

Low Cost Smart Home Hub

Architectural design and implementation



Presented by:

Ngonidzashe Mombeshora

Prepared for:

Dr. Jane Wyngaard

Dept. of Electrical and Electronics Engineering

University of Cape Town

Submitted to the Department of Electrical Engineering at the University of Cape Town in partial fulfillment of the academic requirements for a Bachelor of Science degree in Mechatronics Engineering.

October 2020

Key words:

IOT; Smart home; Cyber-physical systems;

Declaration

1. I know that plagiarism is wrong. Plagiarism is to use another person's work and pretend that it is one's own.
2. I have used the IEEE convention for citation and referencing. Each contribution to, and quotation in, this report from the work(s) of other people has been attributed, and has been cited and referenced.
3. This report is my own work.
4. I have not allowed, and will not allow, anyone to copy my work with the intention of passing it off as their own work or part thereof.

Ngoni Mombeshora

Signature: ***N.MOMBESHORA***

Date: 11/11/2020

Acknowledgements

I would love to extend my most sincere gratitude to the following people who have been the my corner stone and pillars of strength and support throughout the during of this project as well as my undergraduate studies:

My family, especially my parents for their unwavering support and sacrifices made to get me to and through University.

Dr Jane Wyngaard for her support as my supervisor. You were always keen, quick and available to provide a helping hand. I could not have asked for a better supervisor.

To my **friends**. You made the journey much more bearable and memorable.

Abstract

The use of smart devices has become more prominent in modern times. Its increased usage has brought the rapid development in this industry and powering the rise of the 4th Industrial Revolution. The need to connect human beings to their surroundings and environment be it for medical purposes, agricultural or industrial, IOT has become the most dominant field within this revolution. Within this domain the need to convert a home into a smart home has become quite popular due its convenience. The device to execute this conversion is the "smart home-hub". This device generally enables a user to get data from and interact with the world around them.

The growing demand for such devices has brought about a rapid development and competitiveness within this industry to satisfy market needs and in so doing an immense amount of heterogeneity has been introduced. Infrastructures, architectures and services provided are innovatively evolving by the day.

This report aims to identify and analyse the different options available to implement the Smart home-hub, taking into consideration cost, security, ease of deployment and usage, flexibility and reliability. The report further narrates the process of implementing a Smart home-hub and its associated APIs (Application Programming Interfaces) from start to finish.

Terms of Reference

During the course of this project I am to:

1. Provide a review of the commercial and theoretical smart home ecosystems. Analysing their communication protocols, security features, APIs, ease of usage and deployment, their scalability capability and costs. In essence a review of the full OSI stack.(Open Systems Interconnection)
2. Design a Smart Home Ecosystem (Hub and API) that is to satisfy defined user requirements and specifications and budget.
3. Implement the designed Smart Home Ecosystem(Hub and API).
4. Test the implementation.
5. Validate the designed APIs by implementing at least two edge devices.(A sensor and an actuator)
6. Demonstrate and validate the home hub and API using a demonstrator user interface such as a Web Application.
7. Make recommendations for further investigations
8. Provide full system documentation.

Contents

Declaration	i
Acknowledgements	ii
Abstract	iii
Terms of Reference	iv
List of Figures	viii
List of Tables	x
1 Introduction	1
1.1 Background of the study	1
1.2 Objectives	1
1.3 Purpose of the study	2
1.4 Scope and Limitations	2
1.5 Plan of development	3
1.6 Report outline	3
2 Literature Review	4
2.1 Smart home	4
2.1.1 Purpose and usage	5
2.1.2 User benefits	6
2.1.3 Smart devices/objects	7
2.1.4 The Hub	7
2.1.5 APIs(Application programming interfaces)	7
2.1.6 The Cloud	8
2.1.7 User Interface	8
2.1.8 Conclusion	8
2.2 IoT Architectures	8
2.2.1 Cloud Computing	9
2.2.2 Fog Computing	10
2.2.3 Edge Computing	11
2.3 Communication protocols	11
2.3.1 Performance Comparison	12
2.3.2 Communication across the OSI stack	12
2.4 Data security and privacy	13
2.4.1 Solutions	14

2.5	Overview	15
2.5.1	The Hub	15
2.5.2	Sensors, actuators and appliances	15
2.5.3	API	15
2.5.4	The cloud	17
2.5.5	IoT Architecture	17
2.5.6	Communications	17
2.6	Conclusion to literature review	18
3	Methodology	19
3.1	Stage 1: Research	19
3.2	Stage 3: Detailed Design	20
3.3	Stage 4: Experimental work and testing	20
3.3.1	Testing	21
3.4	Stage 5: Results and Conclusions	22
3.5	Stage 6: Recommendations	22
4	Detailed Design	23
4.1	User requirements (URs)	23
4.2	Functional Requirements (FRs)	25
4.3	Work breakdown structure (WBS)	27
4.4	Architectural design and overview	28
4.4.1	Level 1: Context	28
4.4.2	Level 2: Container	28
4.4.3	Level 3: Component	30
4.4.4	Level 4: Code	32
5	Implementation and practical work	35
5.1	Server-side API	35
5.1.1	Adding users	36
5.1.2	API endpoints	37
5.1.3	Server-side API documentation	37
5.2	Hardware API	39
5.2.1	Communications	39
5.2.2	Setting up a new device: A sensor	40
5.2.3	Setting up a new device: An actuator	41
5.2.4	Hardware API Documentation	43
5.3	User Interface	43
5.4	Testing	44

6	Results and Discussion	45
6.1	Problem Statement	45
6.2	User requirements	46
6.3	Market comparison	46
6.4	Limitations	49
7	Recommendations for future work	50
7.1	Latency concerns	50
7.2	Data compression	50
7.3	PCB design	50
7.4	Operating system	51
7.5	Encryption/Decryption	51
7.6	Data driven performance tests	51
7.7	Penetration testing	52
8	Conclusion	53
	References	54
	Appendix	58
9	Appendix	58
9.1	Plan of development	58
9.2	Literature review	59
9.3	Design	60
9.4	Implementation and experimental work	61

List of Figures

2.1	Simplistic diagram of smart home components [1]	5
2.2	Smart home use cases [2]	5
2.3	General smart home IoT architecture [3]	9
2.4	The Fog Extends the Cloud Closer to the Devices Producing Data [4]	11
2.5	Communication protocols across the Edge, Fog and Cloud platforms [5] . . .	13
3.1	Stages taken to see the project to completion.	19
3.2	Agile prototyping [6]	21
3.3	V-Model [7]	21
3.4	Test Driven Development [8]	21
4.1	Work breakdown structure	27
4.2	Context diagram	28
4.3	Container diagram: Expansion of Smart home ecosystem and its respective component	30
4.4	Component diagram: Expansion of the hardware API and the server side API	31
4.5	UML activity diagram: Showing sequence of activities taking place when a user changes a device status (i)	32
4.6	UML activity diagram: Sending a command from the user interface to an edge device (ii)	33
4.7	Finite state machine. State 00:idle, State 01: User input handler, State 11:Edge device input handler	34
5.1	"Device model" setup and information parameters	35
5.2	User "Parent" is able to control their device using PUT method available to them	36
5.3	API admin interface can be used to add users to the smart home hub and setting their permissions.	37
5.4	Documentation for server-side API	38
5.5	Documentation for server-side API: PUT method example. Documentation allows a user to test the API by sending test HTTP requests, receive and view the API response	38
5.6	MQTTCommunications class	40
5.7	Moisture sensor setup, connected to the ESP3266 WiFi module	41
5.8	Desk Lamp setup, connected to an ESP3266 WiFi module	43
5.9	Screenshot of the user interface set up	44
7.1	Data driven performance testing procedure	52
9.1	Plan of development	58
9.2	Smart home use cases [9]	59
9.3	Trello dashboard for keeping track of progress	60

9.4	An http GET request to the API with incorrect authentication information provided returns an Error 403 Forbidden	61
9.5	An http POST request to the API with correct authentication information provided returns response 201 Created	61
9.6	An http PUT request to the API with correct authentication information provided returns response 200 OK Note the name and status change	61
9.7	An http DELETE request to the API with correct authentication information provided returns response 204 No Content(This is a successful delete)	62
9.8	Example MQTTClass usage- Each edge devices becomes an MQTTClass object to inherit communication capabilities. N.B homehubRPi is the name of API-1 from the design diagrams.	62

Listings

1	homehubRPi API usage	40
2	Example code for setting up an actuator to receive commands from a user	42
3	Example code for moisture sensor edge device	62

List of Tables

I	Requirements for the project	2
II	Security requirements for a secure and trustworthy Smart Home [10] . .	16
III	User requirement 1	23
IV	User requirement 2	23
V	User requirement 3	24
VI	User requirement 4	24
VII	User requirement 5	24
VIII	User requirement 6	24
IX	User requirement 7	25
X	User requirement 8	25
XI	Functional requirement 1	25
XII	Functional requirement 2	25
XIII	Functional requirement 3	26
XIV	Functional requirement 4	26
XV	Functional requirement 5	26
XVI	Functional requirement 6	26
XVII	Functional requirement 7	27
XVIII	Functional requirement 8	27
XIX	Available and exposed API endpoints	37
XX	Market comparisons of smart home systems, both open source and proprietary	48

Nomenclature

API Application Programming Interface

IoT Internet of Things

M2M Machine to machine

MVP Minimal Viable Product

MVP Most Viable Product

OSI Open Systems Interconnection

1 Introduction

”When wireless is perfectly applied, the whole earth will be converted into a huge brain, which in fact it is, all things being particles of a real and rhythmic whole. We shall be able to communicate with one another instantly, irrespective of distance. Not only this, but through television and telephony we shall see and hear one another as perfectly as though we were face to face, despite intervening distances of thousands of miles; and the instruments through which we shall be able to do this will be amazingly simple compared with our present telephone. A man will be able to carry one in his vest pocket.”

Nikola Tesla 1926

1.1 Background of the study

With IoT, Nikola Teslas’ description and prediction of the mobile phone in 1926 can be extended beyond simple human communication. Not only are humans able to communicate with each other instantly but they are able to communicate with the environment around them anywhere and anytime. They can communicate with and effect action on the environment at the simple push of a button or tap of a screen or by speaking a few words. The rise of IoT and smart devices has brought about several issues such as the demand for privacy being of important concern. Other concerns arising from consumers are that of data security, data ownership, ease of operation and reliability.

This project aims to address the above concerns using engineering methodologies by researching the currently available technologies, implementing a Smart Home Ecosystem by making use of designed APIs, testing and performing demonstrations using a demonstrator human interface. The resulting Smart Home Ecosystem should meet the specific user requirements and functional requirements set out in 4.1 and 4.2 respectively.

1.2 Objectives

The objective of this study is to critically analyse the available IoT technologies for use in Smart Home applications, including hardware and software communication stacks, available system architectures including edge, fog and cloud computing; and how the use of APIs can provide flexibility, scalability, ease of use and many other advantages to the challenges of building secure human and machine interfaces.

From the Terms of reference a summary of the main project deliverable are outlined in Table I. These requirements are further broken down expanded into user requirements and functional requirements in the Design section. Section, 4.

Table I: Requirements for the project

Requirement Number	Description
R01	Review of existing technologies
R02	Design and implementation of the Smart home hub and API
R03	Full system documentation

1.3 Purpose of the study

The purpose of the study is to propose a feasible IoT platform solution that aims at minimising the current problems being faced by customers and developers. These problems include flexibility, ease of use and deployment, security and cost as stated by the first statement in the terms of reference. The key goal is to provide easily extensible APIs. One of the major challenges for consumers is that commercial options are closed source and operate within their walled gardens making it difficult for consumers to mix and match different components. Companies do this to simplify their security architecture but at the cost of that a customer cannot use brand X with sensor Y or brand Y with sensors from X.

The proposal of this platform will come about by researching the current existing solutions and developments made thus far and then designing a suitable solution, implementing and creating an MVP (Minimal Viable Product) as proof of concept.

1.4 Scope and Limitations

Although the projects aims to propose and implement an IoT solution for a Smart Home Hub. An MVP is to be created so as to test the functionality of the system as a whole. The main deliverables of this project and hence scope are limited to the demonstrator hub and the APIs. These will then be used implement the MVP.

A budget capped at R1500 may be used to obtain components that are not already owned by the University. This is a tight budget meaning every component used should be easily available and the final product to be delivered is relatively cheaper than competitive Smart Home Hubs already available on the market. A time constraint of 13 weeks has also been set meaning that the number of tasks that can be performed is also limited.

This project will likely not implement any front-end frameworks although these will be taken into consideration during API design and a simple or mock-up one will be used to provide a proof of concept and for testing purposes. regarding interaction with the server-side API. The goal is to produce a **low cost** smart home hub, therefore constrained devices ¹ should

¹Constrained devices: Small devices with limited CPU, memory, and power resources these are generally cheaper and therefore more cost effective. Ideal for the limited budget

be considered.

1.5 Plan of development

Figure 9.1 in the Appendix shows the details of tasks to be performed over the duration of the project as a timeline. The first four weeks constitute of researching, understanding the problem and conveying a theory and literature survey. Afterwards a methodology and solution design is to be done spanning a period of two weeks. Four weeks have been allocated after this for practical work and implementation, this should be concluded by the end of week ten. The remaining three weeks will be used to conclude the project by writing up results, documentation and recording videos. Report writing is to be performed at every step throughout the project duration. This plan of development is not set in stone and can be reviewed and altered during the project.

1.6 Report outline

The remainder of this report is structured as follows. Section 2 reviews available academic and consumer literature, showing a narrative of the current existing solution along with their advantages and disadvantages. Section 3 outlines the engineering techniques to be used in tackling the project. Section 5 shows and describes the practical work done whilst implementing the design proposed in Section 4. This is followed by Section 6 which discusses the results of the experimental work in relation to the set objectives. Lastly Section 7 will provide details on carrying the work forward and suggestions for further development. The report will be concluded in Section 8.

2 Literature Review

This section discusses the existing commercial and research smart home ecosystems. It goes on to analyse the overall architectures in use along with their pros and cons. A detailed and low level technical review is performed, by delving into their hardware usage, the range of wireless communications that are available and are in use, their sensors and actuator choices, the APIs available and their respective HIDs (Human interface device). This is a detailed thematic review of the literature relevant to the problem. A Conclusions subsection follows, summarising the aforementioned and providing details on reasons as to what factors will influence the final project design to be made.

2.1 Smart home

The definition of a smart home is largely dependant upon its main use case. Some definitions that provide insight into what a smart home might be are as follows: *“a residence equipped with computing and information technology, which anticipates and responds to the needs of the occupants, working to promote their comfort, convenience, security and entertainment through the management of technology within the home and connections to the world beyond”*. [11] This definition by Aldrich emphasizes the idea of equipping a regular home with computing and automation capabilities to aid the occupants with their different needs. A users needs can range from comfort and convenience to security and to the provision of services such as health monitoring.

Balta-Ozkan et al. defined a smart home from a more technical point of view: *“a smart home is a residence equipped with a high-tech network, linking sensors and domestic devices, appliances, and features that can be remotely monitored, accessed or controlled, and provide services that respond to the needs of its inhabitants”*. [12] This definition brings out some important characteristics of a smart home, firstly the idea of sensors and actuators, which can be remotely controlled, accessed and monitored. A more general definition of a smart home would be a home equipped with an automated or automatable intelligence system that would allow a user to interact and communicate with the environment around them. Essentially an IoT system (Internet of things) where in this case things refer to sensor devices or actuator devices. The idea of an IoT ecosystem in terms of its architecture is further explored in Section 2.2. The components required to make a smart home begin with the “smart object” or “things”. These are used for sensing and actuating.

The figure 2.1 shows simplistic representation of the main components that make up a smart home ecosystem.

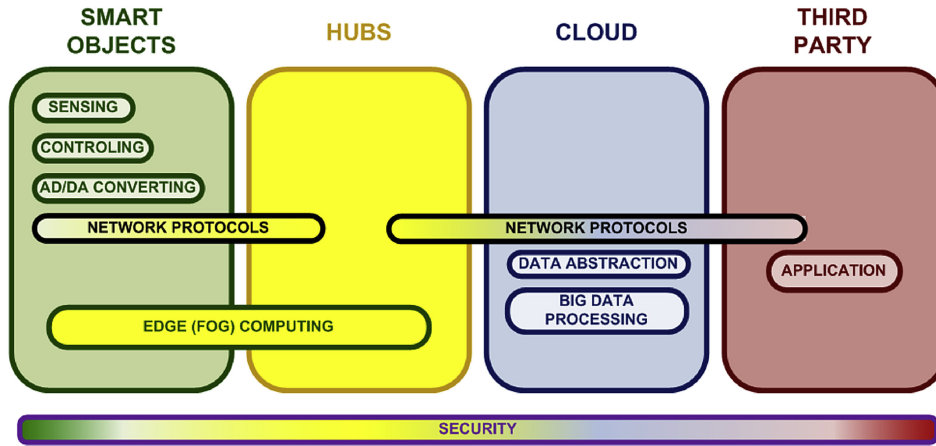


Figure 2.1: Simplistic diagram of smart home components [1]

2.1.1 Purpose and usage

As explored in the definitions above in 2.1 it is possible for a smart home to serve several different purposes depending on what the user requires. Its usage can range from sensing temperature, humidity or light intensity readings within their home, what one decides to do with this data is up to them, one might intend on controlling these readings hence not just monitoring but actuating as well. M.R Alam et al. [2] categorised smart homes according to their different use cases. Figure 2.2 This ranges from comfort which includes remote access and control of smart devices. Health care which could be local and remote monitoring of say an elderly patient or an infant. Security is also another use case, one may intend on being able to remotely access cameras in their homes, or maybe implement restricted access to specific devices, appliances as well rooms.

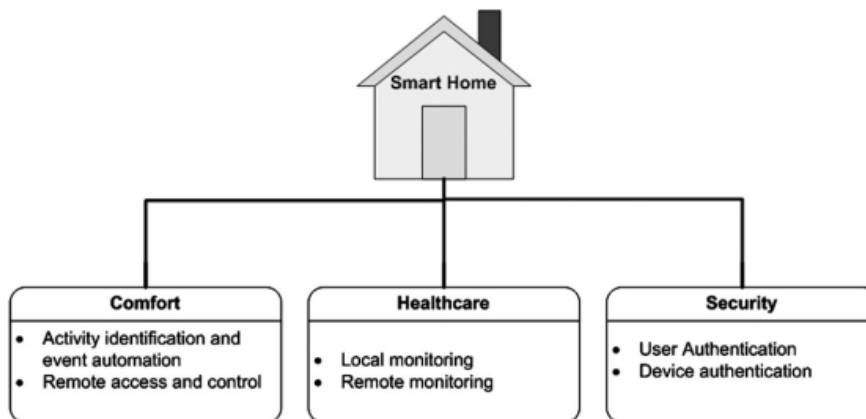


Figure 2.2: Smart home use cases [2]

Orwat et al. further emphasizes the idea that smart home ecosystems may be used for a variety of purposes, he stated that fully-automated devices have the potential to improve the quality of life and encourage the independent living of residents, especially for an ageing population through constant health management, and they may even provide virtual medical assistance in cases of need. [13] It empowers users to remotely control household appliances and decrease the burden of everyday household activities. [14]

2.1.2 User benefits

Marikyan.D [9], Table 9.2 in the Appendix summarises some of the benefits of a smart home in relation to the services offered, immediate advantage and the long term impact. Marikyan.D partitioned smart home benefits according to four main categories. These are health related benefits, environmental benefits, financial benefits, and psychological well-being and social inclusion. All these are use cases in which smart home technologies are increasingly providing value to users.

Some users who can benefit from this technology are:

- People living alone who are unable to seek help in emergencies(unconsciousness, falls, strokes, myocardial infarction, etc.) [14]
- People who want to maximize their home security, and restricting access to appliances.
- Elderly or disabled people who suffer from cognitive (Alzheimer disease, dementia, etc.) and/or physical (visual, hearing, mobility,speech, etc.) impairment. [14]
- People who wish to remotely control of home functions and appliances.
- People who need help in daily life to perform personal care activities (eating, toileting, getting dressed, bathing, etc.) and instrumental activities (cooking healthy meals, dealing with medication, and doing laundry) [14]
- People who desire some home management insights such monitoring energy usage by appliances.
- Informal (family, friends, neighbor people) or formal (care provider) caregivers for the elderly or the handicapped.
- People living in rural and remote communities or in urban communities with inadequate health service provision. [15]
- People who suffer from chronic disease, and who need continuous monitoring (diabetes, cancer, cardiovascular disease, asthma,COPD, etc.) [14]
- People involved in tele-health care undertaking health care at a distance or tele-medicine,with physicians practising ‘virtual visits’ [16]

2.1.3 Smart devices/objects

Also referred to as "Things" in the term "Internet of Things", are networked devices² capable of sensing, actuating, communicating and if powerful enough, processing the retrieved data. They should be able to perform A/D and D/A conversions³ acting as a bridge between the physical space and the digital domain. [17]

2.1.4 The Hub

The Hub is responsible for collecting and/or processing raw data from the smart-objects [1] and acting as a middle man between the smart objects and the cloud as well as providing a communication channel between the user and the smart objects. [18] These smart objects are also referred to in literature as "edge devices". Communication protocols that can be utilised by the hub are explored in subsection 2.3. Ideally the hub should process this raw data into useful information to be forwarded to the cloud, this reduces the the amount of data to transport. [19] The hub is one of the main deliverables for this project alongside the APIs to be implemented. The hub comes with some electrical components attached to it. These can include some communication modules such as Bluetooth, Z-wave, Zigbee and WiFi. A microprocessor also comes attached to carry out all the operations required by the hub, specifications of these microprocessors would largely be influenced by the use case and the tasks required for the hub. These can range from being a simple communication conduit to implementing AI⁴ and machine learning on the edge which would require more computing power. Not all IOT architectures require a hub but most smart home solutions do because they may use the communications mentioned above and not just WiFi. Smart home architectures are discussed in Sub-section 2.2.

2.1.5 APIs(Application programming interfaces)

This can be considered to be a part of the cloud with reference to Figure 2.1 although APIs can be implemented within the hub as well. Such decisions pertain to the chosen architecture used, explored in 2.2. APIs are the bridge between the smart devices and hub to the user via the internet. They implement a communications channel that would otherwise be impossible or difficult to construct. For example a smart device equipped with a WiFi module can directly send data to the internet, but a smart device that can communicate via only Bluetooth would not be able to forward it data to the internet. This is where the API comes in, and one its main advantages becomes apparent. It can introduce heterogeneity and inter-operability within the IoT ecosystem by providing an abstraction layer between the hub and smart devices and between the hub and the internet. APIs are a means of providing

²Networked device are devices that are capable of connecting to and operate within a network

³Analogue to digital and digital to analogue conversion

⁴Artificial Intelligence

automation, flexibility and platform independence. Some examples of home automation APIs include GetGrowingAPI [20] made for smart gardens, MinutAPI [21] and Lightwave Smart Series API [22] just to name a few. API communication protocols are further explored in Sub-section 2.3.

2.1.6 The Cloud

All data from different sources is accumulated, stored and retrieved to and from the cloud (household data, sensor readings, sensor and actuator states etc.). [23] The cloud should provide massive data storage and processing infrastructure. [24] Gubbi et al (2003) [25] emphasizes the importance of the cloud by stating that it promises high reliability, autonomy and scalability for the next generation of IoT applications. And now seven years later, this statement cannot be considered to be more true with services such as Amazons' AWS [26] and Microsoft Azure [27] just to name a few.

2.1.7 User Interface

The third party/ application layer is the HID (Human interface device). This ranges from a web-application to a mobile application or both of them working in tandem. This is where information is delivered to the end users (this could be in the form of notifications, device control, charts, graphs or dash boards) [24] This is particularly useful, since customers will be able to gain insights about the different appliances in their home, and enabling the users to control them intuitively. This layer is out of the scope of this report, although it is taken into account when designing the cloud layer/ API. The application layer should be able to easily receive requested information from the smart devices by interacting with the API, provide the user with a view of this information, and it should also allow a user to send commands to the smart devices.

2.1.8 Conclusion

As discussed, the described benefits of smart home are extensive, these benefits can be considered as justification to the growing popularity of the smart home over the past recent years. The smart home ecosystem technology will continue to grow in demand and complexity, due to increasing user needs. The reasons as to why research in this area now start to become apparent. The following sections will deal with the more technical aspects of the smart home system.

2.2 IoT Architectures

This sub-section explores the different IoT Smart home ecosystem architectures that are currently being employed in the industry and in research. It will draw to light the challenges

faced when designing such architectures and the advantages and drawbacks of each employed architectural design.

The architecture of a smart home is largely influenced by the way which devices communicate with one another, where the information from edge devices is stored, how this information is processed for decisions to be made and how the user might interact with edge devices and vice versa. [3] Several IoT architectures have been proposed and investigated in previous studies. The most general IoT architecture in use is the cloud-based architecture shown in Figure 2.3. Most architectures in use are a build-on from this general "cloud-centric" concept.

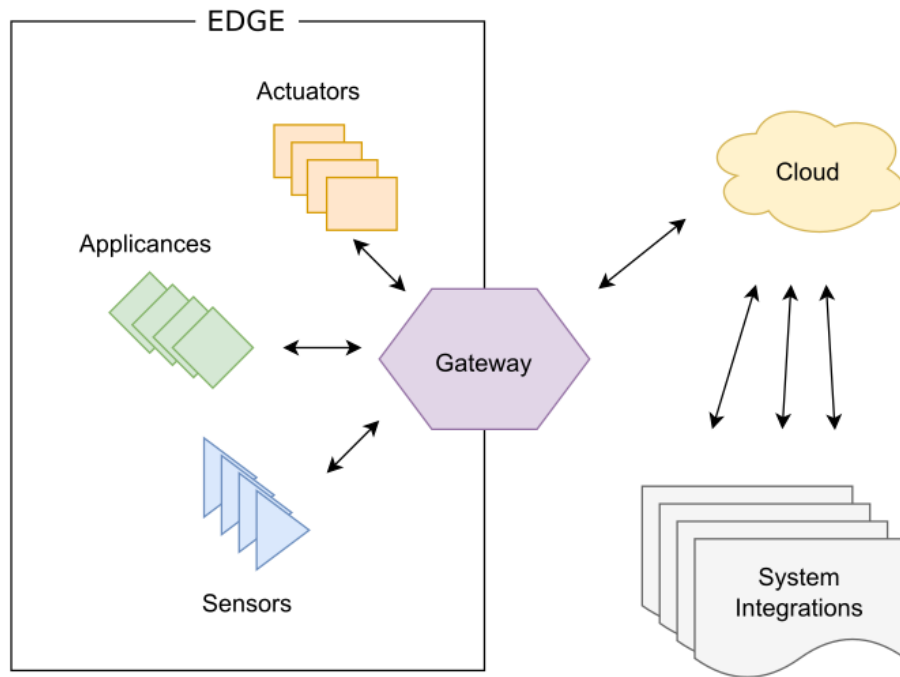


Figure 2.3: General smart home IoT architecture [3]

2.2.1 Cloud Computing

The National Institute of Standards and Technology (NIST) defines cloud computing as “a model for enabling ubiquitous, convenient, on-demand network access to a shared pool of configurable computing resources (e.g., networks, servers, storage, applications, and services) that can be rapidly provisioned and released with minimal management effort or service provider interaction”. [28] Within the Smart home ecosystem the purpose of the cloud is to cluster all data from edge devices, provide storage and processing power.

The cloud essentially provides means to increase availability, reliability and security, while at the same time taking advantage of higher computation power and scalable architecture. The cloud can integrate with many third party services, such as data visualization, smart

home device management, or user access and role management. [3] Due to its high reliability, scalability and autonomy, cloud computing has gained popularity over the years, it has a guaranteed long term support. [25] And hence it would be a plausible choice to use such an approach in designing a Smart home ecosystem.

In recent years some drawbacks of cloud computing have started appearing. One of concern is the fact that although CPU processing power and memory have increased over the years, bandwidth for memory and data transmission has not been able to keep up to this trend. It has become a bottleneck, an inhibitor for the full exploitation of the cloud in terms of its performance. [3, 29] It has become problematic to be able to send the numerous magnitudes of raw data obtained from edge devices to the cloud for processing without trading of the systems latency overheads. (larger data sets require more time to transmit). This is a huge price to pay for especially for critical applications such as a smart home. When a user requests information, they desire it immediately. Fortunately, the more recent, fog computing (explored in sub-section 2.2.2) aims at mitigating this problem.

2.2.2 Fog Computing

This is a more recent paradigm of computing first coined by Cisco in 2014. [4] as an extension to the general cloud computing model. As mentioned above it aims to overcome the IoT-cloud bottleneck of limited bandwidth. Fog computing does this by bringing the cloud closer to the edge devices, (Figure 2.4) thus bringing in the advantages of cloud computing along with it. The advantages including data storage and processing capabilities. This way the need to send numerous amounts of raw data is no more. Instead raw data from the edge is pre-processed in the Fog layer and either metadata/processed data is then forwarded to the cloud. This greatly reduces the bandwidth required for data transmission. With reference to Figure: 2.3 fog computing could be realised by the gateway, and with reference to this research project the gateway would represent the Home **hub**.

The main objectives of fog computing are: [30]

- Reducing the amount of data sent to the cloud leading to ...
- a decrease in network latency and hence ...
- Improving system response time whilst overcoming the low bandwidth concern.

A major drawback of Fog computing, is that faced with many emerging technologies, and that is privacy and security concerns. [3] Adding a new layer between the IoT/edge layer introduces several vulnerability points of exploitation especially in the communications domain. [31, 32] This reasoning is explored in sub-section 2.3 where communication protocols alongside their security concerns are explored.

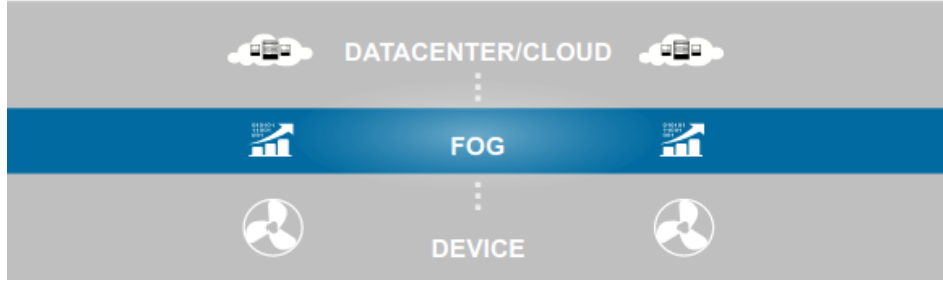


Figure 2.4: The Fog Extends the Cloud Closer to the Devices Producing Data [4]

2.2.3 Edge Computing

Edge computing pushes the intelligence, processing power, and communication capabilities of an edge gateway or appliance directly into edge devices. [30] (The edge computing domain is shown in Figure 2.3) These edge devices could include sensors, actuator or appliances and realising edge computing can be done via several ways, an example would be by attaching so Ble Module with some processing power and memory. The edge device is now capable not only of sensing, but processing, making decisions, and communicating to and from the fog layer or in this case: the home hub as well. The resulting system is more decentralized further decreasing the risk of single-point-of-failure. [3]

Similar to the Fog computing paradigm multiple case studies analysed by Shi et al. [33] and [34] have shown that there are open issues regarding privacy and security in the edge computing paradigm. [3]

2.3 Communication protocols

Despite the wide spread usage of the HTTP communication protocol, the currently used protocols in various domains of IoT, fog and cloud domains are de-facto fragmented with many different solutions. [5] A reason for this is that there are many different requirements that need to be met in the different layers of the IoT ecosystem. For example, in search of achieving heterogeneity in the Edge layer with regards to the communication protocols of edge devices, issues such as security, latency and inter-operability need to be taken into account. Add the Fog and Cloud computing layers and the task of dealing with such issues becomes of utmost concern.

As a result no single messaging/communication protocol will be enough to cover the entire communication on the combined IoT-F2C architecture built by bringing together IoT, fog and cloud systems. [5]The aim of this section to achieve a seamless inter-operable, integrated and coordinated communication scheme suitable for the smart home ecosystem.

Recognizing the fact that one single messaging protocol will not be enough to cover the entire

communication on the combined IoT-F2C architecture built by bringing together IoT, fog and cloud systems, the goal of this section is to unveil open issues and challenges towards a seamless inter-operable, coordinated and integrated smart home ecosystem.

Of the several communication protocols that literature has brought to the forefront, which allow for machine-to-machine communication (listed below) all of them follow one of two main themes, which are Publish/Subscribe and Request/Reply. The main difference between the two is that the former is asynchronous in nature and the latter is not, communication only occurs at known times, i.e there is no reply to where no request has been made. Communication protocols of particular interest are shown in Figure 2.5.

2.3.1 Performance Comparison

[5] researched the above communication protocols and carried out a performance comparison in terms of (1) Security (2) Energy consumption (3) Bandwidth consumption and throughput and (4) Latency. All these are key areas of concern when making decisions for an IoT-F2C architectural designs. The results of his research were that, the most prominent developer choices were MQTT and HTTP. These are currently the most widely used and adopted. The reason for this is that MQTT and HTTP REST are currently comparably more mature and more stable IoT standards than other protocols. For many IoT developers, MQTT and HTTP are protocols of choice in their IoT, fog and cloud implementations. [5]. It should be noted that no communication protocol has been originally designed for a combined IoT-fog-cloud systems, and there is no unifying standard, and this alone is an ongoing area of research and development. [5]

2.3.2 Communication across the OSI stack

Figure 2.5 [5] shows the communication protocols listed above and their most suitable positioning within the OSI stack. It can be seen that some protocols are more suitable than others given a specific area of the stack. For example MQTT and CoAP are more suited for communication across the IoT/edge layer to the fog layer. This is due to their much lower bandwidth requirements and available QoS(Quality of Service) options and the Request/Reply model of communication that they use which is more suitable in the edge environment as it would be ideal to autonomously receive updates from the edge devices. Figure 2.5 also shows that some protocols such as REST HTTP and DDS can carry out communications across the whole OSI stack, which could lead to simpler architectural design, but this would not provide the best solution, for example in the Smart Home System, HTTP would not be ideal for the IoT/Edge layer due to its higher payloads and power consumption which are not suitable for constrained devices.

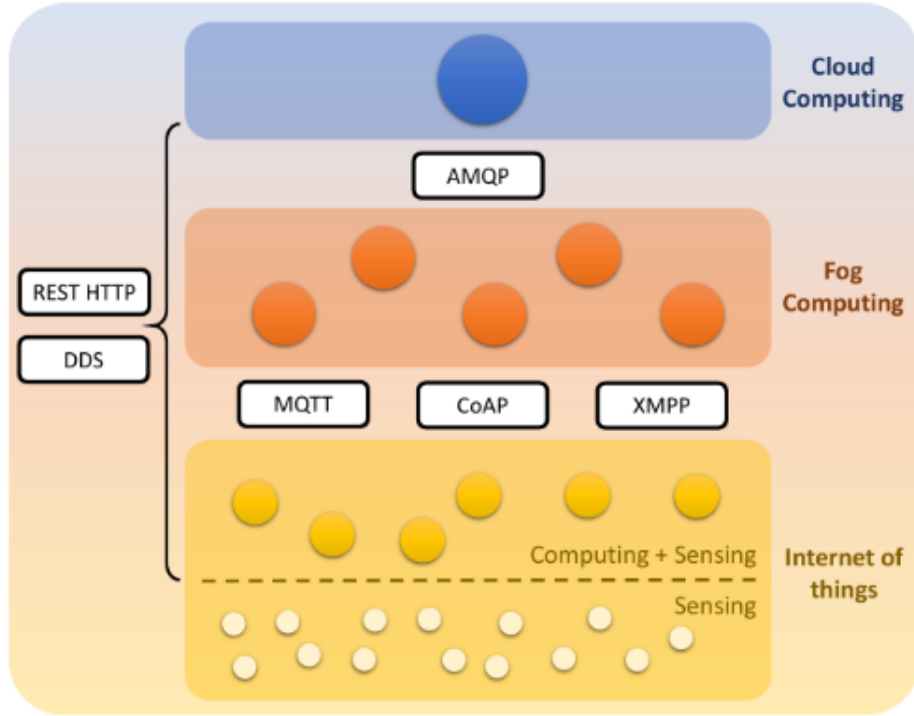


Figure 2.5: Communication protocols across the Edge, Fog and Cloud platforms [5]

2.4 Data security and privacy

The concept of smart home would have not been possible without pervasive computing and multitude of sensors scattered around a house. Unfortunately, the use of these devices, which are usually connected to the Internet (directly or indirectly) and/or use wireless communication, opens up new opportunities for attacks to the security and privacy of the people living in the smart home. [3] This sub-section brings to light some of the security and data privacy concerns when designing a IoT ecosystem as these need to be taken to account when designing the smart home system.

As defined by [35] privacy is the ability of an edge device to protect the personal data of a user. Personal data is considered protected if the user can have the power of deciding where data should be processed. If the data is processed locally the chances of this data being intercepted are close to zero [36], but incorporating the cloud within the ecosystem, personal user information is more vulnerable as it has to be transported to a remote location.

Generally a system is vulnerable to two types of threats, these are internal and external. [3]. Internal threats are possible when the cyber-criminal is with the physical range of the home/local network. Whilst external breaches are possible via an internet connect. This means introducing the cloud and fog computing paradigms increases the risk of data breaches. Zheng et al. [37] summarised the main issues that security deals with:

- Avoiding data breaches (making sure that unauthorized entities cannot access the data);
- Authorization (defining entities that have access to the data)
- Ensuring the privacy of the user

The list of threats that a cyber-criminal can pose on a smart home is inexhaustible but to name a few of the most common ones:

- Eavesdropping: if the cyber-criminal gains access to the victims router, they can intercept traffic coming to and out of the home, they may also be able to use special hardware that can intercept data coming from sensors (which usually use wireless communications.) This attack is passive, but can lead to a physical after for example the cyber-criminal obtains on the habitants' habit. [37]
- Impersonation: Similar to eavesdropping, this is when the cyber-criminal after obtaining the victims credentials tries to act on his/her behalf. [37]
- Dos (Denial of service) :This is when a cyber-criminal hampers the normal operation of sensors/routers, by sending numerous requests at once to the device, or send corrupted messages that the device cannot process and therefore ends up crashing; This prevents the victim to gain access to their home through the internet.
- Software exploitation: Mostly occurs due to the negligence of smart home owners not taking the necessary security precautions when setting up their devices, for example changing the passwords from their default password.
- Ransomware :Cyber-criminals gain access to a victim's device, encrypt the information stored on the drives with a secret key, and then ask for ransom to provide the secret key for decrypting the information. [37]

2.4.1 Solutions

The CIA triad model [38] has been proposed to ensure the privacy and security of a home system. CIA stands for Confidentiality, Integrity and Availability. Confidentiality and Integrity ensure data privacy guarantees, while the availability assure that an edge node is available to share its resources when required. [35]

One of the main issues in the fog and edge computing layers is that of authentication. In a dynamic IoT system, where edge devices can join and leave the network without any restrictions, a mechanism that ensures authentication must be implemented. [35] The same applies to the cloud layer. No user should be able to gain access to edge devices, or cloud APIs without authentication. Puthal D. et al. [39] proposed a solution to securely authenticate edge devices. They assume the cloud is always deployed in the secure environment, so the cloud should initiate the authentication process. The CIA model was further expanded

by [10](Table:II) showing the necessary security requirements to provide and secure and trustworthy home service. These requirements are to be carried forward when implementing the final Smart Home Hub and API for this project.

2.5 Overview

Upon reviewing the literature pertinent to the problem statement for this project, this Sub-section provides finer details of the technologies and decisions made for going forward with the project. That includes details of the hardware choices made, API choices made, User interfaces, the IoT architecture to be implemented and finally the communication protocols chosen.

2.5.1 The Hub

Due to project hardware constraints and project time limit, it would not be possible to design the hardware implementation of a smart home hub. This would include, choosing an appropriate micro-controller and adding required hardware component to it. Designing the PCB (Printed circuit board) and manufacturing it. Instead a Raspberry Pi 3 Model B+ will be used to imitate the Home Hub as this is already available. It provides all the required functionalities for the operations that a Home Hub would be required to perform.

2.5.2 Sensors, actuators and appliances

Simple sensors and or actuators attached to processing/comms modules will be used. For example a humidity sensor can be attached to an Arduino/Wifi-Module such as the ESP8266. This provides the sensor/actuator with a means of communication and limited processing capabilities. A communication channel between the Edge layer(sensors/actuator) and the Fog layer (Hub/Raspberry Pi) can be realised.

2.5.3 API

An API is going to be designed and created from scratch because currently there are no suitable APIs for the IoT architecture to be implemented. The API design should enable a user to request sensor information, or to send requests to the edge devices via the fog layer/Hub. The API should be a conduit between the user and the Edge and Fog layers, essentially the Smart Home ecosystem as a whole. A RESTful API is to be implemented. The reason for this is that, REST (Representational State Transfer) - can be used over nearly any protocol, when used for web APIs it typically takes advantage of HTTP. [40] It is widely used and provides long term support. It can be easily adopted and further improved by developers or even clients.

Table II: Security requirements for a secure and trustworthy Smart Home [10]

Category	Security Requirements
Confidentiality	User's privacy data delivered during the smart home devices inter-communication, and the key information used for the encryption algorithm should be managed securely to prevent potential exposure to the outside.
	In case sending the data generated from the smart home device into another device, the transferring data should be converted into cipher text form.
	To prevent the replication and modification by an outsider, Identification information of a smart home device should manage securely.
	Smart home devices should provide a highly secure password setting function and periodic password change functionality.
	Home Hub must strengthen security via a robust and complex password setting
Integrity	To maintain the reliability and safety of the smart home device, unauthorized device or user access should not be allowed.
	User's privacy data delivered during the smart home devices inter-communication, and the key information used for the encryption algorithm should not be forged or tampered.
	In case sending the data generated from the smart home device into the outside or another device, data integrity should be provided.
	Through the mutual authentication between devices constituting a smart home service, reliable communication environment must be configured.
Availability	To respond to security threats such as cyber-attacks and hacking, external attack detection capabilities must be equipped.
	The security features for software update of smart home device must be provided.
	Device security policy settings features that reflect a variety of device characteristics and specifications must be considered.
	In order to grasp a physical status (e.g. theft, loss, add, disposal) of smart home devices correctly, device management system should be provided.
	Periodic status monitoring of a smart home device, and unnecessary remote access blocking should be provided. In addition, if abnormal operation is generated from smart home devices, appropriate response and event history about abnormal behavior should be accompanied.

2.5.4 The cloud

The cloud layer would be hosting the API, and a remote server/ Virtual Machine provided by the University is going to be used to realise the cloud layer. This cloud layer shall be de-centralised by providing the Hub(Raspberry Pi) with some processing capabilities and API access.

2.5.5 IoT Architecture

The most important goals of this project is to realise a **low cost** Smart Home solution and hence the architecture chosen should be able to realise this goal. That means using the cheapest available resources in the most efficient way, such that they could still compete or be even more superior to what the market already offers. Due to the concerns mentioned in sub-section: 2.3.1 with regards to Energy consumption, Bandwidth consumption and latency, an IoT architecture must be proposed to minimise these concerns. It becomes imperative that the cloud is to be decentralized, introduce the Fog and computing layers mentioned in section:2.2. This in turn reduces the bandwidth requirements as data is "trimmed" before being transferred to the cloud for final processing and storage. Energy consumption is also reduced as less power will be used for communications. Respectively latency will be reduced as well due to the fact that the amount of data moving is reduced. An IoT-F2C⁵ architecture is the initial proposal made to realize this goal. Further details are provided in the Design section 4.

2.5.6 Communications

M2M (machine to machine) For machine to machine communications (That is edge devices to hub) the MQTT protocol is going to be used. MQTT is the most energy efficient protocol out of the protocols mentioned in subsection 2.3. On top of being energy efficient it also provides security since it runs on top of the TCP protocol⁶, which ensures reliability. [5]It is also light weight and has proved to be the most prominent solution for constrained environments. [5]

API communications RESTful HTTP is going to be used for API communication for reasons stated in 2.5.3. The smart hub running on a raspberry pi would be able to call an HTTP client enabling to it to use the HTTP protocol to communicate to and fro with the the cloud server.

⁵Edge to Fog to Cloud architecture

⁶TCP (Transmission Control Protocol) is used for organizing data in a way that ensures the secure transmission between the server and client. It guarantees the integrity of data sent over the network, regardless of the amount.

2.6 Conclusion to literature review

This section has covered the literature relevant to the project being undertaken. It began by putting into context what a smart home ecosystem is, what it does and what it is capable of doing. Covering all the components and technologies that bring about a smart home. The section then went on into a more technical analysis of the smart home, covering relevant literature, and research. Showing the numerous possible routes that can be taken to implement such an ecosystem. Lastly the overview section outlined the decisions made following the aforementioned research and provided suggestions on possible solutions to realise the goal of this project, which is to create a low cost smart home hub.

3 Methodology

This section discusses and outlines the methodology and approach taken to see this project to completion from inception. Figure 3.1 shows the stages of the approach taken. The stages are further explained below.

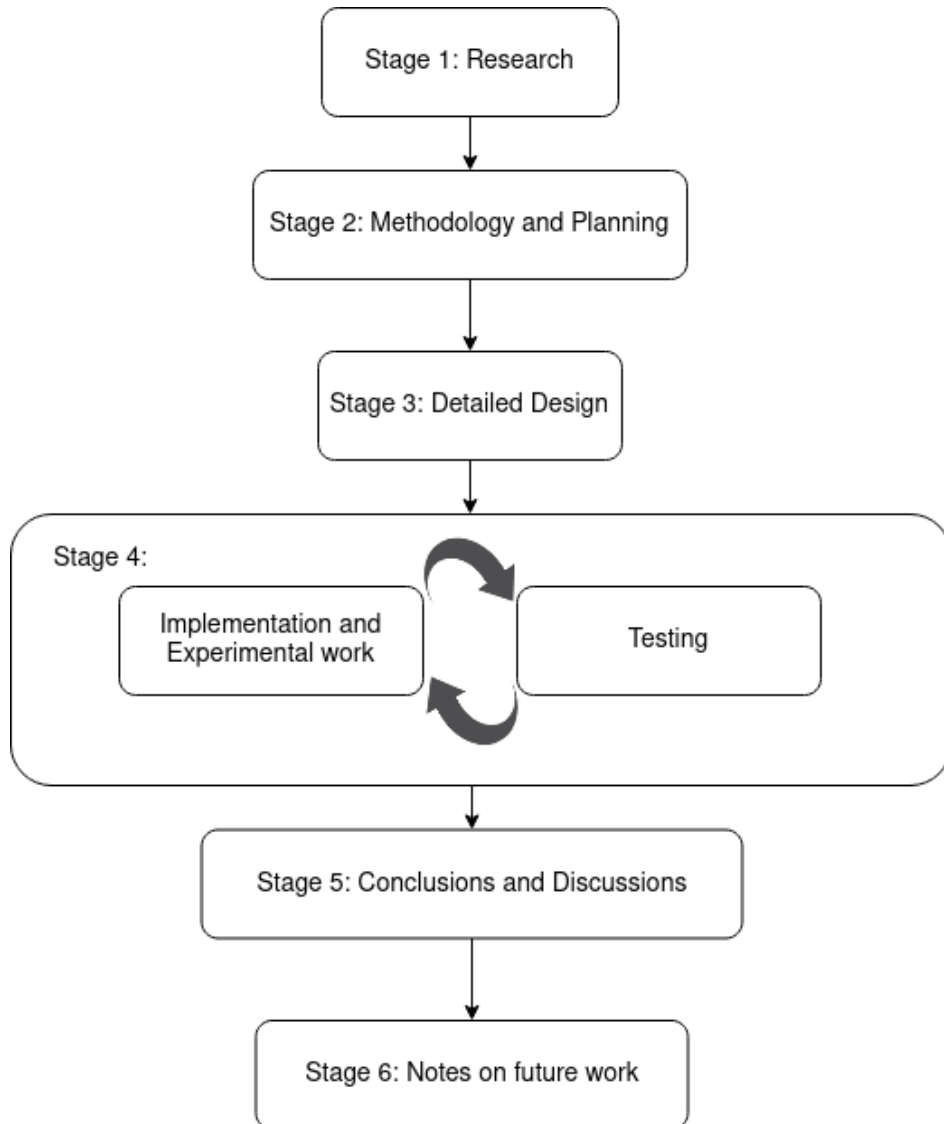


Figure 3.1: Stages taken to see the project to completion.

3.1 Stage 1: Research

To be able to design and implement a low cost smart home hub, adequate research needs to be carried out. This would provide insight into the technological progress made in the field of IOT thus far. Such insights would then enable decision making on the design and implementation

to be done in the following chapters. Research on smart home ecosystems has been carried out, including the associated hardware domain and software domain. This research covered the technical aspects pertinent to the project, this includes IOT architectures, communication modes and protocols as well as security and privacy concerns. All this has been included in the literature review, Section 2.

3.2 Stage 3: Detailed Design

Section 4 will cover the technical design aspects of the project which are to be implemented practically. The design decision made in this section are to be implemented in Stage 4, the experimental work. Design includes both the hardware design and software design. A work breakdown structure is going to show all the practical blocks of work to be implemented so as so finally realise the complete product.

3.3 Stage 4: Experimental work and testing

As stated in the literature review, the IOT industry is immensely dynamic, growing and changing rapidly. Technological advancements are being made at a fast pace and smart home systems are changing just as much as the technology is improving with time. For this reason, the most suitable engineering methodology to run the experimental work is the Agile methodology, Figure 3.2. Agile methodology is a type of project management process, mainly used for software development, where demands and solutions evolve through the collaborative effort of developers and their customers. [41]. The agile methodology was created as a response to the inadequacies of traditional development methods such as the Waterfall method. For a highly competitive market due to the fact that technology is improving and changing rapidly, continuous improvement during the product implementation is required, "to stay on top of the game" This avoids creating a final product that is out-dated and no longer desirable on the market.

The V-model (Verification and Validation), Figure 3.3 is to be adopted as well. The reason for this is that it provides a means of testing at each stage. While the agile prototyping methodology involves testing the V-model clearly links specifications to the final output. It places emphasis on testing and verification of smaller, complete tasks before incorporation into a larger system, which assists with risk mitigation.

The backlog will constitute of the tasks defined in the WBS (Work breakdown structure) detailed in section 4. After the implementation of each task, verification and validation is then performed to ensure the final product being produced to the agile prototyping still meets specifications.

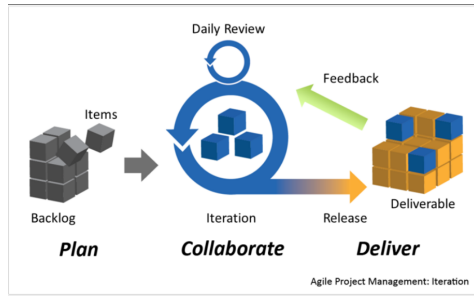


Figure 3.2: Agile prototyping [6]

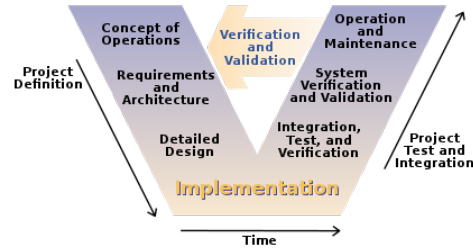


Figure 3.3: V-Model [7]

3.3.1 Testing

As stated above, testing is to be done in an iterative manner each time a task from the work breakdown structure is implemented. Verification and validation is to be performed every time a task is to be added onto the final product. A TDD (Test Driven Development) approach is to be adopted for testing. The test based approach allows for better program design and higher code quality as well as reducing development time. According to the IEEE Software publication [42], implementation of Test Driven Development reduces the percentage of bugs by 40 - 80 percent, which consequently means that less time is required for fixing them. This approach proves to be most desirable for a such project that has a time constraint. Figure 3.4 shows the TDD flow to be implemented throughout Stage 4 of the project. If time permits more testing will be done on the finished product. Such tests include "data driven performance tests" and "penetration testing". These tests will ensure the robustness and security respectively of the final product.

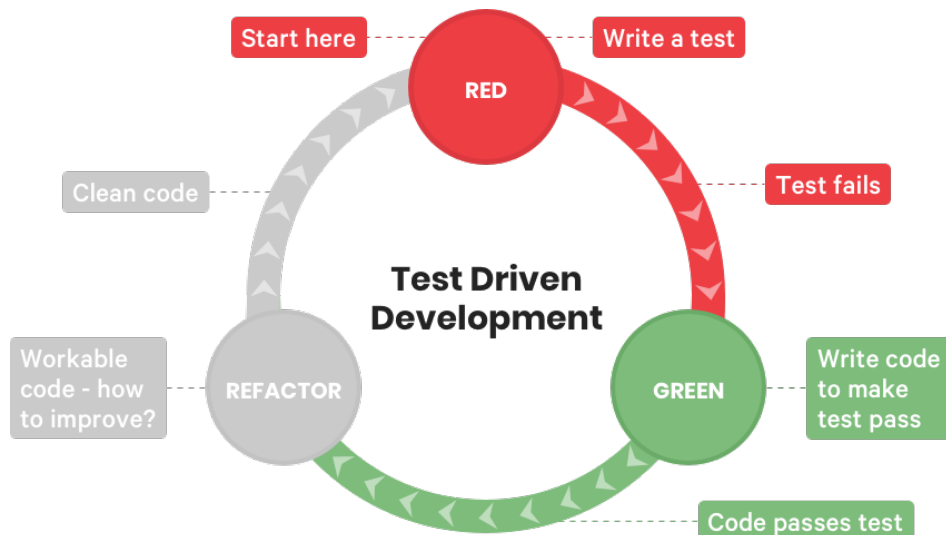


Figure 3.4: Test Driven Development [8]

3.4 Stage 5: Results and Conclusions

A discussion on the results obtained will be carried out, and conclusions drawn. The results are to be compared against the set of initial project requirements to assess the overall progress made and whether or not these have been met. This is done in section 6.

3.5 Stage 6: Recommendations

Suggestions on carrying on of the project (improving the product) will be made. This may include parts of the project that might have been under-developed or required more time to research and implement. This will be addressed in section 7.

4 Detailed Design

This section covers the design aspects to be considered for project implementation. To ensure the project requirements are met, reference to the problem statement in Section 1 and the literature review, Section 2 have to be made. Covered in this section are the user requirements, these are essentially the project requirements that are to address the problem statements specified in the Introduction and reviewed in the literature review. A list of functional requirements is also drawn out. Functional requirements pertain to the practical engineering work to be carried out/what the system must do so as to satisfy the user requirements.

A WBS (Work Breakdown Structure) is presented. This breaks down the project into smaller chunks/ blocks of work to be implemented so as to realise the finalised product deliverable. The tasks set out in the WBS form the "Backlog" as stated in the Methodology section, with reference to the usage of the Agile prototyping methodology, Section 3, sub-section 3.3

Hub and API designs are then proposed in such a way as to meet the user and technical requirements that have been set out. An architectural design, 4.4 is presented using the C4 model which provides a holistic presentation of the design. All these are to be implemented during the experimental work.

4.1 User requirements (URs)

Table III: User requirement 1

<u>UR01</u>	Machine Interface
Requirement	Should allow user to add new devices to the network
Refined by	FR01
Verification	Verified through testing

Table IV: User requirement 2

<u>UR02</u>	Human Interface
Requirement	Should allow user to control and monitor devices registered to the network
Refined by	FR02
Verification	Verified through testing

Table V: User requirement 3

<u>UR03</u>	Affordability-Low cost
Requirement	Final product should be relatively cheaper than similar products available on the market
Refined by	FR02
Verification	Verified through price comparison with what is currently available on the market

Table VI: User requirement 4

<u>UR04</u>	Control and monitoring interface
Requirement	Product should provide an intuitive user interface to access the system while making use of the developed machine interface.
Refined by	FR03
Verification	Verified through testing

Table VII: User requirement 5

<u>UR05</u>	Reliability and Availabilty
Requirement	System should be available whenever required on demand,and provides reliable information.
Refined by	FR04, FR08
Verification	Verified through testing, realiability verified through simulations. Results should be reproducible when testing given a known simulated environment.

Table VIII: User requirement 6

<u>UR06</u>	Security and privacy
Requirement	User data as well as user identity should be safeguarded. Not unauthorised individuals should be able to view user data, and a users identity should not be made public.
Refined by	FR05
Verification	Verified through testing.

Table IX: User requirement 7

<u>UR07</u>	Ease of deployment
Requirement	System/product to be produced should be easy to deploy. That means easy to setup and get it to work first time off.
Refined by	FR06
Verification	Verified through testing.

Table X: User requirement 8

<u>UR08</u>	Interoperability and heterogeneity
Requirement	System/product be able to understand and make use of data from other systems or sub-systems
Refined by	FR07
Verification	Verified through testing.

4.2 Functional Requirements (FRs)

These functional requirements listed below describe what the system must do in order to satisfy the user requirements.

Table XI: Functional requirement 1

<u>FRO1</u>	Devices and hub connected to the network
Requirement	With raspberry pi(acting as the hub) connected to the home network(via WiFi), devices connected to the hub (either via SPI/I2C/UART) indirectly inherit a connection to the network
Refines	UR01
Verification	Verified through testing.

Table XII: Functional requirement 2

<u>FRO2</u>	Front-end framework
Requirement	allows user to gain access to their devices directly from the a mobile phone or web application.
Refines	UR02, UR03
Verification	Verified through testing.

Table XIII: Functional requirement 3

<u>FRO3</u>	API
Requirement	The front-end frame work would be able to make use of the API to gain access to devices. The API is the communication bridge between the machine interface and the user interface.
Refines	UR04
Verification	Verified through testing.

Table XIV: Functional requirement 4

<u>FRO4</u>	Cloud
Requirement	The cloud provides data storage and processing capabilities. If a user desires access to the network devices from outside the network, the cloud can provide this, meaning improved availability. A user can access devices from within the network(via the hub) or from outside the network (via the cloud)
Refines	UR05
Verification	Verified through testing.

Table XV: Functional requirement 5

<u>FR05</u>	Restricted access to user data, devices, and information
Requirement	Requesting a username and password, improves security.
Refines	UR06
Verification	Verified through testing.

Table XVI: Functional requirement 6

<u>FR06</u>	Containerisation
Requirement	Enables deployment of applications without launching an entire virtual machine (VM) improving the ease of deployment.
Refines	UR06
Verification	Verified through testing.

Table XVII: Functional requirement 7

FR07	Standard data-packet packaging
Requirement	Use a known method of packing data to be transported enables other systems/ subsystems to make sense of the data, this improves interoperability. e.g JSON format.
Refines	UR08
Verification	Verified through testing.

Table XVIII: Functional requirement 8

FR08	Sensor and actuators availability
Requirement	To improve the overall availability of the system, sensors and actuators need to have more than one communication channel. If the hub happens to malfunction or cannot receive data, sensors should be able to transmit this data straight to the cloud. Meaning if one channel has malfunctioned the other channel can be used, hence improving availability.
Refines	UR05
Verification	Verified through testing.

4.3 Work breakdown structure (WBS)

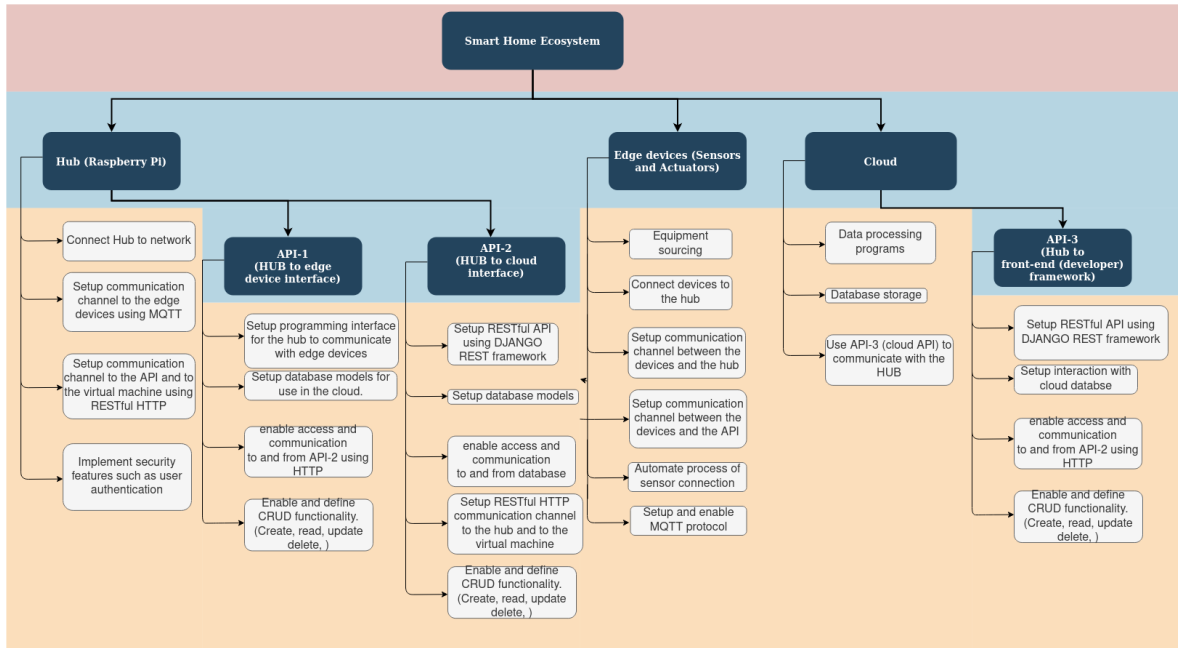


Figure 4.1: Work breakdown structure

As stated in the Methodology section the Backlog of work to be done is taken from the WBS. A Trello dashboard Figure, 9.3 is used to keep track of the done to be done and the completed tasks. N.B For a task to be regarded as complete full coverage testing must have been carried out as well verification and validation.

4.4 Architectural design and overview

The C4 model [43] for visualising software architecture is to be used in this sub-section to illustrate the Architectural design for the smart home ecosystem. The C4 Model has 4 levels to it, namely, Context, Container, Component and (optionally) Code (e.g. UML class) diagrams, with the latter going into more detail than the former.

4.4.1 Level 1: Context

A System Context diagram provides a starting point, showing how the software system in scope fits into the world around it world in terms of the people who use it and the other software systems it interacts with. [44]

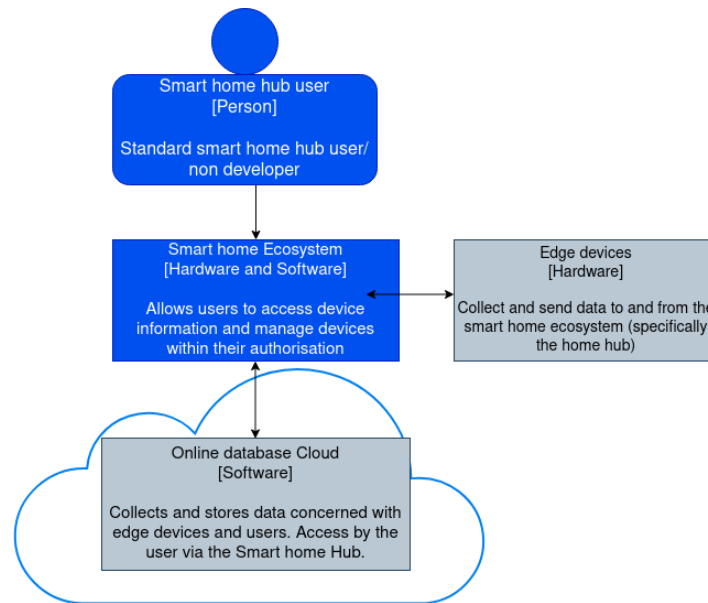


Figure 4.2: Context diagram

4.4.2 Level 2: Container

The Container diagram shows the high-level shape of the software architecture and how responsibilities are distributed across it. It also shows the major technology choices and how the containers communicate with one another. [43]

With reference to Figure 4.3, a web application and a mobile application are shown as user

interfaces for a standard smart home hub user. A standard user is one who is a non-developer. They would receive the smart home with already installed and pre-configured devices. All they need to interact with are the front-end user interfaces to handle their devices. Also shown is the Smart home hub, this is basically a script running on the Raspberry Pi and connects the devices and database to the user via the user interface. It uses the two APIs shown.

API-1 being the hardware API which handles everything hardware related, this includes handling hardware communications between the Raspberry Pi and edge devices. API-2 is the Server-side API and mainly handles communications with the front-end user interfaces and API-1. API-2 is only access pathway to the online database. API-1 and API-2 are further broken down in the component level and explained further.

The design also allows for non standard users to fully utilize the open source architecture to create a personalised bespoke smart home. They can access API-2 to create their own front-end system and user interface. They can also access API-1 and the Raspberry Pi to manually add devices and configure their functionality as they desire.

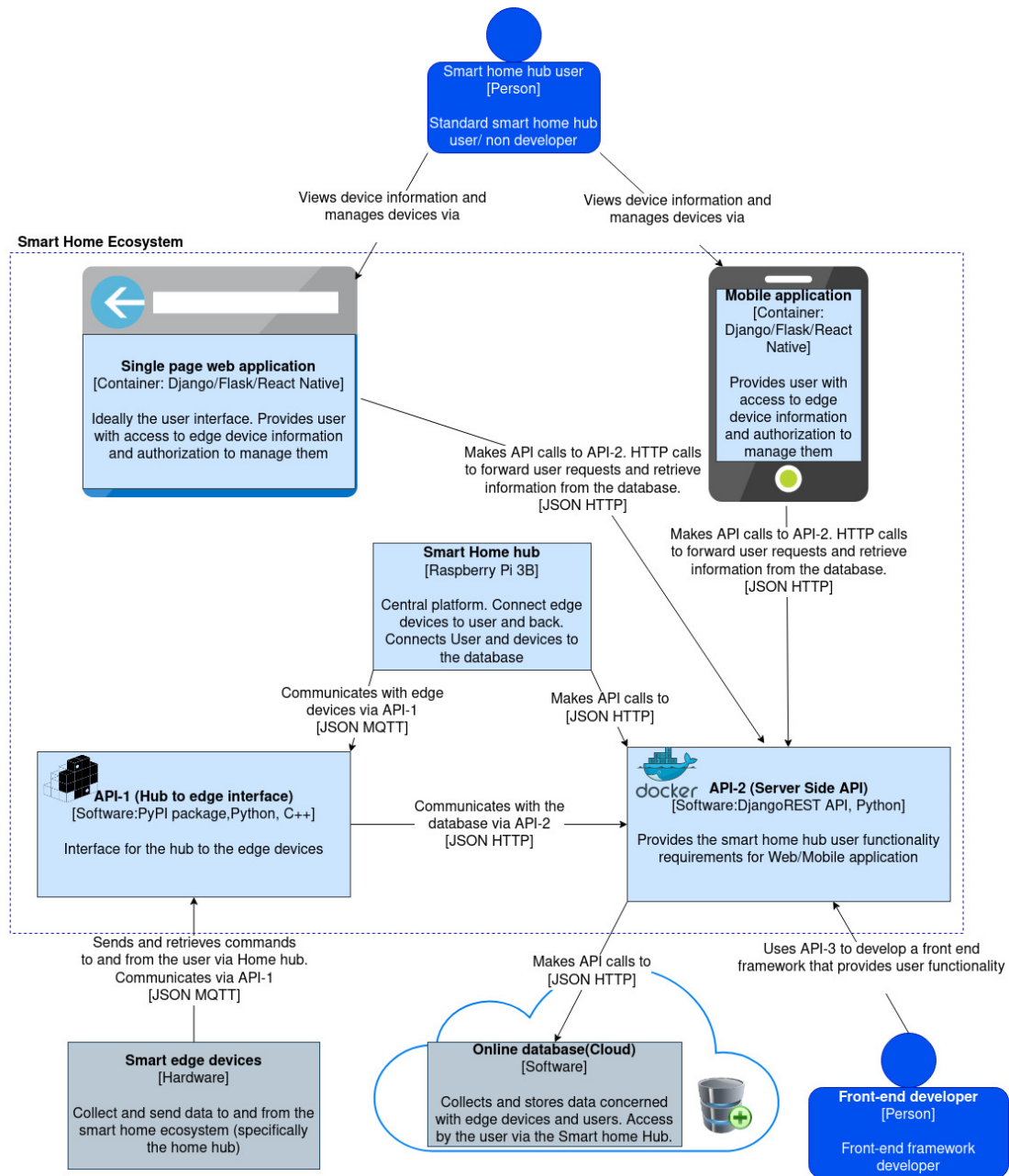


Figure 4.3: Container diagram: Expansion of Smart home ecosystem and its respective component

4.4.3 Level 3: Component

The Component diagram shows how a container is made up of a number of "components", what each of those components are, their responsibilities and the technology/implementation details. The PyPI package API(hardware API/API-1) and the Server-side API(Django REST framework/API-2) are going to be shown in further detail as this are the main dependencies

that the main smart home hub script will be using to tie the whole ecosystem together.

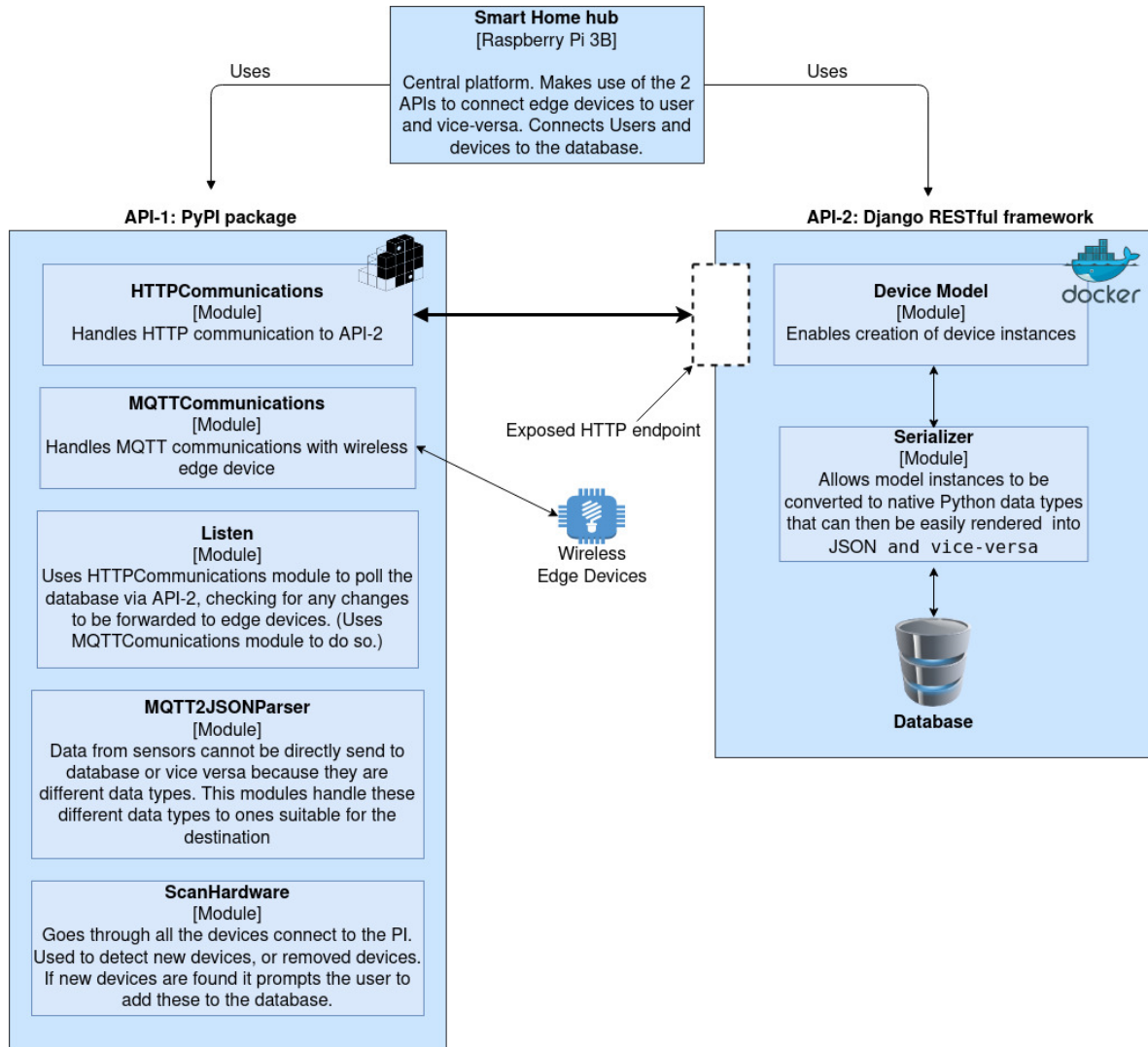


Figure 4.4: Component diagram: Expansion of the hardware API and the server side API

With reference Figure 4.7, there are 5 modules currently available for the hardware API that the smart home hub can make use of. The purposes of these modules are also shown above. The server-side API currently has one model, which is the Device model, this is explored further in the code section 4.4.4 The diagram also show how the two APIs are interconnected and work together using HTTP as the mode of communication. The PyPI hardware API is available at [45], [46] and the server-side API is available at [47]. It should be noted that the server-side API is containerised using Docker, the main reason for this is for the ease of deployment. The API can be hosted on a public sever/local machine/offsite virtual machine. The design for this project will have the API running on a local home machine and on an off-site virtual machine provided by the University. The advantage for this is that if one of

the servers is down, or if a user is not on the home network, they can still access their smart home via the other. This greatly improves availability on demand, which is a critical user requirement.

4.4.4 Level 4: Code

The code section shows how each component is implemented. Because of the large amounts of detail that each component has, this sub-section will only show the most important attributes and methods that bring out the key functionality of the architecture designed. These functionalities are (i) Sending a command from the user interface to an edge device and (ii) Edge devices forwarding the data to the user interface. Representation is done using UML activity diagrams.

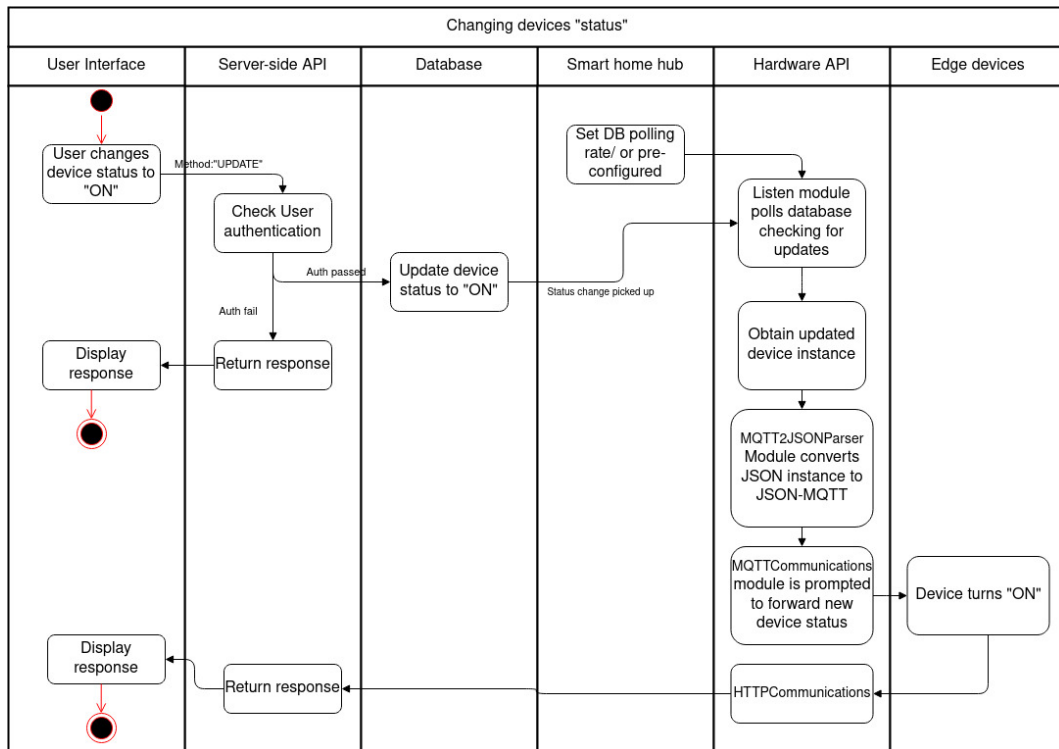


Figure 4.5: UML activity diagram: Showing sequence of activities taking place when a user changes a device status (i)

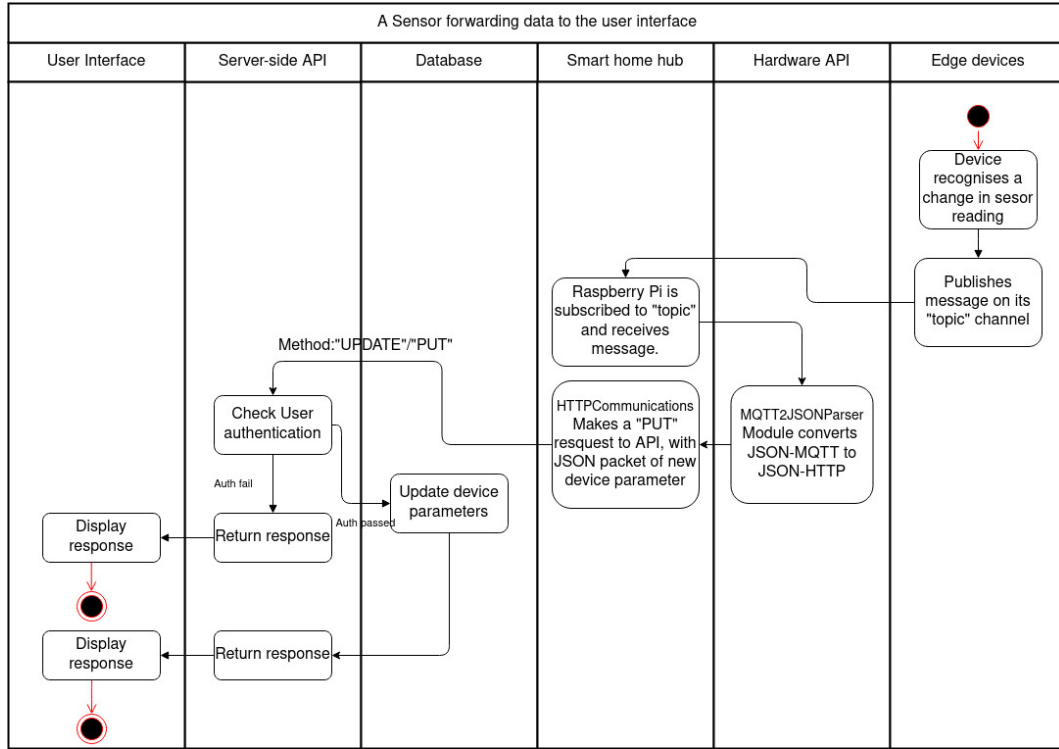


Figure 4.6: UML activity diagram: Sending a command from the user interface to an edge device (ii)

Raspberry Pi Finite State Machine

The Raspberry Pi State Machine shown below is used to described the way the smart home hub(Raspberry Pi) is programmed. A Mealy state machine [48] is used as this is more suitable for the scenario, the output depends on the current present state as well as the present input. Such a state machine reacts to inputs faster and has asynchronous output generation. The state machine will have three states.

1. State 00: Idle state. Constantly polls the database as in Figure 4.5, recognise any database changes and treats it as input. Idle state also waits for updates from edge devices and treats this as inputs.
2. State 01: User input handler.
3. State 11: Edge device input handler

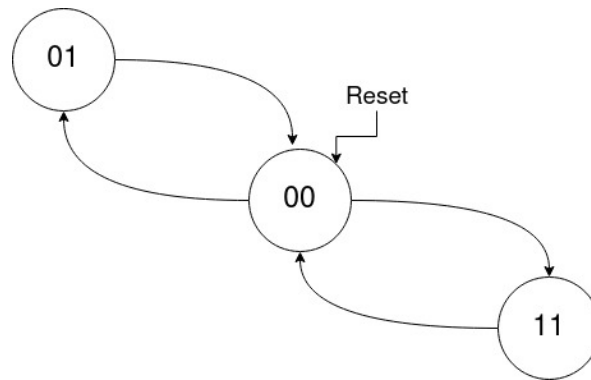


Figure 4.7: Finite state machine. State 00:idle, State 01: User input handler, State 11:Edge device input handler

5 Implementation and practical work

This section outlines the technologies used to implement the designed IoT architecture describe above in Section 4. The section starts off with the server-side API implementation followed by the hardware API, Edge devices and lastly the User Interface.

5.1 Server-side API

A RESTful HTTP framework was used to implement this API using Django REST framework all written using Python. [49]. A "Device" model was set up, and every edge device would be an instance of this model and stored in the database with the various information parameters shown in Figure 5.1.

```
6 class Device(models.Model):
7     name = models.CharField(max_length=20, blank=False, default='device name')
8     category = models.CharField(max_length=10, blank=False, default='sensor')
9     location = models.CharField(max_length=30, blank=False, default='')
10    status = models.CharField(max_length=10, blank=False, default='')
11    reading = models.CharField(max_length=10, blank=True, default='')
12    created = models.DateTimeField(auto_now_add=True)
13    owner = models.ForeignKey(
14        'auth.User', related_name='devices', on_delete=models.CASCADE, default="not set")
15
16    def __str__(self):
17        """Return a human readable representation of the model instance."""
18        return "{}".format(self.name)
19
20    class Meta:
21        ordering = ('created',)
22
```

Figure 5.1: "Device model" setup and information parameters

It should be noted that each device is associated with an "owner" parameter. This parameter is used to implement security measures such as checking for authentication and authorisation. The setup for security and authorisation is as follows:

- Within the smart home network, all users can view device information (e.g parents and children)
- Only a user with admin privileges can operate and control all the devices (e.g parent)
- Normal users without admin privilege can only operate and control devices that are registered to their name.
- Every-time a request is made to the API, a username and password are required to ensure security.
- Unless a user is "admin" only the owner of a device can change its properties or control

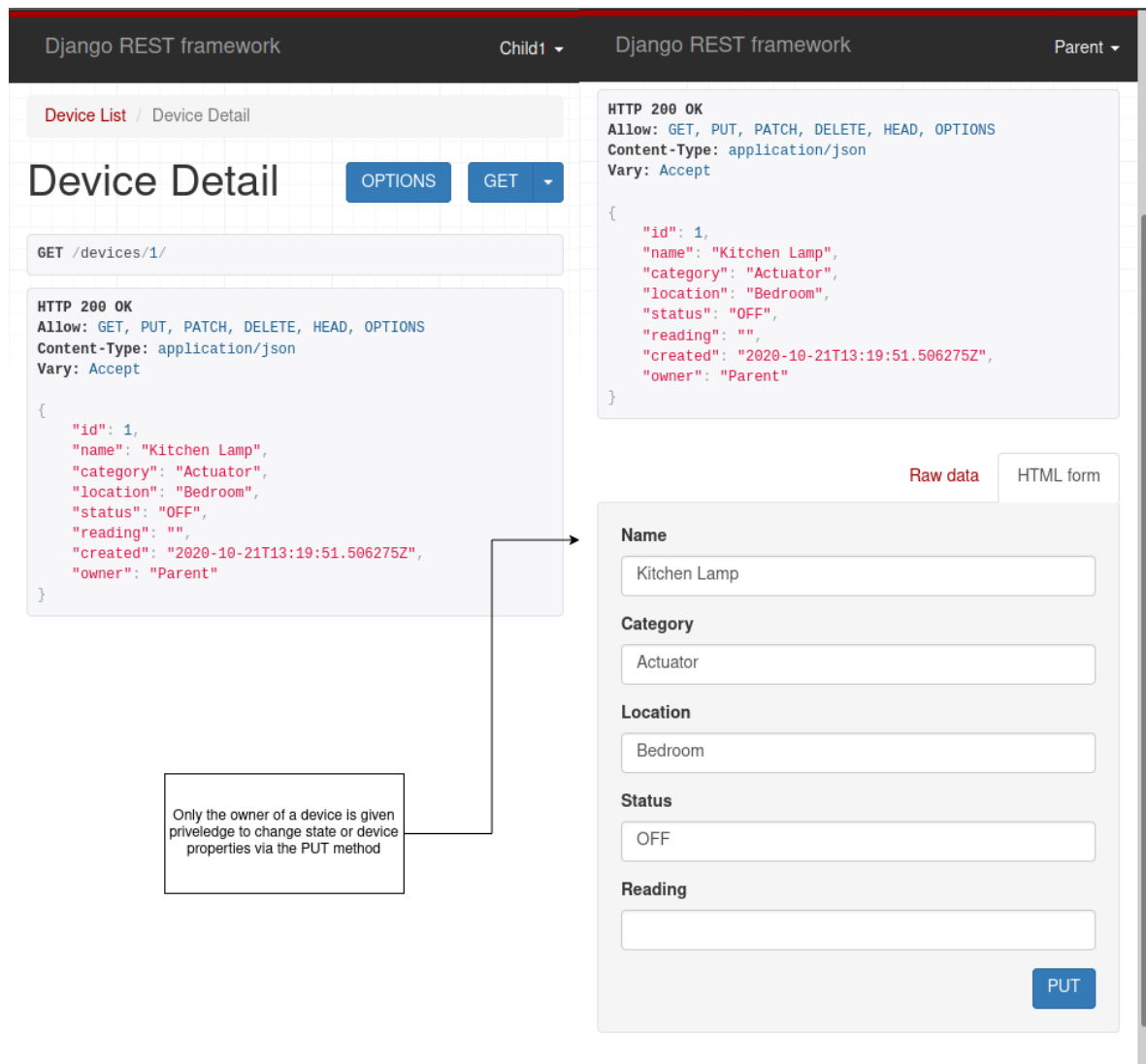


Figure 5.2: User "Parent" is able to control their device using PUT method available to them

Figure 5.2 demonstrates the functionality described above. Only the owner of a device is given permission to change its state or properties via the PUT method. The same is true for the DELETE method. Non-owners can only do as much as viewing a devices' properties.

5.1.1 Adding users

Adding users to the smart home hub can be done easily using the prepared API's Admin user interface. Figure 5.3. Permissions for each user can also be set up. This can be done for each individual user or by creating groups with certain permissions and privileges and then simply adding users to a group.

The image shows two parts of the Django administration interface. On the left is the 'Add user' form, which includes fields for Username, Password, and Password confirmation. It also has a 'Permissions' section on the right with checkboxes for 'Active', 'Staff status', and 'Superuser status'. Below these are two panels: 'Groups' and 'User permissions', each with 'Available' and 'Chosen' lists. The 'Groups' panel shows a search filter and a 'Choose all' button. The 'User permissions' panel shows a list of permissions like 'admin | log entry | Can add log entry' and a 'Choose all' button. At the bottom of the form are three buttons: 'Save and add another', 'Save and continue editing', and 'SAVE'.

Figure 5.3: API admin interface can be used to add users to the smart home hub and setting their permissions.

5.1.2 API endpoints

Available and exposed API endpoints and their usages are shown below in Table XIX.

	Endpoint	Methods available	Usage
1	admin/	GET	Access admin interface
2	devices/	GET, POST	GET devices list, or Create and add new device via POST
3	devices/<int:pk>/	GET, PUT, DELETE	GET specific device information, or Create and add new device via POST
4	users/	GET,POST	View list of users and their associated devices, Create new user.
5	users/<int:pk>/	GET, PUT, DELETE	Edit a specific user, delete a user devices
6	api-auth/	GET,POST	Login page
7	documentation/	GET	Provides all the API documentation,example usage. Allows test runs.

Table XIX: Available and exposed API endpoints

5.1.3 Server-side API documentation

The `documentation/` endpoint allows a user to view and "try-out" all the available API endpoints. This makes it easier for developers both on the back-end and client-side. A screenshot of the documentation endpoint is show in Figure 5.4 and Figure 5.5.

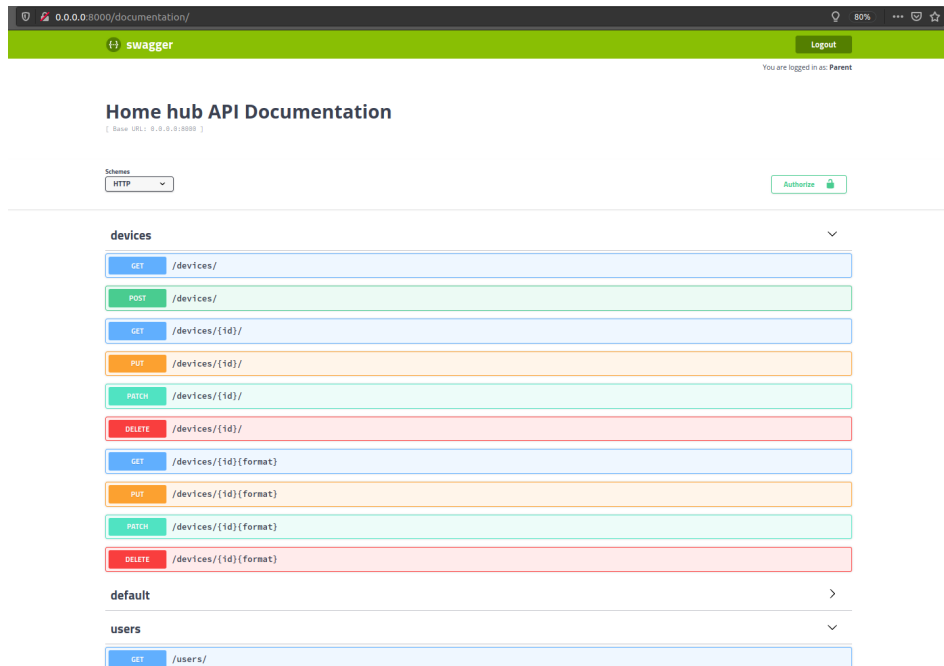


Figure 5.4: Documentation for server-side API

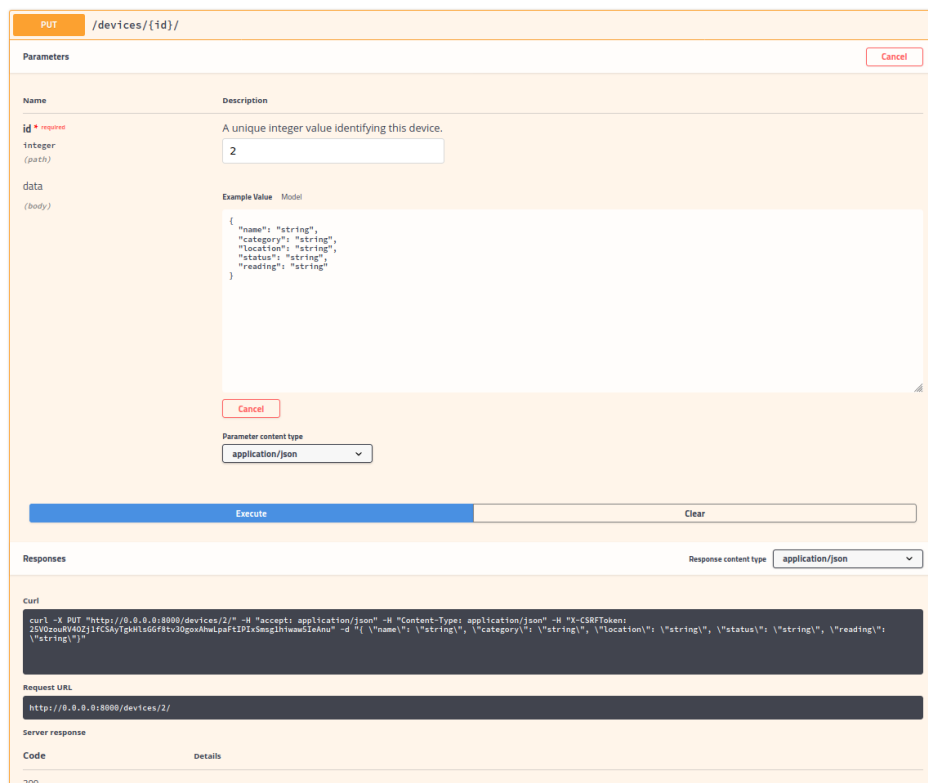


Figure 5.5: Documentation for server-side API: PUT method example. Documentation allows a user to test the API by sending test HTTP requests, receive and view the API response

Currently the API uses CRUD functionality, which stands for Create, Read, Update and Delete. All the methods correspond to POST, GET, PUT/UPDATE and DELETE respectively, in the API. All these methods and their expected responses are shown in Section 9 Figure 9.4 to Figure 9.7.

The API has been containerised using Docker. Containerisation greatly increases the ease of deployment and expansion of the API. One can have as many APIs running on several different servers and each one them connected to the same smart home hub. This enables users to access their smart home ecosystem from within the network and outside the network. This API has been made open-source and is available for development on GitHub at [47].

5.2 Hardware API

As stated above the hardware API handles all the edge devices and is responsible for the communication between the home hub and the edge network of devices. It serves as a means of abstracting the user from the lower level C/C++ code required control I/O on the Raspberry pi and also to the MQTT and HTTP communication operations, i.e an HAL (Hardware Abstraction Layer). Figure 4.7 shows the currently available modules and their purposes.

5.2.1 Communications

Figure 5.7 shows the MQTT class written to implement all the required MQTT communications. The class comes with an available "run" and "pub" methods which subscribe and publish respectively. Of particular importance are the callback functions. A user can use these functions to implements desired actions within their smart home hub. For example the `on_message` callback function has been set to create an `httpCommunications` instance (client) that will forward the data received via MQTT, to the database. The "topic" and "payload" fields in MQTT-class correspond to the "id" and "status" field in the HTTP-class. The `http_put` function also allows the payload to be a JSON format with containing device parameters that will be updated. An example usage case of this class is shown in the Appendix, Figure 9.8.

```

11 class MQTTClass(mqtt.Client):
12
13     def on_connect(self, mqttc, obj, flags, rc):
14         print("rc: "+str(rc))
15
16     def on_message(self, mqttc, obj, msg):
17         client = ConnectToAPI([
18             APIUrl='127.0.0.1:8000', data='/devices']
19         client.http_put(id=msg.topic, status=msg.payload)
20         print(msg.topic+" "+str(msg.qos)+" "+str(msg.payload))
21
22     def on_publish(self, mqttc, obj, mid):
23         print("mid: "+str(mid))
24
25     def on_subscribe(self, mqttc, obj, mid, granted_qos):
26         print("Subscribed: "+str(mid)+" "+str(granted_qos))
27
28     def on_log(self, mqttc, obj, level, string):
29         print(string)
30
31     def run(self, MQTT_BROKER, MQTT_PORT, MQTT_TOPIC, **kwargs):
32         self.connect(MQTT_BROKER, MQTT_PORT, 60)
33         self.subscribe(MQTT_TOPIC, 0)
34         rc = 0
35         while rc == 0:
36             rc = self.loop()
37         return rc
38
39     def pub(self, MQTT_BROKER, MQTT_PORT, MQTT_TOPIC, MQTT_MSG, **kwargs):
40         self.connect(MQTT_BROKER, 1883, 60)
41         self.publish(MQTT_TOPIC, MQTT_MSG)
42

```

on-message callback function invokes the httpCommunication class to forward received data to the database.

Figure 5.6: MQTTCommunications class

5.2.2 Setting up a new device: A sensor

To setup a new edge device, be it a sensor or actuator, both the APIs need to be used by the smart home hub (RPI). An MQTTClass object is set up on the Raspberry Pi for each edge device, this way each edge device, inherits access to the home hub as well as the database.

The device being set up in this instance is a soil moisture sensor, being read from by the ESP8266 WiFi module. The WiFi module can then communicate with the Raspberry Pi using MQTT. On the Raspberry Pi an MQTTClass object is created which subscribes to the topic published by the sensor and forwards this data to the database every time a message is received. Listing 3 in the appendix shows code for setting up an edge device and enabling MQTT communications. The MQTT class then goes on to update the database via the front-end API. Every edge device sensor would be required to have such functionality.

After setting up the edge device, the next step would be to create an instance of it on the Raspberry Pi. Listing 1 shows how this is done, by importing the created hardware API called "homehubRPI" and calling the relevant class, MQTTClass. Below, the "mqttc" object is now "Subscribed" to the sensor.

```

1
2 from homehubRPI.mqttCommunications import MQTTClass

```

```

3
4 mqttc = MQTTClass()
5 rc = mqttc.run(MQTT_BROKER="192.168.1.61",
6               MQTT_TOPIC="1", MQTT_PORT=1883)
7 print("rc:"+str(rc)) #for debugging help. Prints all logs in the terminal

```

Listing 1: homehubRPi API usage

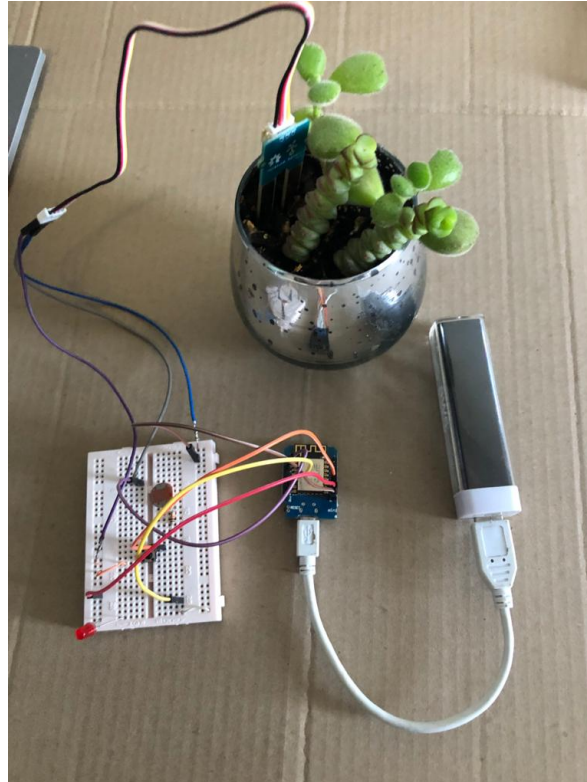


Figure 5.7: Moisture sensor setup, connected to the ESP3266 WiFi module

5.2.3 Setting up a new device: An actuator

Setting up an actuator takes the same steps as setting up a sensor described above but with one major difference. Unlike a sensor that only performs the task of sensing and publishing a message, an actuator should be able to receive commands as messages from the user. This means an actuator should have MQTT subscribing functionality. Listing 2 shows the simple steps to setup an actuator for subscribing. The call back function simply defines what actions to performs every time a message is received. This is performed asynchronously. In this case the desk lamp is turned either ON or OFF.

In this example the actuator being set up is a normal desk lamp attached to the ESP3266. It is subscribed to the Raspberry Pi IP Address to its specific topic.

```

1 void callback(char* topic, byte* payload, unsigned int length) {
2   Serial.print("Message arrived in topic: ");
3   Serial.println(topic);
4   Serial.print("Message:");
5   String messageTemp;
6   for (int i = 0; i < length; i++) {
7     Serial.print((char)payload[i]);
8     messageTemp += (char)payload[i];
9     Status += (char)payload[i];
10  }
11  // Status = (payload);
12  Serial.println();
13  Serial.println("-----");
14  if (messageTemp=="off"){           //relay switch used in "normally-open" mode,
15    digitalWrite(ledPin, HIGH);      // hence the inversion
16  }
17  else if (messageTemp == "on"){
18    digitalWrite(ledPin, LOW);
19  }
20 }
21 void setup{
22   // Connect to MQTT Broker
23   // client.connect returns a boolean value to let us know if the connection was successful.
24   if (client.connect(clientID)) {
25     client.publish(mqtt_topic, "Test publish to broker...");
26     Serial.println("Connected to MQTT Broker!");
27   }
28   else {
29     Serial.println("Connection to MQTT Broker failed...");
30   }
31 }
32 client.subscribe("MQTT_TOPIC");
33 client.setCallback(callback);

```

Listing 2: Example code for setting up an actuator to receive commands from a user

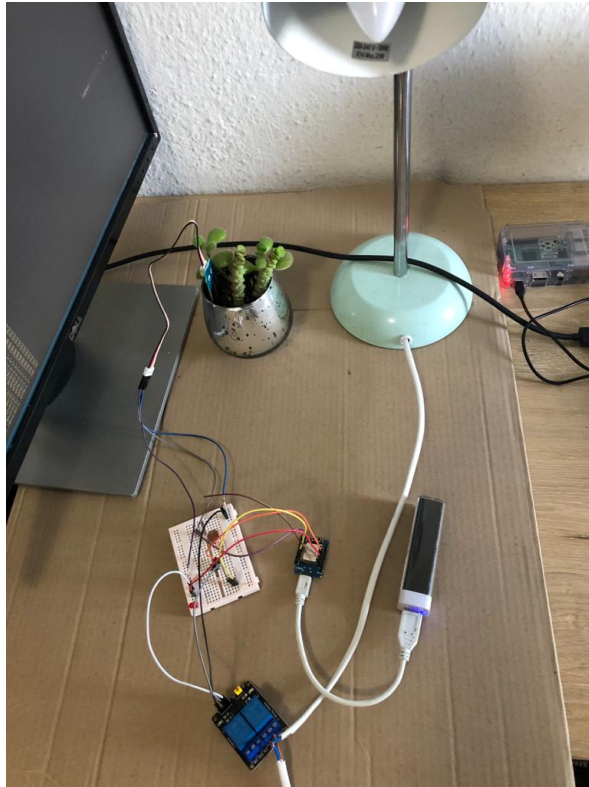


Figure 5.8: Desk Lamp setup, connected to an ESP3266 WiFi module

5.2.4 Hardware API Documentation

Documentation for the hardware API has been generated using pydocs, and can be viewed in the browser or terminal. To access the docs fetch the repository from GitHub, enter the docs folder and open `homehubRPi.html`. Also included within the documentation is documentation of packages/ APIs used when creating the hardware API.

The hardware API is available at [45]. It is also open source and the source code is available for the community on GitHub at [46]

5.3 User Interface

A very simple user interface was setup using Gatsby and React JavaScript to demonstrate the implemented solution and its ease of use. It also acts as a proof of concept with regards to the functionality of the API. Figure 5.9 shows a snapshot of the user interface. The user interface set up is highly scalable and can be further improved to provide functionality such as graphs, device insights and more. This is available and free to use on GitHub at [50].

Smart Home Hub

Moisture Sensor

Sensor Status : ON

Sensor Reading : 54%

ON

OFF

REFRESH

Light Bulb

Sensor Status : ON

Sensor Reading : Non

ON

OFF

REFRESH

Sensor Status : OFF

Sensor Reading :

ON

OFF

REFRESH

Sensor Status : OFF

Sensor Reading :

ON

OFF

REFRESH

Sensor Status : OFF

Sensor Reading :

ON

OFF

REFRESH

Figure 5.9: Screenshot of the user interface set up

5.4 Testing

The TDD (test driven development) methodology described in 3.3.1 was used for testing, Testing the APIs was done through unit testing, each function written was tested and checked to see if it performed as required. The complete smart home ecosystem was designed as a proof of concept for the architecture designed but also served as means of testing the implementation performed and final product created as a whole. Time constraints did not allow for further testing using the methodologies described in 3.3.1.

6 Results and Discussion

This section discusses the outcomes produced from the implementation of the proposed system architecture. This section starts off by comparing the final product with relation to the problem statement. It is essential for every project that the problem statement is addressed. Afterwards the same is done for the user requirements outlined in sub-section 4.1. The section then goes on to compare the final product to other solutions on the market, and finally addresses limitations, if any, of the proposed final product.

6.1 Problem Statement

The problem statement has been outlined in the Introduction section 1, sub-sections 1.1,1.3. The major areas of focus were to deliver a smart home infrastructure that enables reliability, security, cost reduction and ease of use without the cost of flexibility.

All these pertinent issues were addressed in the design section. Containerising the front-end API using Docker enabled the API to be deployed on various server, in this case on a local Linux machine, and an off-site Linux server provided by the University. This meant that the smart home system was accessible from within and outside the local home network. It also added robustness to the system, providing a back-up solution in case one server is down. This makes the system reliable and improves availability on demand. Containerising the API also meant for ease of deployment. After getting the project files, only two terminal commands are required to get the docker container and server running.

Security was also a major concern and was dealt with at every stage of system design. The MQTT protocol of communication can be set up such that for every message sent, a username and password are required. It also provides 3 QOS (Quality of Service) functionalities that a user can implement. Some data such as sensor data that has been published might not be of much importance and hence not require an ACK(acknowledgement message). But some actions like switching on a device require an ACK message from the device itself to ensure that the message was received and the action performed. All the functionalities can be utilised by the user according to their own discretion. The front-end API carries out authentication and authorisation for every HTTP request made. This means that one cannot gain access to the smart home system without any login credentials. Even edge devices performing HTTP request to the API will need to provide authentication information before their requests are approved.

The architecture designed is highly flexible as it can have many use cases. This architecture can not only be for the smart home control, but can be extended to farms, offices and various other IOT use case scenarios such as Autonomous and Connected Vehicles, Smart Watches and Fitness Trackers and even drones for industrial search and rescue.

6.2 User requirements

The proposed solution satisfies all the set user requirements listed in 4.1. The solution provides a machine interface, a human interface, and allows for control and monitoring of peripheral devices. The inter-operability and heterogeneity requirements was satisfied as well, by using the Raspberry Pi, a variety of edge devices can be supported alongside their various communication techniques. All the following communications are supported: WiFi, Bluetooth, SPI, I2C, UART. By using some additional hardware attached to the Pi, other protocols such as Zigbee, Z-wave, LoRaWAN can also be easily supported.

6.3 Market comparison

One of the requirements for the final product deliverable was that it should be a cost reduction solution, meaning it should be within the same price range as other already existing smart home technologies on the market. Table XX shows the comparison of the proposed solution and final product against some of the currently popular smart home systems available on the market.

Product name	Strengths	Weaknesses	Cost /ZAR
Amazon Echo [51]	Sought after voice control	Comes with a lot of privacy concerns, Amazon records all audio after the "wake up word"	2438.76
		Not flexible, only compatible with Amazon approved sensors which is not desirable for DIY enthusiasts. Currently, only a few manufacturers offer Alexa-certified sensors: Samsung SmartThings, Sylvania Lightify, and Philips Hue, Sony etc	
		Requires an active internet connection at all times as it is a cloud based device.	
		Users not residing in the US or UK have been experiencing incompatibility issues.	
OpenHAB [52]	Easy installation and setup	Slow pace of develop due to its rigorous approval tests. Might not keep up with latest technologies being released.	free
	Easy to use REST APIs, Integratable with numerous different technologies	Flexibility comes with some complexity, this might be difficult for people with little technical knowledge.	
	Highly flexible.		
	Stable platform at the cost of pace of development		
	Provides hassle Raspberry Pi image:openHABian		
	Large active community, quick responses		
	Pluggable architecture		
	Runs everywhere: Linux, macOS, Windows, Raspberry Pi, PINE64, Docker, Synology,iOS, Android		
	Good up-to-date documentation		

Table XX continued from previous page

Product name	Strengths	Weaknesses	Cost /ZAR
HomeAssistant [53]	Supports a lot of devices e.g Alexa, Google, Apple Home support	Requires some IT knowledge	free
	Fast pace of development, weekly releases	Due to the fast releases, high risk of breaking your instance with every release. Tests are not rigorous	
	Suitable for DIY enthusiasts	Documentation sometimes not up-to-date. Community not very active, slow responses	
Proposed product	Two main APIs that work together. Easy to use APIs, integrateable with many framework and architectures, including interfacing with OpenHABs RESTful API, GitHub, Slack, DigitalOcean, Travis CI etc		free
	Security assurance, work offline locally, only requires an active network		
	Initially build for the Raspberry Pi, can be tweaked to suit various other boards, such as Beaglebone, Arduino, etc		
	Highly flexible.		
	On-demand availability and reliability		

Table XX: Market comparisons of smart home systems, both open source and proprietary

6.4 Limitations

Although the designed architecture fared well against other solutions and products on the market and managed to address the problem statement, it did not come without its drawbacks. Proposed solutions to address these limitations shall be addressed in the Recommendations section 7.

The biggest limitation of the design was the lack of asynchronicity in the cloud layer (i.e. front-end API). The HTTP communication protocol takes the request/response protocol. This means that if a user wants to see any of his/her devices' information, a request has to be made to the API. The only way for values to be constantly updated onto the user interface is through polling. The drawbacks for this is that a user's information is not updated in real time and might be delayed. This means that such a design is not suitable for more time-sensitive, time-critical operations within the smart home system. Because of this communication protocol, operations that need to travel from one end to the other (i.e. edge to user) had undesirable latencies ranging from 1-3 seconds, depending on the polling rates set.

Setting higher polling reduces the latency problem, but comes at a cost. The most important of power usage, and CPU usage. These become important especially when dealing with constrained edge devices that have limited processing power and a finite power source. The Django REST framework implemented on a local server can only support 1 request at a time, this means that if multiple devices/users make requests at the same time, only one request will be accepted and handled; the rest would not be acknowledged. Meaning requests might need to be repeated and this is not ideal.

7 Recommendations for future work

This paper entails the development of a smart home architectural design and its implementation. After implementing the proposed it was evident that the design could be further improved. This section addresses the limitations of the proposed solution set out in the Results and discussion section above, Section 6 by making suggestions on further research and practical implementations require to better the product. These recommendations are grouped into their subsection below.

7.1 Latency concerns

The high latency issues of the proposed solution was due to the use of a request/response type protocol. This evidently had an undesirable effect. To address this concern other communication protocols would have to be implemented. The REST framework created does not need to be deprecated but rather just change the way it communicates. Better communication protocols would be one that allow for asynchronous communications to take place. One such protocol is the web-socket protocol [54], usually used for chatting application like Whatsapp, this protocol is much more suitable for real-time applications such as this one. Web-sockets allow for a full-duplex communication channel to be created after a single handshake between the client and the server.

7.2 Data compression

Explore methods of compressing data from edge devices before it is transmitted. Currently on raw data is being transmitted. It would prove more ideal to be able to send metadata as well. This may prove useful for the open source community at large and for developer users. It should be taken into account that the devices be used are constrained devices. Data compression would enable more data to be transmitted considering that the MQTT communication protocols has a bandwidth constraint.

7.3 PCB design

One of the major objectives of this project was to create a low-cost and affordable solution. This means that the solution had to be relatively cheaper than competing products that are currently on the market. Sub-section 6.3 shows the prices of competing products. The architectural design proposed had a competitive edge as it could be run locally and still be available offline, only requiring an active network connection. One problem to using a Raspberry Pi and other small board computers is that they can get fairly pricey. The Raspberry Pi Model 3B+ costs about ZAR600. This is fairly cheaper than other smart home hubs, but this price can be further reduced. By designing a PCB, one has the ability to construct a board that only performs the tasks required. For example such a PCB can

have and on-board WiFi module, Bluetooth module, Zigbee module and a processor that is capable of performing only the required functions. This would in turn produce reduce the cost drastically.

To do this one might start off by coding all the required functionality using a hardware descriptive language, e.g Verilog on an FPGA (Field Programmable Gate Array) and running simulations. Such simulations would provide insight on what specifications the PCBs processor should have, such as RAM, supply voltage, processor architecture (RISC/CISC), clock speed, power dissipation, on-chip oscillators, real time clock, number of UARTs etc.

7.4 Operating system

Currently the operating system running on the Raspberry Pi is Raspbian. Although the Raspberry Pi is typically used with the Linux operating system, it's not necessarily the best choice for applications requiring low latency, predictable response to external events. [55]. It is recommended that the proposed solution should be implemented on more suitable operating system. A more light-weight system such as ChibiOS/RT [56] would be a good start as it is suitable for the more real-time operations and provides desirable functionalities such as cross compilation, thread context switching, concurrency control, interrupt processing and device driver development.

7.5 Encryption/Decryption

Explore ways to implement data encryption and decryption. This ensures an extra layer of security if it so happens that the system has been breached, an attacker would not be able to make sense of any data being communicated or store in the database. This in turn, a users privacy.

7.6 Data driven performance tests

Although unit tests were written for every function, a downfall to unit tests is that they are hard coded and hence some edge cases might be missed. To ensure the system performs as specified under all conditions with absolute certainty, the system should be tested exhaustively. Data driven performance tests are a way of ensuring this.

Data Driven Testing is a software testing method in which test data is stored in table or spreadsheet format. Data driven testing allows testers to input a single test script that can execute tests for all test data from a table and expect the test output in the same table. [57]. Figure 7.1 shows how such a procedure would be undertaken. Another suggestion for this this procedure out would be writing "edge device simulation scripts", and "User simulation scripts" that interact with the application under test.

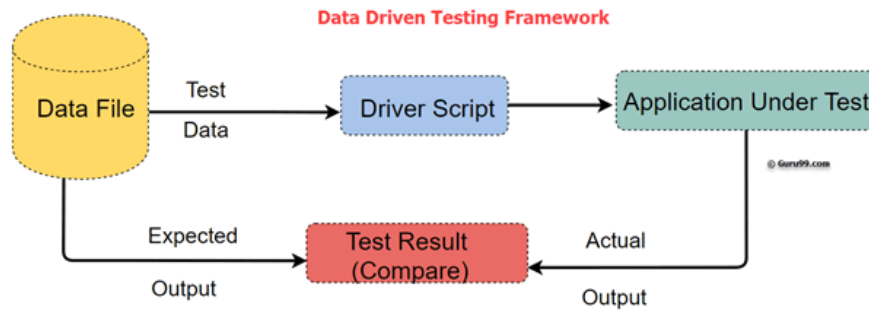


Figure 7.1: Data driven performance testing procedure

7.7 Penetration testing

For the same reasons that data driven performance tests need to be performed, penetration testing also known as Pen testing would ascertain the security of the proposed solution. An IoT penetration test is defined as the assessment and exploitation of various components present in an IoT device/network solution to help make the device/network more secure. [58] This method of testing would make the product more secure, ensuring a user privacy.

8 Conclusion

This section concludes the report on smart home ecosystems. The objectives and requirements stated for the project in Section 1.2, Section 4.1 were all achieved through the following ways:

1. An in depth review on smart home ecosystems was carried out. This review covered issues pertinent to the problem statement including IOT architectures, communication protocols, and data security and privacy.
2. Upon reviewing the literature, a design of a smart home ecosystem was proposed. This architectural design, was able to meet all the requirements stated in 4.1 and 4.2.
3. The designed accounted for issues of concern which were reliability, security, cost reduction, ease of use and flexibility. All these criteria were satisfied as explored above in Section 6.
4. After the design of a suitable architecture, the implementation was then carried out. Two APIs were designed and created. One being a hardware API and the other a server-side API. By adopting these well designed APIs a smart home ecosystem was then implemented on the Raspberry Pi acting as a demonstrator hub.
5. A simple user interface was set up to demonstrate the functionality of the smart home ecosystem as a whole.
6. Full documentation was provided for the APIs created.
7. A review of the finished product was carried out and suggestion for further improvements were proposed.

In conclusion, the designed and implemented solution satisfied all the requirements and specifications. It can be easily expanded and improved by individuals and the open-source community. It is easy to use and upon adoption of the architecture designed an APIs provided, creating a smart home ecosystem for personal use simply becomes a DIY project.

References

- [1] B. L. R. Stojkoska and K. V. Trivodaliev, “A review of internet of things for smart home: Challenges and solutions,” *Journal of Cleaner Production*, vol. 140, pp. 1454–1464, 2017.
- [2] M. R. Alam, M. B. I. Reaz, and M. A. M. Ali, “A review of smart homes—past, present, and future,” *IEEE transactions on systems, man, and cybernetics, part C (applications and reviews)*, vol. 42, no. 6, pp. 1190–1203, 2012.
- [3] D. Mocrii, Y. Chen, and P. Musilek, “Iot-based smart homes: A review of system architecture, software, communications, privacy and security,” *Internet of Things*, vol. 1, pp. 81–98, 2018.
- [4] “Fog computing and the internet of things: Extend the cloud to where the things are,” 2014. [Online]. Available: https://www.cisco.com/c/dam/en_us/solutions/trends/iot/docs/computing-overview.pdf
- [5] J. Dizdarević, F. Carpio, A. Jukan, and X. Masip-Bruin, “A survey of communication protocols for internet of things and related challenges of fog and cloud computing integration,” *ACM Computing Surveys (CSUR)*, vol. 51, no. 6, pp. 1–29, 2019.
- [6] Difference between agile and lean. [Online]. Available: <http://www.differencebetween.net/business/difference-between-agile-and-lean/>
- [7] T. F. E. Wikipedia. V-model. [Online]. Available: <https://en.wikipedia.org/w/index.php?title=V-Model&oldid=852259910>
- [8] Why use test driven development: 6 benefits for your project. [Online]. Available: <https://www.codica.com/blog/test-driven-development-benefits/>
- [9] D. Marikyan, S. Papagiannidis, and E. Alamanos, “A systematic review of the smart home literature: A user perspective,” *Technological Forecasting and Social Change*, vol. 138, pp. 139–154, 2019.
- [10] J. Han, Y. Jeon, and J. Kim, “Security considerations for secure and trustworthy smart home system in the iot environment,” in *2015 International Conference on Information and Communication Technology Convergence (ICTC)*, 2015, pp. 1116–1118.
- [11] F. Aldrich, “Chapter “smart homes: Past, present and future”, inside the smart home,” 2003.
- [12] N. Balta-Ozkan, R. Davidson, M. Bicket, and L. Whitmarsh, “Social barriers to the adoption of smart homes,” *Energy Policy*, vol. 63, pp. 363–374, 2013.

- [13] C. Orwat, A. Graefe, and T. Faulwasser, “Towards pervasive computing in health care—a literature review,” *BMC medical informatics and decision making*, vol. 8, no. 1, p. 26, 2008.
- [14] M. Chan, E. Campo, D. Estève, and J.-Y. Fourniols, “Smart homes—current features and future perspectives,” *Maturitas*, vol. 64, no. 2, pp. 90–97, 2009.
- [15] S. M. Finkelstein, S. M. Speedie, G. Demiris, M. Veen, J. M. Lundgren, and S. Potthoff, “Telehomecare: quality, perception, satisfaction,” *Telemedicine Journal & e-Health*, vol. 10, no. 2, pp. 122–128, 2004.
- [16] T. L. Finch, M. Mort, F. S. Mair, and C. R. May, “Future patients? telehealthcare, roles and responsibilities,” *Health & social care in the community*, vol. 16, no. 1, pp. 86–95, 2008.
- [17] J. Byun, B. Jeon, J. Noh, Y. Kim, and S. Park, “An intelligent self-adjusting sensor for smart home services based on zigbee communications,” *IEEE Transactions on Consumer Electronics*, vol. 58, no. 3, pp. 794–802, 2012.
- [18] Q. Zhu, R. Wang, Q. Chen, Y. Liu, and W. Qin, “Iot gateway: Bridging wireless sensor networks into internet of things,” in *2010 IEEE/IFIP International Conference on Embedded and Ubiquitous Computing*, 2010, pp. 347–352.
- [19] B. Xu, L. Da Xu, H. Cai, C. Xie, J. Hu, and F. Bu, “Ubiquitous data accessing method in iot-based information system for emergency medical services,” *IEEE Transactions on Industrial informatics*, vol. 10, no. 2, pp. 1578–1586, 2014.
- [20] Getdrowing. [Online]. Available: <https://www.getgrowing.app/api-doc/>
- [21] Minutapi. [Online]. Available: <https://api.minut.com/v1/docs/>
- [22] Lightwave smart series api. [Online]. Available: <https://lightwaverf.com/blogs/news/smart-series-api>
- [23] V. Biljana L.Risteska Stojkoska Kire, “A review of internet of things for smart home: Challenges and solutions,” 2017.
- [24] L. Da Xu, W. He, and S. Li, “Internet of things in industries: A survey,” *IEEE Transactions on industrial informatics*, vol. 10, no. 4, pp. 2233–2243, 2014.
- [25] J. Gubbi, R. Buyya, S. Marusic, and M. Palaniswami, “Internet of things (iot): A vision, architectural elements, and future directions,” *Future generation computer systems*, vol. 29, no. 7, pp. 1645–1660, 2013.
- [26] Amazon web services. [Online]. Available: <https://aws.amazon.com/>
- [27] Microsoft azure. [Online]. Available: <https://azure.microsoft.com/en-us/>

- [28] T. G. Peter Mell, “The nist definition of cloud computing recommendations of the national institute of standards and technology,” 2011. [Online]. Available: <https://nvlpubs.nist.gov/nistpubs/Legacy/SP/nistspecialpublication800-145.pdf>
- [29] C. Carvalho, “The gap between processor and memory speeds,” in *Proc. of IEEE International Conference on Control and Automation*, 2002.
- [30] “Fog computing vs. edge computing.” [Online]. Available: <https://info.opto22.com/fog-vs-edge-computing>
- [31] S. Yi, Z. Qin, and Q. Li, “Security and privacy issues of fog computing: A survey,” in *International conference on wireless algorithms, systems, and applications*. Springer, 2015, pp. 685–695.
- [32] I. Stojmenovic and S. Wen, “The fog computing paradigm: Scenarios and security issues,” in *2014 federated conference on computer science and information systems*. IEEE, 2014, pp. 1–8.
- [33] W. Shi, J. Cao, Q. Zhang, Y. Li, and L. Xu, “Edge computing: Vision and challenges,” *IEEE internet of things journal*, vol. 3, no. 5, pp. 637–646, 2016.
- [34] W. Shi and S. Dustdar, “The promise of edge computing,” *Computer*, vol. 49, no. 5, pp. 78–81, 2016.
- [35] S. Dustdar, C. Avasalcai, and I. Murturi, “Invited paper: Edge and fog computing: Vision and research challenges,” in *2019 IEEE International Conference on Service-Oriented System Engineering (SOSE)*, 2019, pp. 96–9609.
- [36] M. Zhou, R. Zhang, W. Xie, W. Qian, and A. Zhou, “Security and privacy in cloud computing: A survey,” in *2010 Sixth International Conference on Semantics, Knowledge and Grids*. IEEE, 2010, pp. 105–112.
- [37] Y. Zheng, X. Ding, C. C. Y. Poon, B. P. L. Lo, H. Zhang, X. Zhou, G. Yang, N. Zhao, and Y. Zhang, “Unobtrusive sensing and wearable devices for health informatics,” *IEEE Transactions on Biomedical Engineering*, vol. 61, no. 5, pp. 1538–1554, 2014.
- [38] S. Nasiri, F. Sadoughi, M. H. Tadayon, and A. Dehnad, “Security requirements of internet of things-based healthcare system: A survey study,” *Acta Informatica Medica*, vol. 27, no. 4, p. 253, 2019.
- [39] D. Puthal, M. S. Obaidat, P. Nanda, M. Prasad, S. P. Mohanty, and A. Y. Zomaya, “Secure and sustainable load balancing of edge data centers in fog computing,” *IEEE Communications Magazine*, vol. 56, no. 5, pp. 60–65, 2018.
- [40] MuleSoft. What is a restful api? [Online]. Available: <https://www.mulesoft.com/resources/api/restful->

- [41] D. Muslihat. Agile methodology:an overview. [Online]. Available: <https://zenkit.com/en/blog/agile-methodology-an-overview/>
- [42] B. D. Tackett and B. V. Doren, "Process control for error-free software: A software success story," *IEEE Software*, vol. 24, no. 03, pp. 24–29, may 1999.
- [43] The c4 model for visualising software architecture. [Online]. Available: <https://c4model.com/>
- [44] The c4 model for software architecture. [Online]. Available: <https://www.infoq.com/articles/C4-architecture-model/>
- [45] Sever-side api-ngonimombeshora. [Online]. Available: <https://pypi.org/project/homehubRPi/>
- [46] Pypi hardware api on github. [Online]. Available: <https://github.com/ngonimombeshora/homehubRPi>
- [47] Sever-side api-ngonimombeshora. [Online]. Available: <https://github.com/ngonimombeshora/Server-side-API>
- [48] The mealy machine. [Online]. Available: https://en.wikipedia.org/wiki/Mealy_machine
- [49] Django rest framework. [Online]. Available: <https://www.django-rest-framework.org/>
- [50] Homehubui. [Online]. Available: <https://github.com/ngonimombeshora/humehub-UI>
- [51] Amazon alexa. [Online]. Available: https://www.amazon.com/gp/product/B0794W1SKP/ref=as_li_tl?ie=UTF8&camp=1789&creative=9325&creativeASIN=B0794W1SKP&linkCode=as2&tag=safety_smart-home-hubs-20&linkId=4b2a1f80d98ab358efe154bd9c28644c
- [52] Openhab. [Online]. Available: <https://www.openhab.org/>
- [53] Iot pen testing. [Online]. Available: <https://www.home-assistant.io/>
- [54] The websocket protocol. [Online]. Available: <https://tools.ietf.org/html/rfc6455>
- [55] Chibios/rt on the raspberry pi. [Online]. Available: <https://www.stevebate.net/chibios-rpi/GettingStarted.html>
- [56] Chibios. [Online]. Available: <https://www.chibios.org/dokuwiki/doku.php>
- [57] What is data driven testing? learn to create framework. [Online]. Available: <https://www.guru99.com/data-driven-testing.html>
- [58] Iot pen testing. [Online]. Available: <https://medium.com/@xesey/iot-pen-testing-1c4849327499>

Appendix

9 Appendix

9.1 Plan of development

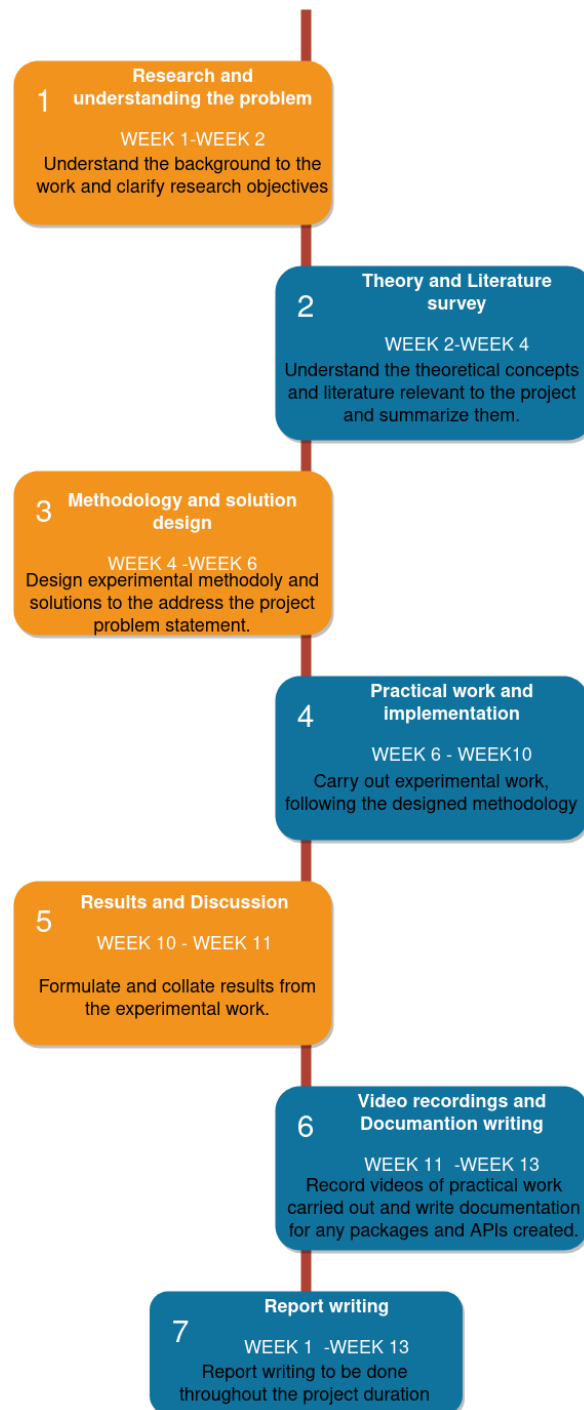


Figure 9.1: Plan of development

9.2 Literature review

Benefit	Service	Immediate advantage	Long-term impact
Health-related benefits	Comfort Monitor Consultancy Support deliver therapy	Care accessibility and availability Users' safety Social connectivity and communication Detection of life threatening events Reduction of medical errors	Promote well-being of ageing and vulnerable people
Environmental benefits	Monitor Consultancy Comfort	Reduce energy usage Feedback on consumption Suggestions how to use electricity efficiently	Environmental sustainability Reduction of carbon emissions
Financial benefit	Consultancy Monitor	Cheaper cost of virtual visits	Affordability of health care Sustainable consumption
Psychological wellbeing and social inclusion	Support	Entertainment Virtual interaction	Overcome the feeling of isolation

Figure 9.2: Smart home use cases [9]

9.3 Design

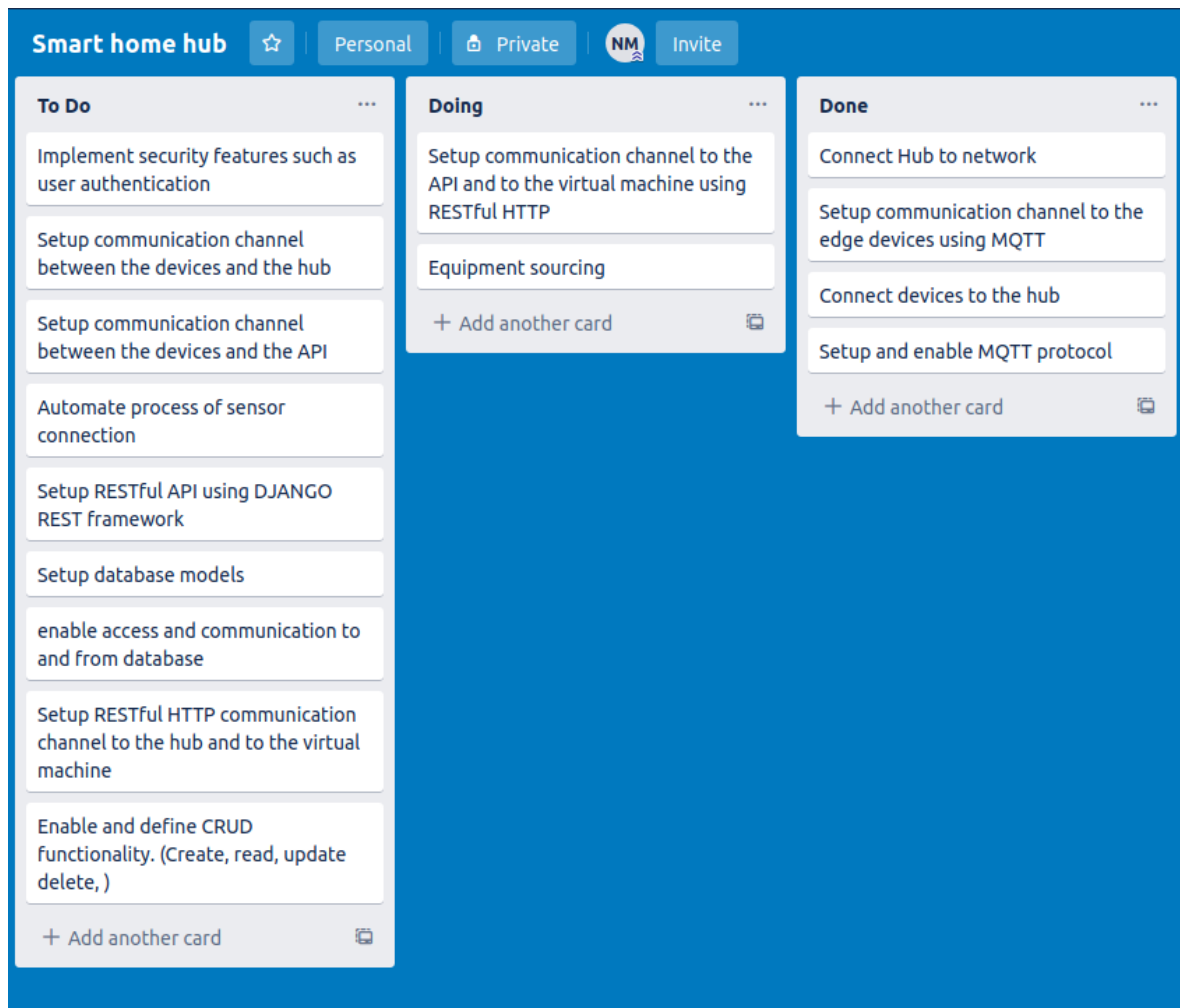


Figure 9.3: Trello dashboard for keeping track of progress

9.4 Implementation and experimental work

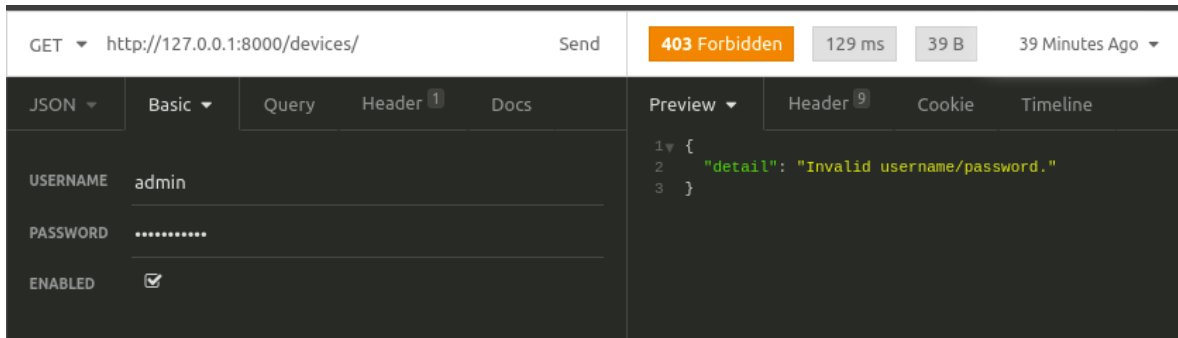


Figure 9.4: An http GET request to the API with incorrect authentication information provided returns an Error 403 Forbidden

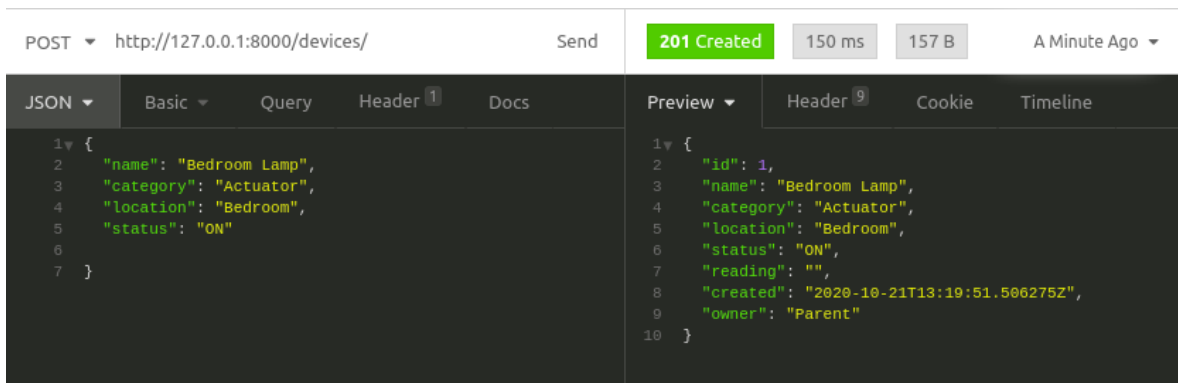


Figure 9.5: An http POST request to the API with correct authentication information provided returns response 201 Created

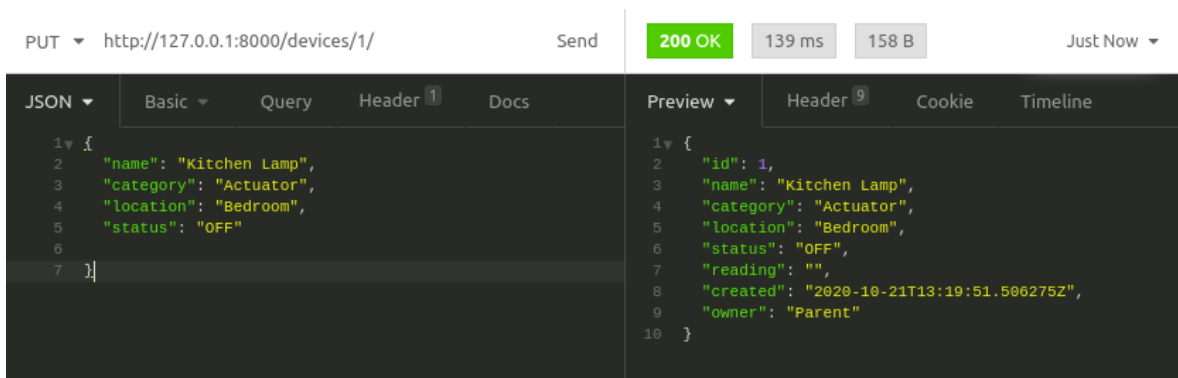


Figure 9.6: An http PUT request to the API with correct authentication information provided returns response 200 OK Note the name and status change

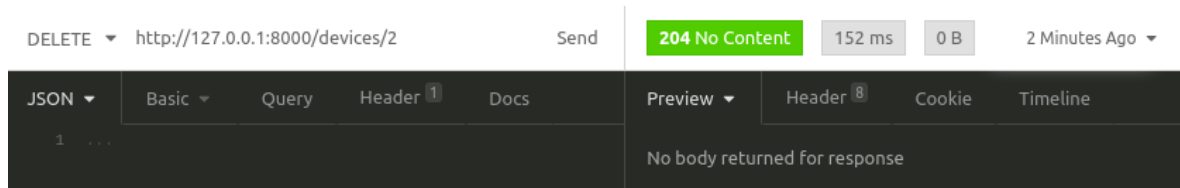


Figure 9.7: An http DELETE request to the API with correct authentication information provided returns response 204 No Content(This is a successful delete)

```

1  from homehubRpi.mqttCommunications import MQTTClass
2
3  mqttc = MQTTClass()
4  rc = mqttc.run(MQTT_BROKER="192.168.1.61",
5                MQTT_TOPIC="helloTopic", MQTT_PORT=1883)
6  # mqttc.pub("192.168.1.61", "helloTopic", "hello_message")
7
8  print("rc:"+str(rc))
9

```

Figure 9.8: Example MQTTClass usage- Each edge devices becomes an MQTTClass object to inherit communication capabilities. N.B homehubRpi is the name of API-1 from the design diagrams.

```

1  /*
2   * ESP8266 Soil moisture sensor example using mqtt
3   * Ngoni Mombeshora
4   */
5
6  #include <ESP8266WiFi.h> // Enables the ESP8266 to connect to the local network (via WiFi)
7  #include <PubSubClient.h> // Acces MQTT functionalities
8  #include <ArduinoJson.h> //json handler
9  //-----
10 int moisture_sensor_pin = A0;
11 float SensorValue=0;
12 const char* Status="ON";
13 float prevSensorValue = 0;
14 //-----
15 // WiFi
16 // Make sure to update this for your own WiFi network!
17 const char* ssid = "AstraEpic";
18 const char* wifi_password = "andrea2020";
19 //-----
20 // MQTT
21 // Make sure to update this for your own MQTT Broker!

```

```

22 const char* mqtt_server = "192.168.1.61";
23 const char* mqtt_topic = "1";
24 // The client id identifies the ESP8266 device.
25 const char* clientID = "192.168.137.15";
26 //-----
27 // Initialise the WiFi and MQTT Client objects
28 WiFiClient wifiClient;
29 PubSubClient client(mqtt_server, 1883, wifiClient); // 1883 is default mqtt port
30 //-----
31
32 void setup() {
33     // Begin Serial on 115200
34     // Remember to choose the correct Baudrate on the Serial monitor!
35     // This is just for debugging purposes
36     Serial.begin(115200);
37
38     Serial.print("Connecting to ");
39     Serial.println(ssid);
40
41     // Connect to the WiFi
42     WiFi.begin(ssid, wifi_password);
43
44     // Wait until the connection has been confirmed before continuing
45     while (WiFi.status() != WL_CONNECTED) {
46         delay(500);
47         Serial.print(".");
48     }
49
50     // Debugging - Output the IP Address of the ESP8266
51     Serial.println("WiFi connected");
52     Serial.print("IP address: ");
53     Serial.println(WiFi.localIP());
54
55     // Connect to MQTT Broker
56     // client.connect returns a boolean value to let us know if the connection was successful.
57     if (client.connect(clientID)) {
58         client.publish(mqtt_topic, "Test publish to broker...");
59         Serial.println("Connected to MQTT Broker!");
60     }
61     else {
62         Serial.println("Connection to MQTT Broker failed...");
63     }
64
65
66 }
67
68 void loop() {
69     // create json representation to forward to database

```

```

70 StaticJsonBuffer<500> JSONbuffer;
71 JsonObject& JSONNencoder = JSONbuffer.createObject();
72
73 JSONNencoder["name"] = "moisture sensor";
74 JSONNencoder["category"] = "Sensor";
75 JSONNencoder["location"] = "bedroom";
76 JSONNencoder["status"] = Status;
77 JSONNencoder["reading"] = analogRead(moisture_sensor_pin);
78
79 //only forward reading to databse when it has changed
80 if (getReading()==1){
81   char JSONmessageBuffer[300];
82   JSONNencoder.printTo(JSONmessageBuffer, sizeof(JSONmessageBuffer));
83   if (client.publish(mqtt_topic, JSONmessageBuffer)){
84     Serial.println("Forwarding reading to database");
85     Serial.println(JSONmessageBuffer);
86     delay(2200); //add delay to avoid sending a message whilst another is being sent.
87
88   }
89   // Again, client.publish will return a boolean value depending on whether it
90   //succeeded or not.
91   // If the message failed to send, we will try again, as the connection may have broken.
92   else {
93     Serial.println("Message failed to send. Reconnecting to MQTT Broker and trying again");
94     client.connect(clientID);
95     delay(10);
96     // This delay ensures that client.publish doesn't clash with the client.connect
97     //call client.publish(mqtt_topic, (JSONmessageBuffer));
98   }
99
100 }
101 else{
102   delay(2200);
103 exit;
104 }
105 }
106
107 int getReading(){
108   float value = analogRead(moisture_sensor_pin);
109   float diff = abs(value- prevSensorValue);
110
111   if (diff>40){
112     Serial.println("true");
113     Serial.println(analogRead(moisture_sensor_pin));
114
115     return 1;
116   }
117   else{

```

```
118     return 0;
119     Serial.println("false");
120
121 }
122 }
```

Listing 3: Example code for moisture sensor edge device

ETHICS APPLICATION FORM



Please Note:

Any person planning to undertake research in the Faculty of Engineering and the Built Environment (EBE) at the University of Cape Town is required to complete this form **before** collecting or analysing data. The objective of submitting this application *prior* to embarking on research is to ensure that the highest ethical standards in research, conducted under the auspices of the EBE Faculty, are met. Please ensure that you have read, and understood the **EBE Ethics in Research Handbook** (available from the UCT EBE, Research Ethics website) prior to completing this application form: <http://www.ebe.uct.ac.za/ebe/research/ethics1>

APPLICANT'S DETAILS		
Name of principal researcher, student or external applicant		Ngonidzashe Mombeshora (MMBNGO003)
Department		EBE- Mechatronics Engineering
Preferred email address of applicant:		
If Student	Your Degree: e.g., MSc, PhD, etc.	Bsc Mechatronics
	Credit Value of Research: e.g., 60/120/180/360 etc.	40
	Name of Supervisor (if supervised):	Jane Wingard
If this is a research contract, indicate the source of funding/sponsorship		
Project Title		Low Cost Smart Home Hub

I hereby undertake to carry out my research in such a way that:

- there is no apparent legal objection to the nature or the method of research; and
- the research will not compromise staff or students or the other responsibilities of the University;
- the stated objective will be achieved, and the findings will have a high degree of validity;
- limitations and alternative interpretations will be considered;
- the findings could be subject to peer review and publicly available; and
- I will comply with the conventions of copyright and avoid any practice that would constitute plagiarism.

APPLICATION BY	Full name	Signature	Date
Principal Researcher/ Student/External applicant	Ngonidzashe Mombeshora		17/08/2020
SUPPORTED BY	Full name	Signature	Date
Supervisor (where applicable)	Jane Wyngaard		17/08/2020

APPROVED BY	Full name	Signature	Date
HOD (or delegated nominee) Final authority for all applicants who have answered NO to all questions in Section 1; and for all Undergraduate research (Including Honours).			
Chair: Faculty EIR Committee For applicants other than undergraduate students who have answered YES to any of the questions in Section 1.			