders and hackers.

Kodali et al. [7] proposed the IoT-based smart security and home automation system, which utilizes certain features of the IoT domain for both communication and control of devices while ensuring their secure operation without interruption. It implemented a security protocol for communication between appliances and the server.

Domb proposed the smart home based on the Internet of Things. It incorporates all the basic features of home automation via communication and control of devices in [26].

This paper [27] by Gupta et al. presents the design and development of an effective system in managing home power consumption. The system primarily gathers information about the various home appliances and monitors and controls their performance for the most efficient power consumption scenario. The authors made use of a Power Line Communication (PLC) with a Power-Controlled Outlet Module (PCOM) to assist in the operation of the system. It consists of a home server connected to the module's network in a star topology arrangement.

However, the lack of an Internet connection is a significant drawback of the system as computation and control of home appliances cannot be made remotely. The lack of remote access to the system poses several problems, the most significant being that the system is less efficient in cases where emergency access is needed.

Ramljak [28] published the security analysis of open home automation bus systems and proffered excellent insight into the vulnerabilities and how to mitigate them effectively. The main focus was security analysis of the OpenHAB framework. It was done by traversing the security architecture and supported features that come with OpenHAB due to the apparent challenge in static code analysis of several used packages in OpenHAB. The author minimized vulnerabilities such as posting data to REST API, which can lead to leakage of user data by setting the X-XSS-Protection HTTP response to "1." The second approach the author used was to run OpenHAB behind the reverse proxy, which redirects client requests to the appropriate server, allowing access to OpenHAB runtime ports 80 (HTTP) and 443 (HTTPS).

Heimgaertner et al. [29] sought to help reduce energy costs and increase the comfort of living by adjusting room temperature according to schedules, rules, and sensor inputs. It utilizes the distributed OpenHAB Distributed Multiuser (DM) setup with extensions introducing user authentication, access control, and management tools for decentralized OpenHAB node deployments.

The limitation of this work is that the new OpenHAB DM is not compatible with existing mobile OpenHAB apps. However, this is mainly because the mobile apps do not currently supply the implementation of user credential functionality as well as provide authentication tokens to the REST API.

Celtek et al. [18] designed and implemented a low-cost, effective wireless home automation system with a general purpose application. The web-based application integrates with existing homes to control temperature, humidity, motion, and luminosity sensors along with the signal conditioning circuitry from user-inputted data. The home automation system consists of three key segments: the sensor nodes (SNs), the actuator nodes (ANs), and the center node (CN).

However, the absence of a mobile application interface reduces the ease of access to the system. The system does not define access roles and privileges for users of the systems, and this makes the system susceptible to unauthorized access. No security measures were taken to encrypt the information sent out by the system over the Internet.

Bhatt and Patoliya in [19] designed and implemented a scalable, cost-effective sensor wireless network for transforming the traditional home into a smart home. They deployed a heterogeneous sensor and actuator nodes based on wireless networking technologies for a home. The system has a scalable architecture, and thus, any number of home appliances can be controlled and monitored on the system. Additionally, the system can easily integrate existing wireless home appliances to present a single interface for the control of all home appliances. The authors employed the use of the ESP8266 WiFi module, and that posed a challenge being that it only provides a single point for analog input. That made it difficult to interface multiple analog sensors to the system. The authors, however, noted that integrating a separate Analog IO Expander circuit with the WiFi module would resolve the issue.

Song et al. [21] designed and implemented a wireless controllable power outlet system for home automation networks. Their power outlet module integrates a Zigbee radio that serves as an actuator node in the home automation networks. The smart home designed by the authors makes it easy to control various home appliances. However, a notable limitation of their research paper is the absence of a security standard for the wireless communication modules.

Karaca et al., in their paper [22], designed a smart home system employing the use of sensors and controller nodes connected to a custom local wireless network—the developed system functions in two primary ways: monitoring and control of home appliances. With the use of an embedded server, users can modify the database over an Internet connection. This embedded server is composed of an ARM microprocessor, security alarm horn, ESP8266 WiFi module, and an SD

card memory. The server is a bridge between the user interface and the sensor nodes/controllers. Server modules feed the user with required information by fetching data from the database, which is updated by the sensor nodes. This system does not implement any security standard to secure the wirelessly transmitted data to protect it from external hacks. Additionally, the authors recommend that future studies should be directed towards learning algorithms that enable adaptive control and machine learning control of parameters such as temperature increments in unit volume and time.

Due to fast steep growth in the usage and reliance on striking features of smart devices, Vikram et al. [23] proposed a methodology to provide a low-cost home automation system using WiFi. It embodies the concept of the internet-working of smart devices. The experimental rig involved the use of the ESP8266 WiFi module, ATmega microcontrollers (μ Cs), Single-Pole Double-Throw (SPDT) relays, Transceiver NRF24L01+ RF, etc. One of the μ Cs was used as a hub, and all sensor readings are sent through the hub from the nodes via the RF modules.

A mobile application was developed with a user interface for controlling the home appliances. This paper failed to address the security aspect of the home automation system. The system does not provide access control levels for the user; hence, anyone can be granted permission to intrude the house. Also, since the paper made an emphasis on their system being WiFi-based, it failed to outline the security measures to handle external intrusions from hackers.

Gyory and Chuah in [24] proposed the design of an IoTOne solution that supports heterogeneous IoT devices and avoids both the security vulnerabilities and the limited device compatibility issues. The IoTOne solution supports heterogeneous IoT devices and provides security checking of IoTApp such that only IoTApp with robust codes would be hosted on the IoTOne App store. However, their system lacks support for COAP (Constrained Application Protocol) and DTLS (Datagram Transport Layer Security) features, which are essential for energy-efficient secure communications.

Shinde and Dube [4] developed an IoT-based smart energy management system to intelligently control the appliances wirelessly rather than just switching devices on or off. In this paper, the power consumption of appliances was measured in intervals of 30 minutes and sent to the Raspberry Pi. A significant drawback of the system is that it was designed for controlling only two appliances, i.e., electric fan and light bulb. Additionally, no security measure was devised, which is not safe due to the rigorous intrusion that is always attempted by hackers.

The paper published by Sowah et al. [30] provides a useful solution to the problem of the programmability and customizability of a modular home automation system. Three main features characterize the system: status, control, and automation of electrical and electronic devices. These three features are achieved through the robust hardware consisting of a router, system controller, wireless communication module, microcontroller module, fan controller module, lamp controller module, and IP camera. Primarily, the system uses Android application as a user interface to enable

communication between all system modules. There is a REST API server to facilitate communication between the database and the WebSocket. While the system works well in achieving the modular design for a home automation setting, it does nothing to secure the wireless network against data breaches and attacks [30, 31].

In these papers [3, 32], the authors designed an automated system for controlling lights and fans. A simple Graphical User Interface (GUI) was designed to make it easy for the end user to access data from a web interface. The system architecture comprised a Raspberry Pi, which serves as the Main Controlling Unit (MCU) and Input/Output Interface (IOI). Raspberry Pi was chosen as the MCU for good reasons—it is user-friendly and relatively cheap. With the Raspberry Pi as the central hub, the input can be fetched from the connected sensors and all the data are sent to the Raspberry Pi. However, the absence of a mobile application interface to the system reduces the ease of accessibility and enhanced functionality of the system. The system lacked security features to prevent network sniffing and session hijacking, which is very common with web-based applications.

All the reviewed papers were notable for successfully internetworking various home appliances and devices for monitoring and control purposes. Most of the papers reviewed were either web-based or Android applications that provided users with an interface to monitor and control the states of their home appliances. This was usually done through reliable and functional wireless communication modules. A few of the papers reviewed considered a security evaluation of the OpenHAB server software. Overall, most of the systems we considered proved to be low cost and user-friendly and proved to be ideal for monitoring and remote control purposes. A few of the systems maintained support for heterogeneous devices and implemented security vulnerability checks.

A closer look at the systems, however, revealed some gaps. Some of the systems failed to provide access control levels for users, making it easy for intruders to be granted access permissions. Other systems were found to be lacking support for multiple devices from different vendors due to compatibility issues. Some of the systems reviewed lacked remote control functionalities, and a few others did not implement any security mechanism for the wireless communication protocols. These gaps identified from the literature review informed our problem statement and possible solution approach in developing a secure wireless home automation system with an open home automation bus (OpenHAB 2) framework by leveraging and improving upon the default server architecture security features to enable privacy by design and control of home appliances via mobile and web apps.

3. System Design and Implementation

In the system design and implementation phase of the proposed solution to the identified gaps in the reviewed literature, both software and hardware tools were utilized. These tools are well suited for achieving our desired goal of building

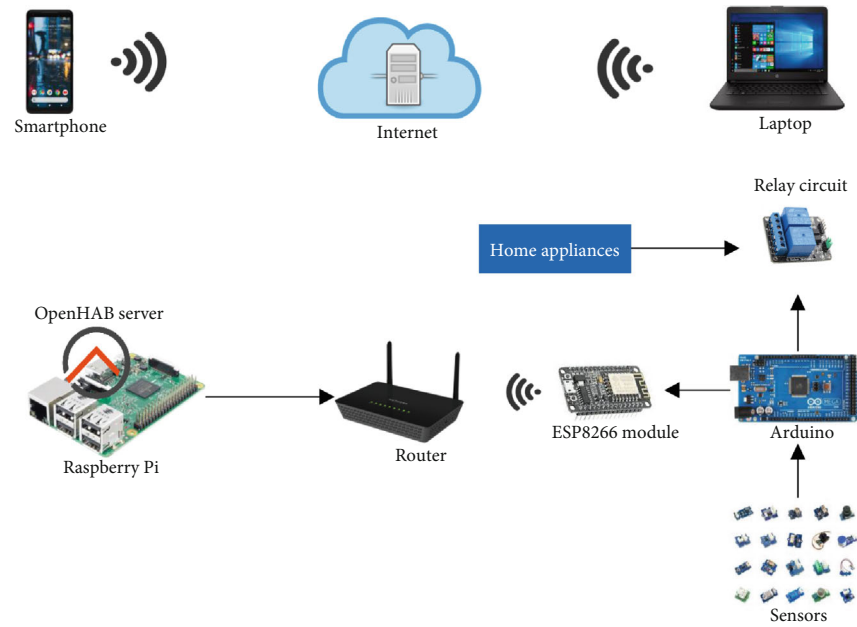


FIGURE 1: System architecture.

a secure wireless home automation system. In building the designed prototype, readily available circuit components were employed. The cost, efficiency, implication, and suitability of each of the components used in the prototype design were carefully weighed. The development tools used in this research paper were effective in helping to deliver the set objectives using the OpenHAB framework.

From the schematic shown in Figure 1, the system has two distinct parts: the wireless control system and the home network system. The smartphone and personal computer, acting as wireless controllers, are connected wirelessly to the network directly or over the Internet to enable remote control of home appliances as well as communication on updated states of the appliances. The Raspberry Pi microcontroller has the instance of OpenHAB software installed on it and acts as the central server for the system. It also hosts the MQTT server, which is a publisher- and subscriber-based protocol that allows multiple devices to communicate with each other over a wireless network [7]. The in-house router logs the IP addresses of the different devices connected to it and further acts as a central connection point for all the devices and appliances. The Arduino microcontroller communicates with the router via the ESP8266 module. It receives and broadcasts the state of the appliances via the MQTT protocol, where all clients can either subscribe or publish their status. An automated or user-queried action is then taken based on the subscribed device state. It is further connected to sensors that pick up signals from the surrounding environment. The Arduino microcontroller receives the command via the ESP8266 and effects the change via actuators (relay circuits and motors), which is essentially the connection point for the various home appliances.

The central server can be accessed by a user authorized by the system with an email address and a password. The connection can either be made through a web interface or

custom-designed Android application. It provides the user with the necessary data stored in the database. Upon gaining access to the central server, based on the information given, the user can then make queries (send commands). Figure 2 is the activity diagram of the overall system architecture that captures the flow of data and relates the communication and control of home appliances to achieve a smart home using OpenHAB 2. Similarly, Figure 3 shows how the system responds to the users' queries and instantaneously sends feedback. The responsiveness of the designed architecture was warranted because, in smart homes, appliances must respond to either switching commands or voice commands.

3.1. Circuit Design and Simulation. The circuit design and simulation using Proteus software was carried out first to ascertain the behavior of the various hardware modules when they are integrated. It included the power supply unit with the relays connected. The current and voltage sensor units are also displayed in Figure 4. The switching operation for energy management utilizes a single channel relay. The appliance switching control schematic using Proteus is shown in Figure 5.

The voltage sensor is used to monitor, calculate, and determine the voltage supply. This sensor can determine the AC or DC voltage levels of the connected appliance. The input to this sensor can be the voltage, whereas the output is the switches, an analog voltage signal, a current signal, an audible signal, etc. Some sensors provide sine waveform- or pulse waveform-like output, and others can generate outputs like AM (Amplitude Modulation), PWM (Pulse Width Modulation), or FM (Frequency Modulation). The measurement of these sensors can depend on the voltage divider.

On the other hand, a current sensor is a device that detects electric current flow and generates a signal proportional to that current flow. The generated signal may be

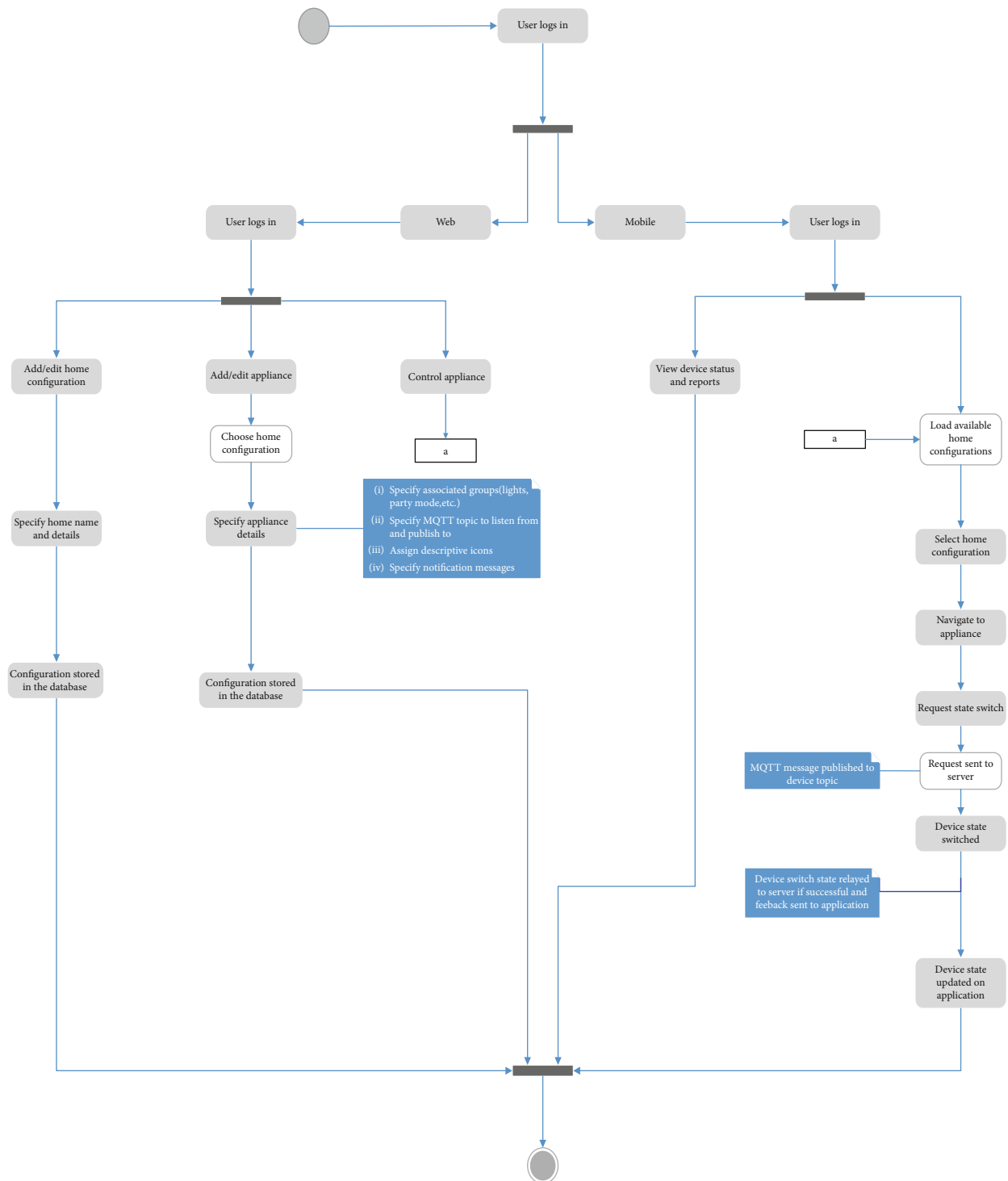


FIGURE 2: Activity flow diagram for the system architecture.

analog voltage or current, which can be utilized to display the measured current in an ammeter or stored for further processing and analysis in a data acquisition system or used for control of home appliances.

3.1.1. Main Hardware Components. Raspberry Pi. The model used in this research paper (Raspberry Pi 3 Model B) hosts a lightweight Linux distribution with OpenHAB installed. It

also has the Mosquitto MQTT broker installed. It communicates with the network by connecting either via Ethernet cable or wirelessly to the router.

Arduino Mega 2560. It acts as a control/logic unit for appliances that do not have a wireless switch. It saves the channel each appliance subscribes to or sensors publish to. It also interprets the MQTT messages sent and forwards the appropriate instructions to specified appliances.

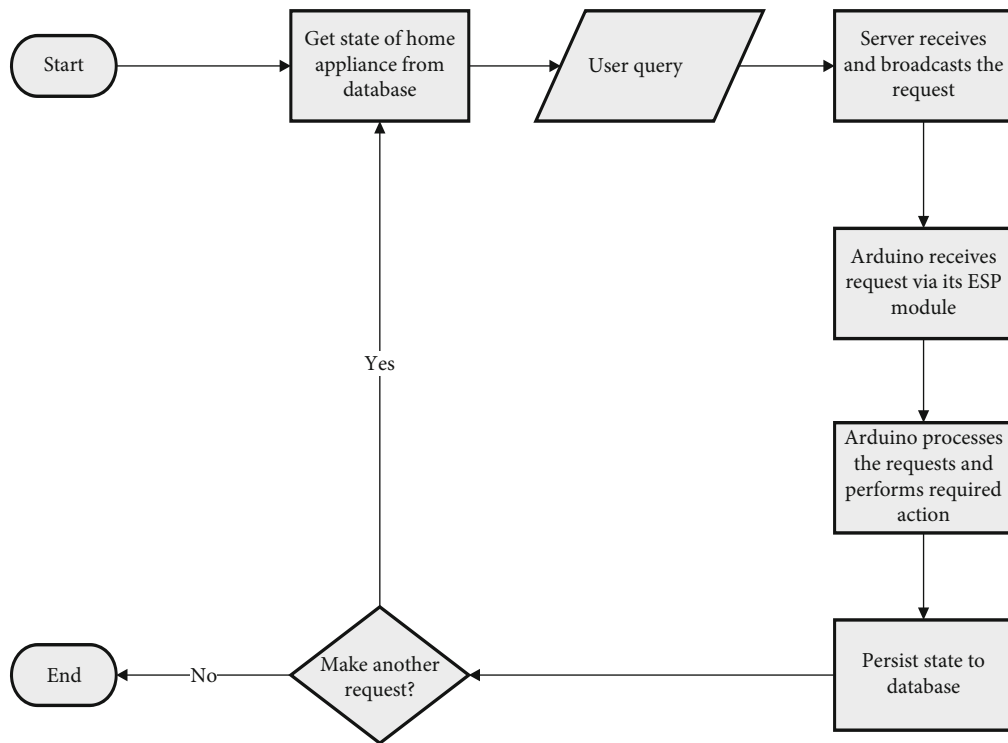


FIGURE 3: System flow diagram.

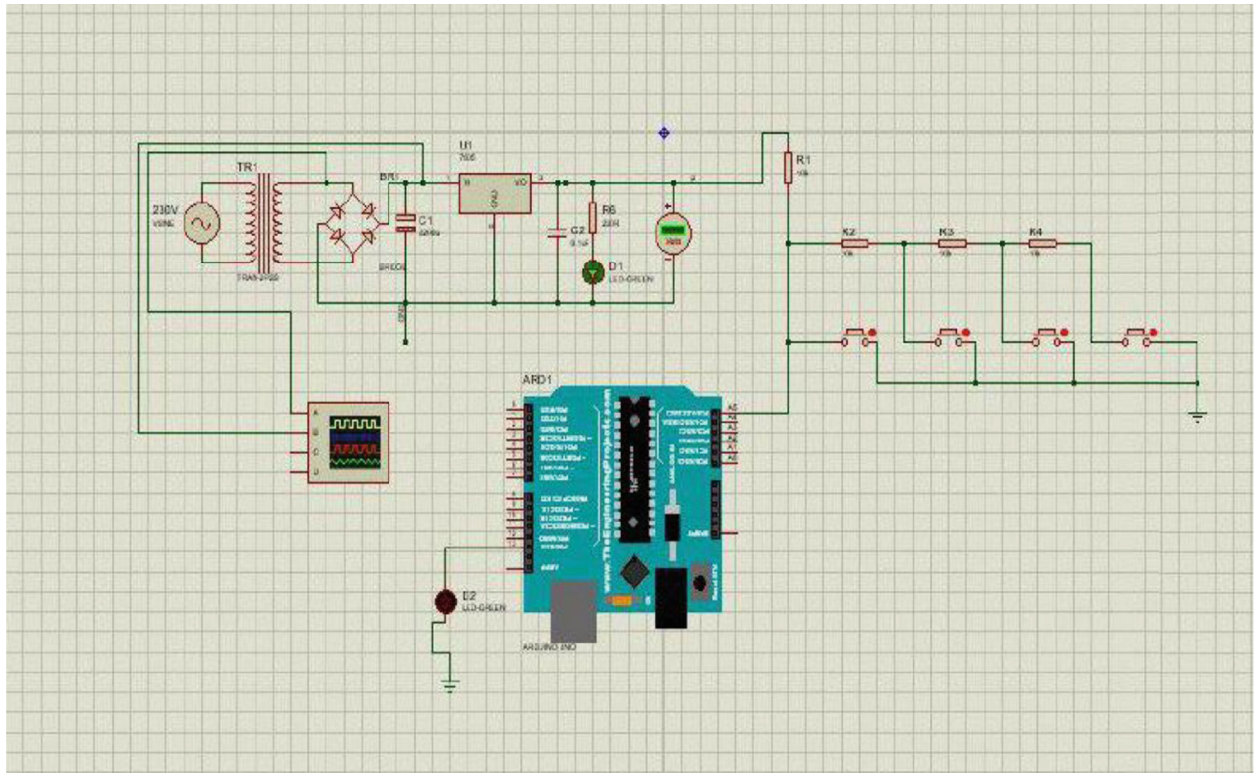


FIGURE 4: Circuit simulation for switching.

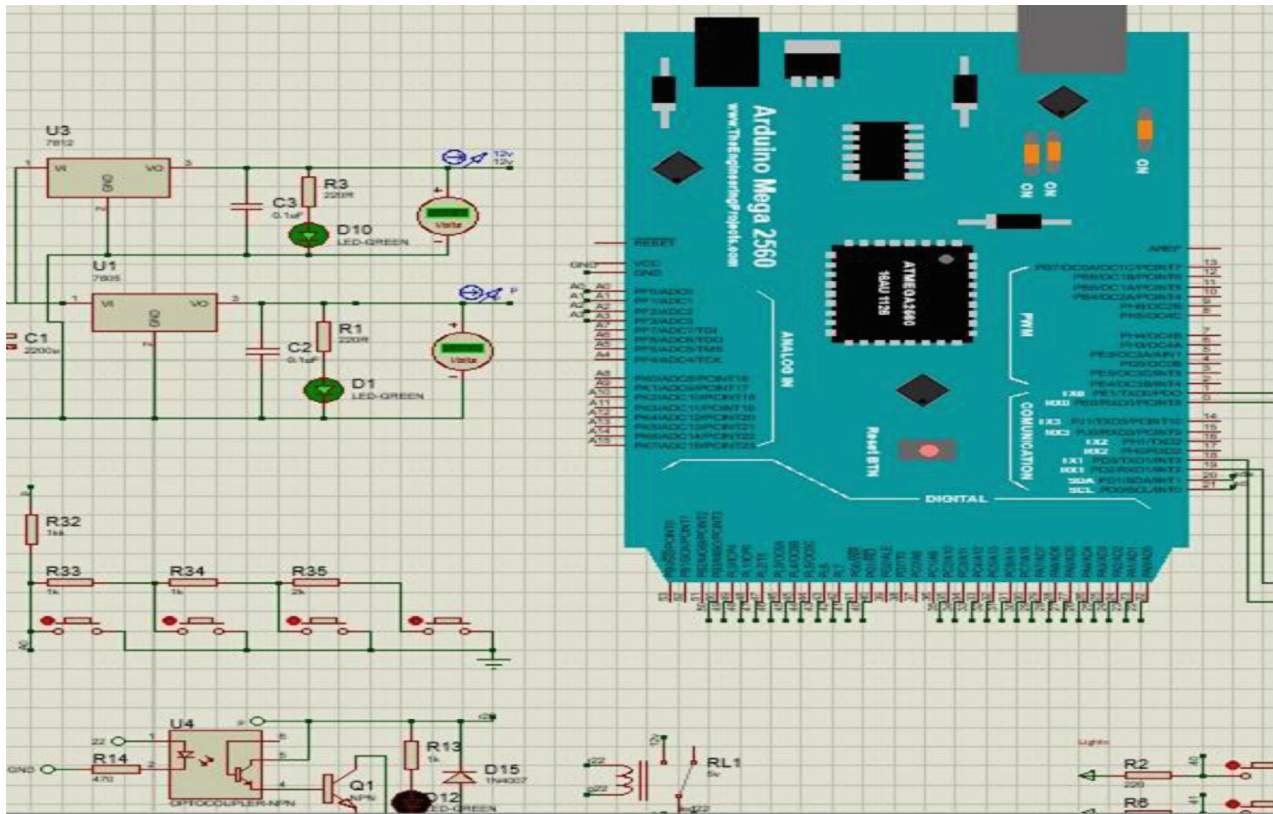


FIGURE 5: Appliance switching control circuit schematic.

NodeMCU (ESP8266) module. It enables wireless communication between the Arduino Mega 2560 and our Raspberry Pi. It also allows individual wireless switches within our system to communicate with the central server.

Relay modules. The relay module used was a 16-channel relay module. The 16-Channel 12V Relay Interface board allows for appliances to be controlled directly from the Raspberry Pi or Arduino microcontroller. This relay module allows a small-level signal to control any normal outlet or high-voltage product rated under 2000 watts.

Other hardware components are jumper wires, LEDs, capacitors, resistors, push buttons, step-down transformers, sensors (temperature, humidity), buzzer module (Piezo speaker), voltage regulator LD1117V33, and NPN bipolar transistors.

4. Hardware Programming and Application Programming

Every part of the system has some level of programming to enable flexibility and adaptable control to achieve the complete home automation system. The design software environments used in this paper for the hardware and application programming include the Arduino Integrated Development Environment (IDE), Android Studio, Visual Studio Code, Proteus Simulation Software, Fritzing, Etcher, PuTTY, and Vim. All code on any Arduino component was done with the Arduino IDE; the Android application was built with

Android Studio, the web view with Visual Studio Code, and configuration on our OpenHAB instance on the Raspberry Pi done over SSH using PuTTY with the Vim editor.

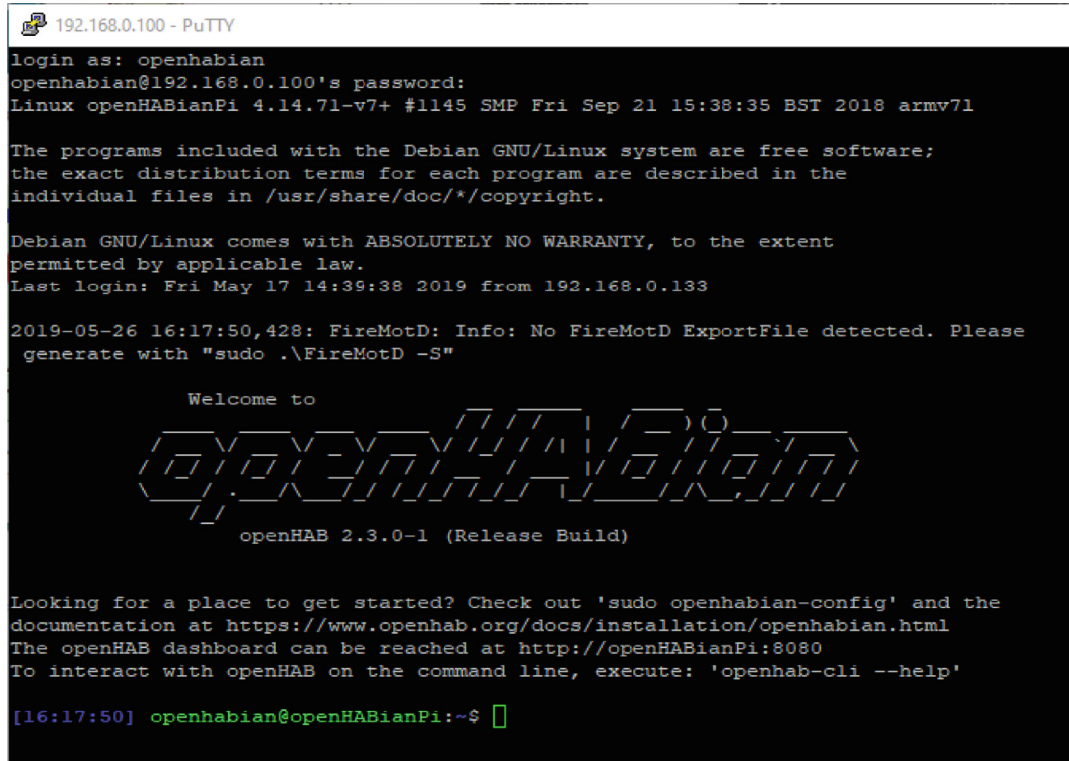
The different sections of the entire developed secure home automation system are as follows.

4.1. The Central Server. To set up the central server, a custom image of Linux distribution with an instance of OpenHAB called *openHABian* is loaded onto a memory card inserted into the Raspberry Pi using Etcher. The memory card is loaded, and the Raspberry Pi is then connected to the Internet to enable updates and initial setup. After 30 to 45 minutes, relative to Internet speed, the process is completed.

The openHABian instance is connected via SSH using PuTTY, as shown in Figure 6.

The configuration for the appliances is implemented in the corresponding files for each of the units below:

- (1) *Items.* This refers to the individual control devices and elements in our system. Each item can be given a name and assigned to groups with specific bindings.
- (2) *Sitemaps.* The sitemap is the interface the user interacts with when the OpenHAB mobile or web application is opened. The various button and information layouts can be controlled from the sitemap.
- (3) *Rules.* This is where the home automation logic is defined. Schedules and conditions required for actions to happen are all defined in this module.



```

192.168.0.100 - PuTTY
login as: openhabian
openhabian@192.168.0.100's password:
Linux openHABianPi 4.14.71-v7+ #1145 SMP Fri Sep 21 15:38:35 BST 2018 armv7l

The programs included with the Debian GNU/Linux system are free software;
the exact distribution terms for each program are described in the
individual files in /usr/share/doc/*/copyright.

Debian GNU/Linux comes with ABSOLUTELY NO WARRANTY, to the extent
permitted by applicable law.
Last login: Fri May 17 14:39:38 2019 from 192.168.0.133

2019-05-26 16:17:50,428: FireMotD: Info: No FireMotD ExportFile detected. Please
generate with "sudo .\FireMotD -S"

Welcome to
  _____
 /  _  _  _  \
|  _ \| | | | | | |
| |_) | | | | |
|  _ \| | | | |
|_| \_|_|_|_|_|
openHAB 2.3.0-1 (Release Build)

Looking for a place to get started? Check out 'sudo openhabian-config' and the
documentation at https://www.openhab.org/docs/installation/openhabian.html
The openHAB dashboard can be reached at http://openHABianPi:8080
To interact with openHAB on the command line, execute: 'openhab-cli --help'

[16:17:50] openhabian@openHABianPi:~$

```

FIGURE 6: PuTTY view after a successful login.

```

//This is the Items file
Group Home Secure Home <house>
Group MasterBedroom Master Bedroom <bedroom_red> (Home)
Group LivingDining Living & Dining Room <sofa> (Home)
Group Kitchen Kitchen <kitchen> (Home)
Group KidsRoom Kids Room <girl_3> (Home)
Group FrontYard Front Yard <lawnmower> (Home)
Group Bathroom Bathroom <bath> (Home)
Group Toilet Toilet <toilet> (Home)
Group Corridor Corridor <corridor> (Home)
Group Garage Garage <garage> (Home)
DateTime Date Date [%1$tA, %1$td.%1$tm.%1$tY] {channel= ntp:ntp:demo:dateTime }
DateTime Time Time [%1$tH:%1$tM] {channel= ntp:ntp:demo:dateTime }
Switch MasterBedroom_Light Light <light> (MasterBedroom, gLight) [ Lighting ] {mqtt= >[home_
broker:master/light/1:command:*.default],<[home_broker:master/light/1:state:default] }

```

CODE Listing 1: Sample Items configuration.

```

sitemap home label= My House
{
  Frame {
    Group item=Lights label= All lighting icon= hue
    Group item=Bedroom label= Bedroom icon= bedroom
    Group item=Office label= Office icon= desk
  }
}

```

CODE Listing 2: Sample Sitemap configuration.

```
//This is the Rules file
rule MasterBedroom_Light
when
  Item MasterBedroom_Light changed
then
  sendNotification( dhalehk@gmail.com ,"Master Bedroom Light turned  +MasterBedroom_Light.state)
end
rule LivingDining_Light
when
  Item LivingDining_Light changed
then
  sendNotification( dhalehk@gmail.com ,"Living & Dining Light turned  +LivingDining_Light.state)
end
```

CODE Listing 3: Sample Rules configuration.

4.2. Wireless Switch Circuit. This module acts as a wireless means of control to be attached to appliances without a direct connection to the main control, the Arduino Mega 2560. The switch is made up of a power supply system, a NodeMCU module, and a 5V rated single channel relay to control 10A 250VAC, 10A 30VDC loads. As shown in Figure 7, the relay terminal (COM, NO, and NC) is brought out with the integrated screw terminal. The LEDs indicate the status of the relay at any given time.

The control signal port of this relay is connected to a digital input/output port of the NodeMCU module. The signal HIGH or LOW, to activate or deactivate the relay module, is based on values received wirelessly by the NodeMCU via MQTT. The value received is based on the NodeMCU configured subscription. Both modules are powered by the power supply unit, which taps from the mains and steps down the voltage to their respective requirements.

Figure 8 shows the implementation of the wireless switch with the relay, NodeMCU module, and power supply system assembled into a single unit.

4.3. Switch Control Unit. Appliances that do not have a wireless connection (wireless switch) to the central server are controlled and monitored via the switch control system. The Arduino Mega 2560 acts as the main brain for control and has all sensors connected directly to it, mostly via serial port communication. To send messages to the central server (Raspberry Pi), readings from the sensors are gathered and managed on the Arduino Mega. The Arduino Mega is then programmed to format the readings together with their respective topics, which are then sent serially to a NodeMCU module connected to the Arduino Mega. The NodeMCU module then forwards the received message to the server wirelessly using the MQTT protocol. On the other hand, to receive messages, the NodeMCU module listens for broadcasts to all the topics of sensors and appliances connected to the Arduino Mega by subscribing to the MQTT broker on the central server. When a new message is published to any of the topics, the MQTT broker broadcasts the received message to all subscribed parties (respective NodeMCU modules).

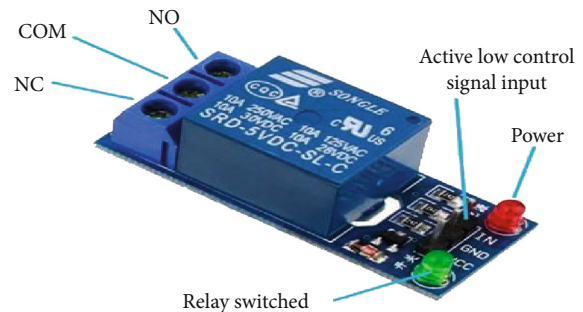


FIGURE 7: A single channel relay circuit.

The message is received and passed serially to the Arduino Mega, which then parses the message to extract the control command. The command is then relayed to the specified appliance the topic belongs to via the actuator (relay) it is connected to. In case that the homeowner is within the premises and does not have immediate access to either an accessible web device or the Android app, the push buttons were used as manual switches in a manner that each press switches the state of the appliance. That is, an LED is turned on if it is OFF and off if it is ON. Input is debounced so that a single press would not appear to the Arduino code as multiple presses. Another reason for implementing the manual switch is to have an alternative method for switching in case the wireless network fails at any point. Figure 9 shows the Raspberry Pi connected to the router. The router records a permanent IP address for the Raspberry Pi. This enables the setup to establish connections between the server and the other appliances.

4.4. Web and Mobile Programming and Interface. The principal requirement of the home automation system to monitor and control the vast number of devices remotely and without hassle is achieved by using both a web interface and an Android application. While building the mobile app, the key consideration here was the platform needed for the development. Within the last four (4) years, Apple iOS and Google's Android operating system have dominated the

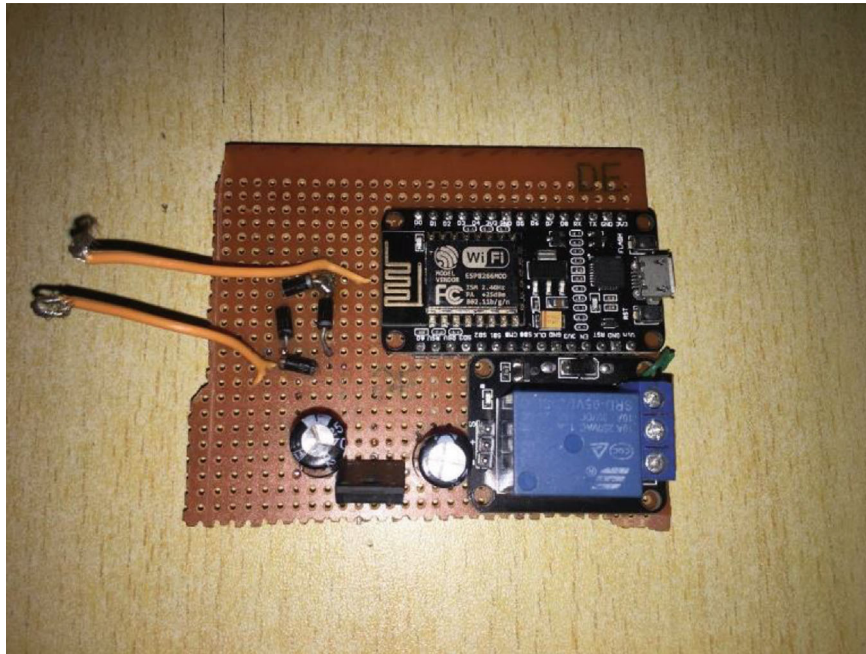


FIGURE 8: Wireless switch prototype implementation.



FIGURE 9: Raspberry Pi server and router setup.

mobile operating system market. Android remains the most popular mobile operating system in the world, having almost 75% of the global market share, according to StatCounter Global statistics [16]. The choice was made to build an Android application to cater for a considerable percentage of the users. While not wholly leaving out iOS in the development process, most of the customization work done on the OpenHAB application was mainly implemented on the Android app and written purely in Java.

In Figure 10, the various use cases for the application are shown. It summarizes the functions available to the home-

owner via the app. To demonstrate the functionality of the Android application, Figures 11 and 12 show authentication, which is a security feature on the smartphone application. For remote connections, even when not at home (local server), the remote connection to the OpenHAB cloud service is required. Otherwise, the user enters their details to be authenticated for access to the local server.

In Figures 13 and 14, the interface for the mobile application is shown. Figure 13 shows the main menu view for the sitemap of a given home. Figure 14 goes on to show the various rooms or spaces on the first floor of the home.

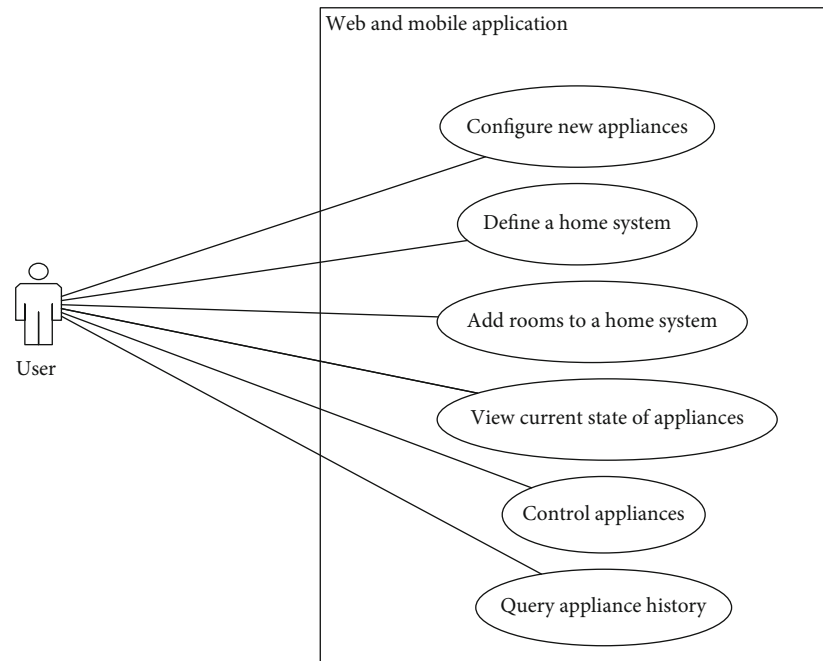


FIGURE 10: Application use case diagram.

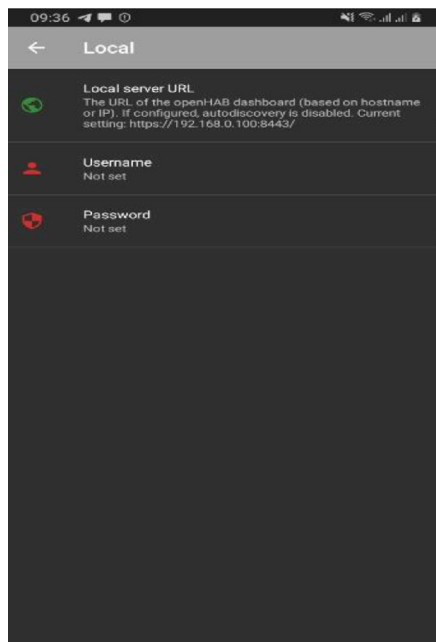


FIGURE 11: Local server connection.

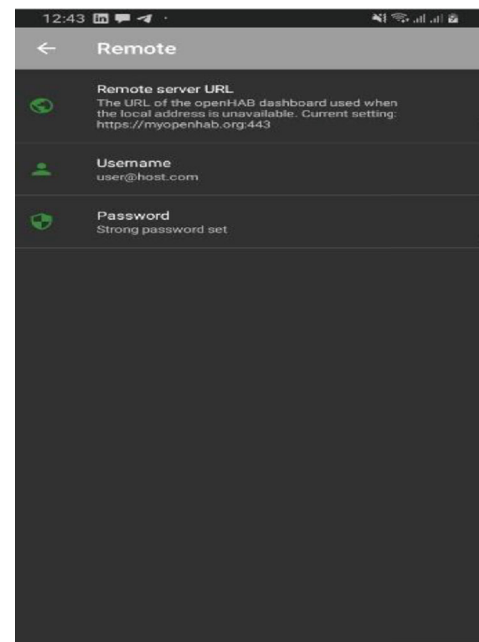


FIGURE 12: Remote server connection.

After selecting the “Corridor,” as shown in Figure 15, all connected appliances in that space, as well as their respective states, are retrieved and displayed to the user. Figure 16 shows the updated state after the light has been switched on.

These demonstrate the requirements of the smartphone application, which accents the importance of the home automation system and conclusively demonstrates monitoring, control, and some level of security.

4.5. Security Implementations by User Authentication and Authorization on OpenHAB. Security is an essential protection against anything that could pose a threat to a system. One of the biggest challenges with OpenHAB perhaps is the nonenforcement of an access control mechanism for its users, thereby making the security of the system dependent on the strength of the wireless network. We developed a simple authentication and authorization model to control access permissions to things on the network.

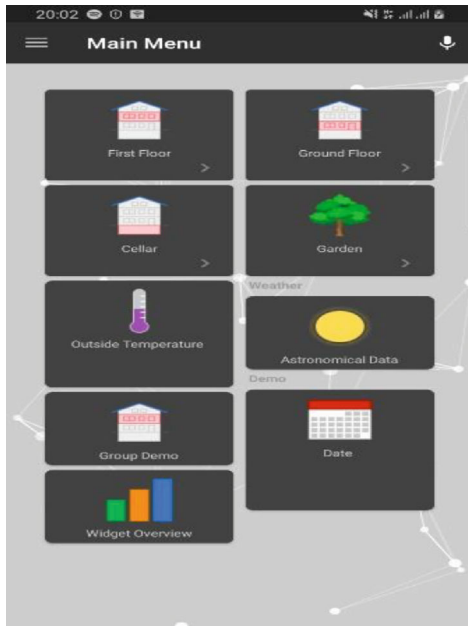


FIGURE 13: Main menu view.



FIGURE 15: Corridor light off.

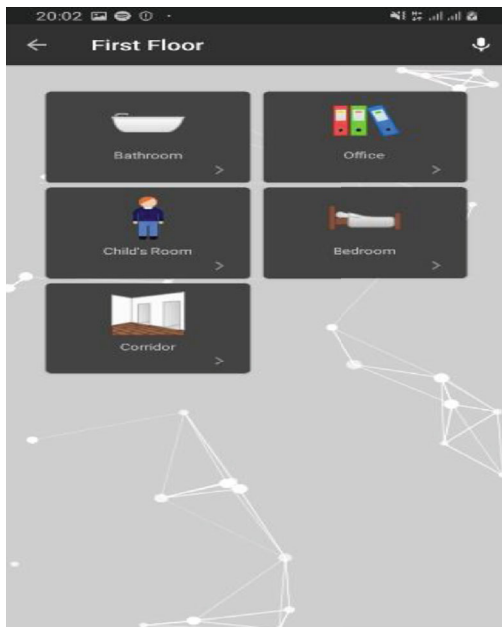


FIGURE 14: First floor view.

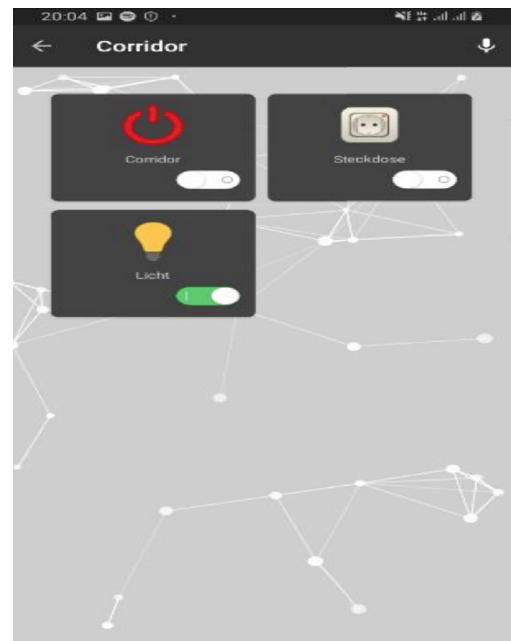


FIGURE 16: Corridor light on.

This was done using the JSON Web Token- (JWT-) based authenticator as a base model for all access control protocols. This base model depicted in Figure 17 is an abstraction of well-defined security policies and rules in our OpenHAB system.

JSON Web Tokens (JWT) are a tool that makes use of cryptography to ensure secure data communication between parties. There are two critical steps in using JWT securely in a web application: (1) sending them over an encrypted channel and (2) verifying the digital signature immediately upon receiving it. The asymmetric nature of the public key cryptography makes JWT signature verification possible. A public

key verifies a JWT was signed by its matching private key to enable data communications securely. No other combination of keys can do this verification, thus preventing impersonation attempts. JWT is a very modern, simple, and secure approach which extends for JSON Web Tokens. JSON Web Tokens are a stateless solution for authentication. Hence, no need to store any session state on the server, making it ideal for restful APIs, thereby making the server implementation lightweight and computationally efficient in the use of resources. Restful APIs should always be stateless, and

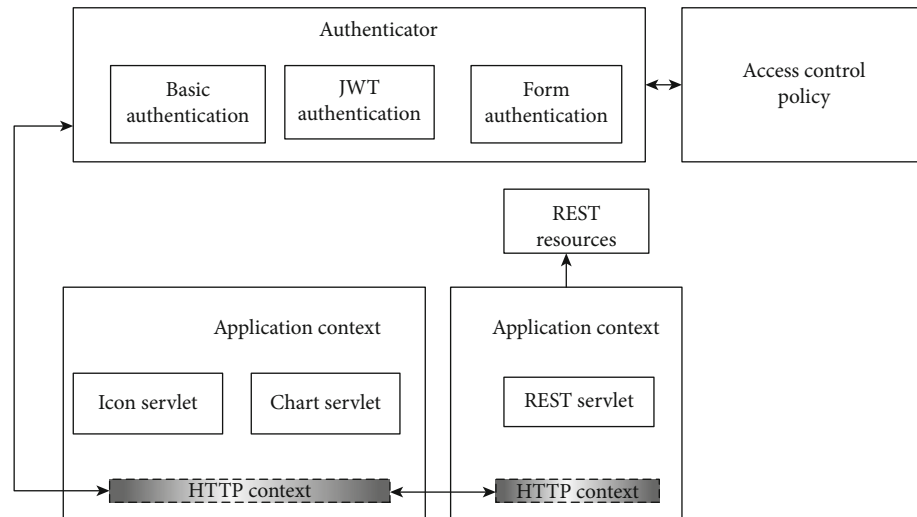


FIGURE 17: Block diagram of authenticators.

the most widely used alternative to authentication with JWT is just to store the user's login state on the server using sessions.

The first step in addressing the security challenges of OpenHAB was to analyze how data is moved through the network. Data flow through the network was monitored using Wireshark, a free and open-source packet analyzer. Based on the packet monitoring and tracing using Wireshark, it was discovered that data is transmitted in two ways, namely, via the cloud and the local OpenHAB host. Recognition of the fact that some devices usually do not need to connect to the Internet to set or reset their states was ascertained based on analysis, and that informed our design choices and protocols to use to establish secure communications. However, connection to the Internet is required in cases where the user might want remote access to things at home. The two ways of data transmission in the network required that we addressed them uniquely. The first scenario involves the case where OpenHAB establishes a connection through the Internet and opens up a lot of known vulnerabilities, one of which is that packets transmitted can be hacked and tampered with. In preventing this, we used the JWT authentication procedure, which we performed in five steps:

- (1) The client sends his/her details through the basic authenticator initialized on the web browser
- (2) The broker extracts the client details, and if it matches a registered client, a *JSON Web Token (JWT)* is generated. It is further appended to his unique ID along with the expiration date and digital signatures. The broker relays the JWT to the client
- (3) The client attaches the JWT on any subsequent request to the server
- (4) For every request, the server extracts the JWT and verifies the digital signature appended to it. If it is deemed valid, the username and password are used to perform authorization on the system for use

- (5) If the JWT is expired, the broker requests user details through basic authentication, and if they are valid, it generates and serves another valid JWT

The second case where OpenHAB communication is done locally between home appliances and the OpenHAB host appeared to be of lesser risk. This is because communication is usually through a wireless network encrypted with AES. Breaking the AES is almost computationally impossible, especially for the AES-256 encryption enforced on the developed system. The sequence diagram of the security implementation is depicted in Figure 18.

4.6. Hardware and Software Integration. The device controller, in two modes, controls the devices. OpenHAB serves to handle the remote mode while the switching circuit developed handles the manual mode. The DHT11 Temperature and Humidity sensor features three pins: VCC, DATA, and GND. The VCC is connected to the 5V, GND to GND of Arduino, and DATA to the analog pin of the Arduino microcontroller. On the Arduino IDE, we imported the *DHT.h library*, which enabled us to convert the analog value read into real-world temperature and humidity values. This data is published via the ESP8266 module to the server on the Raspberry Pi and stored securely for remote monitoring.

5. Testing and Results

The developed system integrated all items to a single switching board to show that the states of devices could be controlled remotely with improved security. From tests conducted, the developed system failure rate is exceptionally minimal, thus offering a high level of security in home automation, and the set objectives of secure wireless switching and internetworking of home appliances and subsequent control were achieved.

5.1. Android Application Testing. The first approach to testing the Android application was to check for stability across

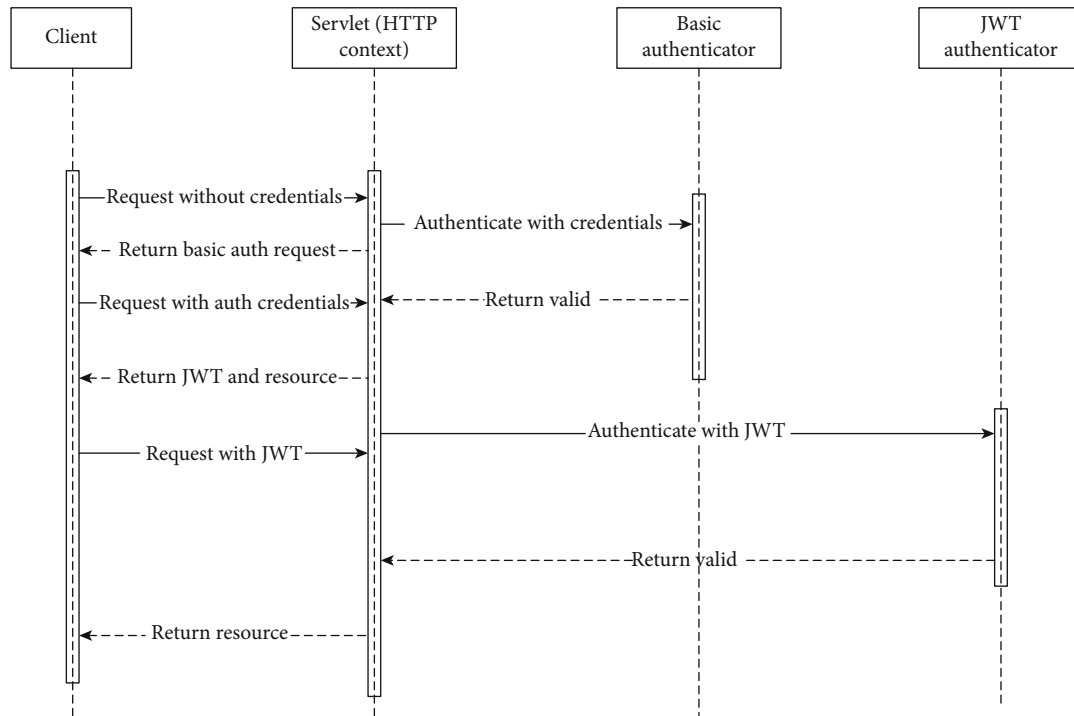


FIGURE 18: Sequence diagram of the security implementation.

a range of Android devices. After looking into the various Android operating systems to determine the relevant ones to test our mobile application, the test focused on a limited set of most popular devices and Operating System (OS) versions since it would be impractical to test out the application on every Android device on the market.

The application was run on one high-end Android device, the Samsung Galaxy S9+, and a midrange device, the Samsung Galaxy A5. The application performance on both devices was similar. No visible performance lags or GUI responsiveness issues.

During mobile application development, the JUnit, a testing framework, was used to test all modules within the application. The test helped us to fix some significant compatibility issues across the different Android operating systems.

Tests performed considered the effect the application running has on battery life. The application had minimal impact on the battery, using less than 1% of battery on standby and about 10% on continuous 5-hour usage. Comparing this with the average of all data-enabled applications, the net battery savings recorded was about 15%. Data consumption was also minimal, with the application consuming only about 5 MB within a 5-hour period of close to constant usage after deployment.

5.2. Web Service Testing. In testing the web service, PostMan and HttpMaster were used. Both applications support dynamic parameters, data validation, and response data. The various requests yielded the expected response codes, hardly ever reporting failures even on very slow and unstable networks.

5.3. OpenHAB Testing. The testing of the OpenHAB implementation focused on the overall system latency. Various switching commands were issued out to the system. On a local network, it was observed that switching requests and responses were much faster than when on the cloud service. Since a guarantee cannot be made for great user Internet speed, this issue was addressed by prioritizing the local server over the cloud server to reduce network latency.

5.4. OpenHAB Security Tests. OpenHAB makes use of bindings, which is a logical software piece that links a thing to OpenHAB. The data transmitted through the REST API calls on the bindings can be eavesdropped upon if the packets through the network are inspected. Wireshark was used to inspect packets and understand how the different bindings transmit data.

Internal communications between things in the network and OpenHAB framework are through a wireless network, and it is often encrypted with AES. An eavesdropper can get the transmitted data if they can decrypt the AES signal data transmitted. That is, in itself, computationally not feasible. Security, in this case, as we tested, was dependent on the strength of the wireless network.

Another communication approach was where OpenHAB was communicating with a remote server in the cloud. In addressing this security challenge, a JWT-based authenticator was implemented in Java. Testing directly on the OpenHAB framework was not an easy task, so we first demonstrated the implementation as an independent deployment. In this independent deployment, we ran the system against various intrusion tests and packet/session hijacking tests using industry-standard tools for which

TABLE 2: Hardware system tests and results.

Test	Expected result	Result	Remarks
Start with the hub, and OpenHAB service configuration is running on the hub without an Internet connection	Initialize the database connection, start OpenHAB service, and connect to cloud service if available. Establish a connection with the MQTT server and update device states	Successful	Start-up takes about 45 seconds on average (out of 20) to start up Raspberry Pi and OpenHAB service
Connect a LAN or WAN to the hub while OpenHAB service configuration is running	Trigger cloud connects service to establish a connection with the cloud server and sync device states	Successful	Cloud service worker detects Internet connectivity, and cloud loads configurations with a connection 10 seconds
Stop the OpenHAB hardware application or power off Raspberry Pi while OpenHAB service is still running	Raise <i>ForcedTermination</i> event in OpenHAB Control Panel logs and publish to <i>ForcedExceptions</i> MQTT topic	Failed some of the time	Failed some of the time. The cause is a delay in publishing before power is cut
Create a new “Home” location in the OpenHAB Control Panel	The location should be added to the local database and posted to and made available on the OpenHAB server	Successful	A new location is immediately available
Add a new appliance in the OpenHAB Control Panel	The appliance with its properties should be added to the local database with OpenHAB service refreshed to show the update as well as send push notification to registered users	Successful	A new appliance was immediately available. The push notifications, however, worked 99% of the time
System vulnerability test	No breaches or vulnerabilities during test runs locally at the client as well as remotely		Firewall bypass testing was a success. The system withstood DNS-level attacks, e.g., switching and routing tests

TABLE 3: Software system tests and results.

Test	Expected result	Result	Remarks
Start the OpenHAB mobile application	Verify OAuth token with the OpenHAB local server and cloud service, while the splash screen is shown, display login state if the token is invalid. Initialize the server connection, synchronize the current device states, and display the main page	Successful	With Internet connectivity, 1.5 secs is required from splash screen to the main screen with a local server and 3.5 secs for an active cloud configuration device information and states
Switching between the local server and the cloud service	Switch from the local server to the cloud service when out of range of the local server and switching back	Successful	Also, prioritizes local server when back in range without compromising connection or device states
Create a new OpenHAB account	Save the user account details and issue the OAuth token	Successful	
Request from OpenHAB mobile to cloud service on OpenHAB web	Verify the OAuth token if it passes, fetches the required data from the database, updates, and returns to OpenHAB service	Successful	One out of about 100 tries does not update on the app after the request is successful. Refreshing the app page, however, shows the updated state
Interact with a hardware control for a device on the OpenHAB Control Panel	Effect change on the hardware device and send an update via MQTT to the OpenHAB server for relaying to all other clients then update cloud service	Successful	
Interact with a control for the power switch of an appliance in OpenHAB mobile	Send change via MQTT to OpenHAB cloud service for relaying to the OpenHAB Control Panel (Raspberry Pi)	Successful	Unless there is no active connection to either the local or cloud server, which is updated when there is a connection

TABLE 4: System security tests and results.

Test	Process	Observation	Remarks
Wireless network penetration testing	Used industry-standard penetration testing tools and frameworks, including Aircrack-ng. The remote penetration testing device was placed within the network. The wireless network audit began with a full sweep of the 2.4 GHz wireless frequencies, where numerous busy networks were found	1. It uses the preshared key for network access by sniffing the network while forcing an existing client off the network 2. Capturing of a WPA2 handshake attack 3. Nonidentification of clients connecting to the hidden SSIDs during the audit period, and therefore, it was not possible to unmask them	A scan was made for Cisco Discovery Protocol (CDP) traffic, which would have disclosed further information about the network. CDP was not found to be running across the public guest network, and VLAN hopping was unsuccessful Risk level: extremely low

the system withstood. The Aircrack-ng software, as a complete suite of tools to assess WiFi network security, was utilized. This tool focuses on different areas of WiFi security, such as monitoring of packet data capture and export of data to text files for further processing, attacking the network using replay attacks, deauthentication, fake access point creation, and packet injection with testing and checking of WiFi cards and driver capabilities. The Aircrack-ng software provides capabilities for cracking passwords or passphrases on Wired Equivalent Privacy (WEP) and Wireless Protected Access (WPA) protocol implementation.

The results of the various tests for the hardware system and integration, as well as software system and integration, are captured in Tables 2 and 3. Table 4 presents the results of the system security tests that were performed using industry-standard tools.

To test all cases in a simulated home environment, a demo house was constructed and fitted with the required hardware. Figure 19 shows the constructed demo house having a model garage, two bedrooms, a kitchen, a corridor, and a dining area with fitted hardware.

All the rooms have an LED serving as an energy-efficient light bulb, relays, and switches for turning lights on and off remotely via the mobile and web applications developed.

Figure 20 also shows the control circuit with the Arduino Mega fitted to enable control of connected devices.

The final setup with enabled connection to the central server is shown in Figure 21.

6. Conclusion

Most homeowners in developing countries lack complete and total control over their homes. They are not able to access vital home automation features such as control and monitoring of home appliances, low-cost security, and efficient energy usage by implication. The OpenHAB 2 protocol is relevant to automating the home while interconnecting appliances based on its flexibility and capability for full customization, to achieve the goals of (1) providing a secure control mechanism for home appliances via a mobile or web application; (2) improving the security of connected home appliances through the system's inbuilt intrusion detection, alarm, and wireless communication data encryption standards; (3) meeting the essential energy management requirement of the household by providing a means to monitor and remotely turn off unnecessary active appliances to conserve energy and reduce electricity bills; (4) offering a wirelessly controlled switch for all home appliances; and (5) making a significant contribution to the existing body of knowledge on secured home automation. This paper presented a secure wireless home automation system that has been designed and implemented with the OpenHAB 2 home automation software framework to meet the set goals. A power supply circuit was designed and implemented for the microcontrollers while another circuit was created for the wireless switching and control of appliances. The OpenHAB server was set up on the Raspberry Pi, and the Arduino was configured to communicate with the OpenHAB server for home automation tasks. Both mobile and web applications



FIGURE 19: Constructed demo house with LEDs integrated.

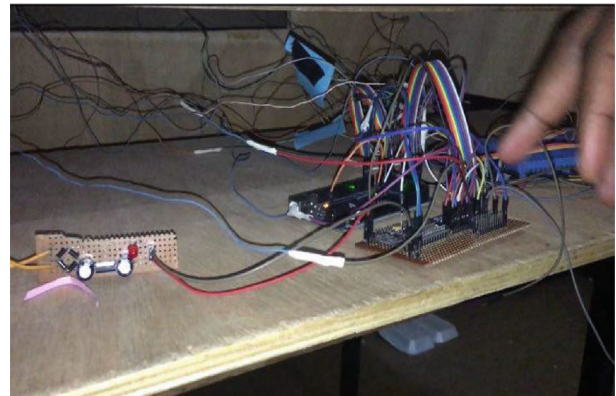


FIGURE 20: Compartment for housing demo house circuitry.



FIGURE 21: Demo house with all LEDs, motors, and sensors integrated.

were developed to control and view the status of home appliances. Also developed was a secure wireless network system enabling communication between the home appliances and the OpenHAB server. The prototype integrated all items to a single switching board as a proof of concept that the states of the device could be controlled and monitored remotely with improved security using JSON Web Tokens. Future research directions could be geared towards integrating the switching circuitry within the real-world appliances to trigger the switching internally instead of just controlling the power

supply to the appliance. Also, learning algorithms to enable adaptive control and machine learning could be implemented to handle sensor parameters automatically (e.g., from temperature, smoke, and other detector systems). Also, future project directions for enhancements could be geared towards the use of machine learning techniques to learn from the homeowner's behavior to optimize the switching of appliance states for energy savings using various optimization techniques.

Data Availability

The data used for software testing and penetration testing are available upon request.

Conflicts of Interest

The authors declare that there is no conflict of interest in this research work and subsequent paper publication.

Acknowledgments

The authors would like to express their profound and earnest gratitude to the entire staff of the Computer Engineering Department, University of Ghana, for the invaluable knowledge and insightful comments on earlier drafts of this manuscript.

References

- [1] J. Jin, Y. Wang, K. Zhao, and J. Hu, "Development of remote-controlled home automation system with wireless sensor network," in *2008 Fifth IEEE International Symposium on Embedded Computing*, pp. 169–173, Beijing, China, 2008.
- [2] A. A. Zaidan, B. B. Zaidan, M. Y. Qahtan et al., "A survey on communication components for IoT-based technologies in smart homes," *Telecommunication Systems*, vol. 69, no. 1, pp. 1–25, 2018.
- [3] M. D'Souza, N. Wilfred, R. Pereira, T. Rayen, and A. Telgote, "Home automation using Internet of Things," in *2017 International Conference on Energy, Communication, Data Analytics, and Soft Computing, ICECDS*, pp. 559–561, Chennai, India, 2018.
- [4] M. Shinde and R. R. Dube, "IOT Based Energy Monitoring and Management System for Smart Homes," *International Journal of Recent Trends in Engineering and Research*, vol. 4, no. 1, pp. 287–295, 2018.
- [5] M. Daneshvar, M. Pesaran, and B. Mohammadi-Ivatloo, "Transactive energy in future smart homes," in *The Energy Internet*, pp. 153–179, Elsevier, 2018.
- [6] R. Teymourzadeh, S. A. Ahmed, K. W. Chan, and M. V. Hoong, "Smart GSM based home automation system," in *2013 IEEE Conference on Systems, Process & Control ICSPC*, pp. 306–309, Kuala Lumpur, Malaysia, 2013.
- [7] R. K. Kodali, V. Jain, S. Bose, and L. Boppana, "IoT based smart security and home automation system," in *2016 International Conference on Computing, Communication, and Automation ICCCA*, pp. 1286–1289, Noida, India, 2016.
- [8] W. Ejaz and A. Anpalagan, "Internet of Things for smart cities: overview and key challenges," in *Internet of Things for Smart Cities*, pp. 1–15, Springer, 2019.
- [9] S. Mahmud, S. Ahmed, and K. Shikder, "A smart home automation and metering system using internet of things (IoT)," in *2019 International Conference on Robotics, Electrical and Signal Processing Techniques*, pp. 451–454, Dhaka, Bangladesh, 2019.
- [10] W. Li, T. Logenthiran, V. T. Phan, and W. L. Woo, "A novel smart energy theft system (SETS) for IoT-based smart home," *IEEE Internet of Things Journal*, vol. 6, no. 3, pp. 5531–5539, 2019.
- [11] W. A. Jabbar, T. K. Kian, R. M. Ramli et al., "Design and Fabrication Of Smart Home with Internet of Things enabled automation system," *IEEE Access*, vol. 7, pp. 144059–144074, 2019.
- [12] TheAmbient, *Around the world in smart home trends*, The Ambient, 2018.
- [13] O. Veras, *Smart cities in Africa: Nairobi and Cape Town*, Africa Business Insight, 2017.
- [14] D. Hendricks, *The history of smart homes*, IoT Evolution World, 2014.
- [15] Research and Markets, *Global home automation and control market 2014-2020 - lighting control, security & access control, HVAC control analysis of the \$5.77 billion industry*, 2015, <https://www.reuters.com/article/research-and-markets-idUSnBw195490a%2B100%2BBSW20150119>.
- [16] *Mobile operating system market share worldwide: concatenated 2015 to 2019*, Statcounter Global Stats, 2019, <https://gs.statcounter.com/os-market-share/mobile/worldwide/2019>.
- [17] January 2017, <https://www.openhabfoundation.org/2017/openhab2>.
- [18] S. A. Celtek, M. Durgun, and H. Soy, "Internet of Things based smart home system design through wireless sensor/actuator networks," in *2017 2nd International Conference on Advanced Information and Communication Technologies (AICT)*, pp. 15–18, Lviv, Ukraine, 2017.
- [19] A. Bhatt and J. Patoliya, "Cost-effective digitization of home appliances for home automation with low-power WiFi devices," in *2016 2nd International Conference on Advances in Electrical, Electronics, Information, Communication and Bio-Informatics AEEICB*, pp. 643–648, Chennai, India, 2016.
- [20] J. Kizza and F. M. Kizza, "Access control, authentication, and authorization," in *Securing the Information Infrastructure*, pp. 180–208, IGI Global, 2008.
- [21] G. Song, F. Ding, W. Zhang, and A. Song, "A wireless power outlet system for smart homes," *IEEE Transactions on Consumer Electronics*, vol. 54, no. 4, pp. 1688–1691, 2008.
- [22] S. Karaca, A. Sisman, and I. Savruk, "A low-cost smart security and home automation system employing an embedded server and a wireless sensor network," in *2016 IEEE 6th International Conference on Consumer Electronics - Berlin (ICCE-Berlin)*, pp. 73–77, Berlin, Germany, 2016.
- [23] N. Vikram, K. S. Harish, M. S. Nihaal, R. Umesh, A. Shetty, and A. Kumar, "A low-cost home automation system using WiFi based wireless sensor network incorporating Internet of Things (IoT)," in *2017 IEEE 7th International Advance Computing Conference (IACC)*, pp. 174–178, Hyderabad, India, 2017.
- [24] N. Gyory and M. Chuah, "IoTOne: integrated platform for heterogeneous IoT devices," in *2017 International Conference on Computing, Networking and Communications (ICNC)*, pp. 783–787, Santa Clara, CA, USA, 2017.
- [25] R. A. Atmoko, R. Riantini, and M. K. Hasin, "IoT real time data acquisition using MQTT protocol," *Journal of Physics: Conference Series*, vol. 853, article 012003, 2017.

- [26] M. Domb, "Smart home systems based on Internet of Things," in *IoT and Smart Home Automation [Working Title]*, IntechOpen, 2019.
- [27] A. Gupta, A. Mudgal, C. Jayaraj et al., "Smart home device and energy management systems," in *2011 Annual IEEE India Conference*, pp. 1–5, Hyderabad, India, 2011.
- [28] M. Ramljak, "Security analysis of Open Home Automation Bus system," in *2017 40th International Convention on Information and Communication Technology, Electronics and Microelectronics (MIPRO)*, pp. 1245–1250, Opatija, Croatia, 2017.
- [29] F. Heimgaertner, S. Hettich, O. Kohlbacher, and M. Menth, "Scaling home automation to public buildings: a distributed multiuser setup for OpenHAB 2," in *2017 Global Internet of Things Summit (GloTS)*, pp. 1–6, Geneva, Switzerland, 2017.
- [30] R. A. Sowah, A. R. Ofoli, M. K. Tetteh, R. A. Opoku, and S. K. Armoo, "Demand Side Management of Smart Homes Using OpenHAB Framework for Interoperability of Devices," in *2018 IEEE 7th International Conference on Adaptive Science & Technology (ICAST)*, pp. 1–8, Accra, Ghana, 2018.
- [31] R. A. Sowah, K. O. Apeadu, A. Ofoli, K. Koumadi, A. Acakpovi, and S. K. Armoo, "Interoperability of heterogeneous appliances in home automation using the AllJoyn framework," in *2018 IEEE 7th International Conference on Adaptive Science & Technology (ICAST)*, pp. 1–9, Accra, Ghana, 2018.
- [32] S. Dey, A. Roy, and S. Das, "Home automation using Internet of Thing," in *2016 IEEE 7th Annual Ubiquitous Computing, Electronics & Mobile Communication Conference (UEMCON)*, pp. 1–6, New York, NY, USA, 2016.