

NOTTINGHAM TRENT UNIVERSITY
SCHOOL OF SCIENCE AND TECHNOLOGY

Context-Aware IoT Smart Plug

by

Ruairidh Taylor

in

2019

**Project report in part fulfilment
of the requirements for the degree of
Bachelor of Science with Honours
in
Computer Systems (Networks)**

I hereby declare that I am the sole author of this report. I authorize the Nottingham Trent University to lend this report to other institutions or individuals for the purpose of scholarly research.

I also authorize the Nottingham Trent University to reproduce this report by photocopying or by other means, in total or in part, at the request of other institutions or individuals for the purpose of scholarly research.

Signature

Ruairidh Taylor

ABSTRACT

This project attempts to implement context-aware functionality within a smart plug by utilising sensors and external web services to gather information about the plug's surroundings allowing it to react accordingly. The system is made up of three components, the Android application, a Firebase web service and the hardware being the plug itself. The contextual type focussed on within this project was temperature; however, as mentioned later in the report can be expanded to other contextual types.

The work introduces the three critical technologies utilised within this system; these are context-awareness, Internet of Things (IoT) and embedded systems. These technologies were explored, referencing work conducted by various published individuals and companies which are relevant to this project.

The use of context-awareness within this system is to gain knowledge of the plugs surroundings to decide if the plug should be on or off based on the device which is being used. This will help increase the user's comfort while minimising the user's interaction with the plug.

The development methodology which was adhered to during the implementation of the system was the Agile development methodology as it allows for quicker development and continuous improvement by gaining constant feedback and testing regularly.

The final system was relatively successful as almost all the core components had been successfully implemented while some of the other functions could not be fully implemented at the time. The only core component which was not implemented was linking the plugs to a user account which meant users could see other users' devices. Testing revealed a few smaller issues which were quickly fixed during the implementation stage.

Future work to be completed on the system had been introduced which discusses fixes for the missing functionality as well as additional features and further work to be completed on the system.

Lastly the Professional, Social, Ethical and Legal issues associated with the project and how they have been approached and then mitigated as to ensure the project being up to standard was discussed.

ACKNOWLEDGEMENTS

I would like to acknowledge everyone who has played a role in my academic accomplishments. First, and most of all, I would like to thank Dr Ahmed M. Elmesiry, for his guidance, assistance and knowledge throughout this year, as without you none of this would have been possible.

I would also like to thank the following:

My mum and dad, Jennifer Campbell and Chris Taylor, you have always been supportive of my choices and have always been there when I needed you.

My friends and colleagues for always being incredibly supportive and helpful; without you I would be in a completely different situation.

Thank you all.

TABLE OF CONTENTS

| | |
|---|------------|
| ABSTRACT..... | II |
| ACKNOWLEDGEMENTS | III |
| TABLE OF CONTENTS | IV |
| LIST OF FIGURES | XI |
| LIST OF TABLES | XV |
| CHAPTER 1..... | 1 |
| INTRODUCTION | 1 |
| 1.1 Introduction | 1 |
| 1.2 Project Goal..... | 1 |
| 1.3 System Testing | 2 |
| 1.4 Report Chapters | 2 |
| 1.4.1 Context | 2 |
| 1.4.2 New Ideas..... | 2 |
| 1.4.3 Implementation | 3 |
| 1.4.4 Results | 3 |
| 1.4.5 Conclusions and Future Work | 3 |
| CHAPTER 2..... | 4 |
| CONTEXT..... | 4 |
| 2.1 Introduction | 4 |
| 2.2 Embedded Systems..... | 4 |

| | |
|--|-----------|
| 2.2.1 Hardware..... | 5 |
| 2.3 Internet of Things..... | 9 |
| 2.3.1 Definition..... | 10 |
| 2.3.2 Market Growth | 10 |
| 2.3.3 Gartner’s Hype Cycle..... | 10 |
| 2.3.4. Internet of Things Stack | 11 |
| 2.3.5 Cloud services..... | 14 |
| 2.3.6 Technologies | 15 |
| 2.4. Context-awareness | 26 |
| 2.4.2 Context-aware services | 27 |
| 2.4.3 Context-aware embedded systems | 28 |
| 2.4.4 Context-aware IoT..... | 28 |
| 2.5 Survey of existing solutions..... | 29 |
| 2.6 Summary | 32 |
| CHAPTER 3..... | 33 |
| NEW IDEAS | 33 |
| 3.1 Introduction | 33 |
| 3.2 Current Issues..... | 33 |
| 3.3 Proposed system | 34 |
| 3.3.1 Functional requirements | 35 |

| | |
|---|-----------|
| 3.3.2 Non-functional requirements | 36 |
| 3.4 Justification..... | 37 |
| 3.5 Potential issues with the proposed system..... | 37 |
| 3.6 Hardware and Software required | 38 |
| 3.6.1 Hardware components..... | 38 |
| 3.6.2 Software required | 40 |
| 3.7 ID3 algorithm..... | 41 |
| 3.8 Product details | 42 |
| 3.8.1 Mobile Application | 42 |
| 3.8.2 Web Service..... | 46 |
| 3.8.3 Context-Aware Smart Plug | 47 |
| 3.9 Methodology..... | 53 |
| 3.10 Testing | 54 |
| 3.11 Project planning | 55 |
| 3.12 Summary | 56 |
| CHAPTER 4..... | 57 |
| IMPLEMENTATION OR INVESTIGATION | 57 |
| 4.1 Introduction | 57 |
| 4.2 Development Phase 1 | 58 |
| 4.2.1 Plugnet Application | 58 |

| | | |
|------------------|---|------------|
| 4.2.2 | Firebase Database | 67 |
| 4.2.3 | Context-Aware Smart Plug | 69 |
| 4.2.4 | User and System Testing | 74 |
| 4.3 | Development Phase 2 | 76 |
| 4.3.1 | Response to Testing | 76 |
| 4.3.2 | Plugnet Application | 81 |
| 4.3.3 | Firebase Database | 87 |
| 4.3.4 | Context-Aware Smart Plug | 87 |
| 4.3.4 | System and User Testing | 91 |
| 4.4 | Development Phase 3 | 93 |
| 4.4.1 | Response to Testing | 93 |
| 4.4.2 | Plugnet Application | 99 |
| 4.4.3 | Firebase Database | 100 |
| 4.4.4 | Context-Aware Smart Plug | 100 |
| 4.5 | Justification of Missing Functionality | 104 |
| 4.5.1 | User Accounts | 104 |
| 4.5.2 | Machine Learning | 104 |
| 4.6 | Summary | 105 |
| CHAPTER 5 | | 106 |

| | |
|---|------------|
| RESULTS / DISCUSSION | 106 |
| 5.1 Introduction | 106 |
| 5.2 Changes made to the algorithm..... | 106 |
| 5.3 Testing Environment | 107 |
| 5.4 System Performance | 107 |
| 5.4.1 Opening the application | 107 |
| 5.4.2 Updating Options | 108 |
| 5.5 System Functionality | 109 |
| 5.5.1 Application functionality | 110 |
| 5.5.2 Firebase web service functionality | 110 |
| 5.5.3 Plug functionality | 110 |
| 5.6 System Usability..... | 111 |
| 5.7 Alternative testing..... | 111 |
| 5.8 Costs | 112 |
| 5.9 Summary..... | 112 |
| CHAPTER 6..... | 113 |
| CONCLUSIONS / FUTURE WORK | 113 |
| 6.1 Introduction | 113 |
| 6.2 Delivery of Projects Aims and Objectives | 113 |
| 6.3 Overall Outcome..... | 115 |

| | |
|---|------------|
| 6.4 Future Work | 116 |
| 6.4.1 Switching Databases | 116 |
| 6.4.2 iOS Application | 116 |
| 6.4.3 Web Interface | 116 |
| 6.4.4 Scheduler | 117 |
| 6.4.5 Additional Sensors | 117 |
| 6.4.6 Additional APIs | 117 |
| 6.4.7 Amazon Alexa and Google Home..... | 118 |
| 6.4.8 Easy Set-Up | 118 |
| 6.5 Evaluation of Professional, Social, Ethical and Legal Issues | 118 |
| 6.5.1 Professional Issues..... | 118 |
| 6.5.2 Social Issues | 119 |
| 6.5.3 Ethical Issues | 120 |
| 6.5.4 Legal Issues | 120 |
| 6.6 Synoptic Assessment..... | 121 |
| REFERENCES | 123 |
| BIBLIOGRAPHY..... | 126 |
| APPENDIX A..... | 129 |
| PROJECT PLANNING DOCUMENT..... | 129 |
| APPENDIX B..... | 140 |

| | |
|---|------------|
| 1 UPDATED GANTT CHART | 140 |
| APPENDIX C..... | 141 |
| 1 MATERIAL DESIGN' DESIGN PRINCIPLES..... | 141 |
| APPENDIX D | 142 |
| APPLICATION FUNCTIONAL TESTING | 142 |
| FIREBASE WEB SERVICE FUNCTIONAL TESTING..... | 144 |
| PLUG FUNCTIONAL TESTING..... | 144 |
| APPENDIX E | 145 |
| 1 USABILITY TEST RESULTS | 145 |
| APPENDIX F | 150 |
| 1 USER MANUAL..... | 150 |

LIST OF FIGURES

| | |
|---|----|
| Figure 1 - Generic embedded system (Programming embedded systems, 2007) | 5 |
| Figure 2 - Common memory types in embedded systems (Barr, 2001)..... | 7 |
| Figure 3 - Hypes cycle for emerging technologies (Gartner, 2018) | 11 |
| Figure 4 - Constrained devices stack (eclipse, 2016) | 12 |
| Figure 5 - Gateways stack (eclipse, 2016)..... | 13 |
| Figure 6 - Cloud platforms stack (eclipse, 2016) | 14 |
| Figure 7 – RFID (Vaculík et al., 2013) | 16 |
| Figure 8 – Example of a Zigbee network (Tomar, 2011)..... | 18 |
| Figure 9 - An example of an IPv6 network with a 6LoWPAN mesh network (Olsson, 2014)..... | 19 |
| Figure 10 – A sample WirelessHART network (Nixon, 2012) | 19 |
| Figure 11 - A sample of an ISA100.11a network (Nixon, 2012)..... | 20 |
| Figure 12 - Rubee vs RFID (Wyld, 2008)..... | 22 |
| Figure 13 - Channels and centre frequencies for 2.4Ghz - (Gauthier, 2019) | 23 |
| Figure 14 - Channels and channel ranges 5Ghz – (Zak, 2019) | 24 |
| Figure 15 - How WiMAX works (Sran, 2009)..... | 25 |
| Figure 16 - Example WiMAX network (Linear, n.d.) | 26 |
| Figure 17 - TP-Link HS110 smart plug (TP-Link, 2019) | 29 |

| | |
|---|----|
| Figure 18 - Belkin WeMo Insight Switch (Belkin, 2019)..... | 29 |
| Figure 19 - Sonoff S20 Smart Socket (Sonoff, 2019)..... | 30 |
| Figure 20 - Eve Energy (Eve, 2019) | 30 |
| Figure 21 - Diagram of context-aware IoT smart plug prototype | 34 |
| Figure 22 - NodeMCU | 39 |
| Figure 23 - 5V 1-Channel Relay Board | 39 |
| Figure 24 - DHT11 Digital Temperature Humidity Sensor | 40 |
| Figure 25 - Initial proposed prototype of Plugnet application | 44 |
| Figure 26 - Top-Level diagram showing the process of the Context-aware smart plug..... | 49 |
| Figure 27 - Top-Level diagram of the default mode | 50 |
| Figure 28 - Top-Level diagram of Fan mode | 51 |
| Figure 29 - Top-Level diagram of radiator mode | 51 |
| Figure 30 - First implementation of Plugnet application..... | 59 |
| Figure 31 - Firebase Authentication console screen | 63 |
| Figure 32 - Firebase real-time database console screen | 68 |
| Figure 33 – NodeMCU Plugnet configuration | 70 |
| Figure 34 - Sign in and create account screens..... | 77 |
| Figure 35 - Device tiles application screen | 78 |
| Figure 36 - Navigation drawer updated..... | 78 |

| | |
|--|-----|
| Figure 37 - Application development (1/3) | 82 |
| Figure 38 - Application development (2/3) | 83 |
| Figure 39 - Application development (3/3) | 84 |
| Figure 40 - Password reset email | 86 |
| Figure 41 - Password reset link..... | 86 |
| Figure 42 - User testing home screen change | 94 |
| Figure 43 - User testing create account screen change | 95 |
| Figure 44 - Changes to the profile screen..... | 96 |
| Figure 45 – Changes to device screen | 97 |
| Figure 46 - Location screen | 99 |
| Figure 47 - Relay connected male and female plug connectors | 103 |
| Figure 48 - Time taken to open the application | 108 |
| Figure 49 - Usability Question 1..... | 145 |
| Figure 50 - Usability Question 2..... | 146 |
| Figure 51 - Usability Question 3..... | 147 |
| Figure 52 - Usability Question 4..... | 148 |
| Figure 53 - Usability Question 5..... | 149 |
| Figure 54 – Configuration of temperature sensor and relay module on the NodeMCU | 151 |
| Figure 55 - Arduino IDE configuration for NodeMCU | 151 |

| | |
|--|-----|
| Figure 56 - Firebase Arduino Fingerprint | 152 |
|--|-----|

LIST OF TABLES

| | |
|---|-----|
| Table 1 - Characteristics of the various memory types (Barr, 2001) | 8 |
| Table 2 - A survey of existing solutions..... | 31 |
| Table 3 - ID3 dataset of temperatures..... | 41 |
| Table 4 - Time taken to open the Plugnet mobile application | 108 |
| Table 5 - Results showing the time taken from user input to a change in the plug | 109 |

CHAPTER 1

INTRODUCTION

1.1 Introduction

The Internet of Things and smart devices are becoming more and more common, with everyday household appliances now including a smart feature. With these devices having a connection to the internet allows for better control, monitoring and analysis. These appliances are controlled with embedded systems; embedded systems are used in almost all devices that use electricity, from medical equipment to cars, embedded systems are designed to perform one or two functions such as opening or closing a valve to the dials on a car.

In a separate area of study, context-awareness is gathering information about the surrounding environment. Context-aware computing does this with the use of sensors and APIs, with the context gathered from sensors and external APIs the system can make decisions on what should happen.

The project is focussing on the three technologies as stated above to produce a context-aware IoT smart plug. This system would allow users to select a device that they have plugged in, using sensors and APIs to get the context of the surroundings would allow the plug to decide on if the plug should be on or off.

1.2 Project Goal

The goal of this project is to construct a context-aware smart plug, in which the user can select the device they have plugged in through a mobile application and with this information the plug can turn on and off based on the context.

Currently, there are no other smart plugs out there that use both APIs and sensors to gather information about the area in which it is located. Other systems that use context-awareness are relatively expensive compared to non-context-aware systems; therefore, the plug should be cheaper or the same price as other smart plugs.

1.3 System Testing

To test the system, the environment was left precisely the same, as this plug would be used for everyday home usage. The tests that were carried out included functional, user, and performance testing.

1.4 Report Chapters

This paper is split into six chapters; each of these chapters have been split further down into sections which focus on a specific area. The chapters are as follows (Excluding the introduction):

1.4.1 Context

The context chapter focuses on background knowledge of the subject areas which would be used to find the appropriate technologies that should be used for the implementation. The chapter also looked at existing solutions; these existing solutions would be then used to help guide the design of the system by focussing on the pros and cons of the other solutions. As no other context-aware smart plugs are currently available, the focus was on popular and unique smart plugs.

1.4.2 New Ideas

The new ideas chapter focuses on the planning and design of the proposed system. The chapter starts by discussing the current issues in which the system

is going to solve, followed by a plan of how the system is going to be created, each section of the system has been planned out separately. The plan for the mobile application is a paper prototype which shows the initial designs of the system, the plan for the plug was shown with data flow diagrams which show how it is going to operate.

1.4.3 Implementation

The implementation chapter focuses on the development of the proposed system, where the implementation was broken down into three development phases to follow the Agile methodology. These three phases have been further broken down to show how each of the essential functions that have been implemented. At the end of the chapter is a section justifying why features mentioned within the new ideas chapter were missing.

1.4.4 Results

The results chapter focuses on the results of the implemented system; the chapter discusses the different tests that have been completed as well as the results obtained through testing.

1.4.5 Conclusions and Future Work

The last chapter focuses on the overall outcome of the entire project, and how the system could be improved for future development. It discusses whether the original aims and objectives of the project have been met, and if not, why? It includes an investigation of the Professional, Social, Ethical and Legal issues which the project has faced.

CHAPTER 2

CONTEXT

2.1 Introduction

The Plugnet project is utilising three core technologies which are being used throughout all aspects of computing. These technologies are embedded systems, Internet of Things, and context-aware computing.

This chapter intends to identify and analyse current research in these fields and use this research to identify a new or relatively untouched area of study. The research will then be used to assist with the creation of a project. The plan is to have a plug that includes context-aware functionality, so the user has as little interaction with the plug as possible.

2.2 Embedded Systems

An embedded system is a “combination of computer hardware and software – and perhaps additional parts, either mechanical or electronic – designed to perform a dedicated function” (Barr, 2007). These systems include a piece of hardware, such as an electrical and mechanical component as well as a software component to control the hardware.

Embedded systems are being used in all most all electronic devices, such as toasters, mobile phones and even cars. The automotive industry is one of the largest consumers of embedded systems, within cars embedded systems are used for the airbag system, traction control and satellite navigation and a whole lot more. As with most technology, embedded systems were not quite as widespread in the past as they are now. According to a study by

MarketsandMarkets “The embedded systems market is expected to grow at a Compound Annual Growth Rate (CAGR) of 4.05% between 2017 and 2023 and is expected to be valued at USD 110.46 billion by 2023”. This estimate is based on the growth of automotive industry migrating to the Electronic Vehicle (EV) market, the growing market for wearable devices, and increased usage of embedded systems in smart appliances for smart homes.

2.2.1 Hardware

As embedded systems are generally only used for maybe one or two specific functions, there is not the need for the typical computer peripherals such as monitor, keyboard and mouse. As well as this they do not need a huge amount of storage, a powerful Central Processing Unit (CPU) or loads of Random-Access Memory (RAM). “This can make it possible to greatly reduce the complexity, size and cost and increase the robustness of embedded systems as compared with general-purpose systems” (The Linux Information Project, 2006). As an embedded system usually has the bare minimum it will have a very low power consumption “of the more than six billion microprocessors produced in 2002, more than 98 percent were used in embedded systems.” (The Linux Information Project, 2006).

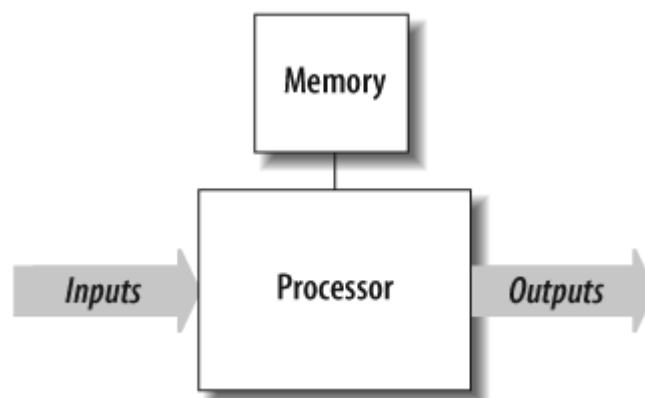


Figure 1 - Generic embedded system (Programming embedded systems, 2007)

An embedded system is made up of the following hardware:

Power Supply – The power supply is an essential part of the embedded system; embedded systems vary on power usage so may use between 1.8 – 5 Volts.

Processor - The processor is the core of an embedded system; it takes the inputs and produces outputs after processing the information. There are a few different types of processors:

- General Purpose Processor (GPP)
 - Microprocessor
 - Microcontroller
 - Embedded Processor
 - Digital Signal Processor
 - Media Processor
- Application Specific System Processor (ASSP)
- Application Specific Instruction Processors (ASIPs)

The two most used processors are the Microprocessor and the Microcontroller:

- Microprocessor – A microprocessor is a single Very Large-Scale Integration (VLSI) chip which includes a CPU. These chips may also include other units such as floating-point processing arithmetic unit, caches and pipelining units that assist with fast processing.
- Microcontroller – A microcontroller is a single-chip VLSI unit which has limited computational capabilities, however, has enhanced input/output capabilities and a number of on-chip functional units which is beneficial for use in embedded systems.

Memory – There are a few different types of memory used in embedded systems:

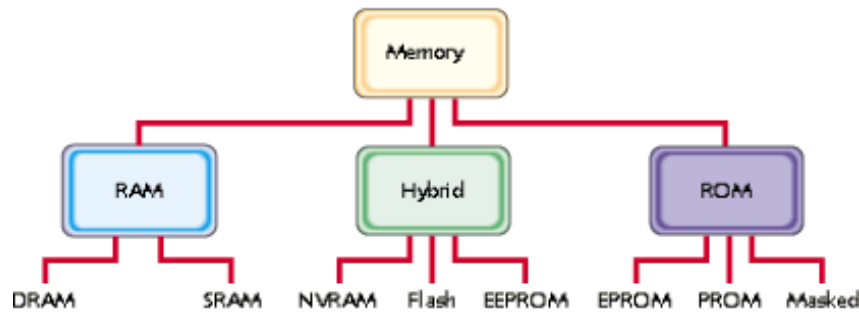


Figure 2 - Common memory types in embedded systems (Barr, 2001)

- Random Access Memory (RAM) – This type of memory is volatile meaning when power is removed it will lose its contents. There are two different types of RAM, Static RAM (SRAM) and Dynamic RAM (DRAM). “The primary difference between them is the lifetime of the data they store. SRAM retains its contents as long as electrical power is applied to the chip. If the power is turned off or lost temporarily, its contents will be lost forever. DRAM, on the other hand, has a concise data lifetime-typically about four milliseconds” (Barr, 2001).
- Read Only Memory (ROM) – This type of memory is non-volatile, so it does not lose its contents when powered off. There are three different types of ROM, Masked ROMs, Programmable ROM (PROM), Erasable and Programmable ROM (EPROM). Masked ROMs are where the contents of the chip must be specified when the chip is ordered; this is advantageous when bulk ordering chips. Programmable ROMs are chips that allow the user to program the chip themselves; however, it can only be programmed once, PROMs are known as one-time Programmable (OTP) devices. Erasable and programmable ROMs are like PROMs apart from it allows the user to erase and reprogram the chip.
- Hybrid RAM/ROM – “Hybrid memories can be read and written as desired, like RAM, but maintain their contents without electrical power, just like

ROM" (Barr, 2001). There are three different types of Hybrid memory; electrically-erasable-and-programmable ROM (EEPROM), Flash and non-volatile RAM (NVRAM). EEPROMs are like EPROMs however instead of removing the data by exposing it to a large amount of ultraviolet light it is removed via an electrical current; this is still not as fast as RAM so would not be used as main system memory. Flash memory combines the best of both RAM and ROM. Flash devices have fast reads, but not writes; they are low cost, high density and electrically reprogrammable. Flash is becoming the most used type of memory within embedded systems. Non-volatile RAM is SRAM with a battery backup.

Table 1 - Characteristics of the various memory types (Barr, 2001)

| Type | Volatile? | Writeable? | Erase Size | Max Erase Cycles | Cost (per Byte) | Speed |
|-------------------|------------------|--------------------------------|-------------------|-----------------------------|----------------------------|-----------------------------------|
| SRAM | Yes | Yes | Byte | Unlimited | Expensive | Fast |
| DRAM | Yes | Yes | Byte | Unlimited | Moderate | Moderate |
| Masked ROM | No | No | n/a | n/a | Inexpensive | Fast |
| PROM | No | Once, with a device programmer | n/a | n/a | Moderate | Fast |
| EPROM | No | Yes, with a device programmer | Entire Chip | Limited (consult datasheet) | Moderate | Fast |
| EEPROM | No | Yes | Byte | Limited (consult datasheet) | Expensive | Fast to read, slow to erase/write |
| Flash | No | Yes | Sector | Limited (consult datasheet) | Moderate | Fast to read, slow to erase/write |
| NVRAM | No | Yes | Byte | Unlimited | Expensive (SRAM + battery) | Fast |

Time-Counters – A counter is used to delay for a specific time interval without affecting the normal code execution.

Communication Ports – Embedded systems have different types of communication ports to communicate with other embedded devices; these usually depend on the size and make of the embedded system:

- Universal Asynchronous Receiver/Transmitter (UART)
- I²C (I2C)
- *Serial Peripheral Interface (SPI)*
- Controller Area Network (CAN)
- Universal Serial Bus (USB)
- Ethernet
- Recommended Standard 232 (RS-232)
- Recommended Standard 423 (RS-423)
- Recommended Standard 485 (RS-485)

Input and Output (I/O) – To get an output of an embedded system input is required; the input is generally a sensor of some sort. The processor is chosen based on the number of input/output devices.

Application specific circuits – Some hardware components are standard such as the (power supply, processor and memory) in embedded systems; however, some embedded systems have particular uses and require different hardware.

2.3 Internet of Things

“Since 2006, there have been more embedded systems, industrial equipment, sensor devices, instruments, meters, cameras, animals, and other objects connected to the Internet than humans using computers, smartphones, and other electronics for information technology” (IC Insights, 2014)

The Internet of Things (IoT), sometimes referred to as the Internet of Everything is the connection of a device which would not usually be connected to the internet being connected to the internet, these devices range from home

appliances to medical equipment. By connecting these new devices to the internet, it creates new opportunities to gain insights; this offers new ways to engage with consumers, drive efficiencies and develop new business.

2.3.1 Definition

There is not a single definition out there for IoT that has been adopted by all countries around the world; the best definition of the Internet of Things would be:

The International Telecommunication Union (ITU) defines the term Internet of Things as “a global infrastructure for the information society, enabling advanced services by interconnecting (physical and virtual) things based on existing and evolving interoperable information and communication technologies” (ITU, 2012).

This is because the International Telecommunication Union is an agency within the United Nations that has set up a Global Standards Initiative (GSI) in the hopes to bring a unified approach for the development of technical standards.

2.3.2 Market Growth

The Internet of Things market is growing massively with “Worldwide spending on the Internet of Things (IoT) is forecast to reach \$745 billion in 2019, an increase of 15.4% over the \$646 billion spent in 2018” (Framingham, Mass, 2019), to continue from this IDC also predicts that by 2022 there will be over \$1 trillion spent on IoT devices.

2.3.3 Gartner’s Hype Cycle

Gartner’s hype cycle for emerging technologies shows a graphical representation of the current stage of different technologies. These start from concept ideas

and continue to maturity. The graph Y-axis shows expectations and the X-axis shows time. IoT has been identified to be within the 'Peak of inflated expectations' this means the technology has been implemented and there is a lot of publicity about it.

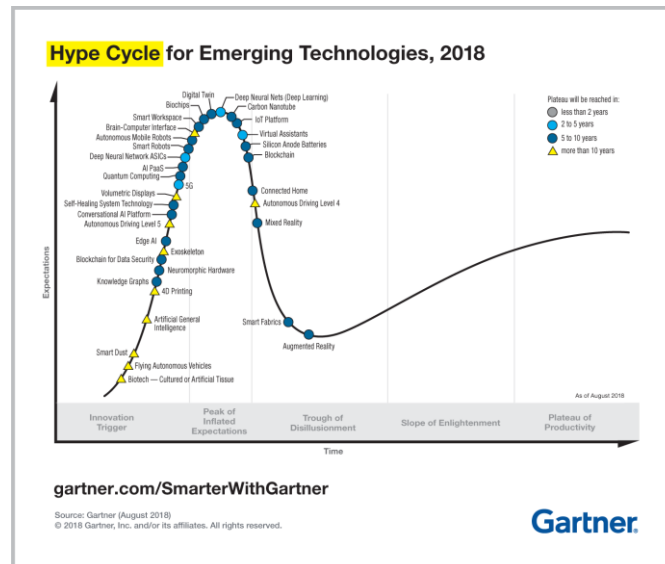


Figure 3 - Hypes cycle for emerging technologies (Gartner, 2018)

2.3.4. Internet of Things Stack

A technology stack is a combination of software and programming languages used to make a piece of software. The Internet of Things stack is continuously being revised, and people have different opinions on what the IoT stack is. Therefore, it was decided that the eclipse' white papers called "The Three Software Stacks Required for IoT Architectures" were to be used where they looked at the new technology requirements and architectures required for IoT solutions. The paper identified three stacks of software needed by any IoT solution:

1. Stack for Constrained Devices: Sensors and Actuators -

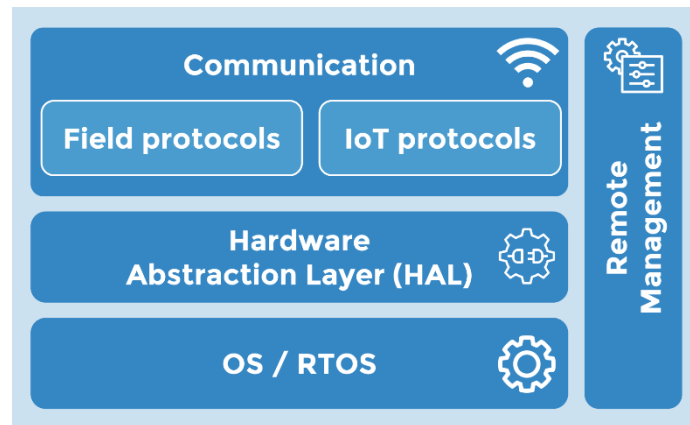


Figure 4 - Constrained devices stack (eclipse, 2016)

- Operating System/ Real-Time Operating System (OS/RTOS) – Most devices will run with a 'bare metal', however, some will have an embedded or RT operating systems that are well suited for small, constrained devices which can provide IoT-specific functionality.
- Hardware Abstraction Layer – This is a software layer that allows access to the hardware features of the microcontrollers (MCU), such as flash memory, serial interfaces, GPIOs, etc.
- Communication – Drivers and protocols are allowing the device to connect to a wired or wireless protocol like Wi-Fi, RFID, Zwave, Bluetooth, etc.
- Remote Management – To be able to access the device remotely allowing the device to be updated and monitored.

2. Stack for Gateways: Connected and Smart Things –

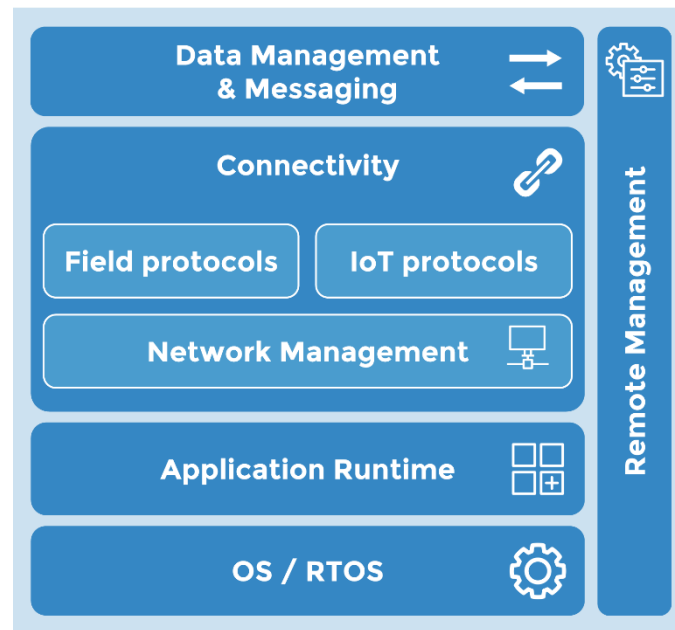


Figure 5 - Gateways stack (eclipse, 2016)

- Operating System – Usually a general-purpose operating system like Linux.
- Application Runtime/ Runtime Environment – IoT gateways can often run application code and allow for the applications to be dynamically updated. E.g. a gateway might have the support for Python, Node.js or Java.
- Data Management & Messaging – Locally support network latency, offline mode and real-time analytics, as well as having the ability to forward information to an IoT platform consistently.
- Remote Management – To be able to remotely configure, provision, start-up/shutdown gateways as well as the applications that are running on these gateways.

3. Stack for IoT Cloud Platforms -

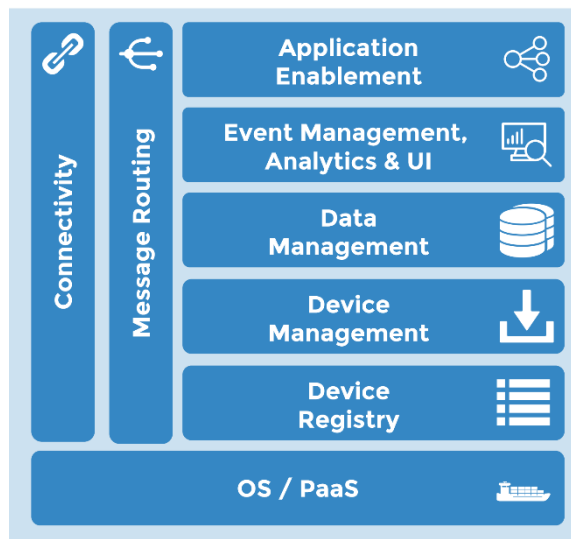


Figure 6 - Cloud platforms stack (eclipse, 2016)

- Connectivity & Message Routing – IoT platforms need to be able to interact with many devices and gateways using a variety of different protocols and data formats, but then normalise it for easier integration to the rest of an enterprise.
- Device Management and Device Registry – A central registry that is used to identify devices/gateways running in an IoT solution and then provision new software updates and manage the devices.
- Data Management and Storage – An elastic data store that supports the volume and variety of IoT data.
- Event Management, Analytics and UI – The ability to consolidate and analyse data, and create reports, graphs and dashboards.
- Application Enablement – The ability to create reports, graphs, dashboards and to use API for application integration.

2.3.5 Cloud services

Microsoft defines cloud computing as the delivery of computing services over the internet. These services range from infrastructure such as servers, storage and networks to software such as Office 365. There are several benefits to using

cloud computing instead of creating your own infrastructure; the main is cost due to not having to purchase equipment as well as the cooling and electricity, this is all cheaper for cloud providers due to economies of scale and elasticity allowing for an infrastructure to grow and shrink as required.

These same benefits also affect IoT systems Hou et al. mentioned that the millions of IoT devices connecting to the internet could generate vast amounts of data, and these devices have minimal capabilities due to their small physical size and energy consumption. Therefore, the cloud is critical to help support the requirements of the millions of IoT devices and provide new and exciting IoT applications for the end-users.

2.3.6 Technologies

The term "Internet of Things" was coined by Kevin Ashton during his time working at the Auto-ID Lab at MIT in 1999, this was to describe a network of objects connected to the internet by Radio Frequency Identification (RFID). However, even though the first definition of the term IoT was related to RFID, other technologies are now being used such as NFC, ZigBee, 6LowPAN, WirelessHART, ISA100.11a, UWB, Rubee, Bluetooth low energy, Wi-Fi, Wi-Max and Zwave.

- Radio Frequency Identification (RFID) – RFID enables identification from a short-range which is typically less than 1 meter, longer ranges which are significantly longer than 1 meter are known as long-range systems. RFID covers a broad range of frequencies between 135kHz and 5.8GHz. RFID systems that operate around 1 cm are known as close-coupling systems; these systems are coupled using both electric and magnetic fields. There are two distinct categories:
 1. Passive - Passive transponders do not have a power supply and require power from an external source such as the RFID reader.

Passive tags contain an antenna that captures energy from the reader which allows the chip to transfer the tag's ID.

2. Active - Active transponders have their energy supply for example in the form of a battery. The magnetic or electromagnetic field received by the reader is not required for powering the chip.

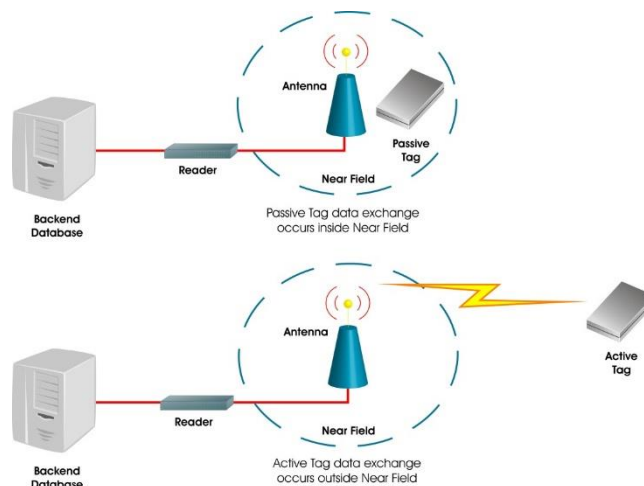


Figure 7 – RFID (Vaculík et al., 2013)

- Near Field Communication (NFC) - NFC) allows data transmission over a short distance which is usually around 20cm using alternating magnetic fields in the frequency range of 13.56MHz. For communication to happen between 2 NFC interfaces, there needs to be a master device and a slave device. The NFC initiator starts communication. Like RFID there are also two categories:
 1. Active - Active mode is where both devices are generating their own RF field. This allows communication in both directions.
 2. Passive – Passive mode is where only one NFC device is generating an RF field to communicate, this device is usually the reader, and the other passive device is called the tag.
- Zigbee – A book by Farahani states that Zigbee is a standard that defines a set of communication protocols for low data rate short-range wireless networking. For communication, Zigbee uses the IEEE 802.15.4 network specification which allows it to communicate with other devices that also

use this specification more efficiently. Zigbee operates within 868MHz, 915MHz and 2.4Ghz frequency bands and has a maximum transfer rate of 250Kb/s. In a paper written by Chellappa et al. states that there are three types of nodes in a Zigbee system:

- Coordinator – The coordinator forms the root of the network tree and can bridge to other networks. There is precisely one coordinator in each network. Its role is to initiate the network by selecting the network parameters such as the unique network identifier, frequency channel and setting other operations parameters. The coordinator can also store information about the network's security keys.
- Router – The router acts as intermediate nodes, relaying information from other nodes. The router can connect to an already existing network; it can also accept connections from other devices and retransmit the data across the network
- End Devices – End devices can be battery-powered or low-power devices. They can collect information from sensors and switches. They can pass information to their parents (either a coordinator or a router) but cannot relay information to other devices. As the end devices do not have much functionality, it makes it a lot cheaper to produce. The machines do not have to be awake all the time while the router and coordinator do. Each of the end devices can have as much as 240 end nodes which are separate applications sharing the same radio.

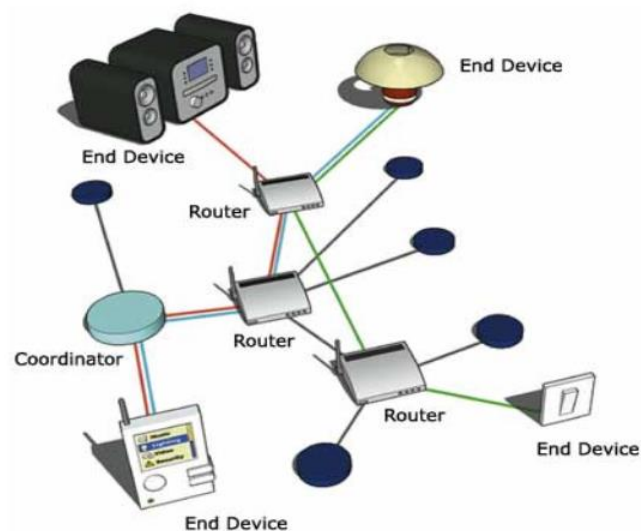


Figure 8 – Example of a Zigbee network (Tomar, 2011)

- IPv6 over Low-Power Wireless Personal Area Networks (6LoWPAN) – 6LoWPAN is a technology in which IPv6 packets can be carried efficiently within small link layer frames, like those defined by IEEE 802.15.4. Figure 9 shows an example of an IPv6 network with a 6LoWPAN mesh network. Connection to the internet is handled by an Access Point (AP) acting as an IPv6 router. In this example, there are several different devices connected to the AP such as the PC and servers. The 6LoWPAN network is connected to the network using an edge router. The edge router handles three actions:
 1. The exchange of information between the internet/intranet and 6LoWPAN devices.
 2. Local exchange of information between devices inside of the 6LoWPAN.
 3. Generation as well as maintenance of the 6LoWPAN network.

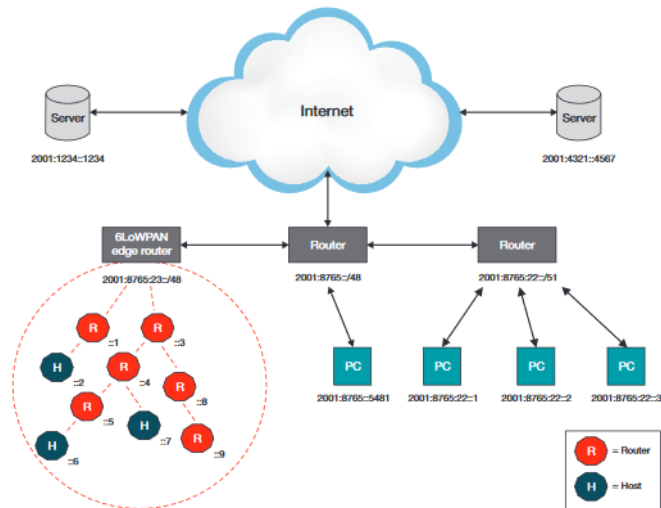


Figure 9 - An example of an IPv6 network with a 6LoWPAN mesh network (Olsson, 2014)

- Wireless Highway Addressable Transfer Protocol (WirelessHART) – WirelessHART is based on the HART communication protocol. A paper by Song et al. states that WirelessHART is a secure networking protocol which operates within the 2.4GHz radio band. The physical layer of WirelessHART utilises the IEEE 802.15.4 network specification. The WirelessHART network layer supports mesh networking technology, where a central network manager is responsible for the configuration of the network and scheduling of all activities happening on the network.

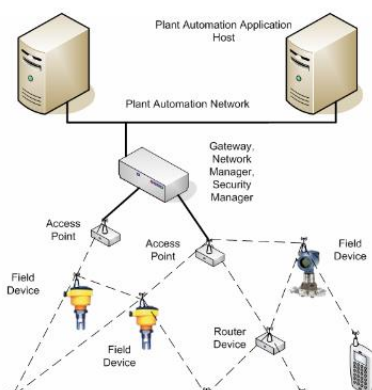


Figure 10 – A sample WirelessHART network (Nixon, 2012)

Figure 10 shows the underlying network devices types which include:

1. Field devices - Perform field sensing or actuating functions.
 2. Routers – All devices must have the ability to route packets within the wireless mesh.
 3. Adapters – Bind wired HART devices into the wireless mesh.
 4. Access Points – Connect the wireless mesh to the gateway.
 5. Handheld devices – Carried by mobile users.
 6. Simple/Redundant gateway – Bridge the host applications.
 7. Network manager – Reside in the gateway devices or be separate from the gateway.
 8. Security manager – Reside in the gateway devices or be separate from the gateway.
- International Society of Automation 100.11a (ISA100.11a) – ISA100.11a defines the protocol stack, system security and system management for use over low-powered, low-rate wireless networks (IEEE 802.15.4). However, the network and transport layers are based on 6LoWPAN, IPv6 and UDP standards. A sample of an ISA100.11a network is shown in Figure 11 –

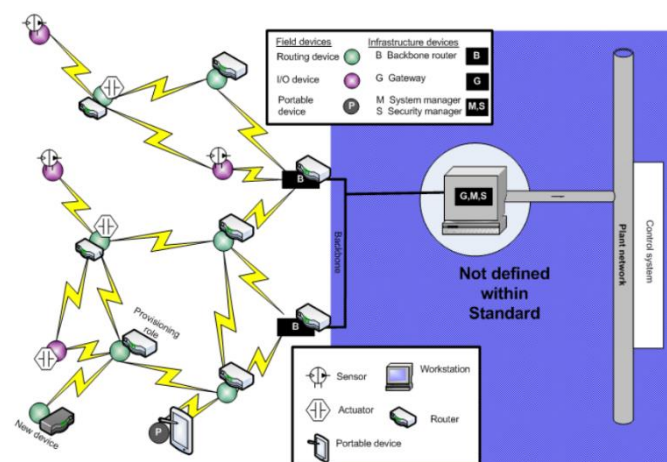


Figure 11 - A sample of an ISA100.11a network (Nixon, 2012)

According to ISA100, there are several roles within an ISA100.11a network:

1. Input/output – Obtains or consumes data (Does not route).
 2. Router – Routes information for other devices operating within the wireless subnet.
 3. Backbone Router – Routes information via the backbone. Mitigates between devices operating in the wireless subnet and devices operating on the backbone.
 4. System Manager – Manages all devices on the network via policy-controlled configurations which are based on the collection of performance parameters reported.
 5. Security Manager – Controls, enables and supervises the secure operation of every device which is present in the network.
 6. Gateway – Provides an interface between the wireless network and in this example the plant network.
 7. System Time Source – Responsible for maintaining the master time source of the network.
- Ultra-Wide Band (UWB) – As the name suggests UWB is a form of transmission that occupies a broad bandwidth, this is typically many Gigahertz (3.1GHz – 10.6GHz), this allows it to have data rates of Gigabits per second. UWB is different from conventional radio frequency and spread spectrum technologies such as Bluetooth and Wi-Fi as stated by Intel as a UWB transmitter works by sending billions of pulses across a broad spectrum of frequencies. The receiver then translates the pulses into data by listening for a familiar pulse sequence sent by the transmitter. In a study by Fernandes et al. states that there are two very different technologies being developed for UWB:

1. Impulse Radio Ultra-Wide Band (IR-UWB) – IR-UWB has been designed with lower complexity and low power consumption compared to MB-OFDM. They are therefore used in energy constrained, short-range wireless applications such as Personal Area Networks (PANs), low-power sensor networks and medical body area network. Because of the broad bandwidths that can be

obtained with UWB radios, they can be utilised for high data rate communication and precise location systems.

2. Multi-Band Orthogonal Frequency-Division Multiplexing Ultra-Wide Band (MB-OFDM UWB) - MB-OFDM has been mostly used in streaming and wireless USB applications with data rates of 480Mb/s. Because of the high-performance hardware required to operate an MB-OFDM UWB radio, these systems will tend to use a lot more power in comparison to IR-UWB.
- RuBee (IEEE 1902.1) – In a conference paper by Dantas et al. RuBee is an updated version of RFID as RuBee operates on a lower frequency radio wave range of 30-900 kHz which allows it to penetrate more objects. Rubee is a two-way wireless protocol which uses long -wave magnetic signals generated by magnetic induction. It is currently being used for data-tracking by using integrated sensors; these sensors do not need line of sight as magnetic field lines are curved in near field as well as being able to penetrate some materials at low frequencies. Although RuBee and RFID are being compared to one another, they are opposite to each other as Rubee almost exclusively utilises magnetic energy, rather than electrical or radio frequencies as shown in Figure 12.

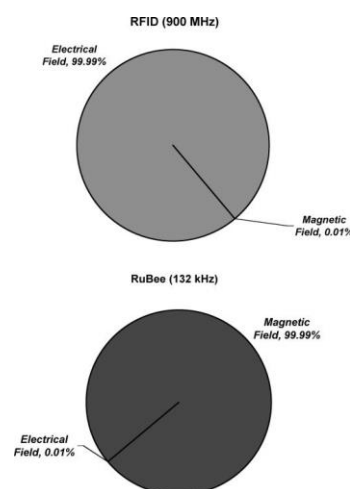


Figure 12 - Rubee vs RFID (Wyld, 2008)

- Bluetooth Low Energy (BLE) – BLE is a low power, short-range radio frequency standard which was developed by Bluetooth Special Interest Group (SIG). This standard was created to facilitate applications which make use of the ultra-low power devices. These devices are usually too small to bear the power consumption as well as costs associated with standard Bluetooth. In a paper by Cao et al. BLE is increasing the lifetime of a system in 3 ways:
 1. Simple device recovery.
 2. Power efficient peak, average and idle modes.
 3. Reliable transmission of data.
- Wireless Fidelity (Wi-Fi) – Wi-Fi is a technology within the Wireless Local Area Network (WLAN) standard IEEE 802.11. The 802.11 standards are continuously growing and evolving over the years as new advancements are made, each new advancement is identified by a new 1- or 2-digit suffix at the end of 802.11, for example, 802.11ac, each now suffix differs in frequency, speed and range. The current frequency bands used within 802.11 include 2.3GHz, 5Ghz and up to 60GHz in 802.11ad.
 1. Wireless frequency 2.4Ghz - In the UK there are 13 channels designated for the 2.4Ghz range, the number of channels will differ from country to country. The spectrum is evenly divided into channels, each channel being 22Mhz wide and have a 5Mhz gap separating the beginning of each channel, channels 1, 6 and 11 do not overlap as shown below in Figure 13 –

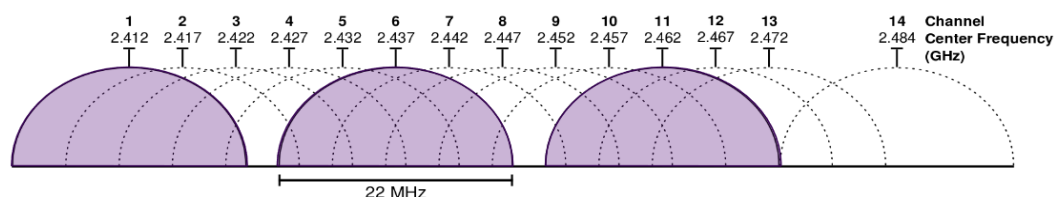


Figure 13 - Channels and centre frequencies for 2.4Ghz - (Gauthier, 2019)

2. Wireless frequency 5Ghz - Like 2.4Ghz, 5Ghz uses channels that are 20Mhz wide; however, unlike 2.4Ghz the channels do not overlap unless changing the channel width to 40/80Mhz. The advantage the non-overlapping provides is not being interfered by saturation. The main difference between 2.4Ghz and 5Ghz is that 2.4Ghz works at longer ranges because longer waves travel further and 5Ghz carries more data as shorter waves carry more data. Information about the channels can be seen in figure 14 -

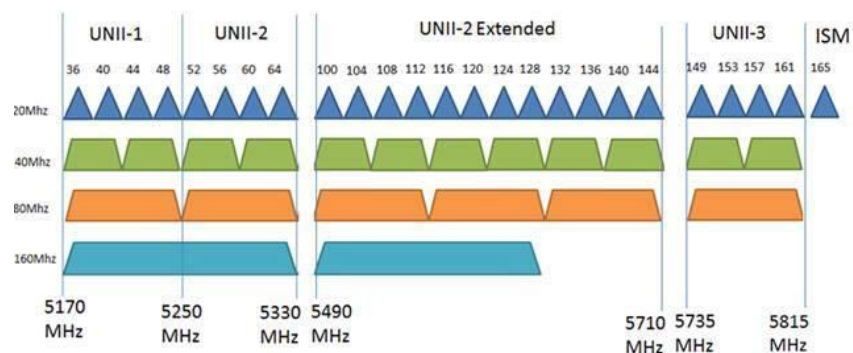


Figure 14 - Channels and channel ranges 5Ghz – (Zak, 2019)

“When electromagnetic waves go through a material they generally get weakened or dampened. How much they lose in power will depend on their frequency and of course the material”
(Maketecheasier, 2019)

- Worldwide Interoperability for Microwave Access (WiMAX) – WiMAX is based on the 802.16 air interface standard. “IEEE 802.16 is the set of standards governing the design of the wireless interface for a standard-based Metropolitan Area Network (MAN)” (Sran, 2009). WiMAX is set to provide higher speed internet, more considerable distances and a more significant number of users than other technologies such as Wi-Fi. A WiMAX system consists of two parts:

1. WiMAX Tower – The tower has a similar concept to a cell-phone tower; it can provide coverage to an area as large as 8,000 square km.
2. WiMAX Receiver – The receiver and antenna could be a small box or PC memory card or built into devices the same way Wi-Fi is. An example is shown in Figure 15 below.

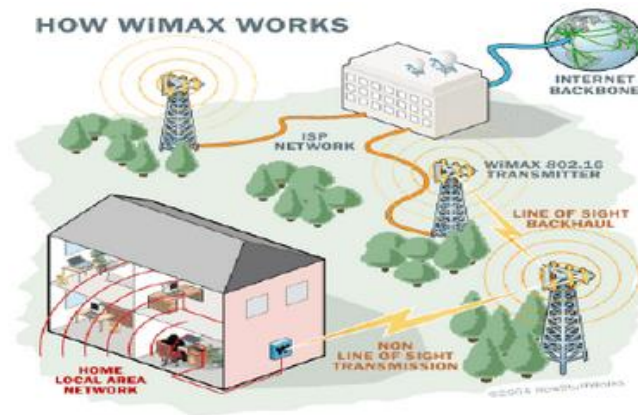


Figure 15 - How WiMAX works (Sran, 2009)

- Z-wave – Z-wave is a low powered wireless communication protocol which uses low-energy radio waves, typically around 900MHz. Z-wave is generally used in home automation for wireless control of IoT devices. Most Z-wave equipment can act as a wireless repeater for other Z-wave devices which is known as a 'mesh network', each of the nodes has a range of around 30 meters. There are two main components:
 1. Controllers – All Z-wave networks are required to have at least one controller. The first controller added to the network is known as the primary controller and is responsible for including and excluding nodes, assigning IDs to the nodes and maintaining a routing table.

2. Slaves – Slaves are nodes such as switches, lights etc. Nodes can be either connected to the mains or battery powered. Nodes that are connected to the mains can be used as repeaters. An example of a WiMAX network is shown below in Figure 16.

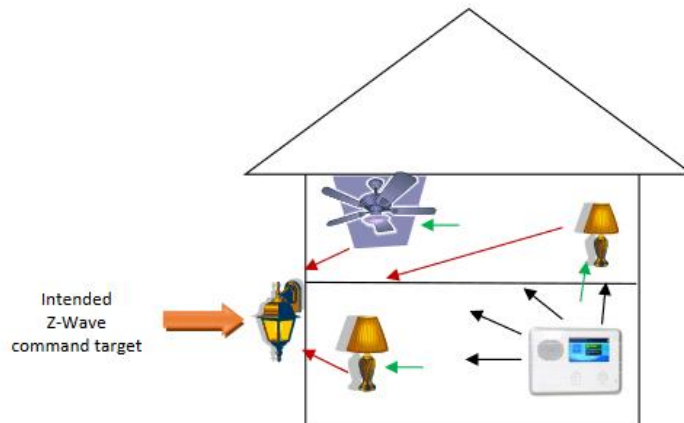


Figure 16 - Example WiMAX network (Linear, n.d.)

2.4. Context-awareness

The term 'context aware' is the ability to gather information about the surrounding environment; this is done using sensors; the system will then process this information and then adapt. According to a paper Schilit et al., context-aware computing is the ability for an application of a mobile user to collect data and then react to the changes of the environment the user is in.

Contextual type refers to the category of context such as time, temperature, speed, location, device, identity, role, activity, process, nearby users. Contextual value is the raw data gathered by sensors, and so the unit will depend on the type of data and sensor, for example, Degrees Celsius and longitude/latitude. Three approaches are traditionally used to develop context-aware applications (Hu et al., 2008):

1. No application-level context model - The application does all the actions, such as communicating with sensors, pre-processes the raw data, evaluates the data and then makes decisions on how to adapt.
2. Implicit context model – The applications are developed with libraries and toolkits to assist with the processing of the context information and gathering and pre-processing the data.
3. Explicit context model – The applications use a context management infrastructure or middleware solution which means pre-processing, storing and reasoning lie outside the application limits.

2.4.2 Context-aware services

Each definition will vary from source to source, techopedia defines it as; “A Web service is a software service used to communicate between two devices on a network” (techopedia, 2019) and IBM defines it as something similar “A web service is a generic term for an interoperable machine-to-machine software function that is hosted at a network addressable location” (IBM, 2019). A web service will include a service provider and a client. Web services feature language transparency, and therefore it does not matter what language the underlying system is written in. For example, an application that is written in a different language can still access the same backend of an application written in another language. This is done using technologies such as SOAP, WSDL XML etc.

Web services allow the integration and interoperability of services provided by different organisations. Therefore, instead of creating large applications, we can now build applications that reuse other services produced by other organisations. An issue when using other organisations services is the fact; they are not aware of the context associated with the application. Being context-aware in this situation allows for the software to be able to improve the response to the use of the software.

A study into context-aware web service systems by Truong, et al. found that It is hard to find a genuinely context-aware web service-based system that's interoperable and secure, which operates across multiple organisations.

2.4.3 Context-aware embedded systems

As embedded systems are getting smaller and more powerful, they are being used a lot more within context-aware computing due to their low costs and low power consumption. To get the contextual value embedded systems will have sensors attached which can gather information about its environment. The raw data collected by these sensors are then processed by the embedded system which then provides an output based on this data.

In a paper written by MIT project Oxygen, they state that the future of computing is human-centred completely mobile. With the availability of embedded systems within the environment, we will not have to carry devices around. "Context awareness is directed effort towards understanding the environment around the user with the help of smart devices embedded within its environment and improve the end user's experience by using this knowledge" (Daftari et al., 2003).

2.4.4 Context-aware IoT

Context-aware IoT is having a self-aware environment using sensors, software and internet connectivity. Using sensors, various objects in the environment can be monitored; network connectivity allows the data to be shared between devices. This allows for increased safety, efficiency and economic benefit for the environment (Carleton University, 2019).

2.5 Survey of existing solutions

Once a decision to create a context-aware IOT smart plug had been chosen, it was necessary to research products that already exist. When creating a new product, it is good to look at what already exists so that they can be developed and improved. Looking at other products showed a clear pattern in what makes a smart plug suitable and what should be avoided. The smart plugs that had been investigated were:

Product 1 – TP-Link HS110 Smart Plug

The first existing product that was looked at was the HS110 smart plug from TP-Link. Like a lot of smart plugs, it is Wi-Fi enabled allowing the user to connect to it over the internet; it enables users to monitor electricity consumption and schedule when the plug is powered on or off all from its Kasa phone application.



Figure 17 - TP-Link HS110 smart plug (TP-Link, 2019)

The Kasa application allows multiple devices to be controlled and monitored from it, including monitoring power consumption from each device and setting individual schedules. As well as being able to control it through the application it can also be controlled via Google Assistant or Amazon Alexa.



Figure 18 - Belkin WeMo Insight Switch (Belkin, 2019)

Product 2 – Belkin WeMo Insight Switch

The next product that was investigated was the WeMo Insight Switch from Belkin. This plug is very similar to the HS110 Smart Plug and contains a lot of the same features such as being able to monitor electricity consumption, create schedules and voice control with Amazon Alexa. On the other hand, it works with IFTTT which adds a lot more

functionality into the plug. However, it is missing out on some features such as it does not work with Google Assistant who is more widely available thanks to Android smartphones.

Product 3 – Sonoff S20 Smart Socket

The next product that was looked at was the S20 Smart Socket by Sonoff. This is also like the other two previously mentioned; nonetheless, it is a lot cheaper, so it does not contain as many features. The plug still has the ability to schedule when the device is to be turned on or off; it still has voice control with Amazon Alexa and Google assistant



Figure 19 - Sonoff S20 Smart Socket (Sonoff, 2019)

however it does not have the ability to see the energy consumption of the device.



Figure 20 - Eve Energy (Eve, 2019)

Product 4 – Eve Energy

The final product that was investigated was Eve Energy by Eve. This smart plug is unique compared to the other smart plugs that were looked at as instead of being Wi-Fi connected it used Bluetooth instead. However, it still contained the same core features that all the other plugs included such as the scheduling, checking power consumption and voice control however instead of Google Assistant or Amazon Echo it uses Siri as it is a part of the Apple ecosystem. This means instead of being able to control the plug from anywhere with only a phone; it requires the user to have an Apple HomePod to be able to control it remotely.

Table 2 - A survey of existing solutions

| Name of product | Features | Strengths | Weaknesses |
|-------------------------------------|--|--|--|
| TP-Link HS110 Smart Plug | <ul style="list-style-type: none"> - Controlled by the user's phone. - Scheduling when the plug is powered on or off. - Monitor electricity consumption of the device. - Voice control with Google Assistant or Amazon Alexa. | <ul style="list-style-type: none"> - Simple set up. - Allows multiple devices to be connected to the application. - Easy to use with voice control. - Electricity consumption monitoring. - Connect to the device from anywhere with an internet connection. - Easy to use Kasa application. | <ul style="list-style-type: none"> - Requires the user to have a smartphone with Android or IOS. - Expensive, especially when buying many of them. - Need to have an internet connection to monitor them. - Bulky. |
| Belkin WeMo Insight Switch | <ul style="list-style-type: none"> - Controlled by the user's phone via their application. - Create rules, schedules and get notifications. - Gain insight into energy usage. | <ul style="list-style-type: none"> - Easy to use and setup. - The application can handle multiple WeMo products. - Insight into home energy usage. - Connect to the device from anywhere providing you have an internet connection. - Works with Amazon Alexa. - Works with IFTTT. | <ul style="list-style-type: none"> - Must have an android or apple device. - Very expensive in comparison to other devices. - Indoor use only. - Bulky design. |
| Sonoff S20 Smart Socket | <ul style="list-style-type: none"> - Controlled by the user's phone. - Scheduling when the plug is powered on or off. - Set timers to only have the plug turned on for a certain amount of time. - Voice control with Amazon Alexa and Google Assistant. | <ul style="list-style-type: none"> - Inexpensive compared to other devices. - Voice control. - A timer feature that can be used to save energy. - Connect to the device from anywhere providing the user has an internet connection. | <ul style="list-style-type: none"> - Internet connection is required to change any timers that may be running. - Requires the user to have an Android or IOS smart device. |

| | | | |
|------------|--|--|---|
| Eve Energy | <ul style="list-style-type: none"> - Voice control via Siri. - Built-in schedules. - Allows the user to view their power consumption - Create scenes that allow the user to control what plugs are on/off. | <ul style="list-style-type: none"> - Works with Apples HomeKit which is easy to use and includes advanced security. - Built-in schedules. - See how much energy the connected devices are using. - Easy setup with Bluetooth. - Compact design. | <ul style="list-style-type: none"> - Only works with Apple iOS and tvOS. - Bluetooth only, no Wi-Fi. - Requires the user to be a part of the Apple ecosystem. - Expensive in comparison to other devices. |
|------------|--|--|---|

2.6 Summary

It is crucial then, to consider all factors that go into the process of creating a context-aware IoT smart plug. From the literature reviewed there is a clear trend in devices moving towards the internet of everything.

Although each of the products evaluated contains some functionality which is planned to be implemented into the Plugnet project, none include any context-aware features. All these devices require some form of user interaction to turn them on or off, whether this is manually or using a timer. Plugnet aims to use embedded systems, IoT and context-awareness to produce a smarter smart plug which reacts to changes in the environment to limit the user's interaction.

CHAPTER 3

NEW IDEAS

3.1 Introduction

The purpose of this project is to create an inexpensive context-aware IoT smart plug. The use of context awareness within the plug will maximise the user's comfort while minimising the user's interaction with the plug.

3.2 Current Issues

The initial concept of the project was the author's idea to stop devices from 'trickle charging' when plugged into an outlet. After more research into the topic of context-awareness within IoT devices, the author found a gap in the market where the plug could be used for more than just turning off a device when it had reached full charge. After looking at the existing solutions, it was clear that there are no similar context-aware products out there that are commercially available which led to the author to propose this system as a proof of concept. With other IoT devices that do utilise context-awareness, there is a considerable price premium that comes with the device compared to similar products as well as this they only use context-awareness inside the device and do not connect to any external web services to gather more information.

Therefore, the project will need to include the following to meet the aims of the project:

- The plug should be context-aware allowing it to make decisions based on the type of device that has been plugged into it and the current device's environment.

- The plug should gather information about its environment using web services to gather information about its location, time and temperature.
- The device should be cheap to make; this will be made easier with the use of embedded hardware as it is relatively inexpensive.

3.3 Proposed system

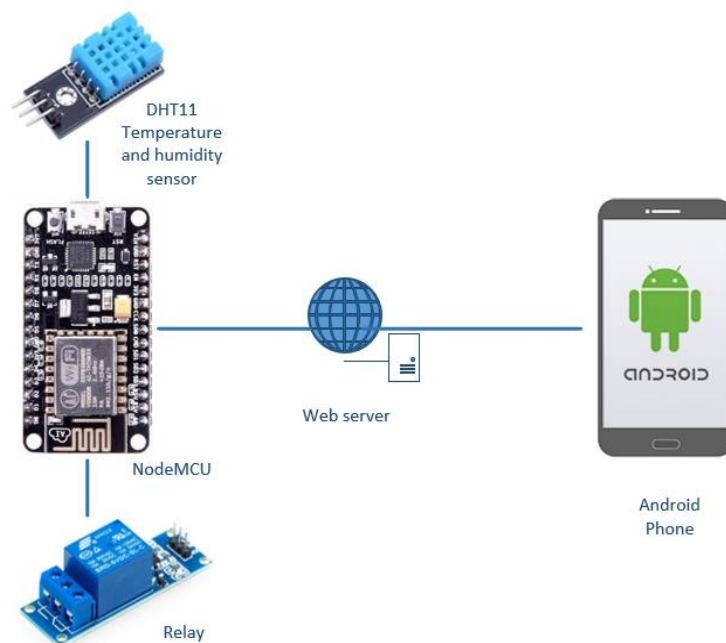


Figure 21 - Diagram of context-aware IoT smart plug prototype

The proposed system will be developed using a NodeMCU, Android application and web service as the primary components. Both the NodeMCU and Android phone will use Wi-Fi to communicate to the web service. The NodeMCU will use sensors and web services to gather information about its surroundings, in this project we will only be using a Temperature/Humidity sensor to keep things simple, as well as a relay to control the flow of electricity to the device.

The system will have three modes which can be selected by the user through the Android application; the modes are 'default', 'fan', and 'radiator' mode. The default mode will turn off any context-aware functionality allowing the user to use the plug as a regular smart plug. Fan mode works by getting the user to

input a threshold temperature which they would like the room to stay below when the temperature sensors on the NodeMCU pass this threshold the plug will turn itself on. The radiator mode is the same as the fan mode except once it passes the threshold the plug will turn off. The system will also utilise external web services to gather information about the external temperature to help make decisions on whether the plug will be turned on or off.

3.3.1 Functional requirements

FR1 – System in action

FR1.1 – The user will be able to control the functions of the plug through the android application.

FR1.2 – The user will be prompted to create an account the first time they open the application.

FR1.3 – The user will be able to log out from the application.

FR1.4 – The application will be clear and straightforward to use.

FR1.5 – The user can select the device which they are plugging in to enable the context-aware functionality.

FR1.6 – Time between user input and action will be less than 5 seconds.

FR2 – User Account

FR2.1 – The user will be able to create an account.

FR2.2 – The users' details will be stored externally allowing them to log in from another device.

FR2.3 – Once the user has logged in, they will not need to again unless the application is uninstalled.

FR2.4 – The user will be able to change the password.

FR2.5 – The user will be able to change the password if they forget via email.

FR2.6 – User details will be stored securely.

FR3 - Features

FR3.1 – User can control the plug from anywhere providing they have access to the internet.

FR3.2 – The embedded system must have a temperature sensor.

FR3.3 – Changing the selected device will change the context of the system.

FR3.4 – The application will show live temperature readings.

FR3.5 – The system should be cheap to produce.

3.3.2 Non-functional requirements

NFR1 – System in action

NFR1.1 – The user will be given instructions on how to set up the device.

NFR1.2 – Users can monitor temperature history through a graph.

NFR2 – Performance

NFR2.1 – The application should not take up too much room on the user's phone.

NFR2.2 – The application will load quickly.

NFR3 - Features

NFR3.1 – Multiple Plugs can be added to the application.

NRF3.2 – The web service will have context-aware capabilities.

3.4 Justification

The goal of this proposed system is to provide context-awareness through both web services and embedded sensors integrated into the device to help reduce energy consumption throughout the home whilst increasing the user's comfort. This would allow a higher number of people to have access to an advanced home automation device as it would be available for a more affordable price. The use of embedded hardware and cloud services will help reduce the overall cost of production and would be reduced as well as being more energy efficient.

Wi-Fi has been chosen for the communication protocol over the other communication protocols which were discussed in chapter 2; this is because Wi-Fi is more available and mainstream than some of the other communication protocols such as Zigbee and WiMAX. Wi-Fi can be obtained in literally any country in the world which allows for the plug to be controlled globally. Finally, most mobile devices only have Bluetooth, and Wi-Fi access meaning for the communication from the mobile to the plug would have to be very close range to use Bluetooth, or multiple communication methods would have to be used.

3.5 Potential issues with the proposed system

There are a few potential issues that concern this project:

- Initial setup – As the plug is going to be using Wi-Fi, there is no way of inputting the users SSID and password into the device to allow for network connectivity. This is going to affect the types of users that could use this product as they would need to be able to flash the memory with the code but with their SSID and password in place. A potential fix for this could be to use another wireless communication protocol such as

Bluetooth as almost all smartphones have Bluetooth, and it allows for direct connection without a password.

- Security – With all the information being stored and accessed over the internet there is the potential issue of the data, more specifically user data being accessed and stolen. There is also the issue of malicious users gaining control of user's devices as all traffic is going through a web server.
- Connectivity – Like all smart devices there is the issue of the user not having internet connectivity meaning if they wanted to turn on or off the device, they would not be able to.
- Web services – Relying on external web services for information can be an issue as if they were to go down it may cause the device not to work correctly.
- Libraries – With embedded systems relying on libraries to gain additional functionality if one of the libraries no longer works it could affect how the device works or stop it altogether.

3.6 Hardware and Software required

3.6.1 Hardware components

NodeMCU –

The NodeMCU is an open source IoT platform which was first developed in 2013 by a Chinese company named Espressif Systems. It includes firmware which runs on the ESP8266 Wi-Fi System on Chip (SOC) and hardware which is based on the ESP-12 module.

The NodeMCU is the firmware, not the development kit. The NodeMCU is like the Arduino with input/outputs on the board; however, unlike the Arduino, it has a Wi-Fi chip directly on the board.

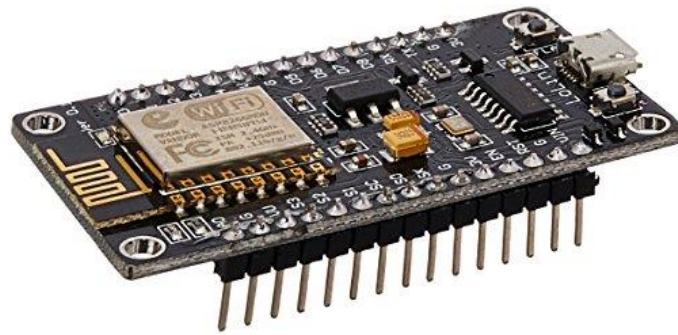


Figure 22 - NodeMCU

The NodeMCU was selected for this project because of its low price, the fact that it has a Wi-Fi chip on the board, the massive amount of community support and finally because of its size.

Relay –

A relay is an electromagnet switch operated by a small electric current (3.3 – 5 volts) which can turn on or off a larger current. A relay controller is used to

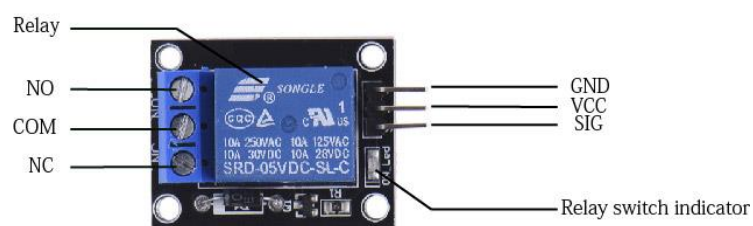


Figure 23 - 5V 1-Channel Relay Board

control a bank of switches by turning on and off magnetic coils under logic control. Relays are used when a circuit is needed to be controlled by a separate low-powered signal.

From figure 23 you can see that there are three output pins –

1. Common (COM) – Common pin
2. Normally Open (NO) – When the relay is not receiving a signal (Low) it is disconnected from the common pin, and when it receives a signal (High) it will connect to the common creating a circuit.
3. Normally Closed (NC) – This is the complete opposite to the NO pin as when it is not receiving it is connected to the common and when it is receiving it is not connected to common.

A relay was selected for this project as the plug is going to have to control the flow of electricity going to the device that is plugged in and will work exceptionally well with the NodeMCU.

Temperature and Humidity Sensor –

For this project, it was decided that a DHT11 humidity and temperature sensor was going to be used due to its ultra-low costs and relatively good accuracy (± 2 °C). It uses a capacitive humidity sensor and thermistor to measure the surrounding air and produces a digital signal on the data pin.

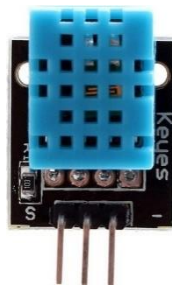


Figure 24 - DHT11 Digital Temperature Humidity Sensor

3.6.2 Software required

Android Studio –

Android Studio is the official Integrated Development Environment (IDE) for Google's Android operating system. This software is essential in creating the Android application as due to the extensive community support which will be

extremely useful in the development of the app. Other IDE's such as XCode could be used for development for iOS devices; be that as it may, the author does not have an iPhone or a Mac to program on.

Arduino IDE –

The Arduino IDE is used to write and upload programs to Arduino compatible boards; however, with the help of third-party cores, it can also be used for other vendor development boards. Because of this, the Arduino IDE is going to be used to program the NodeMCU. There are other IDEs which could be used for this such as Eclipse and UECIDE; after all, there is more community support around the Arduino IDE.

3.7 ID3 algorithm

The Iterative Dichotomiser 3 (ID3) is an algorithm used to generate a decision tree from a dataset. This algorithm is often used in machine learning and can be used in this project with using the 'Temperature history' data, the current outside temperature, and it can work out when the best time is to turn on or off the plug. So, from the data set shown in the table below the algorithm can produce a decision tree to work out if the plug should be on or off, however, this is subject to the user's threshold temperature they have set:

Table 3 - ID3 dataset of temperatures

| Internal Temperature | External Temperature | Threshold Temperature | ON / OFF | |
|----------------------|----------------------|-----------------------|----------|----------|
| | | | Fan | Radiator |
| Hot | Hot | Hot | OFF | OFF |
| Hot | Hot | Cold | ON | OFF |
| Hot | Cold | Hot | OFF | OFF |
| Cold | Hot | Hot | OFF | OFF |

| | | | | |
|------|------|------|-----|-----|
| Hot | Cold | Cold | OFF | OFF |
| Cold | Cold | Hot | OFF | ON |
| Cold | Hot | Cold | OFF | OFF |
| Cold | Cold | Cold | OFF | OFF |

From this basic set of data a decision tree would be made using the ID3 algorithm this would use the current temperature data and work out if the plug would be turned on or off, this would be a lot more complex in the actual development as it would be using real temperatures instead of 'Hot' and 'Cold'.

3.8 Product details

Looking back at the system diagram in Figure 21 shows that the system is made up of 3 distinct sections: the mobile application, the web service, and the context-aware smart plug (hardware). The function of the mobile application is to allow control over the hardware such as setting the device up and turning the plug on or off. The web service is going to be an intermediate between the mobile application and the plug, while the context-aware plug is going to act based on information from the web service as well as information from the sensors.

3.8.1 Mobile Application

The role of the Plugnet application is to allow the user full control over the smart plug such as turning the plug on or off, changing the mode and setting threshold temperatures.

3.8.1.1 User Interface

Before development could be started on the application, it is crucial to design what the application will look like, so development is more straightforward. The user interface for Plugnet application is intended to be familiar and straightforward by using commonly used application templates. The application will consist of several screens which will allow the user to log in, create an account, change the password and edit the plugs details etc. Below in Figure 25 is the prototype for the Plugnet application.

When creating the application-specific design guidelines will be followed to make sure the application is consistent, predictable, and accessible which will make the application more enjoyable for users.

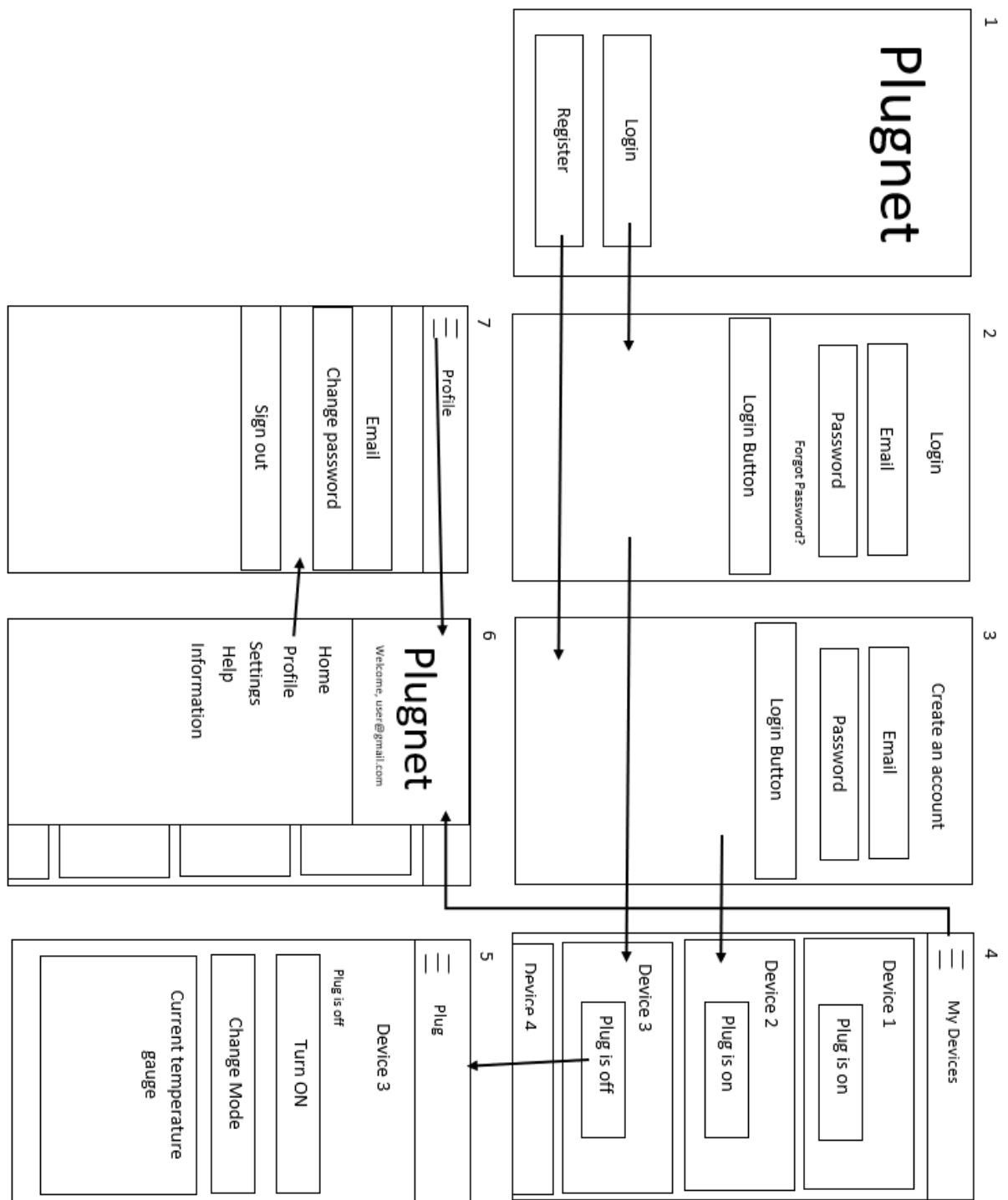


Figure 25 - Initial proposed prototype of Plugnet application

Looking at Figure 25 you can see the basic flow of most of the screens which will be included within the application; each screen has been numbered. The first page the user will see when they open the app is the create account or login in

page; this will give the option for the user to create an account or log in. From here it will take them to one of 2 pages (2 and 3) where they will enter their details and will be taken to screen 4, this screen will show all of the Plugnet smart plugs the user has and will display some information about the status of the plug such as whether it is on or off and the mode that it's in. These tiles will be clickable taking the user to page 5 where it will show more information about the plug and allow for device details to be changed. Several other screens will be included such as profile, settings, help and information.

3.8.1.2 User Account

For user accounts, the mobile application will use Google's Firebase Authentication as this allows for user data to be stored securely in the cloud.

Firebase has been chosen for user login for a few of reasons:

- Security – As all the storing of users' personal information is done through a Google-owned company it will be done to a high standard.
- Speed – Most Back-end as a Service (BaaS) will be a lot faster than one developed by someone with little experience.
- Ease of implementation – Firebase and Android studio are closely intertwined which allow for Firebase to be implemented to any application developed on Android studio with a click of a button.

3.8.2 Web Service

Both the mobile application and the smart plug would have to have access to this. Therefore, it is essential that the information stored on these could be read by either device. XML and JSON are both data-interchange format languages allowing for any device to read the data. Fortunately, Firebase offers a real-time non-SQL database that uses JSON, since Firebase is already going to be used for the login implementing the real-time database is going to be a lot simpler.

The use of Firebase real-time database has the same advantages as Firebase authentication as it is secure, fast and easy to implement.

3.8.2.1 Database

The database had to be designed very differently to other databases as it is a no-SQL database and so it uses an entirely different format. JavaScript Object Notation (JSON) is a language readable by humans as it is stored in plain text which allows for it to be read into any programming language. The first database will need to include all the smart plugs. Therefore, a unique ID will need to be given to each device, and then each ID will contain all the information needed by the plug and user, something like this:

- Unique ID
 - Temperature
 - Plug Status
 - Mode
 - Threshold
 - Name

There will also be a second database in which all the previous temperatures will be stored which allow for analysing and modelling the data to predict when the plug should and should not be on. This database will look something like this:

- Unique ID
 - Date
 - Time
 - Temperature

Because the Firebase database is “real-time” every client which is connected it synchronised in real time across all platforms. This means that when the user turns on or off the plug, it will be updated in real time so the plug will update almost instantly.

3.8.3 Context-Aware Smart Plug

The role of the context-aware smart plug is to gather information about the environment; in this case, it is the temperature and turn on or off accordingly. The plug should also react to user input like changing the status, mode and threshold temperature.

3.8.3.1 Connecting to the web server

NodeMCU' have a lot of community support and so there is a lot of information on how to retrieve information from the Firebase real-time database, and there are a couple of different libraries that can be used to assist with the connection. Because Firebase uses JSON, these libraries also require the use of a JSON library. The information from the database can then be read into local variables on the NodeMCU.

3.8.3.2 Changing the mode

The mode of the plug will consist of default mode, plug mode and radiator mode; these modes will affect how the plug operates based on the temperature and the threshold temperature the user has entered. A diagram is shown below -

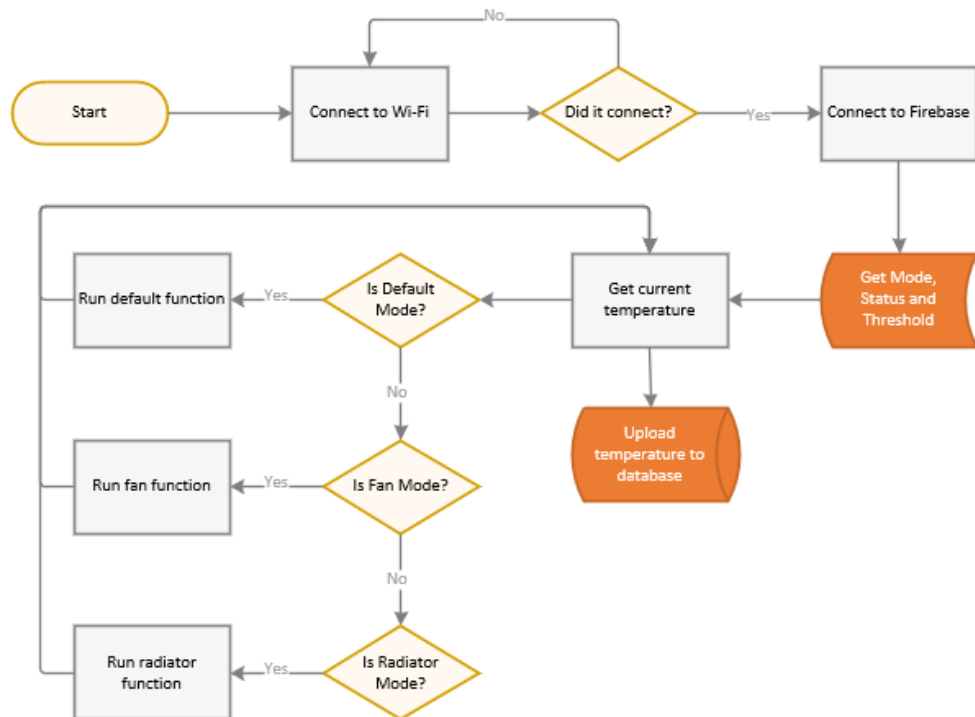


Figure 26 - Top-Level diagram showing the process of the Context-aware smart plug

3.8.3.2.1 Default mode

The default mode does not have any context-aware functionality and so only relies on the status of the plug (whether it is turned on/off). This is shown in the Figure below -

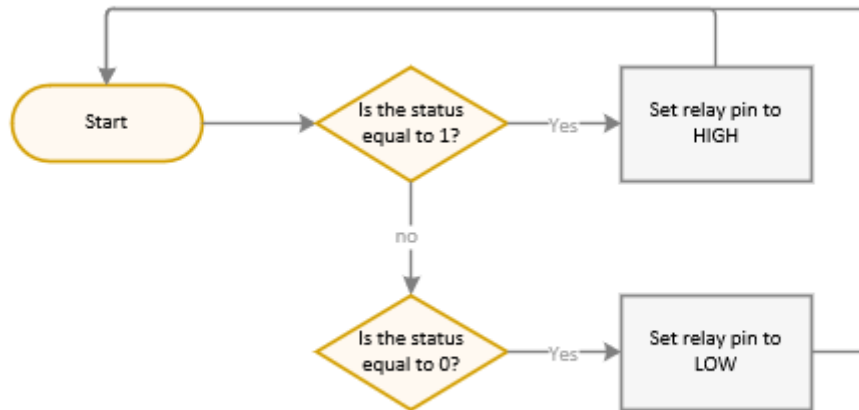


Figure 27 - Top-Level diagram of the default mode

3.8.3.2.2 Fan mode

The Fan mode relies on both the temperature read from the plug, the external temperature which will be obtained from an API and the threshold temperature set by the user. In the case when the internal temperature is higher than the threshold temperature and the external temperature is greater than or equal to the internal temperature the plug will turn on which would turn on the fan and cool down the room. Once the room gets cold enough (threshold is higher than the temperature) the plug will then turn itself off. The idea behind using the external temperature is that if it is colder outside, and warmer inside the plug will not turn on as eventually, the room will cool down to the temperature of the outside which will save the user money and electricity as they will not be wasting it on trying to cool down the room.

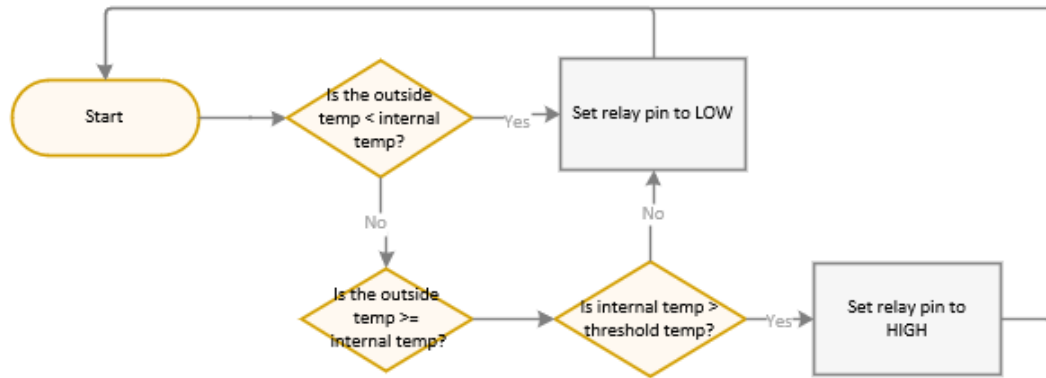


Figure 28 - Top-Level diagram of Fan mode

3.8.3.2.3 Radiator mode

The radiator mode is very similar to the fan mode apart from the temperatures are switched around and so if the temperature is less than the threshold the radiator will turn on which would heat the room. Once the temperature is higher than the threshold, the plug will turn off. The idea behind using the external temperature is that if it is warmer outside and colder inside the plug won't turn on as eventually, the room will heat up to the temperature of the outside which will save the user money and electricity as they won't be wasting it on heating the room.

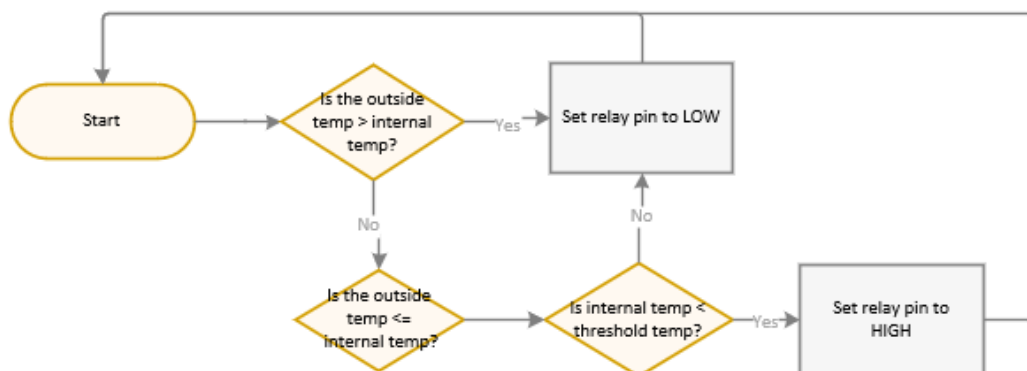


Figure 29 - Top-Level diagram of radiator mode

3.8.3.3 Collecting internal temperature

The internal temperature is collected using the temperature sensor connected to the NodeMCU and will require the DH11 library to be able to convert the signal to a temperature which can be read by humans. As stated above the temperature will be used in the 'Fan mode' and the 'Radiator mode' in each of these cases when the relay is set to 'HIGH' or 'LOW' the current status and the current temperature in degrees Celsius is updated in the database so the user will have the up to date information on the Plugnet app.

3.8.3.4 Collecting the external temperature

The external or outside temperature will be collected using a temperature API which will get the current temperature of a specific location. The location will either be got using a different API or by the user setting it within the application.

3.9 Methodology

The methodology which is going to be used throughout the project's development is Agile development. Agile development was chosen over other methodologies like Waterfall, and Spiral as Agile focuses on producing working software that responds to change rather than following a strict plan. According to the Agile Manifesto, there are 12 principles behind the Agile development methodology:

1. The number one priority is to satisfy the customer through early and continuous delivery.
2. Be open to a change in requirements, even if it is late in the development.
3. Deliver working software frequently, focussing on delivering as soon as possible.
4. Developers must work in conjunction with the business people daily.
5. Build projects around highly motivated individuals.
6. The most effective way of relaying information is through face-to-face meetings.
7. Progression is measured by working software.
8. Agile processes promote sustainable development.
9. Agility is enhanced through attention to technical excellence and sound design.
10. Simplicity (Reduce the amount of work needed to be done).

11. Self-organising teams produce the best architecture, requirements and designs.
12. Regularly reflect on how to become more productive and then adjust behaviour accordingly.

3.10 Testing

In order to make sure the entire system has been produced to a high enough standard at the end of the project, multiple tests would have to be carried out on both the individual components and the system as a whole. The tests carried out would range from users testing the application to make sure the navigation and style standards are met; to testing the entire system and each of the components ensuring the functionality is met.

As much of the system is closed off, testing the areas where users can input their data into the application would be an excellent place to start as it would allow the testing of handling exceptions and errors. Because, the database is JSON the data doesn't have a data type assigned to it so if incorrectly handled would cause both the application and the NodeMCU to crash, as a mismatch of data types would cause the software to crash, which would require the data in the database to be changed which the user would not be able to do. Therefore, by catching these issues before they cause any issues with the application and plug would save a significant amount of time from the users and the database administrators who would work at Plugnet.

Another factor that would require testing would be the usability of the application, making sure a more extensive range of users would be able to use the application. As the development methodology is Agile, this would allow for the application to be tested, improved and tested again allowing for a quicker development while increasing usability of the application. These tests would

make the application available for a broader range of users as they would influence the layout, navigation, text size and fonts etc.

Also, testing speeds and response times to see if there are any bottlenecks within the system which could be improved upon in future development. This could be how long it takes to launch the application; how long it takes to turn on or off the plug from the application and how quickly it takes to measure and react to the external temperature.

Last of all, testing how the database handles multiple clients accessing the database and how the application, plug and database react to simultaneous changes. As changing the same value in the database within succession could cause the application, database or plug to show the wrong value.

3.11 Project planning

At the start of the project, a Project Planning Document (PPD) was created to help with the timing of the project using a Gantt chart by splitting down each section into more manageable time components. Both the PPD and the Gantt chart can be found in Appendix A.

However, due to the time taken to complete other coursework as well as other unforeseen circumstances which were not accounted for within this timeframe for the project planning documents Gantt chart. This caused a lot of the research, planning and development to be pushed back and so tasks that were to be completed in the first term had been pushed to the beginning of the second term. A second Gantt chart has been created with the updated times based on the expected time to research, learn and develop the software; this can be found in Appendix B.

3.12 Summary

This chapter introduces the proposed system that is going to be produced as part of the project, the hardware and software required as well as how each component is going to be implemented and why it is needed. It also discusses the development methodology and testing methods that will be used throughout development. A new Gantt chart was also created to help structure the development process and split each section into manageable steps.

CHAPTER 4

IMPLEMENTATION OR INVESTIGATION

4.1 Introduction

This section of the report will focus on the development of the system; this will be split into three development phases allowing for development, feedback and testing. Within these three development stages it will be split into four main sections:

- The development and functionality of the Android application, which will take the longest to develop due to the amount of code required. Moreover, it will be changed the most during the testing phases from a design point of view.
- The development of the Firebase database, this will also be linked to the development of the application and the plug as they will both be accessing it.
- The development and functionality of the Context-aware smart plug, this will go through many changes through the development process as functionality is added and testing is done.
- Testing the system at that current stage of development to help guide and improve the system in the later stages of development.

Each of these sections will be broken down into smaller sub-sections which will focus on the specific functionality of the individual components and then testing them to make sure they are working as required. Once these testing phases have been completed there will be an overview of the system and a discussion of missing functionality which was intended to be a part of the system.

The chapter is split into these 3 phases as the methodology chosen for this project is Agile development which is an iterative process which responds to change and feedback during the development process. Therefore, it was decided that at 3 points throughout the development of the project user feedback would be received for the design of the application and testing would be carried out on both the app and the plug.

4.2 Development Phase 1

In the first development phase, the goal was to get some of the core functionality implemented to make sure the system could feasibly be implemented.

4.2.1 Plugnet Application

In chapter 3.7.1.1 the initial design was created to help guide the development style of the application; however, this only shows seven screens where there is going to be a lot more. In Android Studio the screen layout is written in XML while the control of the layout is written in Java. Each of the layouts that have been implemented into the Plugnet app is either an 'Activity' or a 'Fragment', the activities are separate screens that can run independently, and the fragments can only run within an activity. Fragments have been implemented across the Plugnet application to reduce the amount of code as the navigation view does not need to be implemented on each of the fragments, only on the activity.

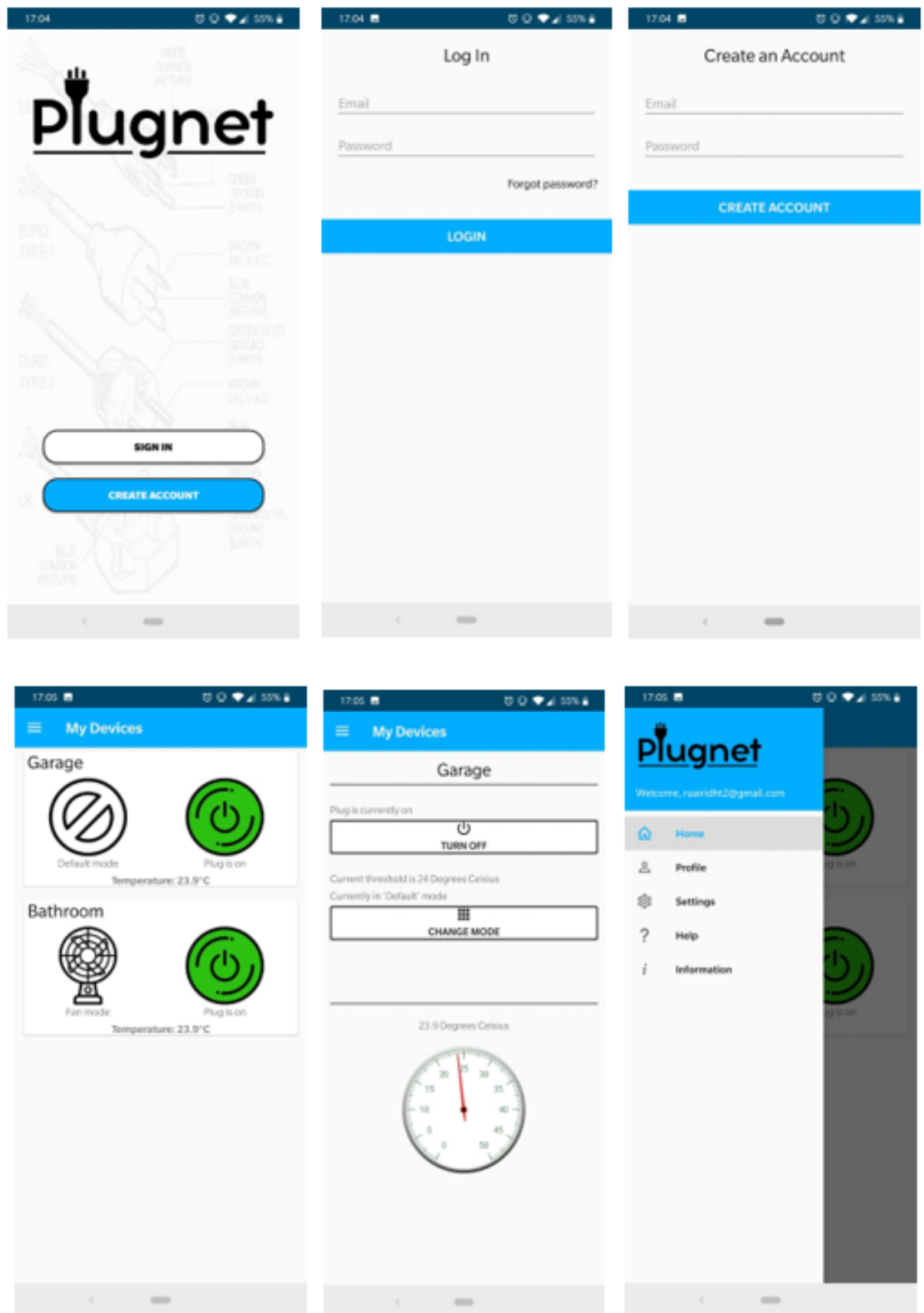


Figure 30 - First implementation of Plugnet application

In Figure 30 the first two screenshots at the bottom of the page which show the devices are using fragments which are used on top of the layout at the bottom right, while the top 3 screenshots are individual activities.

4.2.1.1 Navigation

At this current point in development, all navigation is completed using buttons and navigation menus throughout the app. Navigation between Fragments and activities differ slightly, as shown below with the first being navigation between Activities and the latter being navigation between fragments.

```
public void onClick(View v) {
    switch (v.getId()) {
        case R.id.signIn:
            Intent intent1 = new Intent(this, LoginActivity.class);
            startActivity(intent1);
            break;
        case R.id.createAccount:
            Intent intent2 = new Intent(this, RegisterActivity.class);
            startActivity(intent2);
        default:
            break;
    }
}
```

```
public boolean onNavigationItemSelected(@NonNull MenuItem menuItem) {

    switch (menuItem.getItemId()) {
        case R.id.nav_home:

            getSupportFragmentManager().beginTransaction().replace(R.id.fragment_container, new HomeFragment()).commit();
            getSupportActionBar().setTitle("My Devices");
            break;
        case R.id.nav_profile:

            getSupportFragmentManager().beginTransaction().replace(R.id.fragment_container, new ProfileFragment()).commit();
            getSupportActionBar().setTitle("My Profile");
            break;
        case R.id.nav_settings:

            getSupportFragmentManager().beginTransaction().replace(R.id.fragment_container, new SettingsFragment()).commit();
            getSupportActionBar().setTitle("Settings");
            break;
        case R.id.nav_help:

            getSupportFragmentManager().beginTransaction().replace(R.id.fragment_container, new HelpFragment()).commit();
            getSupportActionBar().setTitle("Help");
            break;
        case R.id.nav_information:

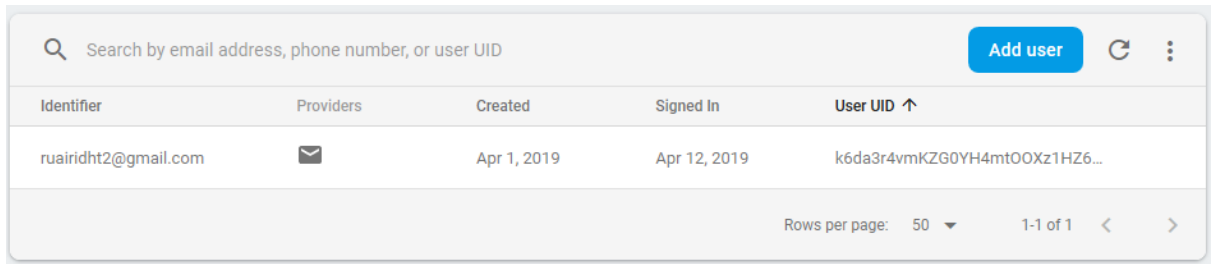
            getSupportFragmentManager().beginTransaction().replace(R.id.fragment_container, new InformationFragment()).commit();
            getSupportActionBar().setTitle("Information");
            break;
        default:
            break;
    }
}
```

```
        drawer.closeDrawer(GravityCompat.START);  
        return true;  
    }
```

In the first code snippet which was taken from the first screen, it shows that each of the buttons has been given a unique name which when clicked will switch from the current activity to the new one. The second snippet of code which was taken from the navigation menu shows that when one of the navigation menu items have been selected, it is replacing a fragment within the activity with another fragment.

4.2.1.2 User Registration and Login

For user registration and login, Firebase authentication has been implemented within the app using the user's email and password. Firebase has been implemented to increase the security of the application as all the user information is stored by Firebase.




| Identifier | Providers | Created | Signed In | User UID ↑ |
|----------------------|---|-------------|--------------|-------------------------------|
| ruairidht2@gmail.com |  | Apr 1, 2019 | Apr 12, 2019 | k6da3r4vmKZG0YH4mt0OXz1HZ6... |

Figure 31 - Firebase Authentication console screen

When the user logs in or creates a new account, it requires them to enter a valid email and password, before the user's details are sent off to Firebase for authentication, the application will check each of the text fields to make sure they have been filled, as shown below:

```
String email = editTextEmail.getText().toString().trim();
String password = editTextPassword.getText().toString().trim();

if (TextUtils.isEmpty(email)) {
    Toast.makeText(this, "Please enter your email",
        Toast.LENGTH_SHORT).show();
    return;
}
if (TextUtils.isEmpty(password)) {
    Toast.makeText(this, "Please enter your password",
        Toast.LENGTH_SHORT).show();
    return;
}
```

The code shows that it is getting the text from the email and password from the text fields and checking to make sure they have been filled out, if they have not a small message will appear at the bottom of the screen telling them to enter either their email or password. If both have been filled in their information will

be sent off for authentication, as illustrated below with the first being signing in and the second being creating a new account:

```
mAuth.signInWithEmailAndPassword(email, password)
    .addOnCompleteListener(this, new OnCompleteListener<AuthResult>() {
        @Override
        public void onComplete(@NonNull Task<AuthResult> task) {
            if (task.isSuccessful()) {
                progressDialog.dismiss();
                openApp();
                Toast.makeText(LoginActivity.this, "Login successful",
                    Toast.LENGTH_SHORT).show();
            } else {
                progressDialog.dismiss();
                Toast.makeText(LoginActivity.this,
                    task.getException().getMessage(), Toast.LENGTH_SHORT).show();
            }
        }
    });
```

```
mAuth.createUserWithEmailAndPassword(email, password)
    .addOnCompleteListener(this, new OnCompleteListener<AuthResult>() {
        @Override
        public void onComplete(@NonNull Task<AuthResult> task) {
            if (task.isSuccessful()) {
                progressDialog.dismiss();
                openApp();
                Toast.makeText(RegisterActivity.this, "Registered
                successfully", Toast.LENGTH_SHORT).show();
            } else {
                progressDialog.dismiss();
                Toast.makeText(RegisterActivity.this,
                    task.getException().getMessage(), Toast.LENGTH_SHORT).show();
            }
        }
    });
```

The code above shows that if the details that have been entered are correct, it will call a function called “*openApp()*” which will take the user to the device screen and will also display a message letting the user know they have logged in or registered successfully. However, if the user did not enter the correct details or there was an error it will show a message letting the user know what went wrong such as –

- The user enters the wrong password when logging in: “The password is invalid, or the user does not have a password”.

- The user enters the wrong email when logging in: "There is no user record corresponding to this identifier. The user may have been deleted".
- The user does not enter a correctly formatted email: "The email address is badly formatted".
- The user does not have a connection to the internet: "A network error (such as timeout, interrupted connection or unreachable host) has occurred".
- The user entered a password that is too short when creating an account: "The given password is invalid. [Password should be at least 6 characters]".
- The user entered an email that has already been used when creating an account: "The email address is already in use by another account".

Once the user has logged in once on the application they should not need to again unless they have deleted and reinstalled the application or they have logged out as when the application is opened it will check if the user has logged in with the following code:

```
mAuth = FirebaseAuth.getInstance();

if (mAuth.getCurrentUser() != null) {
    openApp();
}
```

This code checks if there is currently a user logged in and if there is it will call a function which will open the activity of the device.

4.2.1.3 Retrieving and uploading to Firebase

The primary function of the application is to control the hardware, check current temperatures, and change the mode and threshold. To be able to do so the application needs to access the Firebase real-time database, and both retrieve and update the database. To be able to get access to this information a database reference is needed to know where to get and upload the information to.

```
DatabaseReference databaseReferenceStatus;  
databaseReferenceStatus =  
FirebaseDatabase.getInstance().getReference("Plugs").child(id).child("Plug_  
Status");
```

The "databaseReferenceStatus" variable is used to get the location of the status of the plug in the database so it can be accessed in other functions such as:

```
public void checkPlugStatus() {  
  
    databaseReferenceStatus.addValueEventListener(new ValueEventListener()  
{  
        @Override  
        public void onDataChange(DataSnapshot dataSnapshot) {  
            String value = dataSnapshot.getValue().toString();  
            int plugStatus = Integer.valueOf(value);  
            if (plugStatus == 1) {  
                cStatus.setText("Plug is currently on");  
                plugButton.setText("Turn off");  
                currentStatus = 1;  
            }  
            if (plugStatus == 0) {  
                cStatus.setText("Plug is currently off");  
                plugButton.setText("Turn on");  
                currentStatus = 0;  
            }  
        }  
  
        @Override  
        public void onCancelled(DatabaseError error) {  
            Toast.makeText(getActivity(), error.getMessage(),  
Toast.LENGTH_SHORT).show();  
        }  
    });  
}
```

```
public void updateStatus(int status) {  
  
    databaseReferenceStatus.setValue(status)  
        .addOnSuccessListener(new OnSuccessListener<Void>() {  
            @Override  
            public void onSuccess(Void aVoid) {
```

```

        }
    })
    .addOnFailureListener(new OnFailureListener() {
        @Override
        public void onFailure(@NonNull Exception e) {
            Toast.makeText(getActivity(), e.getMessage(),
Toast.LENGTH_SHORT).show();
        }
    });
}

```

This code is taken from the fragment which is shown at the bottom middle of Figure 30. The first function that checks the current status of the plug, if the plug is on the value will be 1 and if it is off the value will be 0. If the value is equal to 1, it will change the text view above the button letting the user know the button is on and will change the text on the button to "Turn off". The opposite will happen if the value is 0 as it will change the text above the button to let the user know the plug is off and will change the text on the button to "Turn on".

The second function is called whenever the turn on or off button is pressed as the `currentStatus` variable which is updated in the first function is used within the button to determine whether the plug is on or off. If the plug is on when the button is pressed it will call `updateStatus(0)`, changing the value in the database, the complete opposite will happen if the plug is already off as it will call `updateStatus(1)`.

Several other functions perform similar tasks such as getting the current temperature, getting the name, and getting the threshold.

4.2.2 Firebase Database

The Firebase database is used as a web service that allows for data to be read and changed in real time, allowing for changes made on the Plugnet application to affect how the plug functions. The Firebase real-time database uses JSON

which allows it to be effectively used as a web service. Currently, there are two datasets within this database, the first being the plug information and the second being a history of temperatures, as shown in Figure 32.

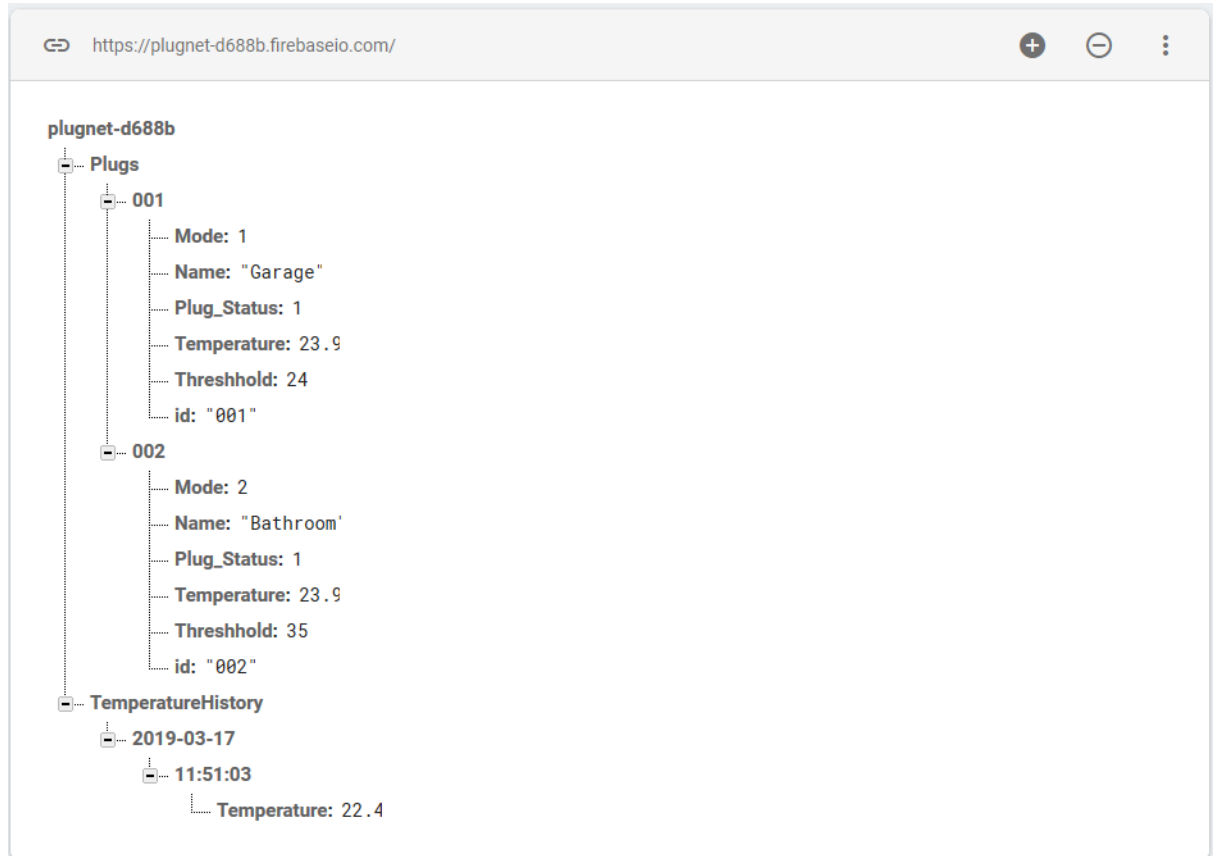


Figure 32 - Firebase real-time database console screen

The plan with the database is to allow the user to add multiple plugs to the database allowing them to control everything from the Plugnet app and so the Name has been added which they can edit within the app to their preference. Each of these variables are used as follows:

- **Mode** - The mode represents the device that has been plugged into the Plugnet plug which can be edited by the user. 1 represents 'default mode' which removes all context awareness from the plug allowing it to function as a regular smart plug, 2 represents 'fan mode' letting the plug know

that a fan is plugged in, 3 represents 'radiator mode' which lets the plug know a radiator is plugged in.

- Name – This has been implemented for the user just in case they have several Plugnet plugs so they can distinguish between them.
- Plug_Status – This is set to either 1 or 0, 1 if the plug is on and 0 if the plug is off, it is used by both the app and the plug.
- Temperature – The plug updates this and read by the application, it can be used to let the user know the current temperature of the room allowing them to get a better idea of their preferred temperature.
- Threshold – This is the preferred temperature set by the user within the application in which the plug will read and try to maintain based on the mode selected by the user.
- Id – This is used within the application to allow information changed within the app to edit the information within the correct plug.

Temperature history is collected every 10 minutes and stored in the database to be later used for training a dataset to predict if the plug should be on or off.

4.2.3 Context-Aware Smart Plug

Using the NodeMCU with the Arduino IDE for the programming requires the Arduino IDE to be set up specifically for this. The first thing that needed to be done was to add a URL to the Additional Boards Manager (arduino.esp8266.com/stable/package_esp8266com_index.json) this allows for the ESP8266 module to be installed so that the NodeMCU board can be selected in board manager. The programming language used by the Arduino IDE is C.

4.2.3.1 NodeMCU Configuration

Both the Relay and the Temperature sensor have three pins, one for ground, one for power and the last one for input/output. Figure 33 shows the current configuration; this is likely going to stay the same throughout development:

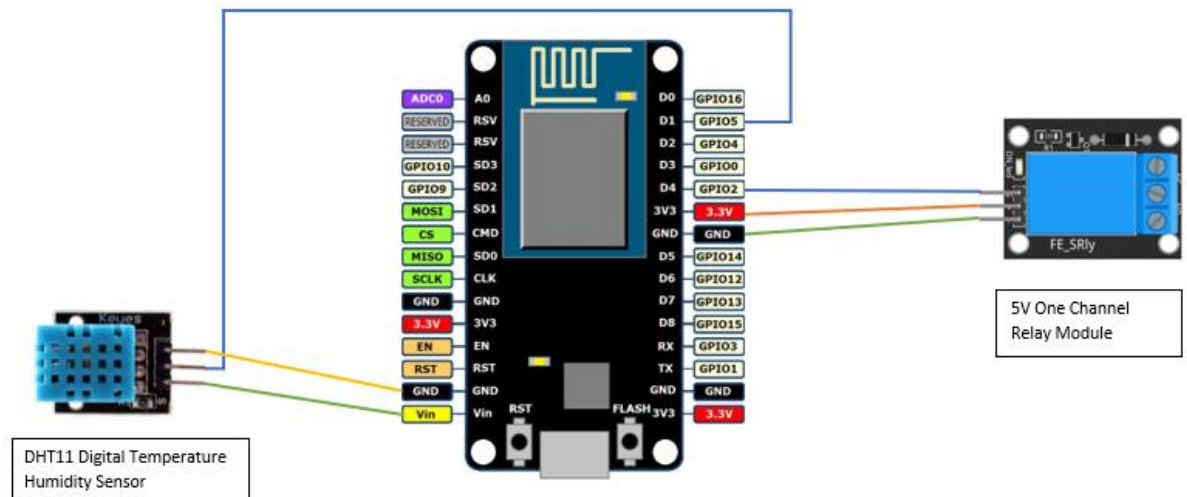


Figure 33 – NodeMCU Plugnet configuration

4.2.3.2 Reading Temperature Sensor

At this stage of development, the plug has some of the core functionality however it is limited in its context-aware capabilities. Currently, the plug can read the temperature and humidity using the DHT11 sensor with the code shown below:

```
1.  float h = dht.readHumidity();  
2.  float t = dht.readTemperature();  
3.  if (isnan(h) || isnan(t)) {  
4.      Serial.println("Failed to read from DHT sensor!");  
5.      strcpy(celsiusTemp, "Failed")  
6.      strcpy(humidityTemp, "Failed");  
7.  }
```

The code above shows the temperatures being read into a float variable once it has been assigned to the variable it checks if the variables are a number using the "isnan" (is not a number) function. To be able to use the DHT11 sensor the DHT Library is needed to be included.

4.2.3.3 Retrieving and uploading to Firebase

The plug also has access to the Firebase real-time database allowing it to upload current temperature information, get the current status and get the current mode. To be able to connect the following values needs to be defined to allow for the plug to know what database needs to be accessed:

```
1. #define FIREBASE_HOST "plugnet-d688b.firebaseio.com"
2. #define FIREBASE_AUTH
   "ap8X2xTGstiJvDPLblJKjuyE9LlGxCsGkFjBoNiM"
3. Firebase.begin(FIREBASE_HOST, FIREBASE_AUTH);
```

On top of this, the plug will need to be connected to the internet, to connect to the internet it will use the following code:

```
1. #define WIFI_SSID "SSID"
2. #define WIFI_PASSWORD "password"
3. WiFi.begin(WIFI_SSID, WIFI_PASSWORD);
```

Both of the ".begin" lines of code are inside of the setup function, once the setup function has completed the main loop function will begin where the majority of the code is either located or called upon. To get and set the data from Firebase uses the following code:

```
1. String ID = "001";
2. String Status = "Plugs/" + ID + "/Plug_Status";
3. n = Firebase.getInt(Status);
4. Firebase.setInt(Status, 1);
```

The code that is used is a lot simpler than that used in the application which does the same thing. The code shows that to set the value it only needs the path of the database value and the value the user would want to input. The ID variable will differ for each plug so that information can be set and got for the individual plugs.

4.2.3.4 Main Code Loop

Being able to get the current temperature and being able to get and set information in the Firebase database allows for a working prototype with less functionality to be created, as illustrated below:

```
1. void loop() {
2.   m = Firebase.getInt(Mode);
3.   n = Firebase.getInt(Status);
4.   timeClient.update();
5.
6.   switch (m) {
7.     case 1:
8.       Serial.print("Default mode");
9.       //default mode
10.      defaultUse();
11.      break;
12.     case 2:
13.       //Fan mode
14.       Serial.print("Fan mode");
15.       defaultUse();
16.       if (thresholdTemperature() < temperature()) {
17.         Firebase.setInt(Status, 1);
18.         Serial.print("LED is on");
19.         digitalWrite(relayInput, HIGH);
20.       }
21.       if (thresholdTemperature() > temperature()) {
22.         Firebase.setInt(Status, 0);
23.         Serial.print("LED is off");
24.         digitalWrite(relayInput, LOW);
25.       }
26.       break;
27.     case 3:
28.       //radiator mode
29.       Serial.print("Radiator mode");
30.       defaultUse();
31.       if (thresholdTemperature() > temperature()) {
32.         Firebase.setInt(Status, 1);
33.         Serial.print("LED is on");
34.         digitalWrite(relayInput, HIGH);
35.       }
36.       if (thresholdTemperature() < temperature()) {
37.         Firebase.setInt(Status, 0);
38.         Serial.print("LED is off");
39.         digitalWrite(relayInput, LOW);
40.       }
41.       break;
42.   }
43. }
```

The first thing that happens is getting the mode, the status and the time, with the mode it checks if its in default mode, fan mode, or radiator mode. If it is set to 1 (default mode) it will call a function called defaultUse() this has two main functions, turning on or off the relay and updating the current temperature:

```
1. if (n == 1) {
2.     Serial.print("LED is on");
3.     digitalWrite(relayInput, HIGH);
4. }
5. if (n == 0) {
6.     Serial.print("LED is off");
7.     digitalWrite(relayInput, LOW);
8. }
9.
10.  Firebase.setFloat(Temperature, temperature());
11.  if (Firebase.failed()) {
12.      Serial.print("setting /number failed:");
13.      Serial.println(Firebase.error());
14.      return;
15.  }
16. }
```

If it is set to 2 (Fan mode), it will also run the defaultUse() function however it will check if the threshold temperature is less than or greater than the actual temperature to either turn on or off the fan. The same applies to the radiator mode which turns on or off based on the actual temperature being lower than the threshold temperature.

4.2.4 User and System Testing

Along with development, there is going to be testing throughout to assist in producing the best product possible; the user testing will help guide the design and functionality of the application while the system testing will help remove any errors the system may be facing.

4.2.4.1 User testing

At each stage of the user testing, a few users will be asked to use the application without any assistance so their feedback will help make the application more user-friendly and potentially spot bugs the system may have.

Five users were asked about the positives and negatives of the application at its current stage –

Positives:

- Likes the name/logo and think it fits well with the purpose of the app.
- Likes the first page (has an excellent background).
- The blue colour used throughout the application.
- Likes the temperature gauge.

Negatives:

- Says “sign in” on one page and “log in” on the other.
- Spread out the login page and change the button to the same as the first screen.
- The tiles that display the status of the plug do not look nice.
- The title at the top does not update when pressing the back button.
- Pressing the back button when logging in takes the user back to the first screen.
- Home does not suit the devices page in the navigation menu, could update that to devices and add a plug icon.
- Tiles could be cleaner and clearer.

4.2.4.2 System testing

From general use of the system, there have been a few bugs and issues that have appeared, these include:

- If the plug loses internet connection, it will not reconnect.
- Turning the plug off on the app when it is in either fan or radiator mode it can turn itself back on if the temperature goes above or below the threshold depending on the mode.
- If the plug fails to update the temperature in firebase, it will cause it to crash.

4.3 Development Phase 2

In this phase development is continuing as well as this, changes have been made based on the user feedback and system testing to keep in line with Agile development methodology.

4.3.1 Response to Testing

This section shows the response to the user and system carried out previously and how the system has changed based on user feedback and changes to the system.

4.3.1.1 Response to user testing

From user feedback on the application, design changes have been made to improve the usability and function. The first thing that was changed was the sign in and create account pages as the users thought they could be improved by changing the buttons to look like the first screen and to fill the screen more by spreading everything out a little, as shown:

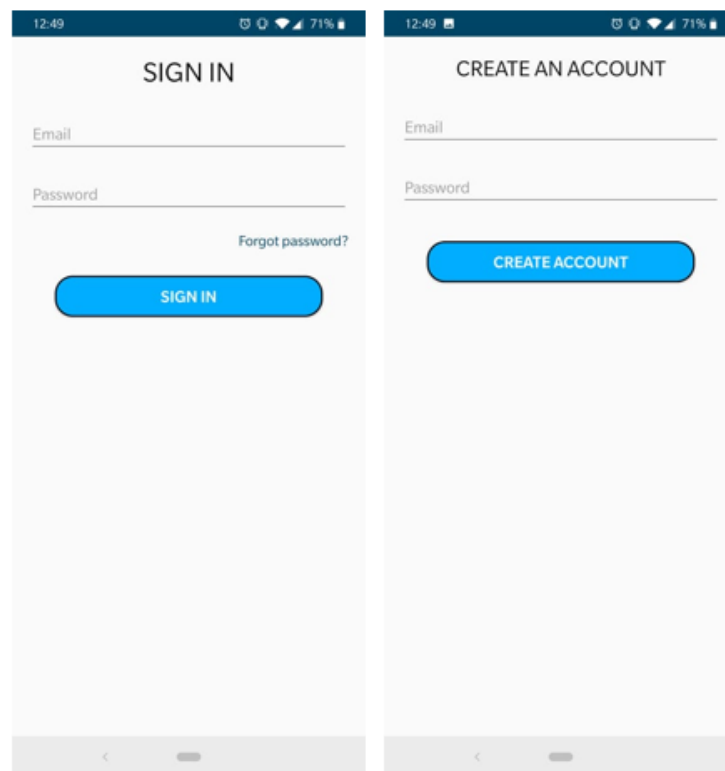


Figure 34 - Sign in and create account screens

Another response to user feedback is their feedback on the tiles that display all the user's devices, as they felt that they were not the most appealing and did not look very "clean". From this feedback an entirely new tile was created which is a bit more subtle, as shown:

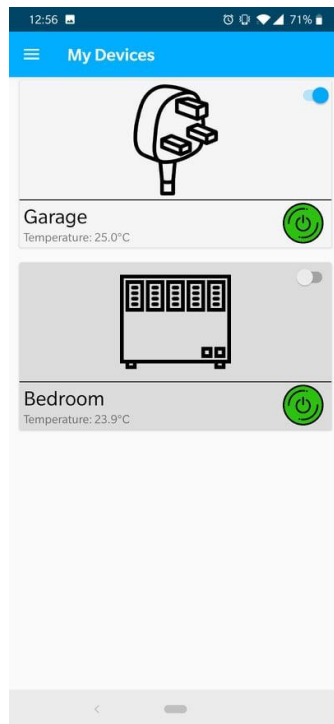


Figure 35 - Device tiles application screen

Finally, the last design change that was made to the app was to change the 'home' button in the navigation drawer to something more appropriate. As the button takes the user to the 'My Devices' page it should be named so:

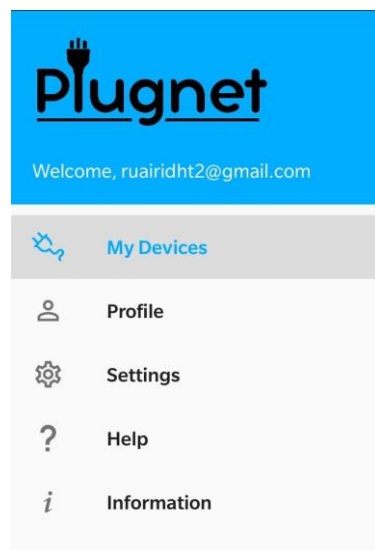


Figure 36 - Navigation drawer updated

Changes to the code also had to be done to fix some of the issues some of the users had with the navigation around the application. Firstly to stop the issue of taking the user back to the starting screen when they press the back button 'finish()' has been called within the openApp function which when called ends the life of the activity it is called in, and so when the back button is pressed it won't take the user to the beginning screen however it will close the app as intended. Another piece of code that has been added is used to update the name on the action bar at the top of the screen:

```
(( AppCompatActivity )getActivity()).getSupportActionBar().setTitle("My  
Devices");
```

This has been added to every fragment and activity, so it updates when the user changes screens, even when the back button is pressed.

4.3.1.2 Response to system testing

Plug losing internet connection – When the plug loses internet connection there was no way of it reconnecting without resetting the plug as the connection was made during the setup function. To fix this, connecting to Wi-Fi now has a function which is called once within the setup function and again when the connection is lost:

```
1. void WIFI_Connect() {
2.   WiFi.disconnect();
3.   WiFi.begin(WIFI_SSID, WIFI_PASSWORD);
4.   Serial.print("connecting");
5.   while (WiFi.status() != WL_CONNECTED) {
6.     Serial.print(".");
7.     delay(500);
8.   }
9.   Serial.println();
10.  Serial.print("connected: ");
11.  Serial.println(WiFi.localIP());
12. }
```

In the main code this is called if it detects that there is not a connection by using the following code:

```
1. if (WiFi.status() != WL_CONNECTED) {
2.   WIFI_Connect();
3. }
```

Plug switching itself back on – When the threshold was higher or lower than the temperature depending on whether it was in fan or radiator mode it would turn itself on or off even if turned off on the application. In response to this, an override switch has been implemented on the app, database and plug to solve this issue, which can be seen in Figure 35 at the top left of the tile a switch has been added to enable and disable the plug. When disabled in the app it goes a darker shade of grey, and the switch is moved to the off position to signal that it has been disabled. In the database, a new value has been added called

'Override' that when set to 0 shows that the plug is off and when set to 1 shows that the plug is on. The code in the plug has also been edited to first get the override status and then to utilise this information, with the override identifier the plug will either turn off or continue to the case statement shown in 4.2.3.3. The code has been implemented as shown:

```
1. override = Firebase.getInt(overrideFB);
2.   if (override == 0) {
3.     Serial.println("Plug is overridden plug is off");
4.     digitalWrite(relayInput, LOW);
5.     Firebase.setInt(Status, 0);
6.     delay(1000);
7.   } else {
8.     switch (m) {...
```

Firestore update failures causing the plug to crash – When the temperature value fails to upload to firestore it causes the NodeMCU to crash, and a reboot is required, therefore, whenever this fails the NodeMCU will reboot:

```
1. Firebase.setFloat(childTemp, temperature());
2.   if (Firebase.failed()) {
3.     Serial.println("Setting temperature in Firestore failed:
4.     ");
5.     Serial.println(Firebase.error());
6.     ESP.restart();
7.   }
```

4.3.2 Plugnet Application

Since the last phase of implementation, several additional screens have been added to the application to make it appear to be more professional as it gives the appearance of legitimacy, these additional screens take the count from 7 to 18:

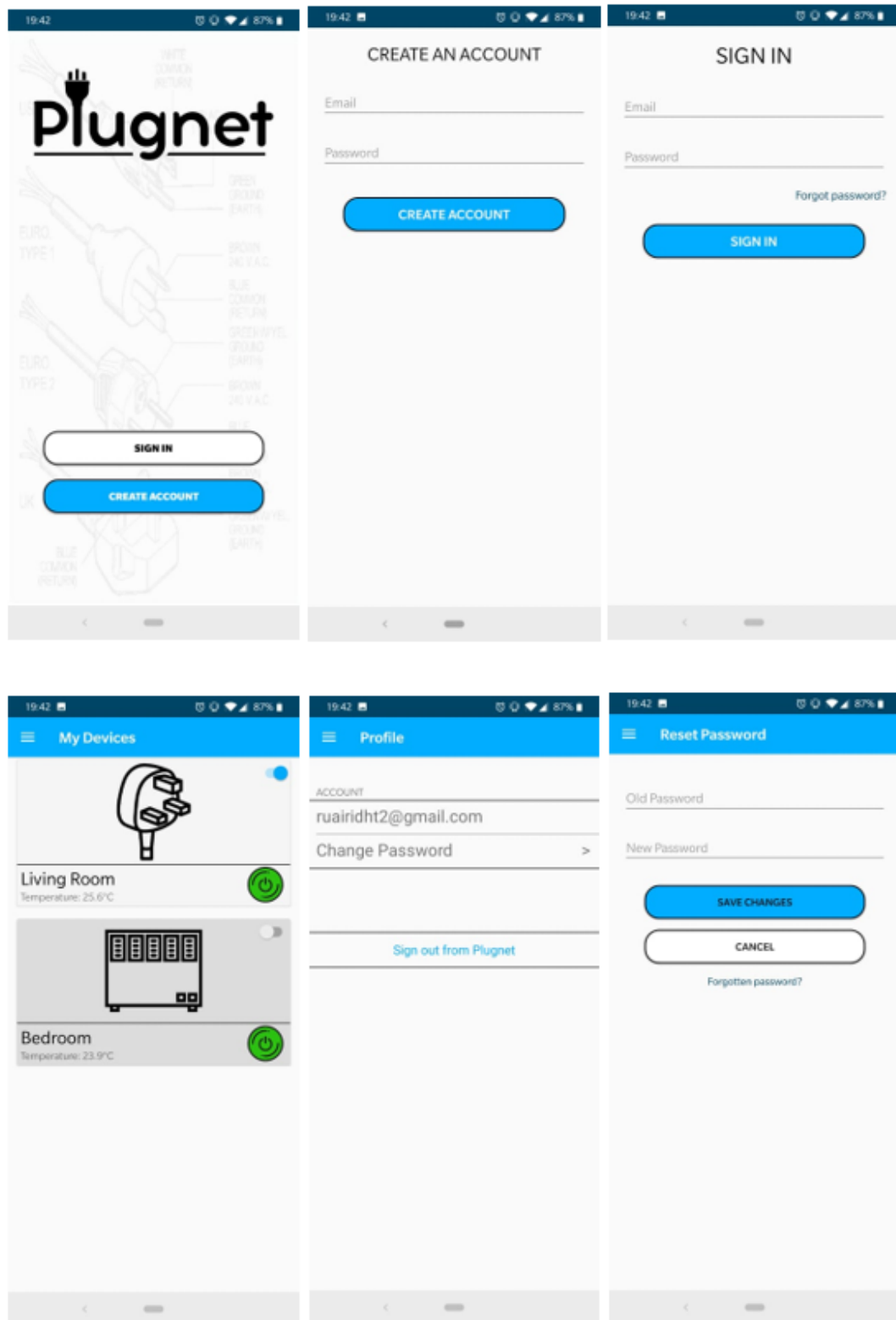


Figure 37 - Application development (1/3)

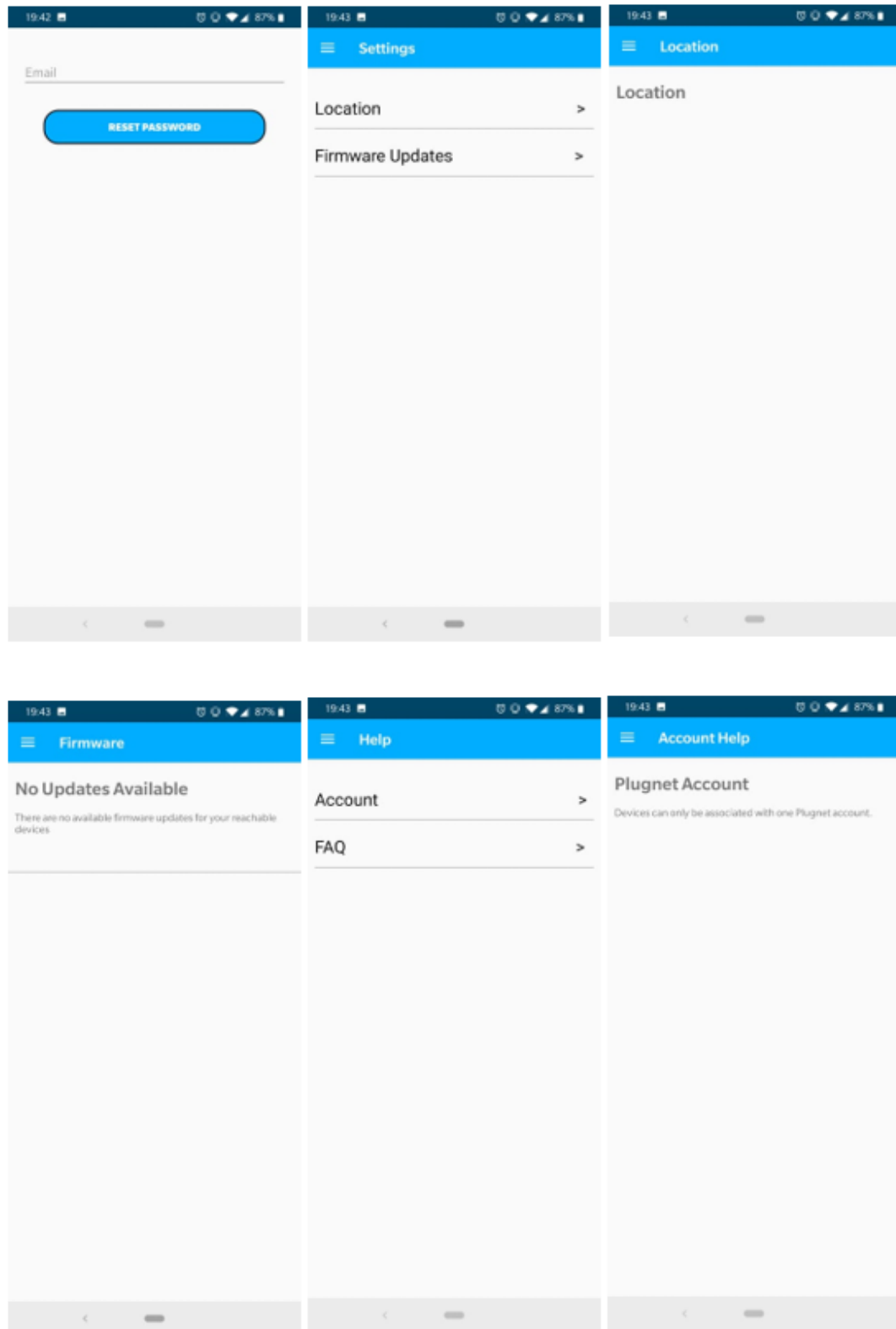


Figure 38 - Application development (2/3)

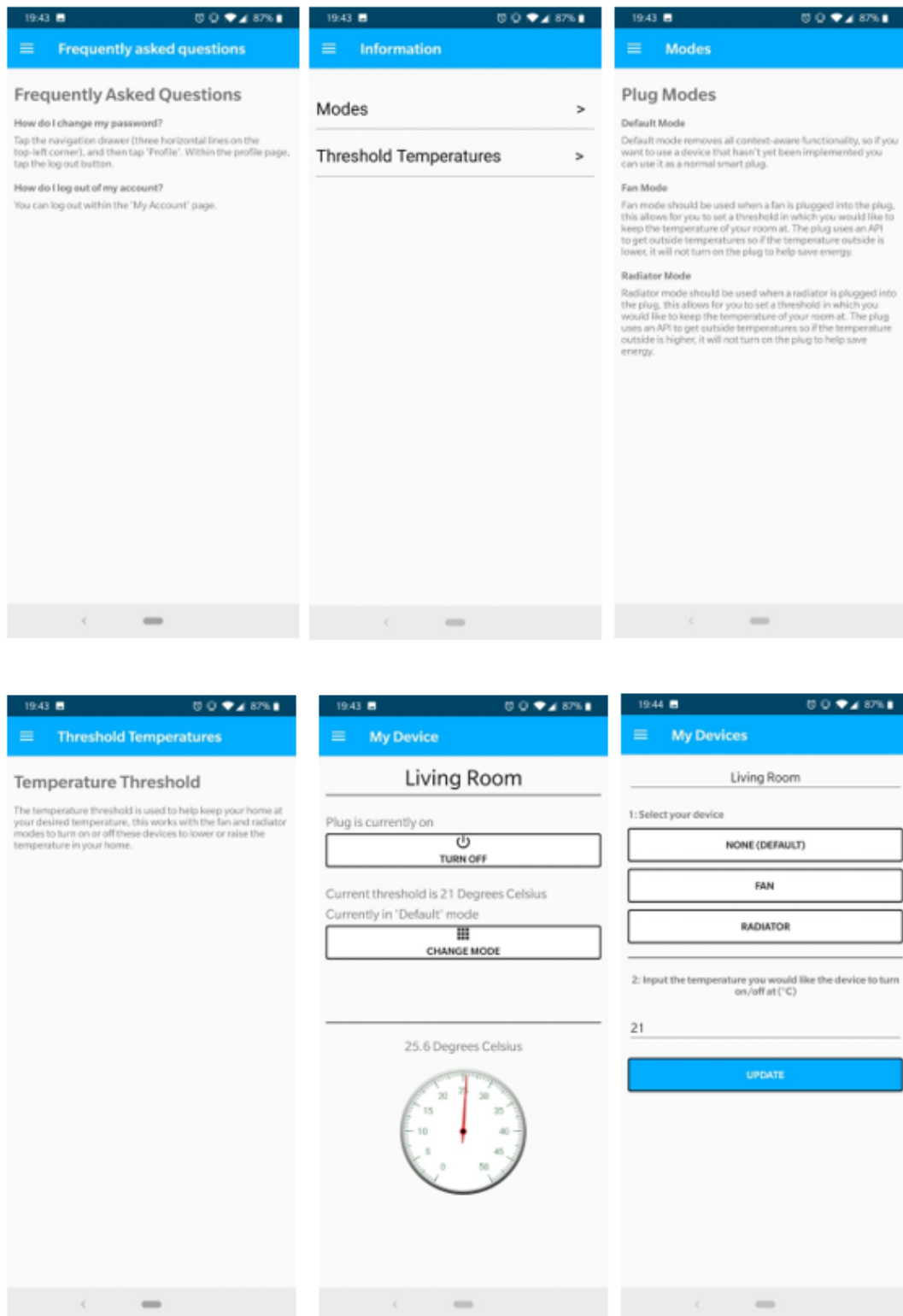


Figure 39 - Application development (3/3)

4.3.2.1 Account passwords

With the addition of these new screens brings new functionality such as the ability for the user to change their password or have their password reset if they have forgotten it. The following code has been implemented for a password change:

```
final String email = user.getEmail();
final String oldPassword = oldPswd.getText().toString().trim();
final String newPass = newPswd1.getText().toString().trim();

AuthCredential credential = EmailAuthProvider
    .getCredential(email, oldPassword);

user.reauthenticate(credential)
    .addOnCompleteListener(new OnCompleteListener<Void>() {
        @Override
        public void onComplete(@NonNull Task<Void> task) {
            if (task.isSuccessful()) {
                user.updatePassword(newPass).addOnCompleteListener(new
OnCompleteListener<Void>() {
                    @Override
                    public void onComplete(@NonNull Task<Void> task) {
                        if (task.isSuccessful()) {
                            Toast.makeText(getContext(), "Password
changed", Toast.LENGTH_SHORT).show();
                            ProfileFragment fragment = new
ProfileFragment();

                            getFragmentManager().beginTransaction().replace(R.id.fragment_container,
fragment).commit();

                        } else {
                            Toast.makeText(getContext(),
task.getException().getMessage(), Toast.LENGTH_SHORT).show();
                        }
                    }
                });
            } else {
                Toast.makeText(getContext(),
task.getException().getMessage(), Toast.LENGTH_SHORT).show();
            }
        }
    });
```

The code above shows that for the use to change their password they need to enter their email, old password and their new password. The email and old password are used to authenticate the user, and if successful their password is of the correct length and format, it will change. If not, it will display a message

letting the user know what they have done incorrectly or if there are any other errors.

If the user has forgotten their password it requires them to enter their email, and if their email is in the list of emails, it will send that user an email asking them to change their email:

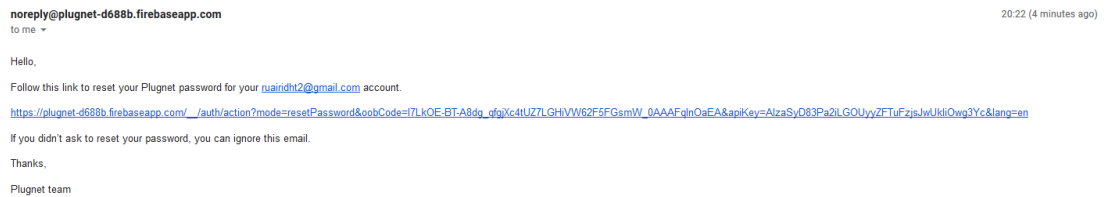


Figure 40 - Password reset email

A screenshot of a web form titled 'Reset your password for ruairidht2@gmail.com'. It features a text input field labeled 'New password' with a blue underline. To the right of the input field is an eye icon for toggling password visibility. At the bottom right of the form is a blue button labeled 'SAVE'.

Figure 41 - Password reset link

All of this is done via Firebase, and the user only needs to enter their email, as shown below:

```
String rpEmail = email.getText().toString().trim();

if (TextUtils.isEmpty(rpEmail)) {
    Toast.makeText(this, "Please enter your email",
        Toast.LENGTH_SHORT).show();
    return;
}

mAuth.sendPasswordResetEmail(rpEmail)
```

```

        .addOnCompleteListener(new OnCompleteListener<Void>() {
            @Override
            public void onComplete(@NonNull Task<Void> task) {
                if (task.isSuccessful()) {
                    Toast.makeText(ForgotPasswordActivity.this, "Reset link
sent to email", Toast.LENGTH_SHORT).show();
                    openApp();
                } else {
                    Toast.makeText(ForgotPasswordActivity.this,
task.getException().getMessage(), Toast.LENGTH_SHORT).show();
                }
            }
        });

```

In the code, it shows it checking to see if the user has entered their email, if so and the email is correctly formatted will send a password reset email as shown in Figure 40.

4.3.3 Firebase Database

In this phase of implementation, nothing other than the addition of the override key has been implemented within the database. In the next phase, the addition of the user ID will be implemented which will add the functionality of only letting one user see their devices, instead of everyone seeing everyone else's devices.

4.3.4 Context-Aware Smart Plug

Since the last phase of development, there have been a few changes made which were mainly fixes to issues which can be seen in chapter 4.1 however there has been some additional functionality added to improve the context-awareness of the system.

4.3.3.1 Temperature API

To help improve the context-awareness of the plug an API to retrieve the current outside temperature has been implemented, this is going to be used to help decide whether the plug will be turned on or off:

The API that was used for this is `api.openweathermap.org` as there is a lot of community support behind this which helped in the implementation.

```
1. String APIKEY = "076836f4d5f3a9696e7983d56fc515ab";
2. String CityName = "Nottingham";
3. char servername[] = "api.openweathermap.org";
```

To use the API, it requires a key to help the developers track what the API is being used for. At this stage of the development the city name is hardcoded but in the next stage the user will be able to change this in the app.

```
1. if (client.connect(servername, 80))
2.   { //starts client connection, checks for connection
3.     client.println("GET /data/2.5/weather?q=" + CityName +
4.       ",UK&units=metric&APPID=" + APIKEY);
5.     client.println("Host: api.openweathermap.org");
6.     client.println("User-Agent: ArduinoWiFi/1.1");
7.     client.println("Connection: close");
8.     client.println();
9.   }
10.  else {
11.    Serial.println("connection failed");           //error
12.    message if no client connect
13.    Serial.println();
14.  }
```

The code above is used in the function which is called every few minutes to get the most up to date temperature and is used to connect to the server and retrieve the weather information. Once connected it will then read that data into a variable called 'result' on the NodeMCU:


```

1. while (client.connected() && !client.available())
2.     delay(1); //waits
   for data
3.     while (client.connected() || client.available())
4.     { //connected or data available
5.         char c = client.read(); //gets byte
           from ethernet buffer
6.         result = result + c;
7.     }
8. client.stop(); //stop
   client
9.     result.replace('[', ' ');
10.    result.replace(']', ' ');
11.    char jsonArray[result.length() + 1];
12.    result.toCharArray(jsonArray, sizeof(jsonArray));
13.    jsonArray[result.length() + 1] = '\0';
14.    StaticJsonBuffer<1024> json_buf;
15.    JsonObject &root = json_buf.parseObject(jsonArray);
16.    if (!root.success())
17.    {
18.        Serial.println("parseObject() failed");
19.    }
20.    float temperature = root["main"]["temp"];
21.    OutsideTemperature = temperature;

```

Once the data has been read into the results variable the square brackets are removed and is then put into a JSON array and then parsed which allows for the temperature data to be filtered out and put into a float variable. This temperature variable is then used in the updated main code loop.

4.3.4.2 Main code loop

With the new external temperature, the main code loop has been updated first to get the external temperature and then implemented in the switch statement to make the plug more context-aware.

```
1. if (counter == 300)
2.   {
3.     counter = 0;
4.     getWeatherData();
5.   } else {
6.     counter++;
7.   }
```

The above code is used to update the temperature every 300 loops of the code which is on average 15 minutes long. This ensures the temperature is relatively up to date without spamming the server for updates.

```
1. case 2:
2.     //Fan mode
3.     Serial.println("Fan mode");
4.     internalTemperature();
5.     if (OutsideTemperature < temperature()) {
6.         Firebase.setInt(Status, 0);
7.         Serial.println("Plug is off");
8.         digitalWrite(relayInput, LOW);
9.     }
10.    if (OutsideTemperature >= temperature()) {
11.        if (temperature() > thresholdTemperature()) {
12.            Firebase.setInt(Status, 1);
13.            Serial.println("Plug is on");
14.            digitalWrite(relayInput, HIGH);
15.        }
16.        if (temperature() < thresholdTemperature()) {
17.            Firebase.setInt(Status, 0);
18.            Serial.println("Plug is off");
19.            digitalWrite(relayInput, LOW);
20.        }
21.    }
22.
23.    break;
24. case 3:
25.     //radiator mode
26.     Serial.println("Radiator mode");
27.     internalTemperature();
28.     if (OutsideTemperature <= temperature()) {
29.         if (temperature() < thresholdTemperature()) {
30.             Firebase.setInt(Status, 1);
31.             Serial.println("Plug is on");
```

```

32.         digitalWrite(relayInput, HIGH);
33.     }
34.     if (temperature() > thresholdTemperature()) {
35.         Firebase.setInt(Status, 0);
36.         Serial.println("Plug is off");
37.         digitalWrite(relayInput, LOW);
38.     }
39. }
40. if (OutsideTemperature > temperature()) {
41.     Firebase.setInt(Status, 0);
42.     Serial.println("Plug is off");
43.     digitalWrite(relayInput, LOW);
44. }
45. break;
46. default:
47.     break;
48. }

```

The above code is the core switch statement which is used to get the current mode of the plug. The algorithm used to find out if the plug should be on or off for the fan and radiator mode has been updated to implement the new external temperature feature. This uses the proposed algorithm in chapter 3.8.3.2 which is has been designed to help save the user money by only turning on when the external temperatures are greater than (plug) or lower than (radiator) the internal.

4.3.4 System and User Testing

This is the second phase of the user and system testing which will be used to help develop and improve the system which will be then implemented in phase

3.

4.3.4.1 User Testing

Again, another five users were asked about the positives and negatives of the application at its current stage –

Positives:

- The new device tiles are much better and clearer.
- The new pages offer a lot of new information that helps use the app.
- The app looks very professional.
- The app has a similar theme throughout.

Negatives:

- Round buttons could be switched the square buttons to make the app uniform throughout.
- The 'Profile' page log out button does not suit the rest of the app as it does not stand out as much.
- The buttons on the 'My Device' page that have the icons are too close to the top.
- In some areas of the app, it uses Degrees Celsius and then in other places it uses °C.

As the application has received very little negative feedback, it is a good sign that the app design is in the right place.

4.3.4.2 System Testing

From general use of the system and some testing that has been carried out, there have been a few bugs and issues that have appeared, these include:

- When the internet is disconnected, the plug will remain on, which means the user will not be able to turn it off. This could be solved by turning the plug off when there is not a network connection.
- The user is unable to see the external temperature which the plug can see, which means they will not know how far off the temperature is from turning on/off the plug.
- When a new user creates an account, they can see other users' devices; a user ID tag can be added to the database so only one user can see their devices.
- When the keyboard is open, and the user presses the update button for the threshold temperature the keyboard will stay open.

4.4 Development Phase 3

This is the last phase of development where everything that can be implemented has been implemented as well as further changes made based on system and user testing.

4.4.1 Response to Testing

4.4.1.1 Response to User Testing

In response to the user's feedback on the buttons being different throughout the application, they were shown a few images on the design of the buttons to see which they preferred as to keep in line with Google's design guidelines of a consistent layout throughout. The guidelines that have been followed throughout the development of the app can be found in Appendix C.

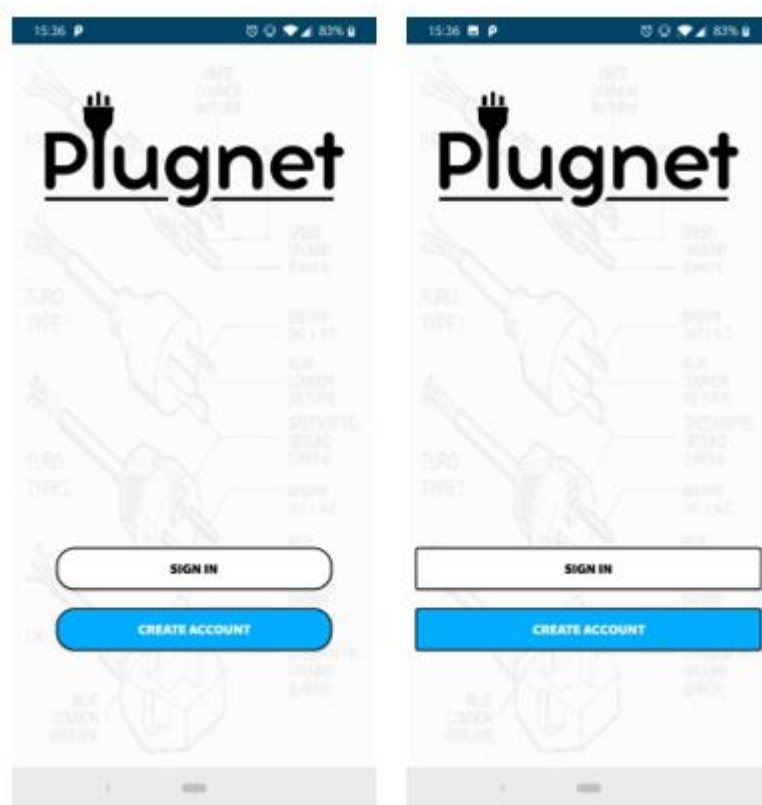


Figure 42 - User testing home screen change

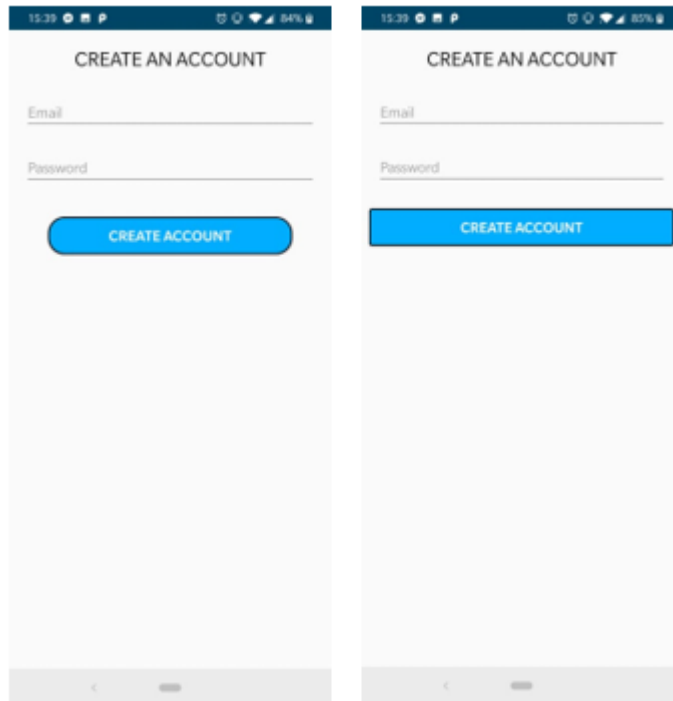


Figure 43 - User testing create account screen change

Users were asked which button layout they preferred more, the rounded and the square from the two Figures 42 and 43 while keeping in mind the fact that the buttons throughout the rest of the app are square as shown in Figure 39. The users thought the round looked more courteous on the home screen while the square looked nicer on the create account page, from this it was decided that the app will use the square buttons as to keep the app layout consistent.

The button on the Profile page has also changed for layout consistency as shown in Figure 44 where the sign out button has been made to look the same:

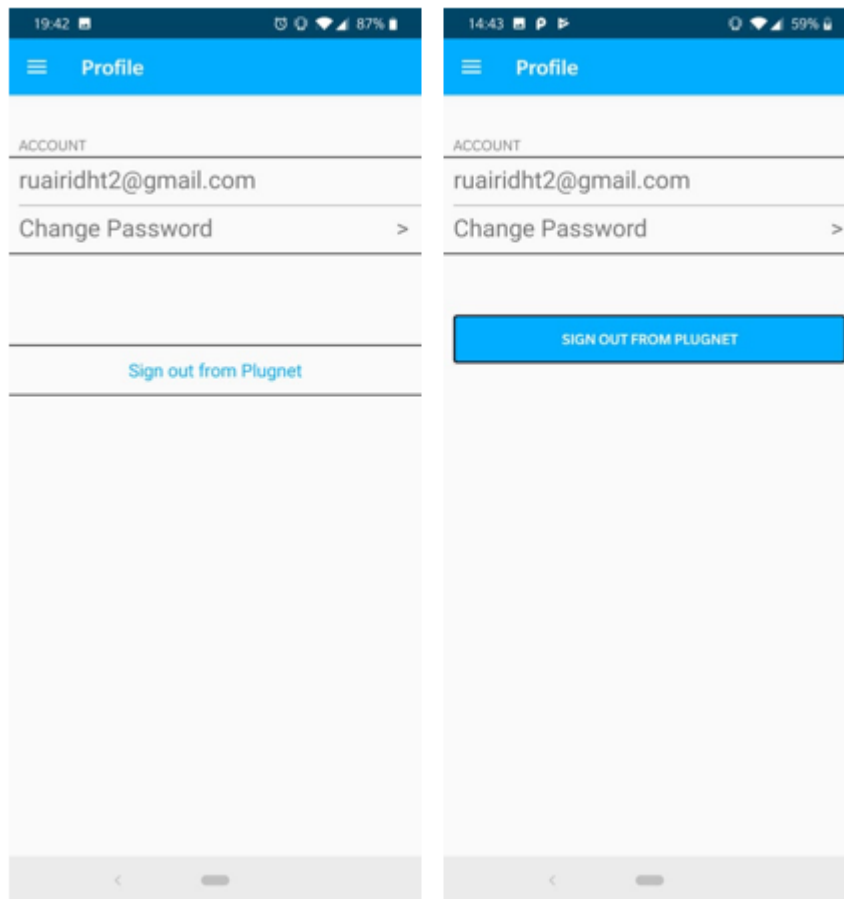


Figure 44 - Changes to the profile screen

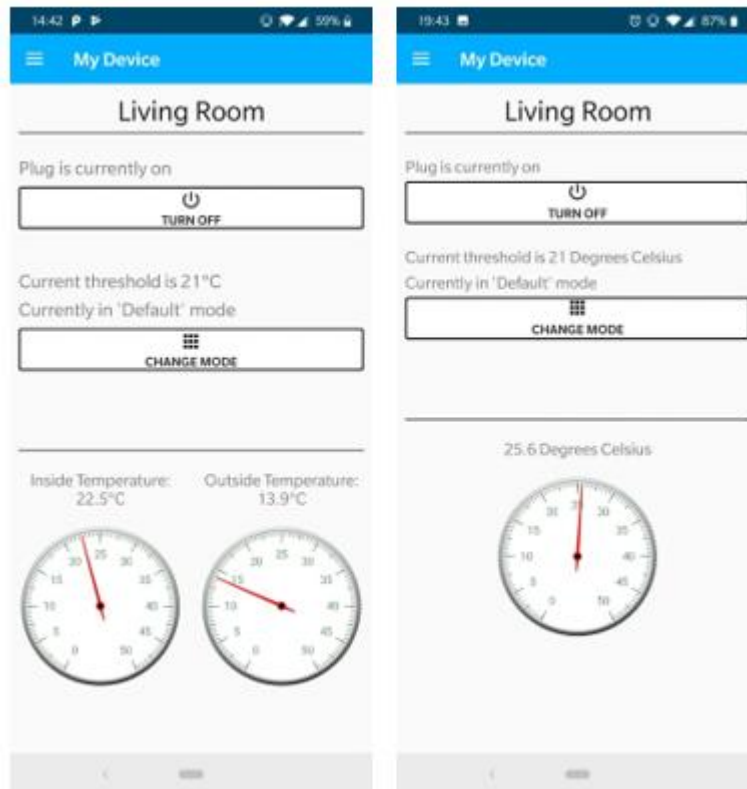


Figure 45 – Changes to device screen

As shown in Figure 45 a few changes have been made from the user testing, firstly the icons shown on the buttons have been moved down slightly so that they are not merging in with the button border. As well as this a second temperature gauge has been added to show the external temperature. Temperatures have also been edited to show '°C' instead of 'Degrees Celsius'.

4.4.1.2 Response to System Testing

Plug staying on when connection lost – When the plug loses internet connection the device will stay on, this could cause issues if the user is not home to manually turn the plug off, to fix this the relay is set to LOW when trying to reconnect as shown:

```
1. void WIFI_Connect() {
2.   WiFi.disconnect();
3.   WiFi.begin(WIFI_SSID, WIFI_PASSWORD);
4.   Serial.print("connecting");
5.   while (WiFi.status() != WL_CONNECTED) {
6.     digitalWrite(relayInput, LOW);
7.     Serial.print(".");
8.     delay(500);
9.   }
10.  Serial.println();
11.  Serial.print("connected: ");
12.  Serial.println(WiFi.localIP());
13. }
```

User cannot see the external temperature from the app – If the user is able to see the external temperature, they will have a better understanding of when the plug is going to be turned on and off. Code has been added to both the application and the NodeMCU, which has been implemented the same as the internal temperature.

Keyboard staying open – When the update button is pressed in the app the keyboard which is used to update the threshold temperature stays open, this has been fixed with the following lines of code:

```
InputMethodManager imm =
(InputMethodManager) getActivity().getSystemService(Context.INPUT_METHOD_SERVICE);
if (imm.isAcceptingText()) {

imm.hideSoftInputFromWindow(getActivity().getCurrentFocus().getWindowToken(), 0);
}
```

The code above checks to see if the keyboard is open by checking if it is accepting text, if it is then it will hide the keyboard, and if nothing is there it will not do anything.

4.4.2 Plugnet Application

In this stage of the development, the only change made to the application other than changes made in the testing was the implementation of the location changing, this feature allows for users to update the city they are in through the application as shown in Figure 46:

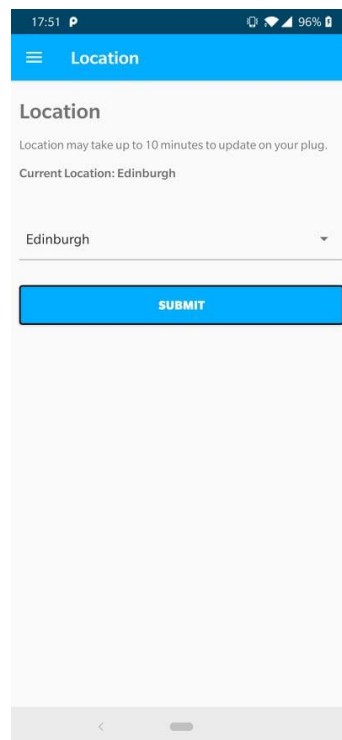


Figure 46 - Location screen

The app uses a drop-down menu or 'spinner' to show a selection of cities located within the UK.

```
btnSubmit.setOnClickListener(new View.OnClickListener() {  
    @Override  
    public void onClick(View v) {  
        String location = String.valueOf(spinner1.getSelectedItem());  
        updateLocation(location);  
        Toast.makeText(getApplicationContext(), "Updated",  
            Toast.LENGTH_SHORT).show();  
    }  
});
```

The code above shows that when the submit button is clicked, it finds the value at which the spinner is currently at and from that gets the cities name, this name is uploaded to Firebase where the plug can read it.

4.4.3 Firebase Database

No significant changes have been made to the database apart from the addition of the location and the external temperature.

4.4.4 Context-Aware Smart Plug

In this final stage of development, a few changes have been implemented to help improve how the plug functions and the plug has been made fully functional.

4.4.4.1 Changes to Fan and Radiator Algorithm

Changes have been made to the fan and radiator algorithm as it was found that it could be improved. Before the changes were made if the temperature outside were colder than the inside the fan would not turn on, this, however, did not account for both the inside and outside temperatures being hot, and cold in the case for the radiator. To fix this issue, the code has been tweaked as shown below:

```
1. case 2:
2.     Serial.println("Fan mode");
3.     if (OutsideTemperature < temperature()) {
4.         if (OutsideTemperature > thresholdTemperature()) {
5.             Firebase.setInt(Status, 1);
6.             Serial.println("Plug is on");
7.             digitalWrite(relayInput, HIGH);
8.         }
9.         if (OutsideTemperature < thresholdTemperature()) {
10.            Firebase.setInt(Status, 0);
11.            Serial.println("Plug is off");
12.            digitalWrite(relayInput, LOW);
13.        }
14.    }
15.    if (OutsideTemperature >= temperature()) {
16.        if (temperature() > thresholdTemperature()) {
17.            Firebase.setInt(Status, 1);
18.            Serial.println("Plug is on");
19.            digitalWrite(relayInput, HIGH);
20.        }
21.        if (temperature() < thresholdTemperature()) {
22.            Firebase.setInt(Status, 0);
23.            Serial.println("Plug is off");
24.            digitalWrite(relayInput, LOW);
25.        }
26.    }
27.
28.    break;
29.    case 3:
30.        Serial.println("Radiator mode");
31.        if (OutsideTemperature <= temperature()) {
32.            if (temperature() < thresholdTemperature()) {
33.                Firebase.setInt(Status, 1);
34.                Serial.println("Plug is on");
35.                digitalWrite(relayInput, HIGH);
36.            }
37.            if (temperature() > thresholdTemperature()) {
38.                Firebase.setInt(Status, 0);
39.                Serial.println("Plug is off");
40.                digitalWrite(relayInput, LOW);
41.            }
42.        }
43.        if (OutsideTemperature > temperature()) {
```

```
44.         if (OutsideTemperature < thresholdTemperature()) {
45.             Firebase.setInt(Status, 1);
46.             Serial.println("Plug is on");
47.             digitalWrite(relayInput, HIGH);
48.         }
49.         if (OutsideTemperature > thresholdTemperature()) {
50.             Firebase.setInt(Status, 0);
51.             Serial.println("Plug is off");
52.             digitalWrite(relayInput, LOW);
53.         }
54.     }
55.     break;
```

The code that has been edited has been highlighted and shows that when in fan mode if the outside temperature is lower than the inside temperature it will check to see if the outside temperature is greater than the threshold if it is then the plug will turn on, if not the plug will turn off. Moreover, the same for the radiator mode but the exact opposite. This change makes more use of the user's threshold and will help to maintain the users preferred temperature.

4.4.4.2 Further Hardware Development

To make the plug fully functional it requires a male connector, a female connector and a wire, where the wire will connect to the relay where the flow of electricity is controlled, the male connector will plug into the wall, and the female connector will be where the user can plug their devices into.

The standard UK cable has three wires:

- Protective Earth (PE) – Green and yellow striped
- Neutral (N) – Blue
- Single Phase: Line (L) – Brown

The Line wire is the wire required to go into the relay, in the case of this plug it one end has been placed into the normally open port, and the other end has been placed into the common port as shown in Figure 47:



Figure 47 - Relay connected male and female plug connectors

4.5 Justification of Missing Functionality

The complete system is functional. Be that as it may, not all the planned aspects of functionality could be implemented. The fact that the system is missing this functionality has affected how this system operates and how users interact with the system.

4.5.1 User Accounts

The idea behind implementing user accounts into the system is to allow for individual users to have access to only their plugs within the database. The issue arose because Firebase Real-time Database does not allow for querying on embedded collections whereas this can be done within Cloud Firestore. However there are no libraries currently available for Cloud Firestore which would make switching over extremely difficult as most of the Plugnet application would need to be changed as well as changing the entire code on the NodeMCU to work with Cloud Firestore, so although it is possible, it would be a challenge this late into the development.

4.5.2 Machine Learning

Another planned piece of functionality which had to be removed was the inclusion of machine learning. Machine learning, was originally going to be used in order to determine what time would be the best to have the plug turned on or off, however, Google Cloud platform which was going to be used as it can be linked with Googles Firebase, but after trying to implement it, it was found that this also only works with the Cloud Firestore and not the Realtime Database. Though, this feature is not detrimental to the system as it functions exceptionally well without it.

4.6 Summary

Overall, the implementation of the Plugnet system was an overall success. As the core functionality of the system was implemented successfully as the user can select the device they have plugged in and control it from the application. The plug uses sensors and API information to get its context and react based on this by turning on and off the plugs at different temperatures. However, a couple of planned aspects were never implemented into the system for reasons outlined in chapter 4.5.

CHAPTER 5

RESULTS / DISCUSSION

5.1 Introduction

The original aim of this project was to produce a plug that has context-aware capabilities, which can be controlled through a phone app. To determine whether the final system has met this original aim, it has had to be exposed to several tests, these test focus on performance, functionality and usability.

Some of the performance testing will be completed manually which means they will only be accurate to a particular deviation; different performance tests have been completed to test the timings of certain aspects of the system such as manually turning the plug off, etc.

5.2 Changes made to the algorithm

In chapter 3.8.3 the algorithm that was initially designed in this chapter to be implemented into the plug, however, during the implantation and testing it was found that it needed to be slightly tweaked as originally in fan mode if the temperature outside was colder than inside the fan would not turn on. This will be an issue if both the inside and the outside are both hot. Therefore, this has been changed to include the threshold temperature, if the threshold is less than the outside temperature the fan will turn on. The same has been changed for the radiator mode where if the temperature inside and outside are both cold then the radiator will turn on.

5.3 Testing Environment

To get the best results possible, they have been obtained in an environment in which it would usually get used (the authors home). The plug has been plugged into a standard UK plug socket which produces around 230 volts at 50Hz. Both the plug and application are connected to the internet through a dual-band router as using a mobile hotspot is not representative of regular use and could affect results.

As the tests were carried out in the authors home, no preparations were needed to set up the system, other than to gain access to the SSID and password of the router.

5.4 System Performance

One of the aspects which have been measured during the testing was its performance. The performance of the system is the amount of time take to complete specific tasks, such as switching between the different modes, updating temperatures, etc.

The total size of the Plugnet phone application is 4.26MB where the average android application is around 11.5MB.

5.4.1 Opening the application

The first test that had been carried out was the time taken (ms) to open the application. To obtain these results the phone was plugged into a computer running android studio, and by filtering the log files by 'Displayed' shows the times taken to open the application. This test was carried out several times to obtain an average; the results are as shown:

Table 4 - Time taken to open the Plugnet mobile application

| Test number | Time taken (milliseconds) |
|----------------|---------------------------|
| 1 | 521 |
| 2 | 553 |
| 3 | 506 |
| 4 | 498 |
| 5 | 493 |
| Average | 514.2ms |

Table 4 illustrates that it takes on average 514ms (0.5 seconds) to open the application. How the results have been obtained are shown in figure 48:

```
14:35:00.953 888-1160/? I/ActivityManager: Displayed com.ruairidh.plugnet/.beginningActivity: +521ms
14:36:42.820 888-1160/? I/ActivityManager: Displayed com.ruairidh.plugnet/.beginningActivity: +553ms
14:38:06.600 888-1160/? I/ActivityManager: Displayed com.ruairidh.plugnet/.beginningActivity: +506ms
14:38:09.918 888-1160/? I/ActivityManager: Displayed com.ruairidh.plugnet/.beginningActivity: +498ms
14:38:12.851 888-1160/? I/ActivityManager: Displayed com.ruairidh.plugnet/.beginningActivity: +493ms
```

Figure 48 - Time taken to open the application

5.4.2 Updating Options

There are a few different options which can be updated on the mobile application by the user, these being the status, the mode, the location and the threshold temperature. This test calculates how long it takes from changing these options on the phone to the plug updating. The purpose of this test is to find out how long on average it will take from the user clicking a button to the plug turning off.

This test will be completed while the plug is in default mode, fan mode and radiator mode and it will be the time taken from the user pressing the turn on/off button to the plug turning off. As the timings are going to be done by the

author, there is going to be a margin of error in these results. The results are as follows:

Table 5 - Results showing the time taken from user input to a change in the plug

| Test Number | Time taken (milliseconds) | | |
|----------------|---------------------------|---------------|---------------|
| | Default Mode | Fan Mode | Radiator Mode |
| 1 | 5716 | 8926 | 8061 |
| 2 | 4848 | 8064 | 8547 |
| 3 | 6638 | 7789 | 7782 |
| 4 | 6536 | 8222 | 8331 |
| 5 | 5547 | 7870 | 7579 |
| Average | 5857 | 8174.2 | 8060 |

There results in Table 5 show that the using the default mode is about 2 seconds quicker than both the fan and the radiator mode, this is expected as these two modes have more tests that are carried out which take longer. All of the above times could be reduced by a couple of seconds removing the delay; however, the short time would not affect the functionality of the system.

5.5 System Functionality

The functionality tests involved making sure the software and the hardware functioned as expected. Tests have been carried out on all three components of the systems which include; the application, the web service, and the plug. The detailed results from the testing can be found in Appendix D.

5.5.1 Application functionality

The application has been thoroughly tested, 20 functionality tests have been carried out on the application ensuring that everything worked as expected and that there are no bugs or glitches. Of the 20 tests, only 1 of them failed.

The test that failed was apart of the update button which is used for updating the name, mode, and threshold. This is because the keyboard used for updating the threshold would not close after the button had been pressed, which caused issues with pressing the back button. This issue was quickly fixed by checking if the keyboard was open and then closing it if it was.

5.5.2 Firebase web service functionality

The second section of the system that was tested was the web service; this also links back to the application functionality as Firebase which is used for the web service is also responsible for the user authentication. The web service is responsible for storing data which can be accessed by both the application and the plug; this can also be updated by them both.

All the tests that were carried out completed as expected and so all passed, as the database are non-SQL it does not have to have a data type assigned to it, and so any data can be entered into the database without causing any issues.

5.5.3 Plug functionality

The final part of the system that was tested was the plug; this part of the system is responsible for obtaining information from firebase, external APIs and onboard sensors. This information is used to either turn on or off the plug as well as update information on the web service so that the user has the most up to date information.

The only test that failed was related to an internet connection issue as when the plug lost its connection it would not be able to connect again due to where the original function was placed, this has now been resolved in the development.

5.6 System Usability

The usability testing phase was completed both during and at the end of the implementation, the tests carried out during the development helped with finding issues with the application as well as giving feedback on the design. At the end of the implementation of the users were asked five questions; these results have been put into graphs which can be found in Appendix E.

The results gained from the questions at the end of the testing were all positive, as all the users liked the design, navigation and functionality of the application. This shows that the final product was up to a high standard and there was little they thought could be improved upon. One of the questions asked if they thought was comparable on design and usability to apps, they would use daily, and of the 5 one of them said yes, and the other four said it is close, but they felt it was missing something.

5.7 Alternative testing

An alternative test that could be carried out would be a Portable Appliance Testing (PAT) as this is a process that tests electrical appliances for safety. This would be useful as the plug contains an electrical component and the safety surrounding this device is unsure. This test, however, was not carried out due to the costs of either renting or buying a PAT machine.

5.8 Costs

The cost of the system is the final aspect which needs to be analysed within this chapter as initially, the idea was to provide a smart plug with context-aware functionality for less-than or the same as an ordinary smart plug. Therefore, the average cost of should be found, this will focus mainly on the low end cheaper smart plugs, as the plugs reviewed in the context chapter were all high end and relatively expensive.

After a quick scan on Amazon, the cheaper plugs would cost between £9 - £14, while the plugs that were looked at during the context chapter cost between £20 - £40.

The cost of the system should consider the overall cost of the parts used to create the plug; this is made up of 1 x NodeMCU, 1 x DH11 Temperature sensors, 1 x Relay, 6 x Wires, and 1 x extension lead, which comes to a total of around £8.39. Therefore, the overall cost of production is lower than the cost of these other smart plugs; however, to make a profit these would have to be marked up slightly to around the cost that they are charging.

5.9 Summary

The implementation of the system has been tested on the performance, functionality and usability. These tests show that all aspects of the system have been tested and improved resulting in a system which performs well, functions as it should and is usable by an average user. However as stated at the end of the last chapter some functionality is missing causing the system to not have all of its original functionality, these areas of improvement have been identified, and thus can be addressed in future developments of the system.

CHAPTER 6

CONCLUSIONS / FUTURE WORK

6.1 Introduction

The purpose of this project was to create an inexpensive context-aware IoT smart plug for an everyday consumer. The use of context-awareness within the system has been developed to maximise the user's comfort while minimising the user's interaction with the plug.

The result of the implementation and testing stage shows that the system is operational. However, some of the functionality was removed to ensure the core functionality works. As the system is missing some of the functionality, the system is mostly but not entirely successful. This chapter discusses the reasons why the project was not entirely successful and discusses the future work to be completed on the system to make it fully functional as well as additional features to be added.

6.2 Delivery of Projects Aims and Objectives

As defined in Chapter 3 and Appendix A of this report, the success of the project is to be decided on how the implementation of this project stack up against the aims mentioned in these chapters. From the original aims of the project found in project planning document to the aims later revised in Chapter 3 a few things have changed, despite the core functionality has stayed the same; this was originally defined as:

"create a prototype context-aware smart plug and an android phone application to allow users to turn on and off their plug from anywhere, providing they have

an internet connection. The hardware must be controlled from the phone application; the phone application must be easy to use.”

From this original aim, it can be said that the implementation has all of the required functionality as the user can control the context-aware plug providing they have an internet connection. However, the missing functionality of linking the plug to a user account has not been implemented so any user that creates an account will have control over any plugs added to the system.

There were also several original objectives of the project which can also be found in Appendix A; the objectives were split into two categories, the necessities and the desirables. There were six necessity’s which were originally outlined, of the six, two were dropped during Chapter 3 as the original idea has been developed upon to improve the context-awareness of the system. There were also three desirable features; none of these made it into the final implementation for the same reason.

In addition to these objectives found in the Project Planning Document, another three system requirements have been added in Chapter 3.3. These are:

- 1. The plug should be context-aware allowing it to make decisions based on the type of device that’s plugged into it and the current device's environment.**

This functionality has been implemented successfully within the system as the user can select from three different modes on the mobile application which device they have plugged in, and this is then uploaded to Firebase where the plug can find out what device it is and act accordingly.

- 2. The plug should gather information about its environment using web services to gather information about its location, time and temperature.**

Web services have been used in this project to collect information about the external temperature using web services, however, during the development phase, the location of the device was altered slightly to be done on the mobile application. The plug is also connected to an NTP server to get the current time and date for the database.

3. The device should be cheap to make; this will be made easier with the use of embedded hardware as it is relatively inexpensive.

In Chapter 5.8 the costs were found to be about the same cost as a low-end smart plug and if these components were to be sourced differently and in bulk could bring the cost way down. Therefore, this objective has been successfully met.

6.3 Overall Outcome

Based on the overall outcome of the implementation, testing and results, it could be said the project was an overall success as the user testing showed extremely positive results. However, as a piece of essential functionality had not been implemented which would affect this system commercially the system is only partially successful.

The functionality that was not implemented was linking a device to a user account; this function would not be an issue if a different Firebase database were created for each system. However, this is not commercially viable. By adding a user ID to each device would allow the application to sort the database based on the user ID and only provide them with devices they own. This would successfully work as an online DIY project as the user could create their own database.

6.4 Future Work

With the implementation of the project not including everything that has been set out, there are several additional features which could be implemented into the system to increase its functionality, scalability and performance.

6.4.1 Switching Databases

Firebase offers an alternative database called Cloud Firestore this database is better than their real-time database in many ways; most importantly it has a more intuitive data model which allows for better querying. With this better querying, it would allow for user IDs to be stored with the device fixing the issue the system currently faces. Firestore also offers faster queries as well as better scalability than the real-time database. However, this would rely on a library being released for ESP8266 that enables data transfer between the two, on top of this would require a lot of the application to be rewritten to switch from one to another.

6.4.2 iOS Application

According to DeviceAtlas iOS users take up 56.59% of the UK mobile market, therefore, potentially missing out on over 50% of the market. This would be relatively simple as Firebase also can be used on iOS devices and so would be as easy as creating the UI. This would, however, have to meet with Apples strict App Store guidelines.

6.4.3 Web Interface

Instead of downloading an application to be able to control the plug, a web interface could be developed to allow users to control their plug from their web

browsers, without the need to use an application. This would also be extremely simple to do as Firebase also allows for web access through APIs.

6.4.4 Scheduler

A scheduling function would be implemented to add additional functionality to the system; it could be used for a wide variety of things such as allowing the user to set their lights to be on a specific time and switch off at a specific time. This could be used for a variety of features and would be relatively simple to set up considering the plug already gets the current time and so would be a matter of adding another mode and configuring the mobile app to add this. Also, this feature could be added at no extra cost.

6.4.5 Additional Sensors

The addition of more sensors could also be used for more functionality as the plug would be able to gather more information about its surroundings. This could include something like a photoresistor which could be used to turn on or off lights based on how bright the room is. There is a lot of additional sensors which could span into different products such as there could be a plug which is dedicated to gas as a gas sensor could be fitted and used to turn off the gas if a gas leak is detected.

6.4.6 Additional APIs

Like the addition of sensors, the addition of APIs could be used to add more functionality as the plug could use location data to find out things about the area in which it is located. An API could be used to find the location of the plug which would take this out of the hand of the user as this is currently configured within the application.

6.4.7 Amazon Alexa and Google Home

As most smart plugs include this feature, it would be advantageous to add this feature as people may buy a plug for voice control. This would allow for users to control their devices with Google Home or Amazon Alexa meaning they would not need to open any interface to control their devices; each device can already be renamed which would make it easier to implement this.

6.4.8 Easy Set-Up

The final suggestion to be made for future improvement of the system is the ability for the user to add new plugs through the mobile app, this, however, would require the use of an additional communication protocol such as Bluetooth. As to give the plugs access to the internet currently requires the user to change the SSID and Password and upload it to the NodeMCU, where an additional protocol would allow for the phone to send these details to the plug.

6.5 Evaluation of Professional, Social, Ethical and Legal Issues

In the Project Planning Document, the PSEL issues were discussed and how they would need to be implemented throughout the entirety of the project. Now since the project is almost over each will be discussed again on how they have impacted the project:

6.5.1 Professional Issues

The first aspect that was defined from the beginning of this project was to act with integrity and respect when working with others, as it states within the BCS Code of Conduct, "act with integrity and respect in your professional relationships with all members of BCS and with members of other professions

with whom you work in a professional capacity” (BCS, 2015). This has been applicable throughout the project and will continue to be followed once the project has finished.

Another aspect that has been followed is to seek feedback, as stated within BCS Code of Conduct also states that one must “respect and value alternative viewpoints and, seek, accept and offer honest criticisms of work” (BCS, 2015). This has been followed throughout the project to help improve the project outcomes.

Finally, the aspect which would be used within the project was that in the interest of the public any data collected or stored about users using the product is kept secure. As it states within the BCS Code of Conduct, “have due regard for public health, privacy, security and wellbeing of others and the environment” (BCS, 2015). To keep in line with this code of conduct all user data was stored using Googles Firebase to ensure that it is stored securely.

6.5.2 Social Issues

The main goal of the design of the application was to make it accessible to as many people as possible, the BCS Code of Conduct states “promote equal access to the benefits of IT and seek to promote the inclusion of all sectors in society” (BCS, 2015). This code of conduct influenced the way the application was tested and implemented to ensure it was made accessible and usable to as many people as possible.

Another social issue surrounding not just this project, but the internet of things is that it is affecting health and fitness as computers are removing the need for us to physically carry out tasks, as well as keeping us rooted to one spot throughout the day. This is not something that could be addressed within this project as it is industry wide.

6.5.3 Ethical Issues

The main ethics issue this project has faced was the safety surrounding the wiring of the plug, and therefore this policy has been upheld "hold paramount the safety, health, and welfare of the public, to strive to comply with ethical design and sustainable development practices, and to disclose promptly factors that might endanger the public or the environment" (IEEE policies, 2018). Following this policy requires the disclosure that there might be some danger when using the system as there is a live current. As to avoid the risk of injury the product has been placed inside of a plastic box which will mitigate the risk of harm. To further add to this, for the demonstration of the project the cables will be safely stored in a plastic socket wall mount box to ensure the safety of everyone.

Another ethical issue faced during the design and development of the project is the effect this may have on the environment. The design of algorithm used was created so that the plug should be only on when it is needed and not any longer, this, in theory, should reduce the amount of electricity used and therefore reduce the carbon footprint.

6.5.4 Legal Issues

As for legal issues surrounding the project, the Data Protection Act, 2018 implements the European Directive known as the General Data Protection Regulation (GDPR) into UK law. The new Data Protection Act 2018 provides that data is to be "handled in a way that ensures appropriate security which includes protecting data against unlawful or unauthorised processing, access, loss, destruction or damage". The changes in the law have impacted the project as of the way the user's information has been stored. Also, the new act has also provided more rights to the consumer which means they will have new extended

rights under the Data Protection Act 2018. One right of importance is the right of access which enables the data subject to ascertain what information is held about them by us. Therefore, if the product was to be released to the public, a feature will need to be added to the application to allow users to request this specifically.

The above act also defines how the data must be legally used, ensuring the user's data is not misused or taken advantage of. In addition to this law, it should also comply with the BCS Code of Conduct which states that confidential data must not be disclosed, ensuring the safety of the data.

6.6 Synoptic Assessment

After completing university my plan is to work within the computing field. However I am unsure of what area I want to focus on because during my placement I enjoyed working within the IT infrastructure area, but while at university I have enjoyed the mobile development module, so my options are open.

The good thing about my project is that it was quite broad as I did a bit of mobile development, a bit of embedded programming and working with external web services. Due to the broadness of the system, there have been many skills gained which can be transferred to the different areas of computing I could work within. These skills include:

Independent Learning -

As the entire report and solution had to be done independently, it has improved my skills of working on my own, figuring things out and then implementing them.

Problem-solving –

Throughout the development and even after I have been faced with issues which needed to be solved, these range from small issues that can be fixed quickly to more significant issues that take days to solve.

Object-oriented programming –

In the development of the application I used object-oriented programming to hold data, the use of object-oriented programming is used across many programming languages and so is transferable.

Research –

Extensive research was completed during Chapter 2; this skill can be transferred to all aspects of computing as research is needed to help develop and solve issues.

Written Communication –

Furthermore, writing this report has helped improve my written communication skills as I had to summarise technical information in a way in which everyone can understand.

REFERENCES

- Barr, M. (2001). Introduction to Memory Types. [online] Available at: <https://www.embedded.com/electronics-blogs/beginner-s-corner/4023326/Introduction-to-Memory-Types> [Accessed 22 Jan. 2019].
- Barr, M. and Massa, A. (2007). *Programming embedded systems*. Beijing: O'Reilly.
- BCS. (2015). *BCS Code of Conduct*. Available: <https://www.bcs.org/category/6030>. Last accessed 14th October 2018.
- Cao, H., Leung, V., Chow, C. and Chan, H. (2009). Enabling technologies for wireless body area networks: A survey and outlook. *IEEE Communications Magazine*, [online] 47(12), pp.84-93. Available at: <https://ieeexplore.ieee.org/abstract/document/5350373> [Accessed 13 Feb. 2019].
- Context-aware IoT. (2019). *Systems and Computer Engineering*. [online] Available at: <https://carleton.ca/internetofthings/context-aware-iot/> [Accessed 16 Apr. 2019].
- Daftari, A., Mehta, N., Bakre, S. and Sun, X. (2010). On Design Framework of Context Aware Embedded Systems. [online] Available at: https://www.researchgate.net/publication/228581512_On_design_framework_of_context_aware_embedded_systems [Accessed 16 Apr. 2019].
- Farahani, S. (2008). *Designing ZigBee networks and transceivers*. Oxford: Newnes.
- Fernandes, J. and Wentzloff, D. (2010). Recent advances in IR-UWB transceivers: An overview. *Proceedings of 2010 IEEE International Symposium on Circuits and Systems*. [online] Available at: <https://ieeexplore.ieee.org/abstract/document/5537916> [Accessed 10 Feb. 2019].
- Hou, L., Zhao, S., Xiong, X., Zheng, K., Chatzimisios, P., Hossain, M. and Xiang, W. (2016). Internet of Things Cloud: Architecture and Implementation. *IEEE Communications Magazine*, [online] 54(12), pp.32-39. Available at:

https://www.researchgate.net/publication/308646413_Internet_of_Things_Cloud_Architecture_and_Implementation [Accessed 16 Apr. 2019].

Hu, P., Indulska, J. and Robinson, R. (2008). An Autonomic Context Management System for Pervasive Computing. *2008 Sixth Annual IEEE International Conference on Pervasive Computing and Communications (PerCom)*. [online] Available at: <https://ieeexplore.ieee.org/document/4517396> [Accessed 11 Feb. 2019].

IEEE. (2018). *IEEE Code of Ethics*. Available: <https://www.ieee.org/about/corporate/governance/p7-8.html>. Last accessed 14th October 2018.

ITU. (2019). *Internet of Things Global Standards Initiative*. [online] Available at: <https://www.itu.int/en/ITU-T/gsi/iot/pages/default.aspx> [Accessed 8 Feb. 2019].

Nixon, M. (2012). A Comparison of WirelessHART™ and ISA100.11a. [online] Available at: <https://www.controlglobal.com/assets/12WPpdf/120904-emerson-wirelesshart-isa.pdf> [Accessed 13 Feb. 2019].

Olsson, J. (2014). 6LoWPAN demystified. [online] Available at: <https://www.ti.com/lit/wp/swry013/swry013.pdf> [Accessed 21 Feb. 2019].

Schilit, B. and Theimer, M. (1994). Disseminating active map information to mobile hosts. *IEEE Network*, [online] 8(5), pp.22-32. Available at: <https://ieeexplore.ieee.org/document/313011> [Accessed 16 Apr. 2019].

Song, J., Han, S., Zhu, X., Mok, A., Chen, D. and Nixon, M. (2008). A complete wirelessHART network. *Proceedings of the 6th ACM conference on Embedded network sensor systems - SenSys '08*. [online] Available at: https://www.researchgate.net/publication/221091410_A_complete_wirelessHART_network [Accessed 16 Apr. 2019].

Sran, P. (2009). Wimax :Its Features and Applications. [online] Available at: https://www.researchgate.net/publication/281291210_Wimax_Its_Features_and_Applications [Accessed 31 Jan. 2019].

Techopedia.com. (n.d.). *What is an Embedded System? - Definition from Techopedia*. [online] Available at: <https://www.techopedia.com/definition/3636/embedded-system> [Accessed 16 Feb. 2019].

The Stationery Office Limited. (2018). *Data Protection Act 2018*. Available: <http://www.legislation.gov.uk/ukpga/2018/12/contents/enacted>. Last accessed 14th October 2018.

The Three Software Stacks Required for IoT Architectures. (2016). [online] Available at: <https://iot.eclipse.org/white-papers/iot-architectures/> [Accessed 16 Feb. 2019].

Tomar, A. (2011). Introduction to Zigbee Technology. [online] 1. Available at: <https://www.cs.odu.edu/~cs752/papers/zigbee-001.pdf> [Accessed 31 Jan. 2019].

Truong, H. and Dustdar, S. (2009). A survey on context-aware web service systems. *International Journal of Web Information Systems*, [online] 5(1), pp.5-31. Available at: <https://www.emeraldinsight.com/doi/full/10.1108/17440080910947295> [Accessed 16 Feb. 2019].

Wyld, D. (2008). RuBee: applying low-frequency technology for retail and medical uses. *Management Research News*, [online] 31(7), pp.549-554. Available at: <https://www.emeraldinsight.com/doi/pdfplus/10.1108/01409170810876107> [Accessed 16 Feb. 2019].

Zak, R. (2016). *How to Find the Best WiFi Channel for 5Ghz Frequency*. [online] Maketecheasier.com. Available at: <https://www.maketecheasier.com/best-wifi-channel-for-5ghz-frequency/> [Accessed 16 Feb. 2019].

BIBLIOGRAPHY

5 Things Dealers Should Know About Z-Wave Systems and Equipment. (n.d.). [online] Available at: <http://www.gocontrol.com/zwave/5-Things-to-know-about-Z-Wave.pdf> [Accessed 19 Feb. 2019].

Agilemanifesto.org. (2019). *Principles behind the Agile Manifesto*. [online] Available at: <http://agilemanifesto.org/principles.html> [Accessed 16 Mar. 2019].

Barr, M. (1999). *Programming embedded systems in C and C++*. Sebastopol, Calif: O'Reilly.

Barr, M. (2013). *Embedded C coding standard*. Englewood Cliffs: Netrino.

evehome. (2019). *Eve Energy*. [online] Available at: <https://www.evehome.com/en/eve-energy> [Accessed 9 Dec. 2018].

Firebase. (2019). *Firebase Authentication* | *Firebase*. [online] Available at: <https://firebase.google.com/docs/auth/?authuser=0> [Accessed 10 Mar. 2019].

Firebase. (2019). *Firebase Realtime* *Firebase*. [online] Available at: <https://firebase.google.com/docs/database/?authuser=0> [Accessed 9 Mar. 2019].

Gcscte.org. (2017). *WiFi Fundamentals*. [online] Available at: <http://www.gcscte.org/presentations/2017/Wifi%20Overview.pdf> [Accessed 16 Feb. 2019].

Ghamari, M., Arora, H., Sherratt, R. and Harwin, W. (2015). Comparison of Low-Power Wireless Communication Technologies for Wearable Health-Monitoring Applications. [online] Available at: <https://ieeexplore.ieee.org/document/7219525/authors#authors> [Accessed 16 Apr. 2019].

Instructables.com. (2018). *Firebase Integrate With ESP8266*. [online] Available at: <https://www.instructables.com/id/Firebase-Integrate-With-ESP8266/> [Accessed 12 Mar. 2019].

Lee, I. and Lee, K. (2015). The Internet of Things (IoT): Applications, investments, and challenges for enterprises. *Business Horizons*, [online] 58(4),

pp.431-440. Available at:

<https://www.sciencedirect.com/science/article/pii/S0007681315000373#bib0055> [Accessed 16 Apr. 2019].

Lineback, R. (2014). *Internet of Things Boosts Embedded Systems Growth*.

[online] Icinsights.com. Available at:

<http://www.icinsights.com/data/articles/documents/740.pdf> [Accessed 16 Feb. 2019].

Madakam, S., Ramaswamy, R. and Tripathi, S. (2015). *Internet of Things (IoT): A Literature Review*. [online] File.scirp.org. Available at:

http://file.scirp.org/pdf/JCC_2015052516013923.pdf [Accessed 16 Feb. 2019].

Odunlade, E. (2017). *ESP8266 Weather Forecaster*. [online] Random Nerd

Tutorials. Available at: <https://randomnerdtutorials.com/esp8266-weather-forecaster/> [Accessed 17 Mar. 2019].

Osoyoo.com. (2017). *Arduino lesson – 1-Channel Relay Module* « osoyoo.com.

[online] Available at: <http://osoyoo.com/2017/08/arduino-lesson-1-channel-relay-module/> [Accessed 23 Feb. 2019].

Panetta, K. (2018). *5 Trends Emerge in the Gartner Hype Cycle for Emerging Technologies, 2018*. [online] Gartner.com. Available at:

<https://www.gartner.com/smarterwithgartner/5-trends-emerge-in-gartner-hype-cycle-for-emerging-technologies-2018/> [Accessed 16 Feb. 2019].

Ramya, C., Shanmugaraj, M. and Prabakaran, R. (2011). Study on ZigBee technology. *2011 3rd International Conference on Electronics Computer Technology*. [online] Available at:

https://www.researchgate.net/publication/261497749_Study_on_ZigBee_technology [Accessed 16 Feb. 2019].

Roboindia.com. (n.d.). *DHT11 Temperature & Humidity sensor on NodeMCU using Arduino IDE | Robo India*. [online] Available at:

<https://roboindia.com/tutorials/DHT11-NodeMCU-arduino> [Accessed 25 Feb. 2019].

Rohan (2019). *Embedded Systems Market worth 110.46 Billion USD by 2023*.

[online] Marketsandmarkets.com. Available at:

<https://www.marketsandmarkets.com/PressReleases/embedded-system.asp>

[Accessed 9 Feb. 2019].

Rouse, M. (2018). *What is Gartner hype cycle? - Definition from WhatIs.com.*

[online] WhatIs.com. Available at:

<https://whatis.techtarget.com/definition/Gartner-hype-cycle> [Accessed 10 Feb.

2019].

Saad al-sumaiti, A., Ahmed, M. and Salama, M. (2014). Smart Home Activities: A Literature Review. *Electric Power Components and Systems*, [online] 42(3-4), pp.294-305. Available at:

<https://www.tandfonline.com/doi/abs/10.1080/15325008.2013.832439>

[Accessed 28 Jan. 2019].

Sonoff.itead.cc. (2019). *Sonoff S20 Socket*. [online] Available at:

<https://sonoff.itead.cc/en/products/residential/s20-socket> [Accessed 12 Dec.

2018].

Stack Overflow. (2019). *Close virtual keyboard on button press*. [online]

Available at: [https://stackoverflow.com/questions/3400028/close-virtual-](https://stackoverflow.com/questions/3400028/close-virtual-keyboard-on-button-press)

[keyboard-on-button-press](https://stackoverflow.com/questions/3400028/close-virtual-keyboard-on-button-press) [Accessed 2 Apr. 2019].

Sundmaeker, H., Guillemin, P., Friess, P. and Woelfflé, S. (2010). Vision and Challenges for Realizing the Internet of Things. [online] Available at:

[https://www.researchgate.net/publication/228664767_Vision_and_Challenges_f
or_Realizing_the_Internet_of_Things](https://www.researchgate.net/publication/228664767_Vision_and_Challenges_f_or_Realizing_the_Internet_of_Things) [Accessed 16 Feb. 2019].

Switch, W. (2019). *WeMo® Insight Switch*. [online] Belkin. Available at:

<https://www.belkin.com/uk/p/P-F7C029/> [Accessed 7 Dec. 2018].

Torchia, M. and Shirer, M. (2019). *IDC Forecasts Worldwide Spending on the Internet of Things to Reach \$745 Billion in 2019, Led by the Manufacturing, Consumer, Transportation, and Utilities Sectors*. [online] IDC: The premier global market intelligence company. Available at:

<https://www.idc.com/getdoc.jsp?containerId=prUS44596319> [Accessed 30 Jan. 2019].

Tp-link.com. (2019). *HS110 | Kasa Smart Wi-Fi Plug with Energy Monitoring*.

[online] Available at: [https://www.tp-link.com/uk/home-networking/smart-](https://www.tp-link.com/uk/home-networking/smart-plug/hs110/)

[plug/hs110/](https://www.tp-link.com/uk/home-networking/smart-plug/hs110/) [Accessed 3 Dec. 2018].

APPENDIX A

Project Planning Document

1.0 - Introduction

Internet of Things (IOT) is the connection of "things" to the internet, these can be computers, phones, kitchen appliances, thermostats, cameras, and medical devices such as heartrate monitors. There are many uses for IOT in home automation, for example this project is going to be a Context Aware IOT Smart Plug. Smart plugs sit between the wall outlet and a connected device, this allows you to control the flow of electricity going to a device from anywhere, providing you have an internet connection. Most of these smart plugs only have the ability to turn on and off and the rest of the features are added through a mobile application. The majority of these application will have a scheduling feature which are mainly used for home security or automation.

Context awareness is the ability to gather information of the device's current environment, such as time, date and location, and automatically adapt to it. For example, it will be able to turn on or off depending on the cost of electricity. These will be implemented into the hardware as it will expand on the current capabilities of current smart plugs.

The IOT market is huge and is growing rapidly, according to Cisco "500 billion devices are expected to be connected to the Internet by 2030" (Cisco, 2016). With the growth in internet connected devices public perception of these devices is changing meaning they are more likely to buy a smart plug. As these devices are becoming more popular there is a huge opportunity to get a budget orientated smart plug with the added feature of context awareness.

2.0 - Aims and Objectives

The aim of this project is to create a prototype context aware smart plug and an android phone application to allow users to turn on and off their plug from anywhere, providing they have an internet connection. The hardware must be controlled from the phone application, the phone application must be easy to use.

Objectives of the project:

Must have -

- An easy to use android application to allow a broad range of users to be able to use the plug. This can be done by asking people to test the application during the development phase.
- Let the user create an account.
- Hardware can be controlled via the application.
- Hardware has context aware capabilities.
- Hardware can be set to turn off when electricity prices rise during peak hours.
- A schedule can be set on the application to turn on or off the hardware.

Would like to have -

- Hardware can analyse the power usage and give user recommendations to reduce this.
- Devices with the application installed can turn off the hardware once it reaches 100% when charging on the plug.
- User can add multiple smart devices to the application.

2.1 - Functional Requirements

FR1 - System in action

FR1.1 – The user will be able to use the android application to control the functions of the smart plug.

FR1.2 – The user will first be prompted to login or create an account.

FR1.3 – The user will be given instructions in setting up the device.

FR1.4 – The system will be user friendly with clear instructions.

FR1.5 – Usage can be monitored and displayed on a graph.

FR2 – User account

FR2.1 – The user will be able to create an account.

FR2.2 – The users' credentials will be stored so they can log in again.

FR2.3 – The system will store device details to a user account.

FR3 – Features

FR3.1 – User can turn on/off the plug from anywhere providing they have an internet connection.

FR3.2 – Hardware has context aware capabilities.

FR3.3 – Automatic switching as prices in electricity fluctuate.

FR3.4 – Lets user add multiple IOT devices

2.2 – Non-functional requirements

NFR1 – Security

NFR1.1 – Users information should be securely stored.

NFR1.2 – Other users can't see other users' information.

NFR2 – Performance

NFR2.1 – The application shouldn't take long to load.

NFR2.2 – The application shouldn't take up more than 1GB on the device.

NFR3 – Features

NFR3.1 – Other types of IOT devices will be supported.

NFR3.2 – Analyse power usage and give recommendations.

NFR3.3 – Application can see battery percentage of device its installed on.

3.0 - Project Scope, milestones, main tasks and deliverables

3.1 – Project scope

This project will be the making a Context Aware IOT Smart Plug, consisting of the smart plug itself and a mobile application. A working prototype will be completed by April 2019. Features of the hardware will

include being context aware meaning it knows the time, date and location
it will have the ability to turn on or off depending on a device's battery,
and the ability turn on or off depending on the time of day to save money
during peak hours.

3.2 – Milestones

| Milestone | Description | Completed on |
|-------------------------|---|--------------|
| Research | Research into embedded device communication, hardware and software requirements and investigating context aware issues. | 06/12/18 |
| Creating application | An android application that will allow control of the hardware, Including testing. | 18/02/19 |
| Building hardware | Creating a plug with context aware capabilities. | 14/01/19 |
| Review Point 2 | Meeting with project supervisor to make sure the project is on track. | 08/02/19 |
| Project Report/ Hand in | A record of everything needed to complete the project including research, planning, design, implementation, testing and analysis. | 25/04/19 |
| Demonstration | Preparing to present and demonstrate the project. | 02/05/19 |

3.3 - Main tasks

To complete the project these are the main tasks that I would need to finish:

- Create a working Wi-Fi enabled smart plug that I can connect to.
- Create an application that allows me to turn it on and off.
- Implement context aware features into the hardware.
- Prioritise features that I want the smart plug to include.
- Implement these features one by one based on priority.

3.4 - Deliverables

By the end of the project I plan on having:

- Review point documents.
- A working smart plug prototype.
- An Android application that can work with the plug.
- Documentation.
- Project report.

4.0 - Sources of information and resources required

When planning a project, the sources of information and resources required to undertake the project must be detailed. This includes the identification of information, such as similar products that have been released, current and future trends, and most importantly information needed to complete the project itself. To complete the project a suitable technology stack and a suitable hardware stack must be researched, ensuring the project concept is achievable and practical.

For the Context Aware IOT Smart Plug it will require a hardware component being the IOT smart plug and a software component being the Android phone application and web server.

Hardware Required –

- NodeMCU.
- Breadboard.
- Wires etc.

Software Required –

- Android Studio.
- Arduino IDE.

For this project a large number of resources will be required as it involves a number of different components and these will all need to be researched. Most of these sources will be found on the internet and will be referenced and cited

throughout the dissertation. As well as the internet I will be using books as they are a great source of reliable information. For example:

- Head First Android Development: A Brain-Friendly Guide - David Griffiths and Dawn Griffiths
- Big Java – Cay S Horstmann
- Programming in Lua – Roberto Leruslimschy
- Designing Embedded Systems and the Internet of Things (IOT) wit the ARM mbed – Perry Xiao
- Context-Aware Computing and Self-Managing Systems – Chapman & Hall/CRC

More sources of information will be found throughout the research and development phases.

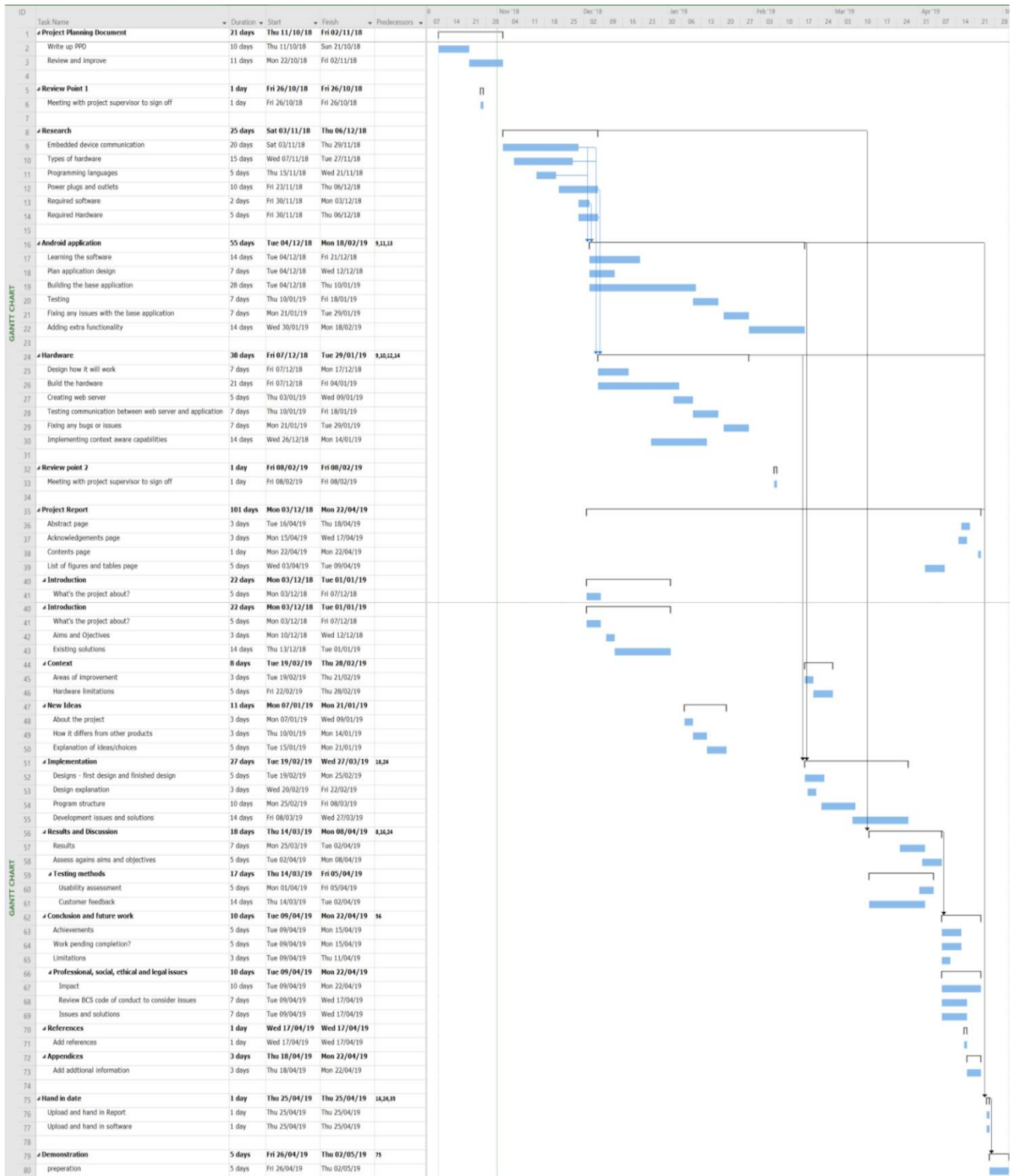
5.0 - Project risks

Risks that could cause me to fall behind on my project plan could be:

| ID | Risk event description and impact area | Risks response description | Trigger | Probability rating | Impact rating |
|-----------|--|--|---|---------------------------|----------------------|
| 1. | Illness/injury – Getting the flu, or breaking a bone etc. | Take care of yourself. | Hygiene, and lifestyle choices. | 6/10 | 5/10 |
| 2. | Performance – Not producing results that meet with project specifications. | Seek advice from lecturers. | Technical knowledge or underestimating what needs to be achieved. | 4/10 | 7/10 |
| 3. | Scheduling – Falling behind schedule. | Put more hours into work or seek advice. | Prioritising other work or not putting in the work. | 4/10 | 7/10 |

| | | | | | |
|-----------|---|--|---|------|------|
| 4. | Shocks or burns from working with electricity. | Learn how to properly deal with electricity in the safest way. | Not following safety instructions. | 1/10 | 8/10 |
| 5. | Loss of Wi-Fi – Fall behind on research, development and testing | Use internet at the university. | Failure to pay the bill or fault from the ISP side. | 2/10 | 3/10 |
| 6. | Limited software resources – some software may be unavailable outside university. | Complete work using the university computers. | Expensive licence costs. | 4/10 | 6/10 |
| 7. | Private computer breaks. | Use the computers or laptops provided by the university. | Hardware deterioration or malfunction. | 1/10 | 8/10 |
| 8. | Damage/loss of a memory stick containing work. | Create multiple copies of the work. | Leaving the stick lying around so it could be potentially lost or broken. | 3/10 | 9/10 |

6.0 - Gantt Chart



7.0 – Evaluation of professional, social, ethical and legal issues

7.1 - Professional

Regarding professional standards and issues, as it states with the British Computing Society (BCS) Code of Conduct, “act with integrity and respect in your professional relationships with all members of BCS and with members of other professions with whom you work in a professional capacity” (BCS code of conduct, 2015). This is applicable throughout the entirety of the project, from working with members of staff to employers and other students during the showcase event. BCS Code of Conduct also states that one must “respect and value alternative viewpoints and, seek, accept and offer honest criticisms of work” (BCS code of conduct, 2015). This allows other people to give feedback on the project without any backlash and will help in the development of the project.

7.2 - Social

When considering social issues within this project it was important that this product would be accessible to as many people as possible. As the BCS Code of Conduct states “promote equal access to the benefits of IT and seek to promote the inclusion of all sectors in society” (BCS code of conduct, 2015). Therefore the project will need to have a simple setup and clear instructions as well as the application being clear and the potential for added disability features. Testing will be heavily carried out throughout the development stage of the project so that the product can be adapted from user feedback, so it serves a larger percentage of people. This is especially important with the rapid growth of IoT devices as more home owners incorporate them into their day to day life, as they would be looking for a product that has an easy setup and user-friendly graphical user interface.

7.3 - Ethical

To avoid any ethical issues, I will be sure to “hold paramount the safety, health, and welfare of the public, to strive to comply with ethical design and sustainable development practices, and to disclose promptly factors that might endanger the public or the environment” (IEEE policies, 2018). As a major part of the project is a plug which involves electricity, it is important that these are upheld as there is a huge potential for injury. To avoid the risks in injury the product will be thoroughly tested, and safety tests will be carried out. As user data will be stored, it is important that the data is stored securely, and that user data is not shared. Communication between the application and the hardware should also be encrypted so that the device is not easily hacked.

7.4 – Legal

The new Data Protection Act 2018 implements the European Directive known as the General Data Protection Regulation (GDPR) into UK law. The new Data Protection Act 2018 provides that data is to be “handled in a way that ensures appropriate security which includes protecting data against unlawful or unauthorised processing, access, loss, destruction or damage”. Therefore, under the new legislation, any user data that is held must be appropriately secured to prevent any of the unfortunate consequences occurring as provided. Also, the new act has also provided more rights to the consumer which means they will have new extended rights under the Data Protection Act 2018. One particular right of importance is the right of access which enables the data subject to ascertain what information is held about them by us. Therefore, a new feature will need to be added to the application to allow users to specifically request this.

References

Cisco. (2016). *Internet of Things*. Available: <https://www.cisco.com/c/dam/en/us/products/collateral/se/internet-of-things/at-a-glance-c45-731471.pdf>. Last accessed 14th October 2018.

BCS. (2015). *BCS Code of Conduct*. Available:

<https://www.bcs.org/category/6030>. Last accessed 14th October 2018.

IEEE. (2018). *IEEE Code of Ethics*. Available:

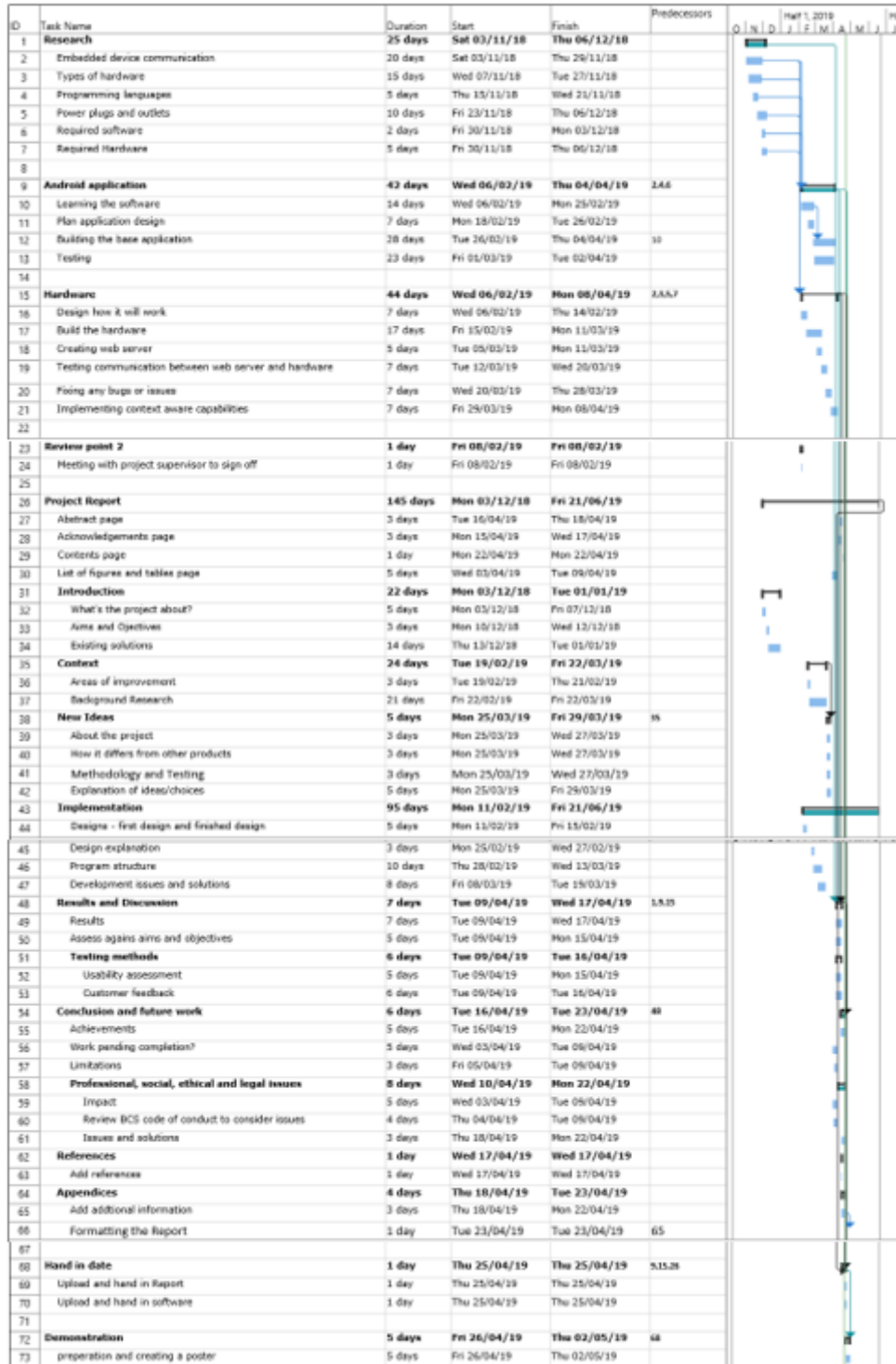
<https://www.ieee.org/about/corporate/governance/p7-8.html>. Last accessed 14th October 2018.

The Stationery Office Limited. (2018). *Data Protection Act 2018*. Available:

<http://www.legislation.gov.uk/ukpga/2018/12/contents/enacted>. Last accessed 14th October 2018.

APPENDIX B

1 Updated Gantt Chart



APPENDIX C

1 Material Design' design principles

The design guidelines below are ones followed throughout the development of the application to ensure that the application is usable.

Material Design layout principles state that the layout of the application should be predictable, consistent and responsive.

- UIs should use intuitive and predictable layouts, with consistent UI regions and spatial organization.
- Layouts are adaptive and react to input from the user, device, and screen elements.
- The Material Design colour system can be used to create a colour theme that reflects your brand or style.
- Motion helps make a UI expressive and easy to use.
- To ensure that your layout is flexible and adapts to different screen sizes, you should use "wrap_content" and "match_parent".

APPENDIX D

Application functional testing

| # | Test | Expected Result | Actual Result | Pass/Fail | Fix |
|----|---|--|---|-----------|-----|
| 1 | Opening the application. | The expected result is that the application opens without crashing. | The application opened and did not crash. | Pass | N/A |
| 2 | Clicking the sign in button. | The expected result is that the sign in page opens | The sign in page opened. | Pass | N/A |
| 3 | Pressing the back button from the sign in page. | The expected result is that it takes you back to the beginning screen. | The beginning screen opened. | Pass | N/A |
| 4 | Pressing the create account button. | The expected result is that the create account page opens. | The create account page opened. | Pass | N/A |
| 5 | Creating an account. | The expected result is that the user's details are given to firebase where they will be authorised, and then be taken to the devices page. | Account created and taken to the devices page. | Pass | N/A |
| 6 | Signing in to Plugnet. | The expected result is if the user has already created an account it should log them in, taking them to the devices page. | Logged in and taken to the devices page | Pass | N/A |
| 7 | Tile load up to date information. | The expected result from this test is up to date information from firebase is obtained and used to update the UI with up to date information. | Shows the current mode, status and temperature of the devices. | Pass | N/A |
| 8 | Pressing the switch on the tile. | The expected result is that the tile should change to a different shade of grey depending on its current position. | Switching it on turned it dark grey while switching it off turned it to a light grey. | Pass | N/A |
| 9 | Clicking on the tile. | The expected result should open the 'my device' page showing more details about the plug that has been clicked on. | My device page opened showing more details about the plug. | Pass | N/A |
| 10 | Clicking the turn on/off button | The expected result is when clicked on should change the text on the button to the opposite of its current status as well as updating the information on firebase. | Button text changed. | Pass | N/A |

| | | | | | |
|----|---|--|---|-----------|--|
| 11 | Clicking the change mode button. | The expected result is that a new page opens showing plug settings than can be changed. | Page opens. | Pass | N/A |
| 12 | Clicking the update button | The expected result is that if all the areas are filled in, then these settings should update in firebase and take the user back to the devices page | Settings updated in firebase; device page opened but the keyboard is still open | Pass/Fail | Software keyboard should close when the button is pressed. |
| 13 | Sliding screen from left to right | The expected result is that the navigation menu opens. | Navigation menu opens. | Pass | N/A |
| 14 | Clicking on the profile, settings, help, and information button in the navigation menu. | The expected result is that all these pages open showing more information. | All the pages opened perfectly fine. | Pass | N/A |
| 15 | Pressing the sign out button | The expected result is that the user is logged out and taken back to the beginning screen. | Logged out and taken back to the first screen. | Pass | N/A |
| 16 | Pressing the change password button | The expected result is that the update password page opens. | update password page opens | Pass | N/A |
| 17 | Updating the password | The expected result is when the user enters their current password and a new valid password it will update. | Password updated. | Pass | N/A |
| 18 | Pressing forgot the password. | The expected result is that the reset password page opens. | Reset password page opened | Pass | N/A |
| 19 | Resetting password. | The expected result is that if a user enters a valid email that is in the Plugnet system, they will receive an email allowing them to update their password. | Email received allowing for a password reset. | Pass | N/A |
| 20 | Error handling. | The expected result is that if the user inputs an invalid email or password they will have a text prompt letting them know what is wrong. | Text prompt letting me know what I did wrong. | Pass | N/A |

Firestore web service functional testing

| # | Test | Expected Result | Actual Result | Pass/Fail | Fix |
|---|-------------------------------------|--|--|-----------|-----|
| 1 | Updating information. | The expected result is that any information updated on firestore would automatically sync with the app and plug. | Information synced with the plug and app causing it to change. | Pass | N/A |
| 2 | Replacing an integer with a string. | The expected result is that both the plug and app will crash. | The plug and app both crashed. | Pass | N/A |
| 3 | Each plug should have an ID. | The expected result is if the ID is removed would cause the application not to function fully. | User cannot update information if the ID is removed. | Pass | N/A |
| 4 | Adding another plug | The expected result is that adding another plug will show up on the user's phone instantly | New plug shows up on users' phone. | Pass | N/A |

Plug functional testing

| # | Test | Expected Result | Actual Result | Pass/Fail | Fix |
|---|-------------------------------|---|---|-----------|--|
| 1 | Update firestore information. | The expected result is that the plug should read and react to the changes made | Reads and reacts to the changes made. | Pass | N/A |
| 2 | Reading internal temperature | The expected result is that the sensor reads the temperature and uses this within the main algorithm. | Gets internal temperature and uses it within the main algorithm. | Pass | N/A |
| 3 | Getting external temperature | The expected result is that it reads the location from firestore and uses that to get the weather data from an API. | Gets location and uses this to get the weather data of that location. | Pass | N/A |
| 4 | Reconnecting to the internet | The expected result is if internet connection is lost the device will reconnect. | The device does not reconnect | Fail | Create a function that checks if connected |
| 5 | Checks the mode | The expected result is if the mode is changed the plug should change to that mode | Mode changes | Pass | N/A |

APPENDIX E

1 Usability test results

To test the usability of the application, a test has been conducted after the implementation of the system had been fully completed. The tests were carried out after leaving the user with the application and plug for a short period of time after this the author asked them five questions designed to find out if the design and usability of the application were to a high standard. Five users were asked the following:

- 1 “Was the application easy to navigate?”

This question was designed to find out if the users thought the navigation around the application was clear and easy to use.

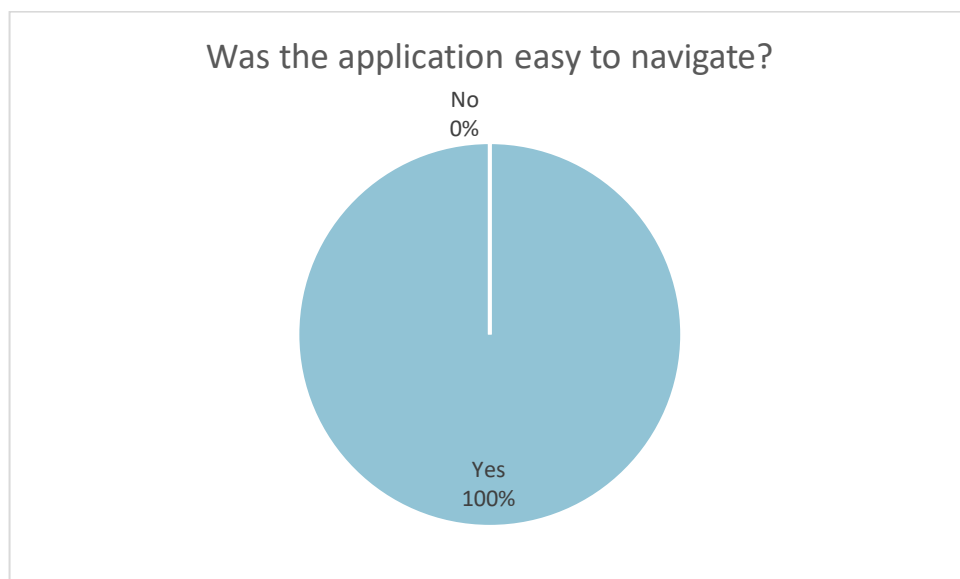


Figure 49 - Usability Question 1

Figure 49 shows that of the five users asked if the application was easy to navigate all of them said yes that it was. This shows that users thought the current navigation of the application is clear and straightforward use.

2 “Do you like the design of the application?”

This question was designed to find out if the users liked the design of the website; these same users have seen the different iterations of the application.

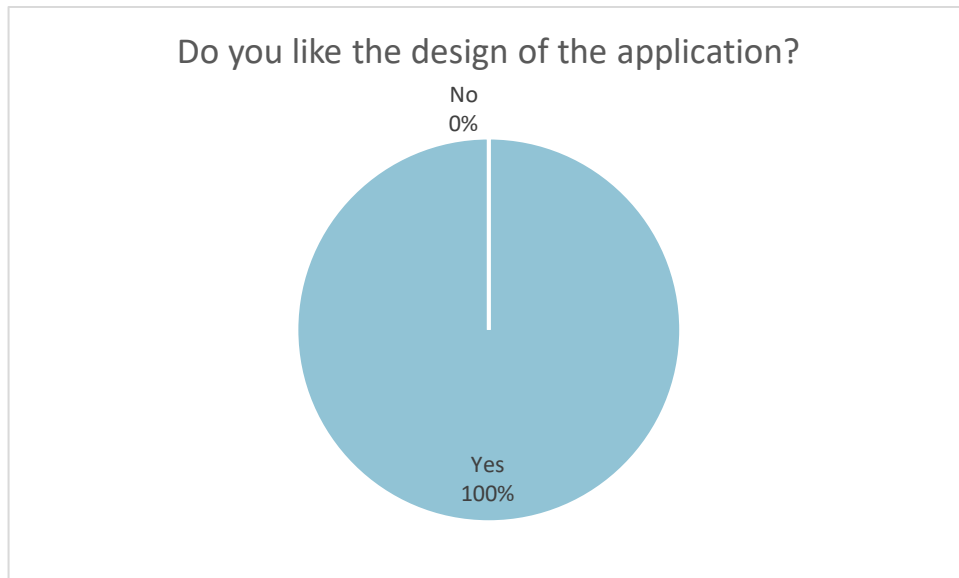


Figure 50 - Usability Question 2

Figure 50 shows that of the five users asked if the design of the application all 5 of them seemed to like it; this shows that the user testing carried out throughout development paid off.

3 “Did you have any issues registering or logging in?”

This question was designed to see if the users had any issues when creating an account or logging in and to see if choosing Firebase authentication was the best option.

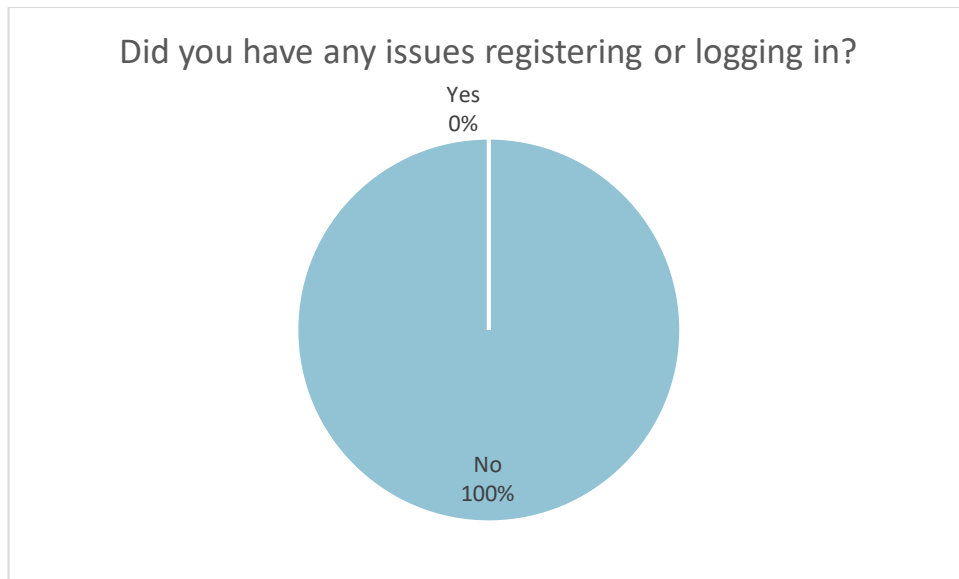


Figure 51 - Usability Question 3

Figure 51 shows that of the five users asked about account issues; none of them seemed to have had any issues, which shows that choosing Firebase was a solid choice.

- 4 "How would you compare this app to others? (10 = as good or better, 1 = a lot worse)"

This question was asked to see how this application stands against other applications the user uses daily.



Figure 52 - Usability Question 4

Figure 52 shows that users were overall happy with the overall design, navigation and style of the application, as one user thought it was as good as the apps they normally use while the other four found that it was good but did not quite compare.

5 "Did you enjoy using the Plugnet application?"

This question was designed to find out if the user was overall satisfied with the quality of the application.

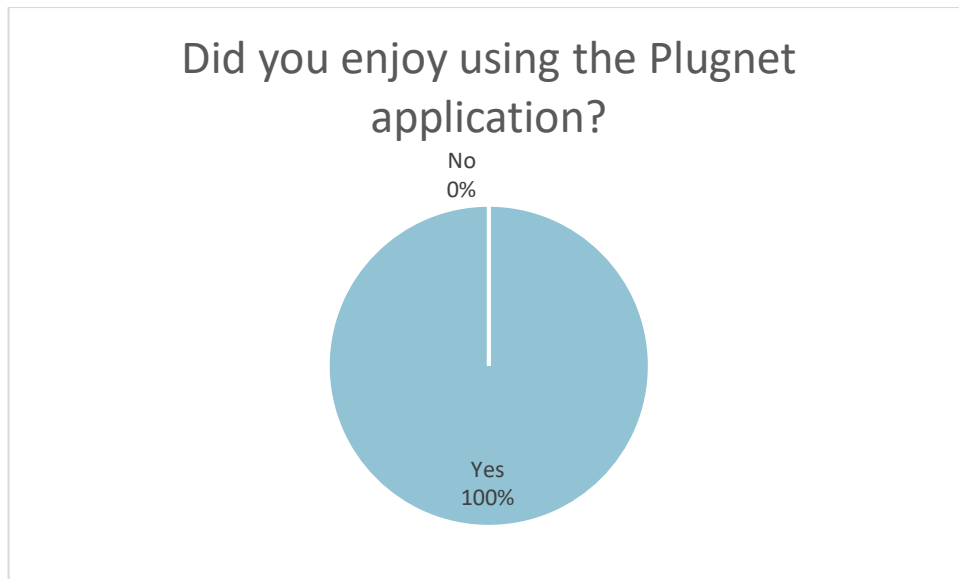


Figure 53 - Usability Question 5

Figure 53 shows that the users were overall satisfied with the quality of the application and that the users genuinely thought the app was good.

APPENDIX F

1 User Manual

The purpose of the manual is to assist others in trying to configure something like the Plugnet system. Styled in the way of an online tutorial.

Plugnet Application

The code can be obtained via the drop box or the USB submitted; this may need to be edited if you want it to work with a different Firebase database, other than that no changes are necessary.

Firebase Database

To use your database a new Firebase project would need to be created ensuring you enable the email/password authentication method. All the database features should be automatically done through the application and the plug. However, if there are any issues then create the database entries and enter default values.

Plugnet Hardware

To set up the plug, you will need a NodeMCU, DH11 temperature sensor and a 5V one channel relay module. These will need to be set up as follows:

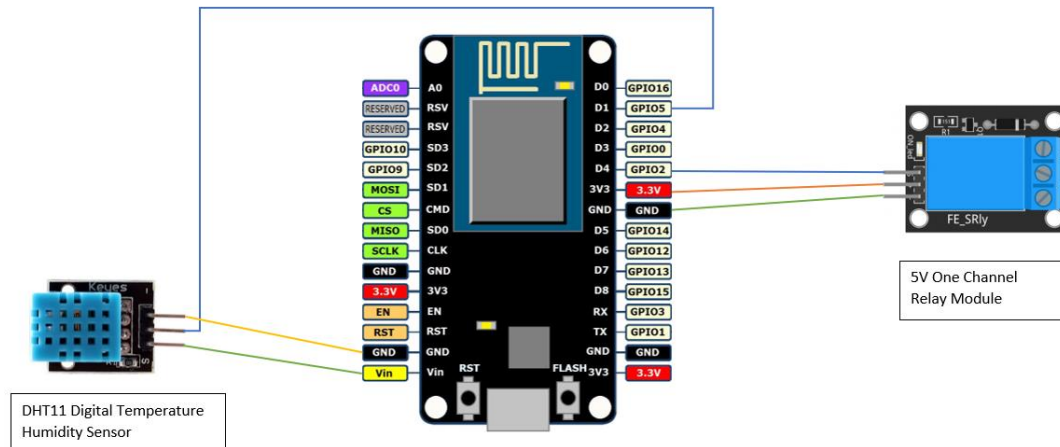


Figure 54 – Configuration of temperature sensor and relay module on the NodeMCU

The code will also be available for this on the drop box or USB.

To get the board manager on the Arduino software this URL needs to be added to the additional board's manager URL:

http://arduino.esp8266.com/stable/package_esp8266com_index.json

and then the tools be configured as shown in figure 55:



Figure 55 - Arduino IDE configuration for NodeMCU

And then the following Libraries are required:

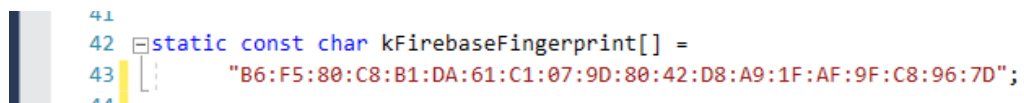
- ArduinoJson (Version 5)
- NTPClient-master
- DHTLib-master
- Firebase-Arduino-Master
- Adafruit-Circuit_Playground
- ESP8266

The firebase Arduino Library will have to be configured slightly as firebase have changed their fingerprint which can be found in:

C:\Users\username\Documents\Arduino\libraries\firebase-arduino-master\src\FirebaseHttpClient.h

The code will need to be changed to include this new fingerprint:

"B6:F5:80:C8:B1:DA:61:C1:07:9D:80:42:D8:A9:1F:AF:9F:C8:96:7D"



```
41
42 static const char kFirebaseFingerprint[] =
43     "B6:F5:80:C8:B1:DA:61:C1:07:9D:80:42:D8:A9:1F:AF:9F:C8:96:7D";
```

Figure 56 - Firebase Arduino Fingerprint

Once all this has been configured it should be able to upload.

Within the code the following changes will need to be made:

- Change SSID and password
- Change the Firebase host and auth