Journal of Engineering Science and Technology Review 15 (3) (2022) 177 - 192

Research Article

JOURNAL OF
Engineering Science and Technology Review

www.jestr.org

Design and Implementation of a Low-Cost Cloud-Powered Home Automation System

Anthony U. Adoghe, Ebere C. Owuama, Victoria Oguntosin* and Boluwatife Morawo

[1]*Electrical and Information Engineering, Covenant University, Ota, Ogun State, Nigeria*

Received 26 January 2022; Accepted 25 July 2022

Abstract

In just a couple of years, the home environment has rapidly introduced network-enabled intelligent digital technology. A synergy of these technologies such as intelligent lighting, entertainment (audio and video), security, heating, ventilation and air conditioning in the home for the automation of control and monitoring activities within that home is typically known as Home Automation today. The Home global home automation market size was approximated to be about USD 45.8 billion in just 2017 and is projected to reach as high as USD 114 Billion at the tail of 2025 with a compound annual growth rate of 12.1% in this forecast period [1]. North America is projected to emerge as the leading region in the global landscape during the forecast period, but West African countries such as Nigeria, Ghana, Cameroon are generally seen to have a much lower adoption rate when it comes to emerging technologies like these. The low adoption rate is due in part to the high cost of implementing a home Automation solution coupled with the general economic state of these developing countries. The interest and investment of the tech industry within the countries also play a role in this context. This paper explores the design and implementation of a relatively less costly cloud-based home automation architecture built partly on open-source technology and widely available resources. The solution was realized using a raspberry pi as the field gateway and primary controller. Arduino Uno microcontroller were used as the secondary controller. The highly robust Microsoft Azure cloud was used, enabling the representation, testing, deploying, and managing applications and systems and services on the cloud and at the edge. In demonstrating the feasibility of the proposed system, three systems were integrated: intelligent lighting, basic access security and remote monitoring. These would be monitored and controlled by a simple mobile app whose communication with the field devices is made possible through the Microsoft Azure IoT solution.

Keywords: Home Automation, Cloud Computing, Azure IoT, intelligent lighting, basic access security and remote monitoring.

1. Introduction

The Fourth Industrial Revolution, popularly known as industry 4.0, is the ongoing automation of traditional, domestic and industrial practices using modern innovative technology. Little to Large-scale machine-to-machine communication (M2M) and the internet of things (IoT) is integrated for increased automation, improved communication and production of smart machines. These machines can be controlled and used to monitor, analyze and diagnose issues with little or no need for human intervention [1]. In essence industry 4.0, is the move towards digitization, with significant technologies including artificial intelligence, robotics, blockchain, 3D printing, blockchain, quantum computing, the Internet of Things (IoT) and others. IoT can be envisioned as a set of connected technologies which include Things, Insights and Actions [2].

A distributed home automation system would take the definition above but applied specifically to the home. A distributed home automation system will consist of various, possibly heterogeneous, computing units, which could also be referred to as nodes, endpoints or simply as devices. These devices work together in order to achieve the desired automation functionalities by sending and receiving messages through some medium. This medium could be on a local IP network, a Bluetooth or BLE network, a ZigBee network, or they could be connected through the cloud [3]. This enables the node devices to operate in a constructive and synchronized manner. Particular emphasis is placed on the relative cost reduction opportunities provided by the tool-set employed in the building of this research. This research demonstrates how certain mundane, repeated activities in a home can be automated, especially when the action doesn't mainly require human intervention. This research employs the 1.2 GHz 64-bit quad-core raspberry pi three (3) running on Windows IoT [4]. This research uses the raspberry pi to act as the gateway device in the perception layer responsible for processing, filtering and uploading the device data directly to the cloud. It would also serve as the boundary device that receives control signals from the cloud and changes the other connected devices' state accordingly.

Additionally, a simple app is developed that allows the end-user to interface with the controls and monitor the sensors' state. Control signals sent from the cloud to the raspberry pi may originate from this app or from some event pre-defined event trigger on the Azure platform. Microsoft Azure's cloud services have created an eco-system where it is very affordable and accessible to build the back-end systems to power the computing, processing analysis, and storage necessary to implement this relatively low-cost solution. This research is built by implementing a five-layered cloud architecture with Microsoft Azure providing the services required to fill in the needs of the processing layer[5].

Following the impact previous industrial revolutions had on organizations and the overall socio-economic progress of

*E-mail address: victoria.oguntosin@covenantuniversity.edu.ng
ISSN: 1791-2377 © 2022 School of Science, IHU. All rights reserved.
doi:10.25103/jestr.153.20

countries, we can assume that Industry 4.0, as well, will leave its own footprints in the ever-changing economy of both developing and developed countries [1]. To involve developing countries in the ongoing industry 4.0 transformation, it becomes crucial for industry giants, SMEs, professionals, and influential persons to propel the move within these countries. It is also essential that educational institutions capitalize on such transformations as the education sector is a significant frontier for engineering and technological innovations. This research is a contribution from a developing country to the existing body of knowledge in the field of IoT and home automation, a contribution that hopes to be an information source for professionals, researchers and hobbyists alike. A cloud-based architecture was chosen in this research because of the prominence of cloud technology in providing scalable digital infrastructure for smart cities while reducing complexities, improving security and allowing great flexibility in cost management.

A smart city is a city that has undergone urbanization through human innovation and technology, with the outcome being a smarter, more efficient, environmentally friendly and more socially inclusive society. The goals of a smart city would be to help monitor and better understand environmental processes improve its attractiveness to citizens and businesses by enhancing and adding city services [2]. A prominent component of such smart cities is the smart home and their interconnectedness within the smart city through the cloud. Good cloud architecture allows better integration of a smart home with a smart city, especially considering data flow and security. A relatively affordable smart home infrastructure deployment provides for a more affordable way to move a city's transformation towards being a smart one. The improved productivity in the lifestyle of citizens that occurs as a result of this transformation would undoubtedly impact the state of the economy of that city of its country positively.

IoT based systems are also known to generate large amounts of information quickly. This situation, depending on the present or progressive scale of a research, could make such generated information unsuitable for processing using traditional SQL-queried relational database management systems (RDBMSs) and also put a strain on an existing internet infrastructure [3] [4]. This tendency necessitates the deployment of resources that can handle such "Big Data" processing requirements. As a result, robust Cloud technologies today are built explicitly for handling the aggregation, analysis and storage of large amounts of data. And with a pay-as-you-use model, where customers only pay for the specific resources used, there is even more control over the cost incurred for subscribing to a cloud platform.

This research aims to study and demonstrate the use of a cloud-based approach to create back-end services for implementing smart home automation features in homes as a relatively low-cost alternative to existing high-cost solutions. The objectives are to: Review literature on existing cloud-based home automation systems; Design a smart home model using mathematical analysis and software simulation tools; Build a prototype model of a smart home with basic remote-control functionality using azure IoT services with Raspberry Pi and Arduino compatible devices; implement data monitoring capabilities into the smart home model; set-up automated e-mail and SMS alerts within predefined contexts; Create a presentation of the working model based on the data collected and the prototype built.

The research's primary focus was to create a working prototype of a cloud-powered IoT system with functions geared towards the home. This research involves planning (and requirements definition); design (and analysis), simulation, and then implementing a cloud-powered home automation prototype system. This order was chosen as it allowed an iterative design process with each successive design iteration informed by the previous iteration's planning decision and simulation result. An extensive literature review was carried out on existing research work relating to IoT, Home Automation and cloud computing. The review generally follows the order of concept or technologies definition, features, comparisons and summary. Planning involved identifying the research requirements; the requirements definitions were guided to implement features within two main application areas in home automation: lighting and security. The planning phase was mainly a 2-stage process, namely exploration and analysis.

The design of the prototype "smart" home is done using Proteus and Azure IoT platform. On complete or successful analysis of the chosen design, the design is set-up and simulated within Proteus. The simulation was used to validate more accurately the chosen design. Some of the software code used by the micro-controllers were tested during this stage to ensure errors in code were eliminated before implementation. Proteus was chosen for circuit simulation because of its mathematical accuracy and the plethora of devices and device variants it is able to simulate. It allowed the visualization and connection design of the electrical and electronic components of the prototype. Azure IoT, visual studio code using C# and C programming language was used for (simulated) device-to-cloud and cloud-to-(simulated) device telemetry. Azure IoT provided APIs which enabled IoT related simulations without the need for physical devices. This was meant to enable companies to decide on the scope and scale of implementation before committing its resources to purchase hardware components;

The simulated design passes this phase if the simulation functions as desired within the simulator and the total cost recorded in the bill of Materials was within budget. There could be undesired results with the simulation or unreasonable cost projection based on components used. In such cases, the desired feature was either re-defined in the planning phase or other designs explored and a new design selected for analysis and simulation.

Upon completion of tests and simulation, a physical circuit was assembled to be as close as possible to the design. This stage was necessary for presenting a working prototype of the research. The Model implements the features of Networked Lighting System, Security Access control, Energy management, Intruder detection, and Humidity and Temperature report. Upon physical implementation, tests were carried out on different sections of the circuit. A section is defined by its function (e.g., A section could be a group of components meant to send temperature and humidity data to its connected microcontroller) and its performance monitored in order to monitor how the design behaves in a real-world scenario. Performance was measured based on various metrics explored in Section 4 of this paper. Rounding up the research, a presentation of the research findings and use cases was made alongside a physical demonstration of the working prototype.

This paper follows the following pattern for its implementation; a total of 5 sections will be reviewed. Section One presents the introduction, the statement of the problem, aims, and objectives of the research, the significance of the research, the methodology used, and the structure of the research. Section Two presents a historical and theoretical literature review on past but relevant works relating to the research, bringing out the gaps, resolution areas, meeting points, and where we differ. Section Three deals with the data gathering, grouping and analysis activities, as well as explains the architecture, design and implementation of the prototype smart home critically; highlighting the methods, tools and technology used in designing and assembling the Model. Information such as circuit diagrams, design architecture, schematics of manufactured components, graphs, etc., is provided here. In Section Four, gathered and analyzed data are presented using use-case diagrams, graphs, images and tables. Results gotten from the design and prototype tests are also visualized. Section Five is the concluding Section which states how well the research objectives were met and drew conclusions from the data analysis and model implementation carried out.

2. Literature Review

The Internet of Things (IoT) describes the network of physical objects or things embedded with software and sensors for connection and exchange of data with other devices and systems over the internet [5]. Technologies from the internet of things can be applied to ordinary household objects, which forms a significant aspect of what is called Home Automation. An automated home is generally aimed at automating as much of a user's life as possible and thereby improving the user's quality of life; it usually comes with a high cost of ownership though and is more slowly adopted in developing countries. The research's major objective is the design and implementation of a relatively cost-friendly, cloud-powered home automation system that is capable of basic automation and control of household appliances. While many local pieces of research have focused on an on-premises device-to-device communication protocol, this research aims to explore and demonstrate a use case of cloud computing in an automated home.

This literature review provides background information on research findings, existing technologies and approaches to implementing a cloud-based home automation system. Reviewed are the following topics: Smart Home Definitions, IoT Communication Protocols, Cloud Computing, Edge Computing, Cloud Computing paradigms, Cloud Computing applied to IoT, IoT messaging protocols, IoT service providers, IoT related APIs, IoT prototyping controllers.

Home Automation is a system of networked hardware devices and sensors that allows computerized or automatic control of typical or specialized home appliances. Another popular synonym for Home Automation is called domotics. This specific area of domotics can also be denoted as Ambient Assisted Living (AAL) to differentiate it from the more universal field of request [6]. Home automation solutions can be categorized into four main areas by application: lighting, entertainment, safety and security, Heat, Ventilation, and Air Conditioning (HVAC). Lighting automation involves creating automated alterations in lighting levels to emphasize architecture, art or affect mood. Entertainment Automation is the installation and configuration of Entertainment systems to allow the easy and automated control of entertainment-related systems in the home. It is able to connect all of the audio/video sources (e.g., Roku) and audio/video sinks (e.g., TV, speakers) throughout the home to a central control system at a fundamental level. All These are done with the goal of achieving low latency at various distances and increased levels of control and zoning from remote platforms and user interfaces [7]. Although TCP use may lessen packet losses under network overcrowding conditions, it fails to completely address the problem of timely delivery to the playback client [8]. Safety and security have to do with the automation, monitoring and control of safety and security equipment to enhance an occupant's safety and strengthen security within the home. Vulnerabilities in Home Automation systems can be inherent to a device's features and implemented technologies chosen by the manufacturer; installations or upgrades performed by a non-professional or a professional's errors can expand the attack surface of the facility. Also, it could be due to the home owner's technical ignorance or lack of interest in investment in the security of installed IoT devices or the network to which these devices belong [9]. Table 1 gives a summary of a few general attacks to which a Smart Home may be vulnerable.

Table 1. General threats and possible attacks in a Smart Home Network

Threat/attack	Brief Description	Consequences	References
Network Packet Sniffers	Intercepting and reading user's data packets as it travels across the network layer	Extracted data such as personally identifiable information (PII) and passwords from data packets can be used to compromise user accounts.	[10]
IP Spoofing	Bad actor hides true identity and masquerades as a different host by forging the source addresses in IP packets.	User session hijacking	[11]
Man-in-the-middle attacks	Replacing original certificate authenticating an HTTPS server with a modified one.	Bad actor can decrypt and reveal entire communication, injecting false/malicious content	[12][13]
Distributed denial of service (DDoS) attack	Deploy multiple attack entities to prevent legitimate use of a service	Service denial, interruption, machine subversion	[14]
Botnet attack	Network of compromised computers used to attack single or multiple targets	A DDoS attack, malware dissemination, phishing,	[12][15]
Ransomware	Malware in a legitimate software that encrypts user data and blocks user's access.	Financial losses from paying off ransom to unlock user data	[16]

Some proactive solutions can be implemented to mitigate or reduce the risk of a successful attack on an IoT network. Some of these preventive measures are Honeypots, inbuilt security on chips, security legislation, exhaustive security testing, Device Authentication, regular updates, Firewalls and Intrusion Prevention Systems. A summary of some of these preventive measures can be seen in Table 2.

Table 2. General security risk prevention methods for IoT

Threat/attack	Brief Description	Benefits	References

Honeypots	Gain info on bad actors by using a bait computer to mimic a target.	Intelligence gain on system vulnerabilities and counteracting without affecting regular infrastructure.	[17][12]
inbuilt chip security	integrating security for IoT hardware/chips during design and manufacturing.	Faster security related execution rate.	[12]
security legislation	Data protection regulation (e.g., GDPR, CCPA) by appropriate bodies.	General awareness of the public and the availability of legal enforcement of accountability on digital privacy.	
Secure Boot	Role-based access control (RBAC) to establish and validate the integrity of the software installed on the IoT device.	Service denial, interruption, machine subversion	[12]
Device Authentication	Device-to-device(D2D) authentication through a machine-to-machine (M2M) channel before other processing actions.	Adds an extra layer of protection and better safeguards user's credentials.	[12]
Firewalls and Intrusion Prevention Systems	Firewall, protocol filtering and deep packet inspection (DPI) to control network traffic.	Controls and secures traffic terminating at the IoT devices.	[12]

Heating, ventilation, and air conditioning (HVAC) systems are systems designed to automate and control environmental (usually indoor climate) conditions to meet the requirements of satisfactory comfort of the people in that environment. Various comfort levels can be achieved by adjusting the HVAC controls (temperature, Humidity, airflow, cleanliness) to fit the needs of different types of buildings like residential, commercial, educational and industrial buildings [18]. The choice of HVAC systems in a specified building will be subject to the building's age, the individual preferences or budget of the building owner, the architectural design of the buildings, life cycle analysis, and the outdoor climate [19]. Essential components needed to set up a functioning HVAC system are Ducts, Control, Mixed-air plenum and outdoor air control, Supply fan, Return air system, Cooling tower, Exhaust or relief fans and an air outlet, Terminal devices, Humidification and dehumidification equipment, Heating and cooling coils, Water chiller, Air filter, Self-contained heating or cooling unit, Boiler, and Outdoor air intake.

This research implements smart lighting with safety and security as these are the most in-demand features in many homes and are more budget-friendly to implement in a proof-of-concept such as the one in this research [7]. Security processed are also handled at the could-side with Azure IoT provisioning access control with Azure Active Directory (Azure AD), Shared access signatures, Per-device security credentials, Azure Device Provisioning Service (DPS) and Azure Security Center for IoT.

Communication within the Home Automation can be achieved through wired or wireless media with protocols guiding how this communication is achieved. The scope of communication within an IoT or Home Automation environment can be identified within the following four type:

1. Device-to-Device (D2D) communication which is a direct communication between two nodes or devices.
2. Device-to-Application (D2A) communication which occurs between devices and an IoT application.
3. Device-to-Gateway (D2G) communication which occurs between a gateway that resides in close proximity to the edge of the network and interacting IoT devices.
4. Device-to-Cloud (D2C) communication is achieved directly between IoT devices and cloud service providers [20].

Wired communication usually proves to be the most reliable in a smart home because it can perform high bandwidth transmission over high-grade communications cable. It is also the more expensive solution [21]. Wired communication protocols most commonly employed in the home automation industry are Universal Serial Bus (USB) [22], Universal Asynchronous Receiver Transmitter and Universal Synchronous Asynchronous Receiver Transmitter (UART/USART) [23], Ethernet [24], Inter-Integrated Circuit (I2C) [21], and Serial Peripheral Interface (SPI) Protocol.

The USB protocol was used for this research because of its asynchronous data transmission ability and simplicity of setting up the interface and associated code. Wireless communication protocols are also widely adopted in the implementation of IoT and Home Automation solutions. Some of the most popular protocols used are IPv6, Wi-Fi, Z-Wave, Zigbee, 6LoWPAN, Bluetooth, BLE, NFC, SigFox, RFID, Cellular and more recently Thread.

IoT devices typically connect to the internet in two ways; some devices have network capability built into them, while others may require a gateway to do so. These devices use communication protocols defined within the 7-layer Open Systems Interconnection (OSI) model. Brief reviews of the Application layer, or layer seven (7) protocols used to realize communication in Home Automation solutions are Message Queuing Telemetry Transport (MQTT) [27][28], Advanced Message Queue Protocol (AMQP) [23][24], Constrained Application Protocol (CoAP) [20][29][30] and Hypertext Transfer Protocol (HTTP) [20][30].

Both wired and wireless technologies have enabled the realization of Home Automation and IoT solutions in general. While wired connection is the better choice for higher bandwidth requirements, wireless connections excel in low power installations and where the cables cannot be easily passed around. This research utilizes the USB for wired protocol with Wi-fi and 4G cellular for wireless. MQTT is used as the Application layer protocol. The offerings of these protocols proved simple and performant enough to handle all the data transfer requirements needed within the research implementation.

Cloud computing is the on-demand distribution of virtualized IT infrastructure—including servers (virtual, physical) storage, databases, networking, software, analytics, and intelligence—over the internet with a pay-as-you-go pricing. Instead of purchasing and maintaining physical data centers and servers, organizations of every type, size, and industry can access these services on an as-needed basis from a cloud provider for faster innovation, flexible resources, and economies of scale [31][32]. In recent years, the concept of cloud computing has been the focus of many in the information industry. Consumers have found more important reasons to move resources into these transparent inter-connected computers intended to allow users admittance to their resources from anyplace in the world as long there is internet connectivity [27]. These virtualized resources can be pooled and divided irrespective of physical hardware boundaries through software abstraction; this technology is called virtualization [33]. Some major cloud service providers include Amazon Web Services (AWS), DigitalOcean, Microsoft Azure, VMWare, Google Cloud, Rackspace Cloud, Alibaba Cloud, IBM Cloud, SAP, Oracle, Salesforce. Some major benefits of Cloud computing are: Agility and time-to-value [32], Performance [31], Cost savings [31][32] and Security [31]. Three different types of cloud services deployable, they are: public cloud [33], private cloud, or hybrid cloud. The evolution of cloud computing over the years has led to a number of paradigms of cloud service offerings, the four most commonly accepted are: Infrastructure-as-a-Service (IaaS) [32] [33], Platform-as-a-Service (PaaS) [27][32] [33], Software-as-a-Service (SaaS) and serverless computing [33][31].

Fog computing is a distributed computing paradigm that fundamentally extends the services existing on the cloud to the edge of the system/network. Closely related to fog computing is edge computing which uses the computation capabilities of devices at the edge. In this scenario, there is no impulsive linking to the cloud and the focus is mainly the device side of the network [34]. Based on this definition, we can refer to Fog Computing as a combination of both cloud and edge computing. Unlike the two aforementioned technologies, though, Fog computing is a completely distributed computing approach and does not entirely depend on any integrated component [35]. This solves to a reasonable degree the latency issue of an exclusively cloud set-up and reduces the high bandwidth requirement by utilizing the unused resources of different devices near the client. Although there is still a dependence on cloud resources to do major tasks, the structure of Fog computing makes it a more efficient candidate for an IoT scenario [34]. Table 3 shows a simple relationship between cloud and fog computing.

Table 3. Cloud vs Fog Computing comparison. Adapted from [34]

Parameters	**Fog Computing**	**Cloud Computing**
Positing of server nodes	At the confined network edge	Within the Web
Delay	Less	More
Range amidst server and client	Single hop	Many hops
Hardware	Less computing energy and storage	Expanding computing energy and storage
Position knowledge	Accepted	Denied
Protection	Difficult to measure	Measured
Hosting	Dispersed	Integrated

The nature of Cloud computing solves a significant challenge many IoT systems face, which is the transport and processing of large amounts of data. The cloud does offer several other features to IoT enabled systems such as Remote monitoring and controlling of devices anywhere in various geographical locations; More performant and efficient data processing; Data storage and routing; Automatic regulation based on pre-defined rules in the cloud application; Higher data integrity and worldwide availability of associated data; High system scalability; Easier multi-protocol implementation using distributed APIs; Available SDKs abstract away lower-level functionality and reduce programming complexity during implementation; Many cloud platforms have multi-language support, giving flexibility of choice; Rich data and analytics services with more intelligent insights; and Robust Machine Learning(ML) capabilities [27].

In [38], a cloud-based home automation system is proposed by Ilker Korkmaz et al. The system makes use of the Android operating system and is designed to be scalable. The operation mechanism implements services on the distributed Google Cloud Platform(GCP) to enable the communication structure in the system, and a web server stores data as required. A similar system was also implemented by Shopan Dey et. al. [39] uses GCP but implements a authentication feature requiring the client to register on the app. The client's registration details are stored on the cloud and used to validate user identity subsequently when the client app is re-opened.

In [20], Singh et al. designed a system that allows a client to conduct a variety of tasks around the house. The goal of the research was to utilize as little energy as possible while reducing human labour. Different parts of technology are incorporated into the smart home system, such as wireless networking and cloud communication. The data that was saved to the cloud was then evaluated.

In [40] Ravi Kishore Kodali et al. builds a prototype Home Automation system focused on wireless home security. The system allows a client to be alerted via a phone call regardless of the client's device internet connectivity status. The home microcontroller, a TI CC3200 LaunchPad, has to be connected to the internet over wi-fi, however. The prototype does not implement any smartphone application but relies instead on digits from a cellphone or smartphone keypad; thus, the system is platform-independent and usable across different operating systems.

In [41] Kumar Mandula er al. designed a home automation system using Arduino as the system hardware controller and Bluetooth as the control communication enabler between smartphones and Arduino microcontroller.

Nicholas Dickey et al. in [42] demonstrates the use of a cloud network called pachube as a communication enabler in a home automation environment. The system features include: mobile phone application, remote controller, power-line communication, cloud networking and wireless communication using X10 and ZigBee protocol to provide the user with remote control of various lights and appliances within their home. The presence of a ZigBee enabled remote control allowed the control of the home devices without requiring an Internet connection.

Omar Hamdan et al. [43] in proposed a smart home automation system with features that allow for a dual-mode interaction. The microcontroller used had an in-built Wireless Access Point (WAP) used to communicate with a home server. The dual-mode of the system uses firstly implements a mobile app interface with sliders and virtual switches to monitor and control the home appliances. The second mode uses a machine learning feature called natural language processing to enable a chat-based text/audio commands interface to monitor and control the home appliances.

The literature review was conducted for papers that addressed some key challenges in home automation systems including control, monitoring and security. Solutions provided in the reviewed papers include availability if remote control with or without internet access, timeliness of information transformation and responsivity of the automate system, security in the home, of the automation devices' network and security of the client app if any. Related works reviewed also addressed redundance or alternative control design to improve the reliability of the system set-up or to provide flexibility of control options.

3. System Design

This section deals with the proof-of-concept and prototyping report based on the research carried out. Information on Design Procedures; Online/Cloud services requirement; House wiring design and architecture; System Architecture; assisting hardware and software tools; cost requirements; and practical use-cases are reported within this section. No cost was incurred using any software or cloud service as all software used within this research were free to use (some with limited capability), offered the privilege of a 30-day trial or was made available through institutional benefits.

3.1 Design Specifications

The design specifications include the functional and non-functional requirements; hardware and software requirements.

3.1.1 Functional and Non-Functional Requirements

Table 4 shows the functional requirements needed for the research.

Table 4. Functional requirement of Home Automation research

S/N	Function Requirement Description
1	The client should be able to monitor home temperature and humidity conditions using the client app.
2	The client should be able to see the state of the lights and the front door with the ability to control them as desired.
3	The client would need to use a legitimate Facebook login to access the client app for every startup. Auto-Login must be implemented.
4	The web interface must provide rich data presentation of data streams existing within the cumulative system.
5	The client should be able to set home into "Home", "Away", "Sleep", and "Awake" mode presets to batch control home devices.
6	The client must be able to set up and secure his device on a local network. Once the first authorization handshake occurs between the client app and field gateway device, no other client app can control the home devices except with the Client's Facebook credentials.
7	The client should be alerted through e-mail and SMS if an intruder breaks into the apartment.
8	Control and monitoring of home devices must be possible in any location where the user has access to an internet connection, provided the home gateway is also active and connected to the internet.
9	The home device must have an alternate means of control when the client app is unavailable or not desired for usage.

Table 5 shows the non-functional requirements of the implemented research.

Table 5. Non-Functional requirement of Home Automation research

S/N	Non-Function Requirement Description
1	The client app user interface should look production-ready.
2	Provision should be made for adding more components to the device side of the research incase more features are desired for implementation.
3	Client app should be cross-platform and should be able to run on
4	Device updates and changes to device configuration should be possible using cloud to device commands/messaging.

3.1.2 Hardware and Software Requirements

Note that not all purchases were needed or implemented within the current scope of this research, meaning some purchases were for redundancy in case of faults and if the scope of the research was to be expanded. Unless otherwise stated in Table 6 all purchases were made in Naira.

Table 6. Hardware Requirements cost table (in NGN).

S/N	Name	Quantity	Unit Cost (NGN)	Total Cost (NGN)
1	Raspberry pi 3b+	1	25000	25000
2	9 in 1 multifunctional Expansion shield	1	3500	3500
3	Arduino Pro Micro	1	3300	3300
4	2.2kΩ 1W resistor	10	15	150
5	Mg90s Servo	2	1700	3400
6	Relay module 2 channel	2	800	1600
7	Female-female jumper wire (set)	1	650	650
8	Male-female jumper wire (set)	1	650	650
9	Male-male Jumper wires (set)	2	650	1300
10	vero board big size stripped	2	300	600
11	Vero board small size stripped	3	150	450
12	Bi-directional logic level converter	3	550	1650
13	7cmx9cm double sided fiberglass PCB veroboard/stripboard	6	350	2100
14	UF5408 Diode	10	50	500
15	LCD 2004 Module	1	2500	2500
16	Plastic keyboard 4x4	1	2000	2000
17	Double Rocker Power Switch	3	300	900
18	10KΩ 1W resistor	10	15	150
19	BC547(NPN)	10	20	200
20	5mm red LED	3	10	30
21	5mm LED blue	5	10	50
22	3mm LED green	5	5	25
23	3mm LED yellow	5	5	25
24	40 pin 2mm single in line (SIL) female header	2	180	360
25	40pin 2mm single in line (SIL) male header	10	50	500
26	PIR motion sensor	2	1000	2000
27	Door magnetic switch normally open	1	400	400
28	Arduino uno r3	3	6500	19500
28	Light Bulb(200W), lamp holder & Type G plug	4	N/A	3000
29	Paper board Cartons	4	N/A	N/A(recycled used ones)
	Total Cost			**76490**

Table 7 shows software packages were used in the realization of this research. Note that while each software used could be replaced by another to perform the same function, certain specific software was more critical to the completion of the requirements of this research.

Table 7. Software tools used in Home Automation Research

S/N	Software Name	Producer	Requirement level	Plugins/Addons	References
1	Google Chrome	Google	Replaceable	N/A	
2	Visual Studio Community (2019)	Microsoft Corporation	Required	Visual Studio Build Tools 2017, Azure development .NET cross-platform development, Game development with Unity.	-
3	Visual Studio Code	Microsoft Corporation	Replaceable	C# support extension, Azure Tools extension, Debugger for Unity extension.	-
4	Arduino IDE	Arduino	Required	N/A	-
5	VNC viewer	ReAlVnc	Replaceable	N/A	-
6	Putty	Simon Tatham, Open-source	Replaceable	N/A	-
7	Proteus	Labcenter Electronics	Required	Arduino Library for Proteus V2.0, Proteus Libraries of Embedded Sensors.	[44][45]
8	Unity 3d	Unity Technologies	Required	Facebook SDK for Unity, Newtonsoft JSON for Unity, AdobeXD to Unity UI converter, M2MQTT for Unity, DoTween, Azure Functions for Unity.	[46][47][48]
9	Edraw Max	Wondershare	Replaceable	N/A	-
10	Software Ideas Modeler	SoftwareIdeas	Replaceable	N/A	-
11	Adobe XD	Adobe Inc.	Required	AdobeXD to Unity UI converter	[46]
12	AutoCad Web App	Autodesk	Replaceable	N/A	-

3.2 Circuit Design

The circuit designs in Figure 1 and 2 was a result of theoretical analysis and software simulation. The simulation circuit's visual connection design was set up and tested in Proteus software. Due to the limitations of a simulation environment, the real-life circuit implementation and result varied to some degree compared to what was simulated.

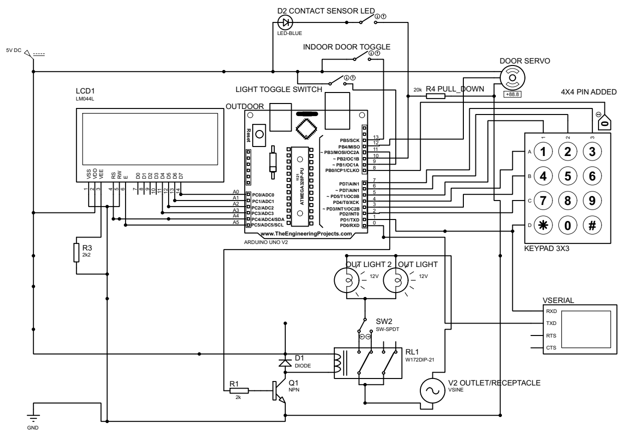


Fig 1. Outdoor devices Control circuit design.

The Outdoor Devices Control Design Review consist of the Arduino to which analogue and digital pins are connected. This microcontroller board has General-Purpose Input Output (GPIO) pins usable as input (or INPUT_PULLUP) or output when interfacing with other components. In the design shown in Figure 1, the Arduino GPIO are mapped according to the following:

1. GPIO digital pins (0-7, 5V): These pins are connected to the keypad and are responsible for scanning inputs made on the matrix designed keypad. The keys on the keypad have beneath each of them a membrane switch. The membranes are connected together according to their row-column configuration by conductive traces underneath the keypad. Each column and also row terminates to a single pin. The column membranes connected pins are set as input (INPUT_PULLUP) and are held HIGH by default; the row membranes connected pins are set as output and held LOW by default, this makes an electrical loop possible in a matrix form. When one of the buttons is pressed, the column pin becomes LOW because the current entering the HIGH column starts flowing into the LOW row pin of that button, as shown in Figure 2. The pressed button row is identified. The Arduino then switches each of the row pins to HIGH while reading all the column pins to detect which column pin returns to HIGH. During the row pin switching, if a column pin goes HIGH again, the Arduino detects that column as the pressed button column and is able to identify the button that was pressed.

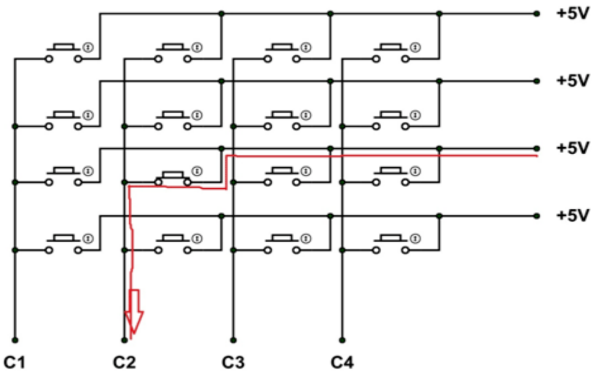


Fig 2. Keypad schematic.

2. GPIO analogue pins (A0-A5): These pins are connected to the LCD module with the following configuration: Analog A0 to A3 serves as data output pins to the LCD module. Analog A4 is used to switch between the LCD's command and data registers. When set to HIGH, the data register is selected, and output from the data output would cause data to be displayed on the screen. When set to LOW, the command register is selected, and the output from the data output pins would be detected as a command, such as positioning the cursor or clearing the screen.
3. GPIO digital pin 9: This pin is connected to an indicator red led. The purpose of the led is to signify to the client that the door is unlocked (LOW) or locked (HIGH). The limiting resistor is chosen as thus:
   LED forward operating voltage ≈ 1.8V
   LED forward current ≈ 20mA
   LED limiting resistance = $\frac{Vin-Vled}{Iled}$ (1)
   $= \frac{5-1.8}{20*(10^{-3})} = 160\Omega$, 220Ω chosen
4. GPIO digital pin 10: This pin is connected to a relay through a BC547 NPN transistor. This set-up is to allow the Arduino to control the outdoor light bulbs. The NPN serves as a current amplifier for the Arduino because the Arduino's maximum output single pin current is 40mA, but the relay requires a current of approximately 70mA or greater. The following design was realized.
   Relay Trigger voltage = 5V DC
   Relay Trigger current ≥ 70mA: **100mA** chosen
   Relay Max AC load(V_2) = 10A@250V AC: **230V** used
   NPN max hFE = 500, **50** chosen
   Arduino pin output (I_b) ≤ 40mA,
   output used: $I_b = \frac{Ic}{hFE}$ (2)
   $= \left(\frac{100}{50}\right) = 2mA$
   NPN limiting resistance = $\frac{Vin-Vbe}{Ib}$ (3)
   $= \frac{5-0.7}{2*(10^{-3})} = 2{,}150\Omega \approx 2.2k\Omega$

An inverted Single Pole Double Throw switch is connected to the Relay output to simulate a 2-way switch and allows toggling/switching of control between the Arduino and the switch.

5. GPIO digital pin 12: This pin serves as output to control the door controller servo. The servo angle is controlled using Pulse Code Modulation (PCM) by the control pulse duration. Two angles were used during the control; 1.5ms pulse for 90° and 1ms pulse for 0° as seen in Figure 3.3 In the servo, a simple potentiometer detects(rotates) and feeds back the servo position (as voltage) to the error detector for comparison (potentiometer voltage vs signal voltage) to the target position. The error margin from the comparison is used to match the servo position with the target position until an error of zero (0) is reached
6. GPIO digital pin 13: This pin is used as an input pin to detect when the indoor door switch is toggled or has changed state. A change in the state of the button causes a toggle of the door controller state from 0 to 90 and vice versa.

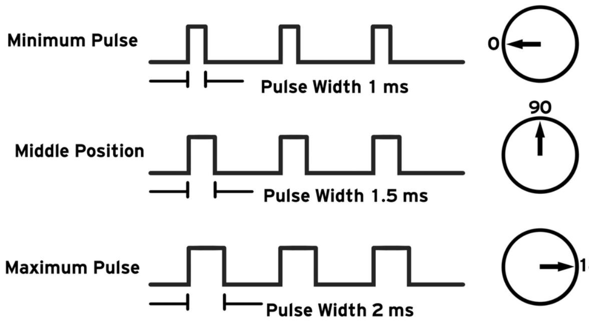


Fig 3. Servo motor control configuration.

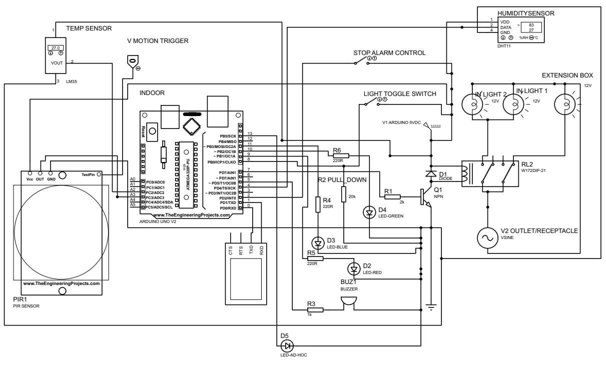


Fig 4. Indoor Devices control circuit design.

The Indoor Devices Control Design Review consist of the Arduino to which analogue and digital pins are connected as explained:

1. Analog pin(A2): This pin is configured as an input for the LM35 temperature sensor. Although a DHT11 exists within the circuit, which could also double a temperature sensor, it is not as accurate. A conversion to the Celsius scale of the value received is seen in (3.4).
   $\left(\frac{val}{1024.0}\right) * 500$ (4)
2. Digital pins (9,10,11&13): These pins are output pins connected to LEDs in the same way GPIO digital pin 9 from Figure 3.1 is connected. The connected LEDs blink in a pattern that signifies intrusion detection.
3. Analogue pin A3: This pin serves as an analogue-to-digital converter (0V – 0:5V-1023) input to the Passive Infrared (PIR) motion sensor. When a heat-emitting object (e.g., human body) appears within the detection area, the infrared energy radiation is detected by the pyroelectric sensor within the motion sensor, and a positive differential change value (as voltage) is sent to the Arduino analogue signal. A negative value when the object leaves the detection area. The Arduino analogue pin can detect the differential value. This value is used to set the motion detection variable within the Arduino code using if statements and a threshold value of 50. Upon startup, it takes about 60 seconds for the motion sensor to properly calibrate itself to its environment before it can properly detect motion. This calibration time has been accounted for by a delay within the Arduino code.
4. Digital pin 7: This pin acts as an output to control the relay. The configuration is the same as the one used in Figure 3.1.
5. Digital pin 5: This pin is an output connected to a piezo buzzer through a 1kohm limiting resistor chosen arbitrarily to remove the sharpness from the tones. The contraction and expansion of the ceramic disc of the buzzer cause vibration of air molecules which is detected

as sound. The buzzer outputs sound when an intrusion is detected, and the tone control frequencies are built into the Arduino code.

6. Digital pin 4: This pin serves as an input to the DHT11 humidity sensor and reads the relative Humidity around it.
7. Digital pin 2: This pin is set as a digital input to a switch. If the switch is toggled while the alarm is active, the alarm is toggled off. The switch will be situated indoors. Note that switching off the alarm does not prevent an alert E-mail or SMS from being sent to the client.

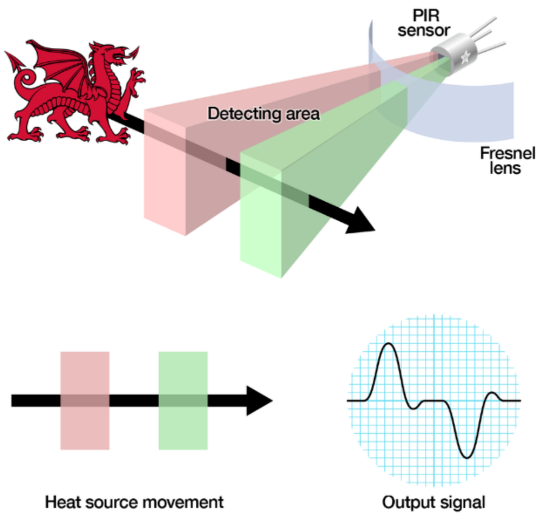


Fig 5. PIR motion detection.

The Serial Communication Scheme implemented consists of a bi-directional data transfer was implemented between the Arduinos and Raspberry Pi. The communication was implemented over Serial using Arduino compatible USB cables. This allowed full-duplex data transfer between the Raspberry Pi and Arduinos. The raspberry Pi communicates with the Arduino by sending a two-digit integer over serial to the Arduino; then, the Arduino reads the data and decodes the corresponding instructions through a switch statement within its program. Figure 6 and 7 show the bi-directional communication scheme. Similarly, the Arduinos send response data to the Raspberry Pi by sending a three-segment string over serial to the Arduino. The strings are differentiated using a semicolon (;) and a colon (:).

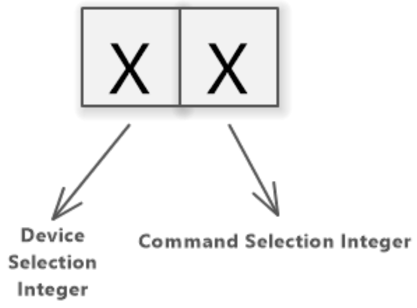


Fig 6. Raspberry Pi to Arduino serial communication scheme.

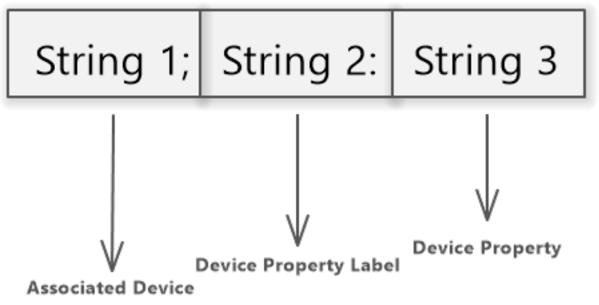


Fig 7. Arduino to Raspberry Pi serial communication scheme.

3.3 Software Design

The software was designed using AdobeXD, Unity3d and Visual Studio Code. A predesigned UI Kit compatible with AdobeXD was downloaded from the internet [49]. A plugin was then used to export the UI contents into a Unity3d friendly set of media files[46]. Extensive editing had to be performed, though, in order to make the user interface interactable by the client. Figure 8 shows the interface of program, called unity used to make the client app.

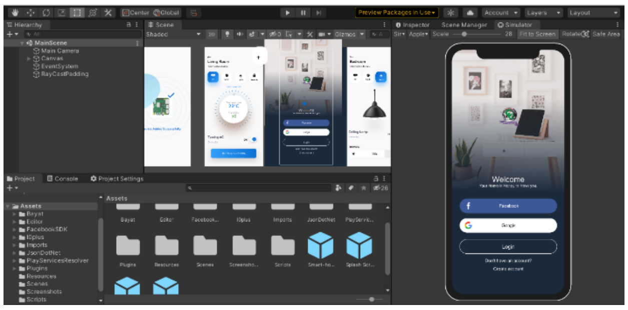


Fig 8. Smart home research app design in Unity. Adapted from [49].

3.4 Online Services Requirement

Table 8 shows the online application and services used to meet the functional requirements of this research.

Table 8. Online services used in Home Automation Research

S/N	Software Name	Producer	Description
1	Twilio	Twilio	Communication API for SMS voice video and authentication.
2	Facebook for Developers	Facebook	Platform that provides APIs to connect businesses with customers.
3	Azure Functions	Microsoft Corporation	On-demand cloud service that provides continually updated infrastructure and resources needed to applications on the cloud[50].
4	Azure Logic App	Microsoft Corporation	cloud-based platform for creating for automating workflows that integrate organization apps, data, services, and systems[50].
5	Azure Time Series Insights environment	Microsoft Corporation	End-to-end IoT analytics platform to monitor, analyze, and visualize your industrial IoT analytics data at scale[50].
6	Azure IoT Hub	Microsoft Corporation	Managed service for bidirectional communication between IoT devices and Azure[50].

7	Azure Recource Group	Microsoft Corporation	A container that holds cloud resources for an Azure solution[50].
8	Azure Application Insights	Microsoft Corporation	An extensible APM(Application Performance Management) service for DevOps professionals and developers[50].
9	Azure App Service plan	Microsoft Corporation	Defines a set of compute resources to run a web app[50].
10	Azure Service Bus Namespace	Microsoft Corporation	A fully managed enterprise grade message broker with publish-subscribe topics and message queues[50].
11	Azure Storage Account	Microsoft Corporation	Provides a unique namespace Azure Storage data and is made accessible from anywhere in the world[50].
12	Azure API connection	Microsoft Corporation	Provides prebuilt API operations to aid steps in your workflows.

3.5 House Wiring
The original house plan was sourced from the interne [51]. The research's house wiring as seen in Figure 9 was designed using a demo version of Edraw Max. Other software such as Microsoft Visio or draw io would also have been good alternatives to visualize the design.

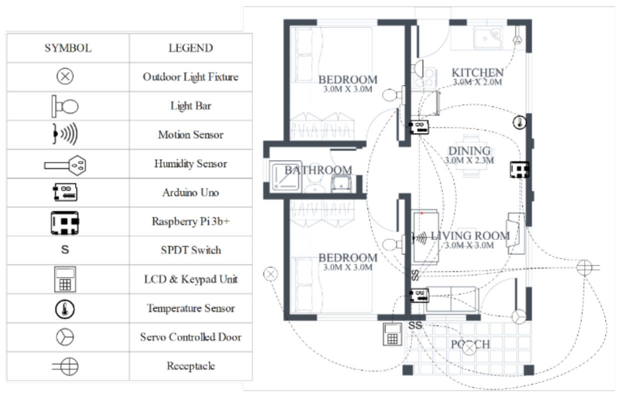


Fig 9. House Wiring Scheme for Home Automation Research.

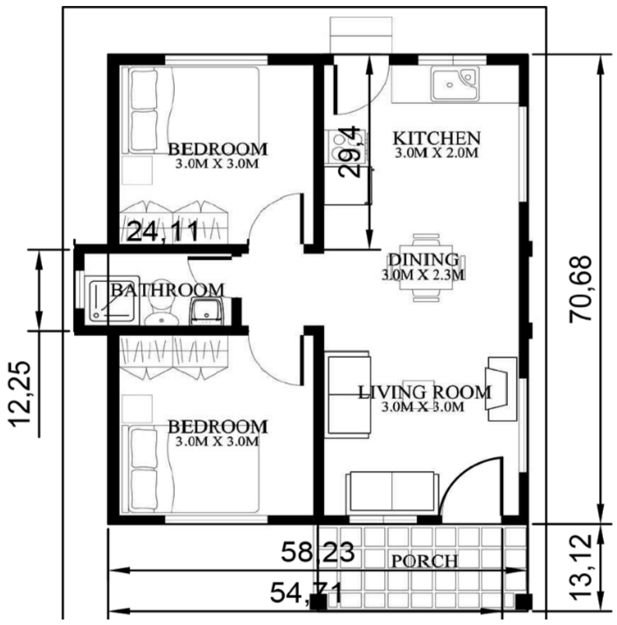


Fig 10. Smart Home research measurements (cm).

3.6 Architecture
The reference architecture is seen in Figure 11 was based on Microsoft's recommended architecture for IoT applications on Azure using platform-as-a-service(PaaS) components [5].

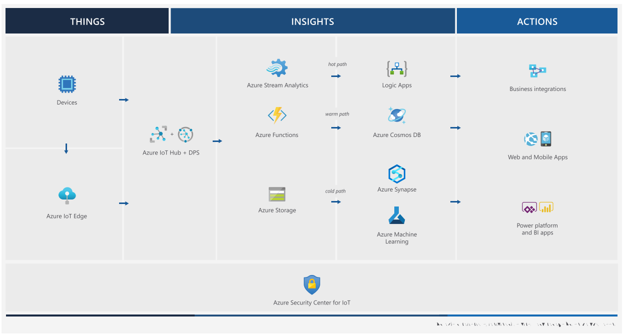


Fig 11. Azure IoT Reference Architecture.

3.7 Security
Earlier reviewed in this work are various concerns pertaining to IoT and Home Automation systems. The following features were implemented into the Smart Home System.

- A New or newly reset field gateway device (raspberry Pi) cannot send telemetry nor receive commands until it first authenticates a client on its local network.
- Newly installed client apps cannot send commands or receive telemetry from any field gateway until they are first authenticated to do so by connecting to the field gateway device on a local network.
- The client will need an authentic Facebook credential to use the app at any point in time, and a login (automatic or manual) will be required every time the client opens the client app freshly.
- A specific field gateway can only be accessed via the first Facebook credential used to authenticate a client unless a data reset is performed on the field gateway device.

All sensitive data are stored in a secure location within the cloud and are only accessible with elevated access privileges or using the client's authenticated app. Table 3.6 summarizes the authentication flow implemented in the research.

Table 9. Smart Home user authentication flow

	Step	Without provider SDK
1	Sign user in	App directs client to /.auth/login/<provider>.
2	Post-authentication	Provider redirects client to /.auth/login/<provider>/callback.
3	Auth session Established	App Service adds an authenticated cookie to the response.
4	Serve authenticated content	Client app (auto)includes auth cookie in following requests.

4. System Implementation, Testing and Results
This section of the work highlights the tests procedures and results of the research implementation. The overall implementation of the system design yields a largely asynchronous system where sections of the research are not tightly dependent on another section, for example, the Azure IoT hub does not require the raspberry Pi in order to functions properly even though data needs to be transferred in a bi-directional manner between both gateways. Due to this asynchronicity, use cases have been chosen as the best way to demonstrate how the system responds to various inputs.

Figure 12 gives an overview of the use case pattern followed during testing.

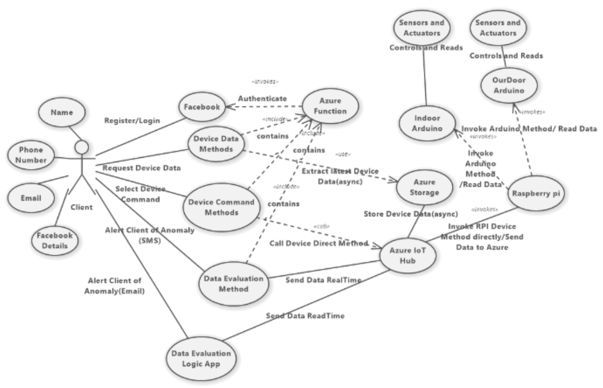


Fig 12. Home Automation System Use Case.

4.1 Circuit Testing

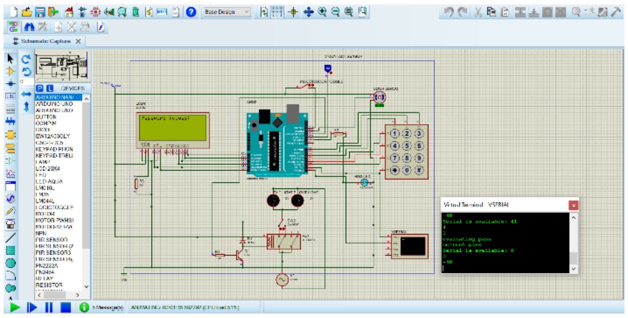
Tests on the circuit were carried out firstly via the Proteus simulator to ascertain that the design would function as expected. Inputs to the simulation was given via a virtual terminal, and outputs were observed via the virtual terminal console logs. Figure 13 and 14 show the simulation in action.

4.1.1 Outdoor Circuit Virtual Test

Table 10 shows the serial output and behavioral response of the outdoor circuit simulation to different serial inputs. The serial input data was provided via a virtual terminal within the simulator.

Table 10. Smart Home Outdoor Circuit Serial Communication Test (virtual)

Input (integer)	Serial output (string)	Expected result	Actual result	Comment
10	mydoorsensor;property2:False	Returns result as serial output	Returns result as serial output	Virtual trigger(set to 0) used to emulate door sensor state
20	mydoorcontroller;property2:0	Returns result as serial output	Returns result as serial output	-
30	N/A	Turns off light	Turns off light	-
31	N/A	Turns on light	Turns on light	-
32	light control accepted	Toggles lights	Toggles lights	-
40	wrong pass	set servo rotation to 0	servo rotation set to -90	Virtual servo is negative 90 degrees out of phase with code designated rotation.
41	correct pass	set servo rotation to 90	servo rotation set to 0	Virtual servo is negative 90 degrees out of phase with code designated rotation.
42	default pass result	N/A	N/A	-



```
Virtual Terminal - VSERIAL
-90
Serial is available: 41
4
1
evaluating pass
correct pass
Serial is available: 0
0
-90
```

Fig 13. Proteus simulation of outdoor circuit.

4.1.2 Outdoor Circuit Virtual Test

Similar to the virtual indoor circuit test, the serial input data was provided via a virtual terminal within the simulator. Test inputs and results can be seen in Table 11.

Table 11. Smart Home Indoor Circuit Serial Communication Test (virtual)

Input (integer)	Serial output (string)	Expected result (virtual)	Actual result	Comment
10	mytemperaturesensor;property2:27.34	Returns result as serial output	Returns result as serial output	-
20	myhumiditysensor;property2:83	Returns result as serial output	Returns result as serial output	-
30	mymotionsensor;property2:676	Returns result as serial output	Returns result as serial output	A virtual 3.3V DC was used to trigger state to high
40	turn off light	Turns off light	No change observed	Simulator took too long to respond
41	turn on light	Turns on light	No change observed	Simulator took too long to respond
42	toggle light	Toggles lights	No change observed	Simulator took too long to respond
50	toggle sound alarm	Toggles Buzzer state	No change observed	-

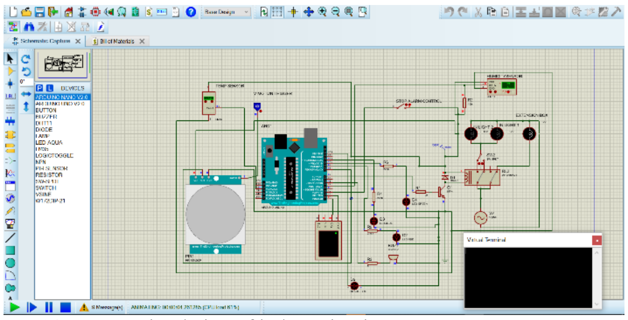


Fig 14. Proteus simulation of indoor circuit.

4.1.3 Outdoor Circuit Serial Communication Test (physical)

Upon building the Model physically, tests were performed on the outdoor circuit by connecting the outdoor Arduino via USB to a laptop computer. This method allowed testing of the circuit as an isolated unit without connecting the Arduino device to the raspberry pi. Table 12 displays the serial inputs and observed outputs.

Table 12. Smart Home Outdoor Circuit Serial Communication Test (physical)

Input (integer)	Serial output (string)	Expected result	Actual result	Comment
10	mydoorsensor;property2:False	Returns result as serial output	Returns result as serial output	Door was closed
10	mydoorsensor;property2:True	Returns result as serial output	Returns result as serial output	Door was open
20	mydoorcontroller;property2:0	Returns result as serial output	Returns result as serial output	Door was closed
20	mydoorcontroller;property2:90	Returns result as serial output	Returns result as serial output	Door was open
30	N/A	Turns off light	Turns off light fixtures	
31	N/A	Turns on light	Turns on light fixtures	-
32	light control accepted	Toggles lights	Toggles state of light fixtures	-
40	wrong pass	set servo rotation to 0	servo rotation set to -90	-
41	correct pass	set servo rotation to 90	servo rotation set to 0	-
42	default pass result	N/A	N/A	-

4.1.4 Indoor Circuit Serial Communication Test (Physical)

Serial tests were similarly conducted on the indoor circuit to the outdoor circuit test. Table 13 shows the test inputs and corresponding results.

Table 13. Smart Home Indoor Circuit Serial Communication Test (Physical)

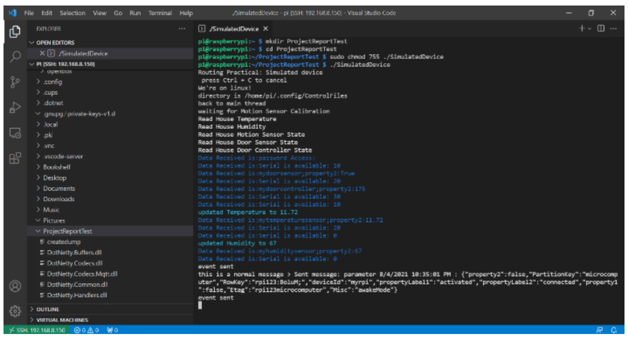
Input (integer)	Serial output (string)	Expected result	Actual result	Comment
10	mytemperatursensor;property2:27.34	Returns result as serial output	Returns result as serial output	-
20	myhumiditysensor;property2:83	Returns result as serial output	Returns result as serial output	-
30	mymotionsensor;property2:False	Returns result as serial output	Returns result as serial output	Motion sensor requires approx. one minute to calibrate
40	turn off light	Light fixtures turn off	Light fixtures turn off	-
41	turn on light	Light fixtures turn on	Light fixtures turn on	-
42	toggle light	Light fixtures toggle state	Light fixtures toggle state	-
50	toggle sound alarm	Buzzer toggles state	Buzzer toggles state	-

4.2 Device Side Telemetry Monitoring

Monitoring the research's activities was performed using Visual Studio Code, Azure IoT hub and Azure Function. Visual Studio Code was used in displaying JSON formatted telemetry console logs being sent to Azure IoT hub as a means of on-site monitoring (Figure 15). An Azure function was used to observe the logs as a means of off-site or remote observation (Figure 16). The Azure Function's primary task was to store the telemetry data in Azure Table storage for asynchronous access by clients' devices but doubled as a log reader. The device-side code was compiled as a .NET5.0 research using the command dotnet publish to Linux command. The compiled files were transferred to the raspberry Pi using the "scp" command.

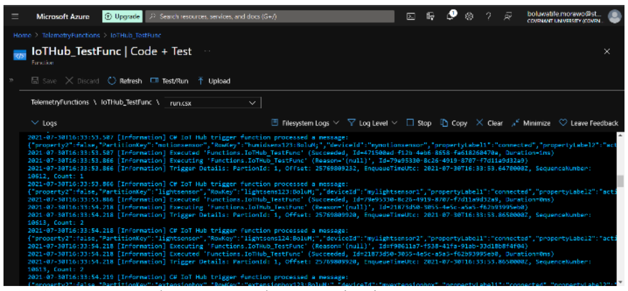
While the Azure Function was used to display the verbose logs, IoT hub monitoring provided flexible graphical information on various measurable metrics (Figure 17). The metrics measured using IoT hub are: Failed direct method invocations, Successful direct method invocations, routing: telemetry messages delivered to endpoints, routing: telemetry messages orphaned and device-to-cloud (D2C) telemetry messages sent.

The graph type was set to a line chart; the measurement time range(X-scale) was set to 24-hours; time granularity was set to local, and the Y-scale was set automatically by Azure IoT hub. Azure Time Series Insights (TSI) was also tested (Figure 18). TSI is built to be a more powerful data analytics and visualization tool service more than just data routed from an IoT hub.


./SimulatedDevice - pi [SSH: 192.168.8.150] - Visual Studio Code
./SimulatedDevice

```
pi@raspberrypi:~ $ mkdir ProjectReportTest
pi@raspberrypi:~ $ cd ProjectReportTest
pi@raspberrypi:~/ProjectReportTest $ sudo chmod 755 ./SimulatedDevice
pi@raspberrypi:~/ProjectReportTest $ ./SimulatedDevice
Routing Practical: Simulated device
 press Ctrl + C to cancel
We're on linux!
directory is /home/pi/.config/ControlFiles
back to main thread
waiting for Motion Sensor Calibration
Read House Temperature
Read House Humidity
Read House Motion Sensor State
Read House Door Sensor State
Read House Door Controller State
Data Received is:password Access:
Data Received is:Serial is available: 10
Data Received is:mydoorsensor;property2:True
Data Received is:Serial is available: 20
Data Received is:mydoorcontroller;property2:175
Data Received is:Serial is available: 30
Data Received is:Serial is available: 10
updated Temperature to 11.72
Data Received is:mytemperaturesensor;property2:11.72
Data Received is:Serial is available: 20
Data Received is:Serial is available: 0
updated Humidity to 67
Data Received is:myhumiditysensor;property2:67
Data Received is:Serial is available: 0
event sent
this is a normal message > Sent message: parameter 8/4/2021 10:35:01 PM : {"property2":false,"PartitionKey":"microcomp
uter","RowKey":"rpi123:BoluM;","deviceId":"myrpi","propertyLabel1":"activated","propertyLabel2":"connected","property1
":false,"Etag":"rpi123microcomputer","Misc":"awakeMode"}
event sent
```

Fig 15. On-site telemetry test logs monitoring.


Microsoft Azure
Home > TelemetryFunctions > IoTHub_TestFunc
IoTHub_TestFunc | Code + Test
Function
Save
Discard
Refresh
Test/Run
Upload
TelemetryFunctions \ IoTHub_TestFunc \ run.csx
Logs
Filesystem Logs
Log Level
Stop
Copy
Clear
Minimize
Leave Feedback

```
2021-07-30T16:33:53.587 [Information] C# IoT Hub trigger function processed a message:
{"property2":false,"PartitionKey":"motionsensor","RowKey":"humidsens123:BoluM;","deviceId":"mymotionsensor","propertyLabel1":"connected","propertyLabel2":"act
2021-07-30T16:33:53.587 [Information] Executed 'Functions.IoTHub_TestFunc' (Succeeded, Id=471508ad-f12b-4eb6-8658-fa618268470a, Duration=1ms)
2021-07-30T16:33:53.866 [Information] Executing 'Functions.IoTHub_TestFunc' (Reason='(null)', Id=79e95330-8c26-4919-8707-f7d11a9d32a9)
2021-07-30T16:33:53.866 [Information] Trigger Details: PartionId: 1, Offset: 25769809232, EnqueueTimeUtc: 2021-07-30T16:33:53.6470000Z, SequenceNumber:
10612, Count: 1
2021-07-30T16:33:53.866 [Information] C# IoT Hub trigger function processed a message:
{"property2":false,"PartitionKey":"lightsensor","RowKey":"lightsens123:BoluM;","deviceId":"mylightsensor1","propertyLabel1":"connected","propertyLabel2":"acti
2021-07-30T16:33:53.866 [Information] Executed 'Functions.IoTHub_TestFunc' (Succeeded, Id=79e95330-8c26-4919-8707-f7d11a9d32a9, Duration=0ms)
2021-07-30T16:33:54.218 [Information] Executing 'Functions.IoTHub_TestFunc' (Reason='(null)', Id=21873d50-3055-4e5c-a5a5-f62b93995eb0)
2021-07-30T16:33:54.218 [Information] Trigger Details: PartionId: 1, Offset: 25769809920, EnqueueTimeUtc: 2021-07-30T16:33:53.8650000Z, SequenceNumber:
10613, Count: 2
2021-07-30T16:33:54.218 [Information] C# IoT Hub trigger function processed a message:
{"property2":false,"PartitionKey":"lightsensor","RowKey":"lightsens124:BoluM;","deviceId":"mylightsensor2","propertyLabel1":"connected","propertyLabel2":"acti
2021-07-30T16:33:54.218 [Information] Executing 'Functions.IoTHub_TestFunc' (Reason='(null)', Id=f98611a7-f538-41fa-91ab-33d18b8f4f04)
2021-07-30T16:33:54.218 [Information] Executed 'Functions.IoTHub_TestFunc' (Succeeded, Id=21873d50-3055-4e5c-a5a5-f62b93995eb0, Duration=0ms)
2021-07-30T16:33:54.218 [Information] Trigger Details: PartionId: 1, Offset: 25769809920, EnqueueTimeUtc: 2021-07-30T16:33:53.8650000Z, SequenceNumber:
10613, Count: 2
2021-07-30T16:33:54.219 [Information] C# IoT Hub trigger function processed a message:
{"property2":false,"PartitionKey":"extensionbox","RowKey":"extensionbox123:BoluM;","deviceId":"myextensionbox","propertyLabel1":"connected","propertyLabel2":"
```

Fig 16. Remote telemetry test logs monitoring.

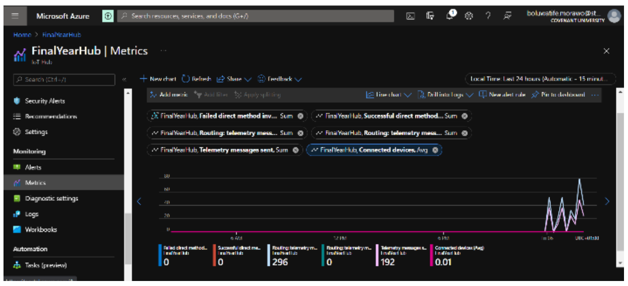


Fig 17. Remote system monitoring (line chart)

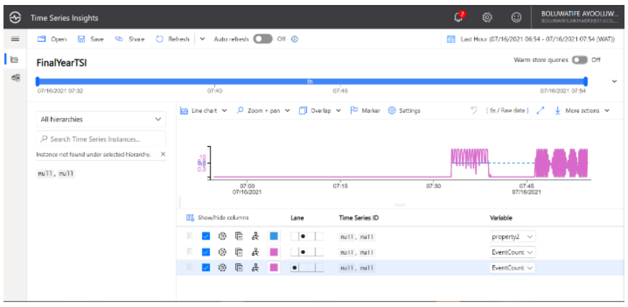
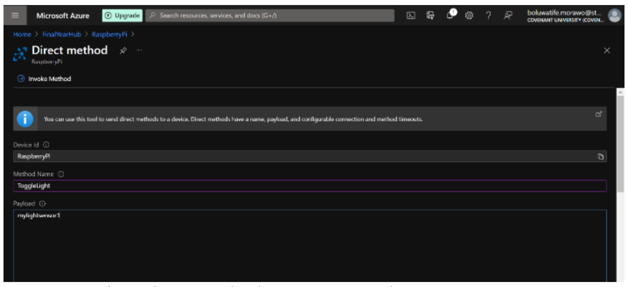


Fig 18. Remote system Monitoring (Time Series Insights).


Microsoft Azure
Upgrade
Search resources, services, and docs (G+/)
Home > FinalYearHub > RaspberryPi >
Direct method
RaspberryPi
Invoke Method

You can use this tool to send direct methods to a device. Direct methods have a name, payload, and configurable connection and method timeouts.

Device Id: RaspberryPi
Method Name: ToggleLight
Payload: mylightsensor1

Fig 19. Device Direct Method Azure IoT-Hub test

4.2.1 Device Direct Method Tests

Device direct method tests were conducted using Azure IoT hub Device direct method call feature (Figure 19). The Arduino microcontrollers were assigned permanent port labels to ensure consistency of recognition by the Raspberry pi even if the Raspberry pi has to restart. Similar to serial communication tests, the method of testing employed allowed the device side set-up to be treated like an isolated used without the need for a test mobile device/tablet to invoke the device direct methods. Table 14 shows the results of the device direct method test.

Table 14. Device Direct Method tests result

Method Name	payload	Expected result	Actual result	Comment
ToggleLight	mylightsensor2	Toggles the outdoor light fixtures	Outdoor light fixtures toggled	-
ToggleLight	Mylightsensor1	Toggles the indoor light fixtures	Indoor light fixtures toggled	-
TogglePresenceMode	1	Toggles the presence mode	Presence mode toggled.	Presence mode toggled between home and away
ToggleSleepMode	1	Toggles sleep mode	Sleep mode toggled	Presence mode toggled between sleepMode and awake mode
ToggleDoorLock	1	Toggles the door state	Door state toggled	-
Door	1	Set the door state to locked if it wasn't already locked	No observed change	-
Lock Door	1	Set the door state to locked if it wasn't already locked	Door state changed to locked physical	The door was previously in an open state
ToggleMotionSensor	1	Deactivate the motion sensor	No observable change	Feature not implemented
ToggleExtensionBox	1	Toggle extension box state	No observable change	Feature not implemented

4.3 Onboarding Tests

After the completion of telemetry and device direct method tests, the client app was compiled to target Android OS and launched on a Tecno tablet. User Onboarding tests were then conducted from the tablet computer. Results of the onboarding tests are as shown in Table 15.

Table 15. User Onboarding test result

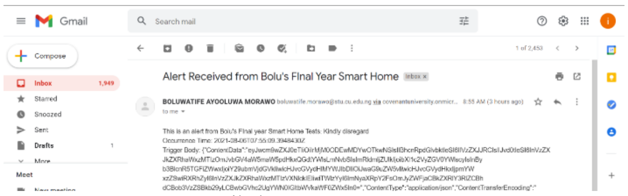
S/N	User Action	Expected result	Actual result	Comment
1	User opens application	The user is taken to the get started screen	The user is taken to the get started screen	-
2	User taps on the get started button	The user is taken to a profile details screen to input basic details	The user is taken to a profile details screen to input basic details	-
3	User inputs basic details and taps on the "Save Profile" button	The user is taken to the login screen	The user is taken to the login screen	The user's phone number was stored for SMS alert purposes
4	User taps on the Facebook login button	The user is taken to a new sub-browser window to login using Facebook credentials	The user is taken to a new sub-browser window to login using Facebook credentials	Presence mode toggled between sleepMode and awake mode
5	User logs in successfully and authenticated app to use Facebook ID	User is taken back to the app windows and to the discover devices screen	User is taken back to the app windows and to the discover devices screen	-
6	User taps on the discovered device UI	User is taken to the app's home page	User is taken to the app's home page	Device data is stored as part of the auto authentication requirement for next app startup.

4.4 Alert Tests

The system was configured to send SMS and E-mail alerts automatically under pre-configured event or conditional context. SMS operations were realized using Twilio API and E-mail alerts were implemented using Gmail SMTP. Figures 20, 21 and Table 16 show the result of the automatic Alert tests.

Table 16. Smart Home alerts test results

S/N	Alert Context	Expected result	Actual result	Comment
1	Room temperature exceeds 40 degrees Celcius	Alert Client by E-mail and SMS	Client device alerted by E-mail and SMS successfully	-
2	Room relative Humidity Exceeds 55%	Alert Client by E-mail and SMS	Client device alerted by E-mail and SMS successfully	-
3	Motion sensor detects movement while the client device is in away mode	Alert Client by E-mail and SMS	Client device alerted by E-mail and SMS successfully	The user's phone number was stored for SMS alert purposes
4	Door sensor detects open door while the client device is in sleep mode	Alert Client by E-mail and SMS and shut door automatically.	Client device alerted by E-mail and SMS successfully	Presence mode toggled between sleepMode and awake mode
5	Motion sensor and door sensor detects a breaking in through the door	Alert Client by E-mail and SMS	Client device alerted by E-mail and SMS successfully	-
6	Indoor and Outdoor lights are on	Inform Client by E-mail and SMS	Client device alerted by E-mail and SMS successfully	Device data is stored as part of the auto authentication requirement for next app startup.

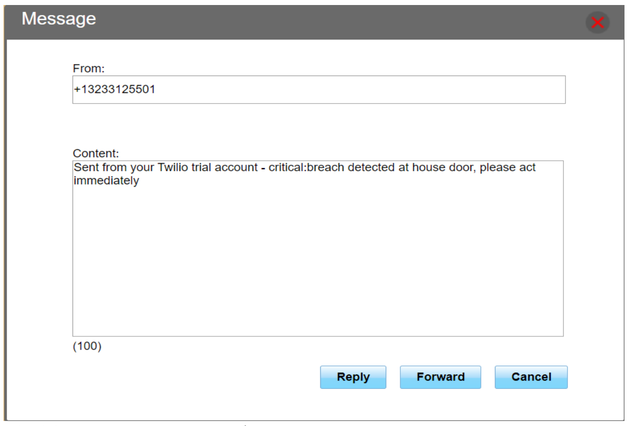


Alert Received from Bolu's Final Year Smart Home Inbox

BOLUWATIFE AYOOLUWA MORAWO boluwatife.morawo@stu.cu.edu.ng via covenantuniversity.onmicr... 8:55 AM (3 hours ago)
to me

This is an alert from Bolu's Final year Smart Home Tests: Kindly disregard
Occurrence Time: 2021-08-06T07:55:09.3948430Z
Trigger Body: {"ContentData":"eyJwcm9wZXJ0eTIiOiIrMjM0ODEwMDYwOTkwNSIsIBhcnRpdGlvbklleSI6IlVzZXJJRClsIJvd0tleSI6InVzZX JkZXRhaWxzMTIzOmJvbGV4aW5maW5pdHkxQGdIYWlsLmNvbSIsImRldmljZUlkIjoibXI1c2VyZGV0YWlscylsInBy b3BlcnR5TGFiZWwxIjoiY29ubmVjdGVkIiwicHJvcGVydHlMYWJlbDIiOiJwaG9uZW5vIiwicHJvcGVydHkxIjpmYW xzZSwiRXRhZyI6InVzZXJkZXRhaWxzMTIzVXNlckIEIiwiTWlzYyI6ImNyaXRpY2FsOmJyZWFjaCBkZXRlY3RlZCBh dCBob3VzZSBkb29yLCBwbGVhc2UgYWN0IGltbWVkaWF0ZWx5In0=","ContentType":"application/json","ContentTransferEncoding":"

Fig 20. Smart Home E-mail alert test



Message

From:
+13233125501

Content:
Sent from your Twilio trial account - critical:breach detected at house door, please act immediately

(100)

Reply Forward Cancel

Fig 21. Smart Home SMS alert test

4.5 Cost Monitoring

Incurred costs from the usage of could service was monitored using the Azure IoT portal and the Twilio portal as seen in Figure 22 and 23.

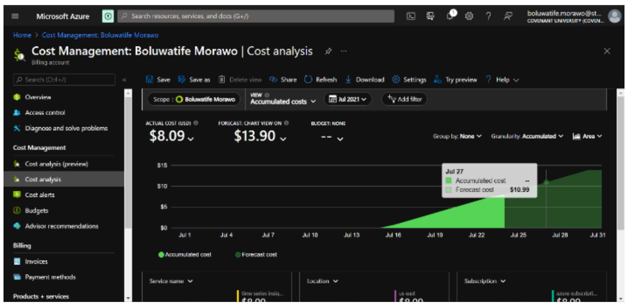


Fig 22. Azure IoT Cost Management portal

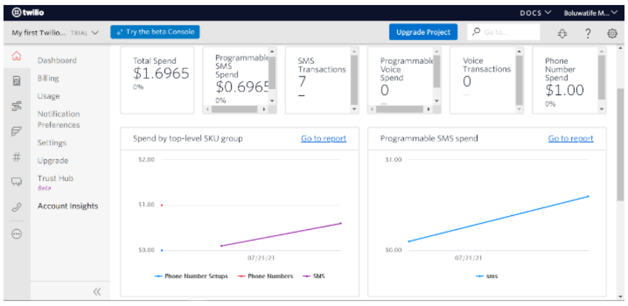


Fig 23. Twilio Cost Management portal

4.6 Conclusion

This section detailed the implementation of the software functional and non-functional requirements; the device side, cloud side and client-side tests. The use of data gathering and processing tools was explained and how they can be used to inform a client automatically of alert worthy events and conditions. Source code developed in this research can be found in [52][53][54]. Figure 24 shows the completed physical prototype.

5. Conclusions

The key achievements within the research scope include: Successful Design of a smart home model using mathematical analysis and software simulation tools; Successful construction of a prototype model of a smart home with basic functionality using azure IoT services with Raspberry Pi and Arduino compatible devices; Successful implementation of data monitoring capabilities into the smart home model; Successful set-up automated e-mail and SMS alerts within predefined contexts; Successful Creation a presentation of the working model based on the data collected and the prototype built; and Successful creation of a user-friendly client app to interact with the prototype smart home.

A good number of opportunities arise when on-site systems become connected to the cloud. Processes once thought of as prohibitively expensive or time-consuming suddenly become more feasible; data insights previously not accessible can be easily accessed, and control can be transferred to any internet covered location on the Earth.

References

1. The free encyclopedia Wikipedia, “Fourth Industrial Revolution,” 2021..
2. Microsoft, “Introduction to the Internet of Things (IoT),” 2021. https://docs.microsoft.com/.
3. T. A. Abdulrahman, O. H. Isiwekpeni, N. T. Surajudeen-Bakinde, and A. O. Otuoze, “Design, Specification and Implementation of a Distributed Home Automation System,” in Procedia Computer Science, 2016, vol. 94, doi: 10.1016/j.procs.2016.08.073.
4. R. P. Foundation, “Raspberry Pi Documentation.” https://www.raspberrypi.org/documentation.
5. M. Corporation, “Azure IoT reference architecture.” https://docs.microsoft.com/en-us/azure/architecture/reference-architectures/iot.
6. [M. H. Moubarak, “Internet of Things for Home Automation,” Researchgate.Net, 2016.
7. H. Yamauchi and A. Lustica, “Audio and video over IP technology,” in Proceedings Elmar - International Symposium Electronics in Marine, 2015, vol. 2015-November, doi: 10.1109/ELMAR.2015.7334512.
8. R. S. Ramanujan, J. A. Newhouse, M. N. Kaddoura, A. Ahamad, E. R. Chartier, and K. J. Thurber, “Adaptive streaming of MPEG video over IP networks,” 1997, doi: 10.1109/lcn.1997.631009.
9. A. Cyril Jose and R. Malekian, “Smart Home Automation Security: A Literature Review,” Smart Comput. Rev., 2015, doi: 10.6029/smartcr.2015.04.004.
10. S. Ansari, S. G. Rajeev, and H. S. Chandrashekar, “Packet sniffing: A brief introduction,” IEEE Potentials, vol. 21, no. 5. 2002, doi: 10.1109/MP.2002.1166620.
11. Z. Duan, X. Yuan, and J. Chandrashekar, “Controlling IP spoofing through interdomain packet filters,” in IEEE Transactions on Dependable and Secure Computing, 2008, vol. 5, no. 1, doi: 10.1109/TDSC.2007.70224.
12. A. Jurcut, T. Niculcea, P. Ranaweera, and N.-A. Le-Khac, “Security Considerations for Internet of Things: A Survey,” SN Comput. Sci., vol. 1, no. 4, 2020, doi: 10.1007/s42979-020-00201-3.
13. F. Callegati, W. Cerroni, and M. Ramilli, “Man-in-the-middle attack to the HTTPS protocol,” IEEE Security and Privacy, vol. 7, no. 1. 2009, doi: 10.1109/MSP.2009.12.
14. J. Mirkovic and P. Reiher, “A taxonomy of DDoS attack and DDoS defense mechanisms,” Computer Communication Review, vol. 34, no. 2. 2004, doi: 10.1145/997150.997156.
15. M. Feily, A. Shahrestani, and S. Ramadass, “A survey of botnet and botnet detection,” 2009, doi: 10.1109/SECURWARE.2009.48.
16. S. R. Zahra and M. Ahsan Chishti, “RansomWare and internet of things: A new security nightmare,” 2019, doi: 10.1109/CONFLUENCE.2019.8776926.
17. Kaspersky, “What is a honeypot?” https://www.kaspersky.com/resource-center/threats/what-is-a-honeypot (accessed Jun. 11, 2021).
18. H. Jang, B. Kang, K. Cho, K. hee Jang, and S. Park, “Design and Implementation of IoT-based HVAC and Lighting System for Energy Saving,” MATEC Web Conf., vol. 260, 2019, doi: 10.1051/matecconf/201926002012.
19. S. Seyam, “Types of HVAC Systems,” in HVAC System, 2018.
20. E. Al-Masri et al., “Investigating Messaging Protocols for the Internet of Things (IoT),” IEEE Access, vol. 8, 2020, doi: 10.1109/ACCESS.2020.2993363.
21. A. Gloria, F. Cercas, and N. Souto, “Comparison of communication protocols for low cost Internet of Things devices,” 2017, doi: 10.23919/SEEDA-CECNSM.2017.8088226.
22. A. Piltch, “USB 4: Everything We Know, Including Apple Support,” 2021.
23. Pianalytix, “Wired Communication Protocols In IoT,” 2020. https://pianalytix.com/wired-communication-protocols-in-iot/.
24. C. E. Spurgeon and J. Zimmerman, Ethernet: The Definitive Guide (Designing and Managing Local Area Networks). 2014.
25. Yida, “UART vs I2C vs SPI – Communication Protocols and Uses,” 2019. https://www.seeedstudio.com/ (accessed Jul. 21, 2021).
26. S. Al-Sarawi, M. Anbar, K. Alieyan, and M. Alzubaidi, “Internet of Things (IoT) communication protocols: Review,” 2017, doi: 10.1109/ICITECH.2017.8079928.
27. C. Pühringer, “Cloud Computing for Home Automation,” Bachelor’s Thesis Clemens Puhringer Vienna Univ. Technol., 2012.
28. A. Chaudhary, S. K. Peddoju, and K. Kadarla, “Study of Internet-of-Things Messaging Protocols Used for Exchanging Data with External Sources,” 2017, doi: 10.1109/MASS.2017.85.
29. V. Karagiannis, P. Chatzimisios, F. Vazquez-Gallego, and J. Alonso-Zarate, “A Survey on Application Layer Protocols for the Internet of Things,” Trans. IoT Cloud Comput., vol. 3, no. 1, 2015.
30. N. Naik, “Choice of effective messaging protocols for IoT systems: MQTT, CoAP, AMQP and HTTP,” 2017, doi: 10.1109/SysEng.2017.8088251.
31. M. Corporation, “What is cloud computing? A beginner’s guide.”
32. Amazon®, “What is cloud computing?” .
33. S. Vennam, “Cloud Computing,” 2020. https://www.ibm.com/cloud/learn/cloud-computing (accessed Jun. 13, 2021).
34. S. H. and N. V., “A Review on Fog Computing: Architecture, Fog with IoT, Algorithms and Research Challenges,” ICT Express, vol. 7, no. 2, pp. 162–176, Jun. 2021, doi: 10.1016/j.icte.2021.05.004.
35. A. V. Dastjerdi, H. Gupta, R. N. Calheiros, S. K. Ghosh, and R. Buyya, “Fog Computing: Principles, architectures, and applications,” in Internet of Things: Principles and Paradigms, 2016.
36. H. J. Cha, H. K. Yang, and Y. J. Song, “A study on the design of fog computing architecture using sensor networks,” Sensors (Switzerland), vol. 18, no. 11, 2018, doi: 10.3390/s18113633.
37. M. Corporation, “Why Azure for IoT.” https://docs.microsoft.com/en-us/learn/modules/azure-iot-strategy-and-solutions/2-why-azure-for-iot (accessed Jul. 21, 2021).
38. I. Korkmaz, S. K. Metin, A. Gurek, C. Gur, C. Gurakin, and M. Akdeniz, “A cloud based and Android supported scalable home automation system,” Comput. Electr. Eng., vol. 43, 2015, doi: 10.1016/j.compeleceng.2014.11.010.
39. S. Dey, A. Roy, and S. Das, “Home automation using Internet of Thing,” 2016, doi: 10.1109/UEMCON.2016.7777826.
40. R. K. Kodali, V. Jain, S. Bose, and L. Boppana, “IoT based smart security and home automation system,” 2017, doi: 10.1109/CCAA.2016.7813916.
41. K. Mandula, R. Parupalli, C. H. A. S. Murty, E. Magesh, and R. Lunagariya, “Mobile based home automation using Internet of Things(IoT),” 2016, doi: 10.1109/ICCICCT.2015.7475301.
42. N. Dickey, D. Banks, and S. Sukittanon, “Home automation using cloud network and mobile devices,” 2012, doi: 10.1109/SECon.2012.6197003.
43. O. Hamdan, H. Shanableh, I. Zaki, A. R. Al-Ali, and T. Shanableh, “IoT-Based Interactive Dual Mode Smart Home Automation,” 2019, doi: 10.1109/ICCE.2019.8661935.
44. Andrew Shaw, “Arduino Library for Proteus V2.0,” 2021. https://www.theengineeringprojects.com/2021/03/arduino-library-for-proteus-v2.html.
45. James Wilson, “Proteus Libraries of Embedded Sensors,” 2020. https://www.theengineeringprojects.com/2020/07/proteus-libraries-of-embedded-sensors.html (accessed Jul. 23, 2021).
46. H. Ito and KlingOne, “AdobeXD to Unity UI converter,” 2020. https://github.com/itouh2-i0plus/XuidUnity.
47. G. P. Viganò, F. Jasche, and C. Pogolski, “M2MQTT for Unity,” 2019. https://github.com/gpvigano/M2MqttUnity.
48. Deadlyfingers, “Azure Functions for Unity,” 2018. https://github.com/Unity3dAzure/AzureFunctions.
49. S. Verma, “Smart Home Mobile App.” https://www.behance.net/gallery/91128375/Smart-Home-Mobile-App.
50. M. Corporation, “Microsoft Technical documentation.” https://docs.microsoft.com/en-us/documentation/ (accessed Jul. 22, 2021).
51. Ulrichome, “Small And Simple House Design With Two Bedrooms,” 2021. https://ulrichome.com/home-plans/small-simple-house-design-two-bedrooms/ (accessed Jul. 22, 2021).
52. M. Bloluwatife, “FinalYearProject.” https://github.com/hayone1/FinalYearProject (accessed Aug. 18, 2021).
53. M. Bloluwatife, “IoT_SimulatedDevice,” 2021. https://github.com/hayone1/IoT_SimulatedDevice.
54. M. Bloluwatife, “AZUREfUNCTIONS,” 2021. https://github.com/hayone1/AZUREfUNCTIONS.

Appendix

Abbreviations

S/N	Parameter
HVAC	Heating Ventilation and Air Conditioning
IoT	Internet of Things
AWS	Amazon Web Services
RFID	Radio Frequency Identification
M2M	Machine to Machine
IP	Internet Protocol
BLE/LE	Bluetooth Low Energy
HCI	Host Controller Interface
HID	Human Interface Device
MAC	Media Access Control
QoS	Quality of Service
SSID	Service Set Identifier
WLAN	Wireless Local Area Networks
MQTT	Message Queuing Telemetry Transport
SQL	Structured Querying Language
UI	User Interface
GPIO	General-Purpose Input Output
SMTP	Simple Mail Transfer Protocol
Iaas	Infrastructure-as-a-Service
PaaS	Platform-as-a-Service
SaaS	Software-as-a-Service
GCP	Google Cloud Platform