

Contents

1	INTRODUCTION	2
1.1	Problem Statement	2
1.2	Aims and objectives	3
1.3	Literature Review	3
1.3.1	Similar Services	3
1.3.2	Scalability	4
1.3.3	Cost	4
1.4	Summary of the report	5
2	ANALYSIS	6
2.1	Feasibility Study	7
2.2	Requirements Gathering.....	8
2.3	Requirements.....	10
2.3.1	User Requirements.....	10
2.3.2	System Requirements.....	11
2.4	Project Plan	12
3	DESIGN.....	13
3.1	System Architecture	13
3.2	Hardware Design.....	14
3.2.1	Ethernet network (Ethernet module ENC28J60)	14
3.2.2	Internal temperature sensor (DS18B20 probe)	14
3.2.3	External temperature sensor (DS18B20 probe)	15
3.2.4	Reed switch (Cylewt reed switch)	15
3.2.5	Rain sensor (AZDelivery rain / snow / humidity detection sensor).....	15
3.2.6	PIR sensor (HC-SR501 sensor)	15
3.3	Database	15
3.4	Android app Layout	16
3.5	Summary	18

4	IMPLEMENTATION AND TESTING	19
4.1	Hardware Components.....	19
4.1.1	Arduino Microcontroller.....	19
4.1.2	Ethernet module ENC28J60	20
4.1.3	Rain Sensor	21
4.1.4	PIR sensor	22
4.1.5	DSB1820 temperature probes	23
4.1.6	Reed Switch	25
4.2	Testing.....	26
4.3	Summary	28
5	EVALUATION.....	28
6	REFLECTION AND CONCLUSION.....	32
6.1	Reflection	32
6.2	Improvements.....	34
6.3	Conclusion	35
7	REFERENCES	36
8	APPENDICES	37
8.1	Appendix A Code.....	37

Abstract

Smart home systems are readily available on the market, primarily being driven by Amazon's Echo variants and Google's Home variants. The aim of this project is to develop a low cost, informative Smart home system that allows the user to view the information such as temperature via an android mobile application that will display the information collected by the system.

The development of this project will rely heavily on research. This report contains information regarding the current smart home systems on the market that have been researched. From this research the development of the project will start to take shape as there has been a comparison made.

The system will be developed and tested according to feedback gained from questionnaires and reviews of potential users. In summary, this report will provide an overview into the steps taken to complete this project.

1 Introduction

The global market for smart home systems has grown substantially each year from 2016. In 2016 the global market size for smart home systems was 24.1 billion U.S Dollars – this grew to 35.9 billion U.S dollars in 2019 and is forecast to be worth 53.45 billion U.S dollars by the year 2022 (Statista). With Amazon being the market leader for smart home systems, selling roughly 70% of the estimated 25 million units in the US alone to date (BCG) it would be wise to delve into their cheapest smart home system – the Echo Dot, priced at £49.99 on their own website (Amazon.co.uk) is currently the cheapest smart home system on the market, tied with Google's Home Mini. The Echo Dot also has a mobile app called Alexa that can be used to view the data that Amazon's AI 'Alexa' produces.

The issue with the Echo Dot is the cost, to gain the full potential of this system or to view some basic information of the home the user will have to purchase other smart devices which will generate an even greater cost. An example of this would be to view the internal temperature of the home – the user would have to either purchase a different smart device from Amazon that can read the temperature such as an Echo Plus which is priced at £139.99 or a smart thermometer that is compatible with the Echo Dot. An example of a device like this would be the SwitchBot Thermometer priced at £19 on Amazon.co.uk.

The solution for this issue is outlined in this document. The following is research into a smart home system that can provide information of the user's home via a mobile app.

1.1 Problem Statement

A large problem many people face is that they will need to pay a lot of money for a smart home system to have the convenience of digital visibility of their home, what is meant by this is being able to view data such as the internal / external temperatures, if their windows are closed, if it is raining or not etc. By people not having this kind of visibility it can lead to security risks such as people forgetting to close their windows or not turning on their alarm – the smart home system can detect these features for the user and display it to an app on their Android device.

1.2 Aims and objectives

The aims of this project are:

- Create a smart home system using Arduino as the microcontroller
- To be cheaper than any other smart home system currently on the market
- Develop an Android app that can communicate with the smart home system to display the data collected to the user

1.3 Literature Review

1.3.1 Similar Services

We are going to explore some of the existing systems which produce a similar outcome to this project. As previously mentioned, the Echo Dot and the Google Home Mini are currently the two cheapest smart home systems on the market at £49.99.

We know the Echo Dot is produced by Amazon, one of the largest companies in the world and the leader of the smart home system industry, while it has a speaker, voice recognition and Alexa built into it, it still seems to be lacking the basic functionality that a user would expect for paying £49.99. It can't give you the basics such as being able to read the internal temperature due to it not already having a thermometer but what are the other downsides to this device? There are a number of downsides to the Echo Dot. We have already mentioned the cost of it being quite high so let's look at the other issues.

One of the problems with the Echo Dot is the compatibility of other smart devices – not all IoT devices are compatible with Amazon's product and will either simply not work with the device or you will require additional devices to gain all of the functionality from the Echo Dot. For example, many smart devices such as cameras or locks will require you to purchase a smart home hub to be compatible with the Echo Dot, this in turn will require additional set up and cost.

The Google Home Mini is very similar to the Echo Dot with the main difference being the integrated AI on the device. While Amazon's product uses Alexa, the Home Mini uses Google's Google Assistant. A review of the products show that the Echo Dot is a better device for compatibility, sound quality and the hardware however the Home

Mini is more accurate for general questioning such as “How long will it take me to travel to Belfast via train?” (techhive.com)

Both of these devices however may have some potential buyers hesitant to purchase the product. This is due to the fact that they are ‘Always Listening’ therefore will be able to pick out key words from your conversations even when you are not interacting with the device itself. It does this so you can be targeted more efficiently by advertising so they can increase their sales (techhive.com). Users aware of this information may be put off purchasing either of these devices as they may not trust a company having your data on record for advertising purposes.

The main reason that users may be put off the idea of purchasing these devices would be due to the high cost of £49.99 as the cheapest smart home systems out there. The entire cost to develop this project comes to \$22.55 (£18.03).

1.3.2 Scalability

The scalability of this project is incredible due to the microcontroller that will be used (Arduino) with the bonus of being able to scale the system at a low cost due to the cheap components required. For example the product can expand by having more wireless sensors such as thermometers and placing them around the home in different rooms, the mobile app could then be developed to be customisable to show you the values of each of those thermometers in specific rooms i.e. The living Room, Kitchen, Bathroom etc. The Arduino microcontroller can also be programmed to accept voice recognition and output a response through a speaker. An example of this would be to ask the system “Is it raining?” and the response through the speaker would be “Yes” or “No” depending of course on the data collected by the rain sensor.

The mobile app itself could be developed to be compatible with iOS which increases the scalability of the project. This is due to the fact that although Android have the much larger global market share of devices (74.45%) as of Jan 2019 (macworld.co.uk) there are still 728 million iPhones in use worldwide (statista.com). This will allow a massive increase of users to be compatible with this system.

1.3.3 Cost

As mentioned before, the cost of the current systems is the main issue. The large upfront cost of £49.99 can only be the start for a lot of users – they will also need to pay for subscriptions such as Amazon’s Prime service which will allow for music

streaming on the Echo Dot and is priced at £7.99 per month or £79 per year (radiotimes.com). As previously mentioned, there are a lot of smart devices that simply aren't compatible directly with the Echo Dot but are compatible with the higher priced version of the Echo – the Echo Plus priced at £139.99. These devices will require an additional purchase of a smart home hub that will allow the Echo Dot to then have visibility / control of them. The smart home hubs can come at a cost – upwards of £60 such as the Logitech Harmony Hub, costing £64.97 and the Samsung SmartThings Starter Kit costing £80.

1.4 Summary of the report

This section will give a brief of the report to give the reader an expectation of what is to come. The report will be broken down into topics and described.

The introduction section has covered an outline of what this project is – a low cost smart home system and has described the importance of such a system. The introduction has also shown the problem statement and why the need for a system like this is necessary as well as the aims and objectives of this project. The literature review has outlined some research that was conducted into this area to show similar services, the scalability of the project and the cost element of a smart home system.

The analysis section of this report follows. The analysis of the project is very important as this section conducts research to find the user requirements, system requirements, functional requirements and non-functional requirements. This is performed by a feasibility study and requirements gathering through the use of a survey which users will fill out. This section will also outline the plan of the project using a Gantt chart and will show the date at which the milestones will be completed.

The design chapter of the report follows the analysis section. The design chapter can only be completed once the analysis has been performed to find the requirements. The design chapter will outline how the low-cost smart home system will be created and will show the system architecture, the hardware design including a description of each component used, the database including an entity relationship diagram and the app layout.

The testing chapter follows the design chapter. The testing chapter is self-explanatory, but this will document how the project has been tested to show that it has been implemented correctly and that any errors have been resolved. Both the hardware system and the software system (application) will be tested.

The evaluation chapter follows the testing chapter. This section will give the reader an entire summary of the work carried out during the project and will critically evaluate it. This section will assess the value of the created system and will include an explanation of issues that arose during the creation of the system.

The reflections and conclusions chapter follow the evaluation of the system. This section will also include issues that arose during the project but will go into detail as to what I learned from them and how I would possibly implement features within the system in the future.

The references plus appendices will follow the reflections and conclusions. The reader will be able to see the references that have been sourced during the creation of this report and the appendices section will provide secondary material to support the main body of this report.

2 Analysis

To accurately understand the problem being addressed the system will need to be analysed to determine:

- User requirements
- System requirements
- Functional requirements
- Non-functional requirements

However, the feasibility of the system must be proven via a feasibility study before these requirements can be discussed.

2.1 Feasibility Study

A feasibility study is “an examination of a situation to decide if a suggested method, plan, or piece of work is possible or reasonable” (dictionary.cambridge.org). The study will consist of interviewing several Echo Dot users with the questions as follows:

Would you say the Echo Dot is reasonably priced?

Interviewee 1 “I wouldn’t agree with the price of what I believe was around £50 when I purchased my Echo Dot – I felt as if that was a little too much money, but I didn’t have any other options as that was the cheapest one available.”

Interviewee 2 “Yes, I purchased mine on sale for £34.99 and think that was a reasonable price for the device.”

Interviewee 3 “No - I was going to purchase the device while it was on sale, but I missed it and had to pay the full £49.99 for it.”

Are you happy with the functionality of the Echo Dot?

Interviewee 1 “Overall I am happy with what the Echo Dot can do but I would like to see some extra features such as being able to see the temperature of my own house, not just the town I live in.”

Interviewee 2 “Yes I would say that I’m happy with the functionality of the Echo Dot.”

Interviewee 3 “No, it doesn’t come with anything for it show information around my home.”

Have you had any compatibility issues with the Echo Dot, if so – what were they?

Interviewee 1 “Yes I have, I have recently bought what I thought was a smart thermometer to address the issue in your last question, but I couldn’t connect it to my Echo Dot, it had its own application I had to download.”

Interviewee 2 “No, I haven’t tried connecting anything other than my phone to it.”

Interviewee 3 “Yes, I purchased a camera for my front door but needed to purchase an additional router for the Dot to be able to connect with it.”

Were you aware at the time of purchasing the Echo Dot that the functionality would be limited as it doesn't have as many in-built features as the Echo Plus?

Interviewee 1 "No I wasn't aware of this, I assumed the only difference was the size of the speaker."

Interviewee 2 "No, I didn't know this – I thought the only difference was that the Echo Dot was smaller."

Interviewee 3 "No, otherwise I would've purchased the Echo Plus as it would have been cheaper than the Dot + buying the smart home router."

Would you consider purchasing a cheaper smart home system that could mainly give you information about your home?

Interviewee 1 "Yes I would, I would especially like to see the temperature of my home as I have a new-born child and need to monitor this."

Interviewee 2 "Yes - It would be great to have something that could monitor my home without having to purchase any additional devices."

Interviewee 3 "Definitely as that is what I wanted from the Echo Dot but didn't receive."

2.2 Requirements Gathering

To understand the requirements of my system I have created a survey for people to fill in that can help me decide what the system will be able to perform.

What functions would you expect to see in a low cost smart home system i.e the data displayed

Answered: 34 Skipped: 4

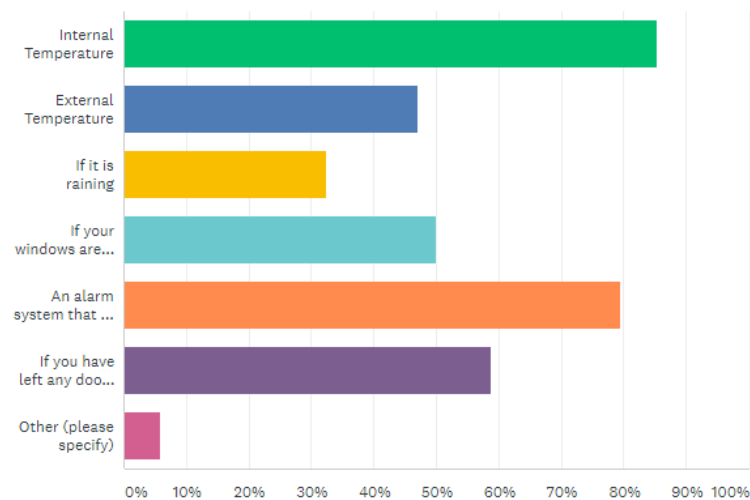


Figure 2.2.1 Survey Question Results

Why would you want to purchase a smart home system?

Answered: 38 Skipped: 0

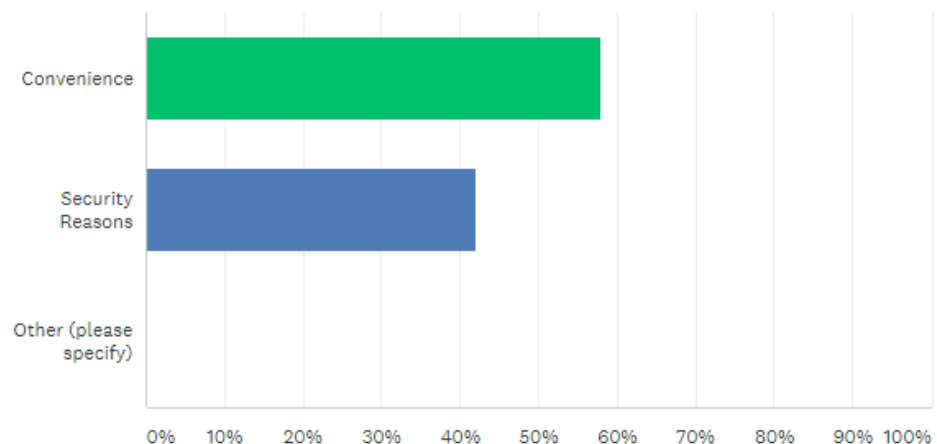


Figure 2.2.2 Survey Question Results

Figure 2.2.1 shows a question within my survey used to determine what requirements of the system that users would expect to see. As you can see the function with the most responses is the system being able to record and display the internal temperature of the home, an issue with the current systems on the market previously highlighted in 1.3.1. The users gave the second most responses for a feature that when enabled, can send an alert to your mobile device if it detects movement.

Figure 2.2.2 shows a question within my survey that outlines the main reasons the users would want a smart home system for – by the responses gathered, more than

half would rather purchase a system for the convenience aspect instead of for the security reasons. That being said, there is not a huge gap between the two factors – with convenience gaining 57.89% of the responses and security reasons with 42.11%.

With the results gathered from these 2 questions alone it is clear that the system will need to be both convenient and secure – making both of these requirements necessary.

How would you prefer to view the data collected by the system?

Answered: 37 Skipped: 1

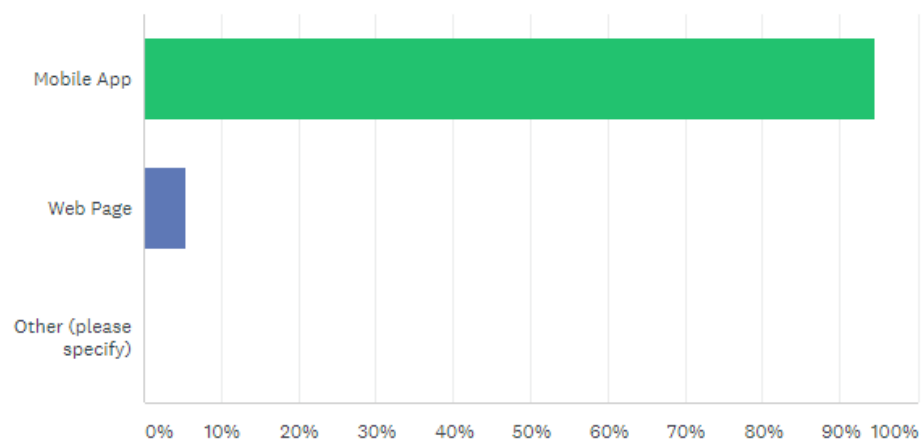


Figure 2.2.3 Survey Question Results

Figure 2.2.3 shows a clearly preferred method of viewing the data collected by the system – this confirms a mobile app will need to be developed instead of a web page for users.

2.3 Requirements

Below is a list of requirements, both user and system sorted by functional and non-functional.

2.3.1 User Requirements

Functional:

- The user can control the system through the mobile app.

- The user can view the data collected through the mobile app.
- The user can enable the alarm through the mobile app.
- The user can force an update of data.
- The user will only be able to view their own data.

Non-Functional:

- The mobile app will be easy to view / use.
- The mobile app will alert the user when the PIR sensor detects movement if it is enabled.
- The system will update the data every 30 seconds.
- The system won't need to go offline.
- Only the user will be able to view the data.

2.3.2 System Requirements

Functional:

- The user should have no reason to interact with the hardware of the system.
- The system must collect new data every 30 seconds.
- The system must be able to communicate with the Firebase database.
- The system must be secure to not allow any intruders to access the data.
- The mobile app must be compatible with Android.
- The system must always have a network connection.
- The system must be easy to install.

Non-Functional:

- The system must be cheap to construct.
- The system shouldn't stand out aesthetically.
- The system must be robust.
- The mobile app must be user friendly.
- The mobile app's UI should be simplistic.

2.4 Project Plan

See below Gantt chart (figure 2.4.1) outlining the project plan and timeframe. This Gantt chart displays the plan of the project and when each element will be completed.

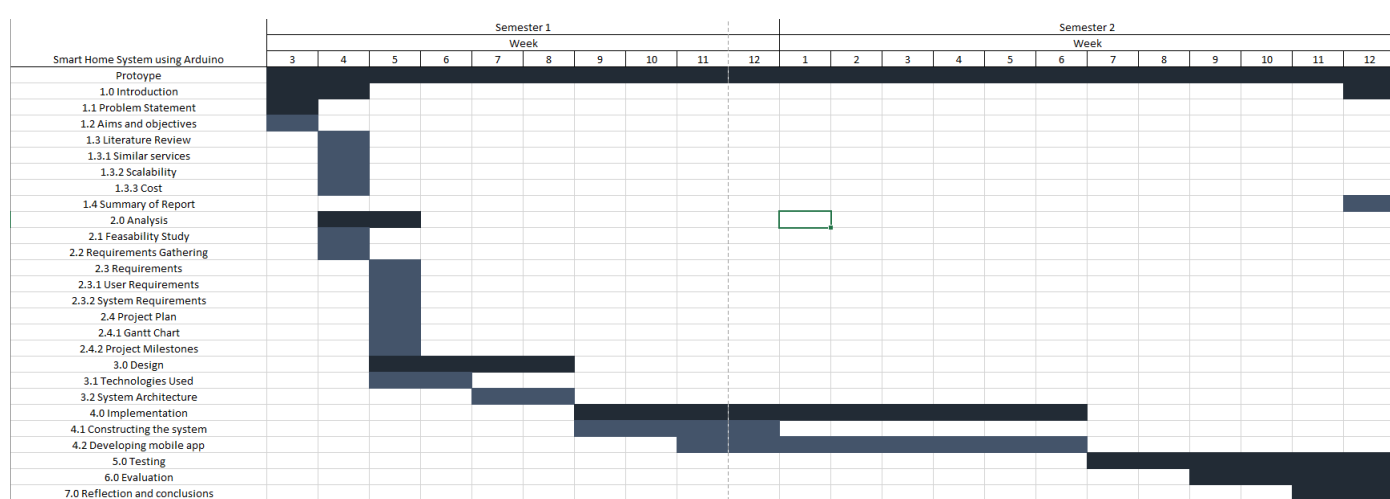


Figure 2.4.1 Gantt Chart

In correlation to the Gantt chart see table 2.4.2 which shows the project milestones, this table shows the date at each milestone will be completed.

Table 2.4.2 Project Milestones

Semester 1	Milestone	Date
Week 5	Project Plan created	24/10/2019
Week 8	Design completed	14/11/2019
Semester 2		
Week 8	Development completed	02/04/2020
Week 12	Testing completed	30/04/2020
Week 14	Prototype showcased	16/05/2020

3 Design

As the requirements of the system have been gathered by performing various analysis methods it is now time to design the smart home system. The following chapter outlines the design of the smart home system including the system architecture, hardware design, database diagram and the Android app layout of the mobile application.

3.1 System Architecture

See figure 3.1.1 outlining the system architecture. The system comprises of 4 main components – the Arduino microcontroller collecting data from the various sensors, sending these results to a PHP file hosted on <http://matthew-webster.com>, the Firebase Realtime database and the android application.

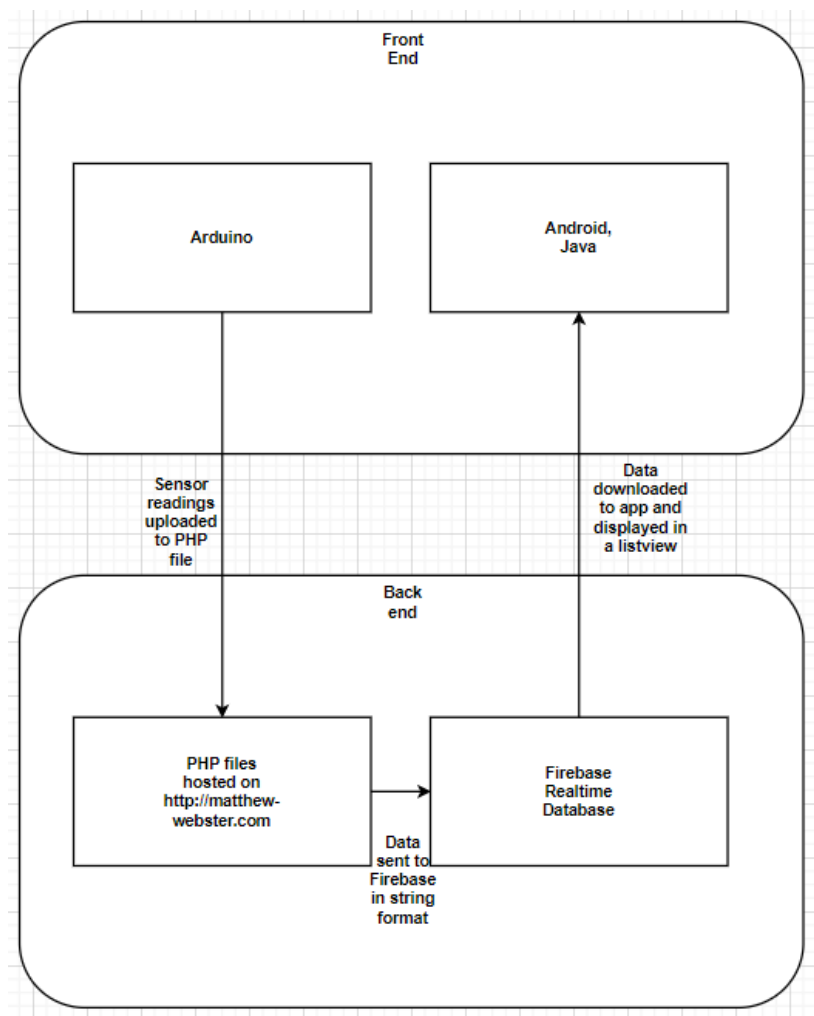


Figure 3.1.1 System Architecture

The Arduino microcontroller will communicate with the Firebase Realtime Database by sending the data collected from the various sensors to the PHP files named

firebaseUpload.php and firebaseLib.php then sent to the Firebase Realtime Database in Realtime over HTTP. The Firebase Realtime Database will then be populated and communicate with the android mobile application by sending it the data. The Android mobile application will be what the end user interacts with and will allow them to view the information about their home. The application UI will be outlined in figure 3.4.1.

3.2 Hardware Design

See below figure 3.2.1 outlining the hardware design of the smart home system. The Arduino microcontroller will be connected to various sensors shown in the figure. The sensors will collect data based from their surroundings and communicate this with the Arduino which will then send it to the PHP files hosted on <http://matthew-webster.com> then to the Firebase Realtime Database via an ethernet network.

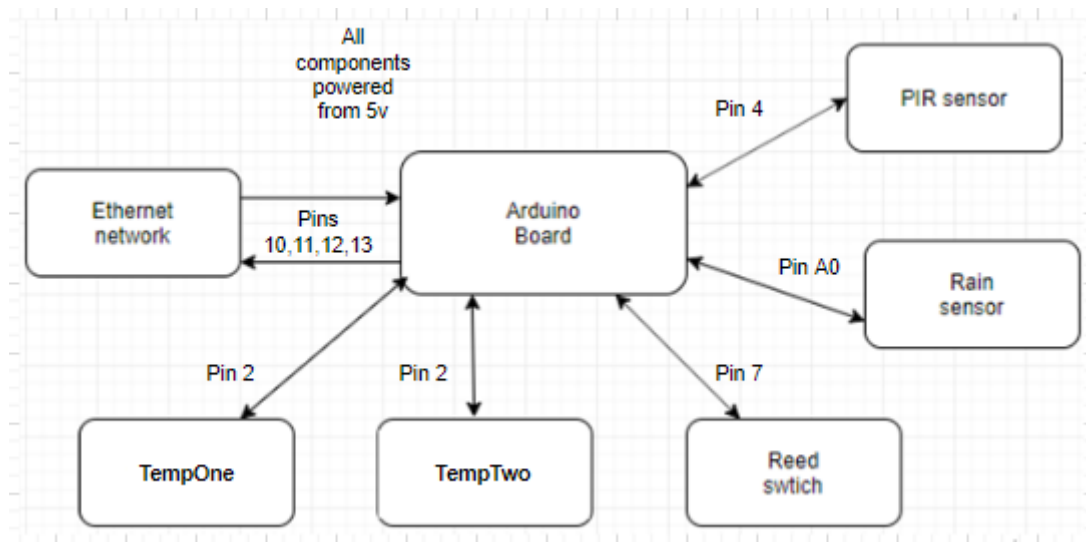


Figure 3.2.1 Hardware design

3.2.1 Ethernet network (Ethernet module ENC28J60)

The Ethernet port will be connected to the Arduino board. This is what will allow the network connection to the microcontroller which will in turn allow it to communicate via HTTP to the Firebase Realtime Database.

3.2.2 Internal temperature sensor (DS18B20 probe)

The internal temperature sensor will be connected to the board and placed within the home. This will collect the temperature data from within the home and send this data to the Arduino microcontroller.

3.2.3 External temperature sensor (DS18B20 probe)

The external temperature sensor will be connected to the board and placed outside the home, which would be best situated on the outside of the window frame. This will collect the external temperature data and send this data to the Arduino microcontroller.

3.2.4 Reed switch (Cylewt reed switch)

The reed switch will be located along the circuit and connected to the window and door. When the switch is closed I.e. does not have a magnetic field running through it this will indicate that the door or window is closed and if it does not have a magnetic field running through it this will indicate the door or window is open. This will allow the microcontroller to determine the open / closed status.

3.2.5 Rain sensor (AZDelivery rain / snow / humidity detection sensor)

The rain sensor will be situated alongside the external temperature sensor, on the outside of the window frame. When this sensor detects moisture, it will then update the Arduino with new data allowing it to determine whether or not it is raining.

3.2.6 PIR sensor (HC-SR501 sensor)

The PIR sensor is an infrared sensor that will be located on the circuit. This sensor can detect movement when the option is enabled by the user in the Android application. If this option is enabled and the sensor detects movement it will send an alarm to the user's mobile informing them of movement within their home.

3.3 Database

See below table 3.3.1 which displays my database in unnormalized form (UNF). UNF is the first stage of database design, this allows the creator to gather all of the attributes, identify the primary key and to name any relations within the database(ryanhmfc.wordpress.com). Performing this stage will allow me to plan my database accurately and precisely covering all the aspects of the database.

Table 3.3.1 Database in UNF format

Database in UNF format	
TempOne	String
TempTwo	String

Reed	String
PIR	String
Rain	String

- TempOne – this will hold the data for the thermometer named TempOne.
- TempTwo – this will hold the data for the thermometer named TempTwo.
- Reed – this will hold the data collected by the Reed Switch to show if the door is open or closed.
- Rain – this will hold the data collected by the rain sensor module.
- PIR – this will hold the data collected by the PIR sensor and display if movement was detected.

See below figure 3.3.2 which shows the entity relationship diagram for the required database. An entity relationship diagram shows the relationships of entity sets stored within a database(smartdraw.com).

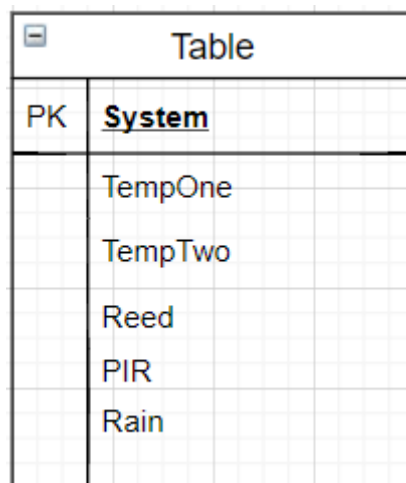
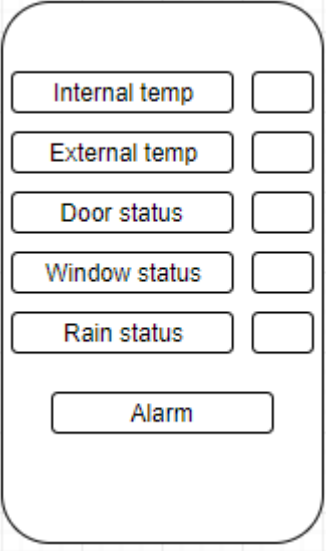
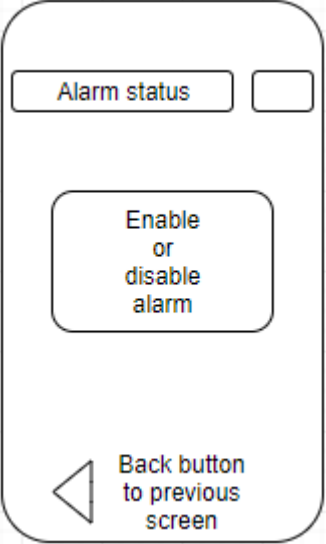


Figure 3.3.2 Entity Relationship Diagram

3.4 Android app Layout

The android application will consist of three screens. The first screen will be the main screen of the application which will display all the data collected by the smart home system to the user. The second screen is accessible from the first screen and will allow

the user to view the status of their alarm and give them the option to enable or disable it. The third screen of the application will be a pop-up screen alerting the user of movement when the alarm is enabled. See below table 3.4.1 outlining these screens.

Screen mock-ups	Features of the screen
 <p>The mock-up shows a mobile app interface with a rounded rectangle. It contains six status items, each with a text label and a small square checkbox to its right: 'Internal temp', 'External temp', 'Door status', 'Window status', 'Rain status', and 'Alarm'.</p>	<p>This screen will allow the user to view all of the data collected by the smart home system. The data will be displayed and updated into a firebase list view on this screen.</p> <p>This screen also allows the user to navigate to the alarm screen.</p>
 <p>The mock-up shows a mobile app interface with a rounded rectangle. It contains three main elements: an 'Alarm status' label with a checkbox to its right, a large button in the center labeled 'Enable or disable alarm', and a back button at the bottom left consisting of a left-pointing triangle and the text 'Back button to previous screen'.</p>	<p>This screen allows the user to view the status of the alarm I.e. whether it is enabled or disabled and will update this data to a textbox on the right of the screen.</p> <p>This screen allows the user to enable or disable the alarm by pressing a button in the middle of the screen.</p> <p>The user can navigate back to the previous informative screen by pressing the back button at the bottom left of the screen.</p>

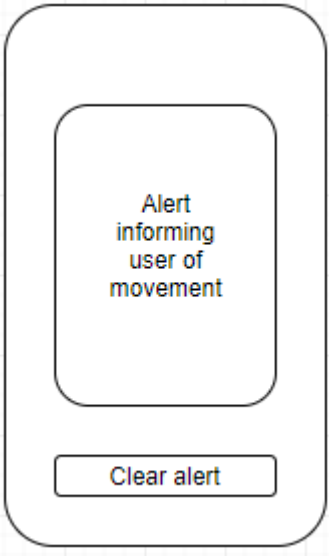
	<p>This screen will pop-up on the user's device whenever the alarm is both enabled and detects movement. The user can clear this alert by pressing on the 'clear alert' button located at the bottom of the screen.</p> <p>When the user presses this button, it will navigate them to the alarm screen where they can choose to disable the alarm if they wish or they can navigate back to the informative screen.</p>
---	--

Table 3.4.1 Screen Mock-ups

3.5 Summary

This chapter outlined the designs that make up the smart home system and has shown the designs for the system architecture, hardware design, database diagrams in UNF format, entity relationship diagram and the android app layout in the form of screen mock-ups. With the design stage of the system completed it will now be time to take action on this and implement the system.

4 Implementation and Testing

The implementation and testing of the planned smart home system is documented in this chapter. This chapter divulges as to why each specific component was chosen and how the system was created. The components that were chosen were tested individually before being connected to the Arduino all together at once. This chapter outlines how the components were tested along with the completed prototype.

4.1 Hardware Components

The selection of the components is crucial to the smart home system functioning as intended. The components that were selected are previously mentioned in 3.2 Hardware Design but will be mentioned again here. The components chosen are as follows:

- Arduino microcontroller
- Ethernet module (ENC28J60)
- Rain sensor
- PIR sensor
- 2 x DS18B20 temperature probes
- Reed switch

As previously mentioned all of these components were tested individually before they were fashioned together to create the entire system, the below subchapters will show the testing results of each hardware component.

4.1.1 Arduino Microcontroller

The microcontroller is the most important component of the project. This is used to run the code and is the component that all other components are connected to and the sensors feed the data back to it. The Arduino was chosen to be the microcontroller for this project as it fits the requirements. The main features that made Arduino a good choice are:

- Cost effective
- Powerful enough for the project
- Allows connections for various sensors
- Has it's own IDE

See below figure 4.1.1.1 outlining the pinout diagram of the Arduino UNO that is being used for this project.

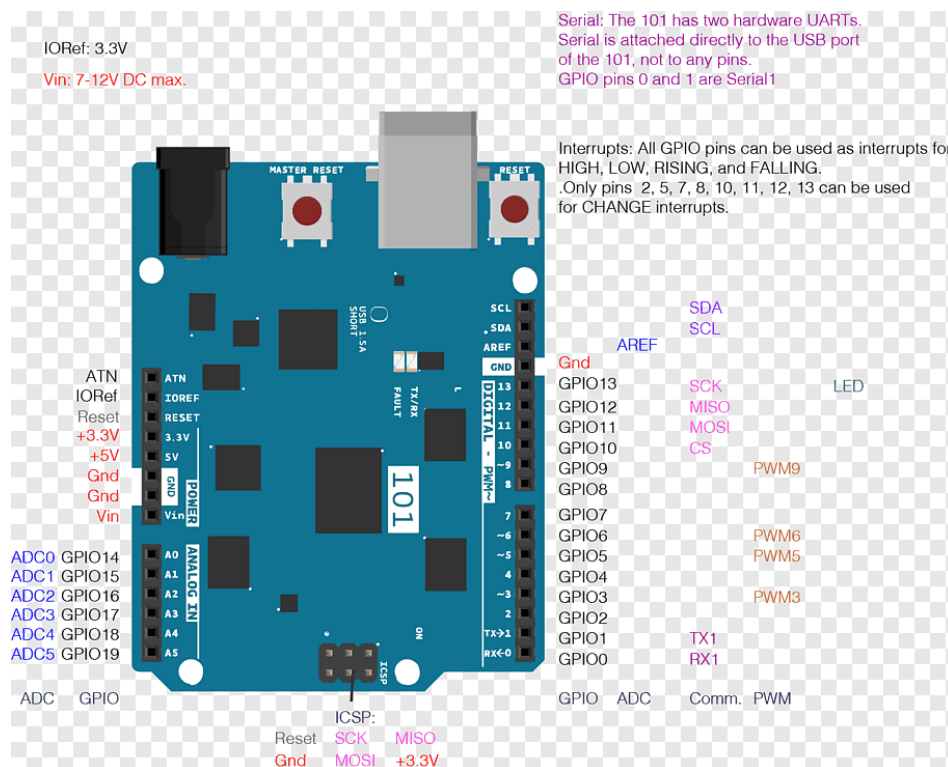


Figure 4.1.1.1

4.1.2 Ethernet module ENC28J60

The Ethernet module is also an extremely crucial component of the smart home system. This is because it is what gives the system and internet connection and how the system communicates with the database hosted on Firebase. Without an internet connection the project simply wouldn't be possible. The ethernet module is an external component to the Arduino that needed programmed before it would be functional. This ethernet module required the Ethercard.h library to be imported to the Arduino IDE. The ethernet module requires 5v of power and is connected to the Arduino on digital pins 10, 11, 12 and 13. Below showing figure 4.1.2.1 is a snippet of code outlining how

the module is assigned a mac address and IPv4 address to allow it to be used to gain internet connection.

```
//Ethernet interface
static byte mymac[] = { 0x74,0x69,0x69,0x2D,0x30,0x31 };
static byte myip[] = { 192,168,0,208 };
```

Figure 4.1.2.1

The Ethernet module was tested to ensure that the system gains an IP address and is able to connect to the internet. An example of this being tested is shown below in figure 4.1.2.2.

```
19:11:59.686 -> Setting up ethernet module.
19:12:00.131 -> IP: 192.168.0.11
19:12:00.131 -> GW: 192.168.0.1
19:12:00.165 -> DNS: 194.168.4.100
19:12:00.165 -> SRV: 160.153.131.190
19:12:00.165 -> Ethernet module set up.
```

Figure 4.1.2.2

The above figure shows the Arduino printing to the serial monitor to display that the ethernet module now has an IP and an internet connection. This is a crucial step to test and ensure is working correctly as without the internet connection no data would be uploaded to the PHP files hosted on <http://matthew-webster.com>.

4.1.3 Rain Sensor

The rain sensor is quite self-explanatory – it is designed to detect rain drops / water and not humidity. This will then feed the data back to the Arduino via the code in figure 4.1.3.1 below.

```
int range = map(sensorReading, sensorMin, sensorMax, 0, 3);

// range value:
switch (range) {
case 0:    // Sensor getting wet
    Serial.println("Raining");
    break;
case 1:    // Sensor getting wet
    Serial.println("Rain Warning");
    break;
case 2:    // Sensor dry
    Serial.println("Not Raining");
    break;
}
delay(1); // delay between reads
```

Figure 4.1.3.1

The rain sensor requires 5v of power and is connected to analogue pin A0 on the Arduino. The rain sensor was also tested to ensure that it was working correctly, this was done by dropping droplets of water onto it and monitoring serial monitor to detect any changes, the results are displayed in figure 4.1.3.2 below.

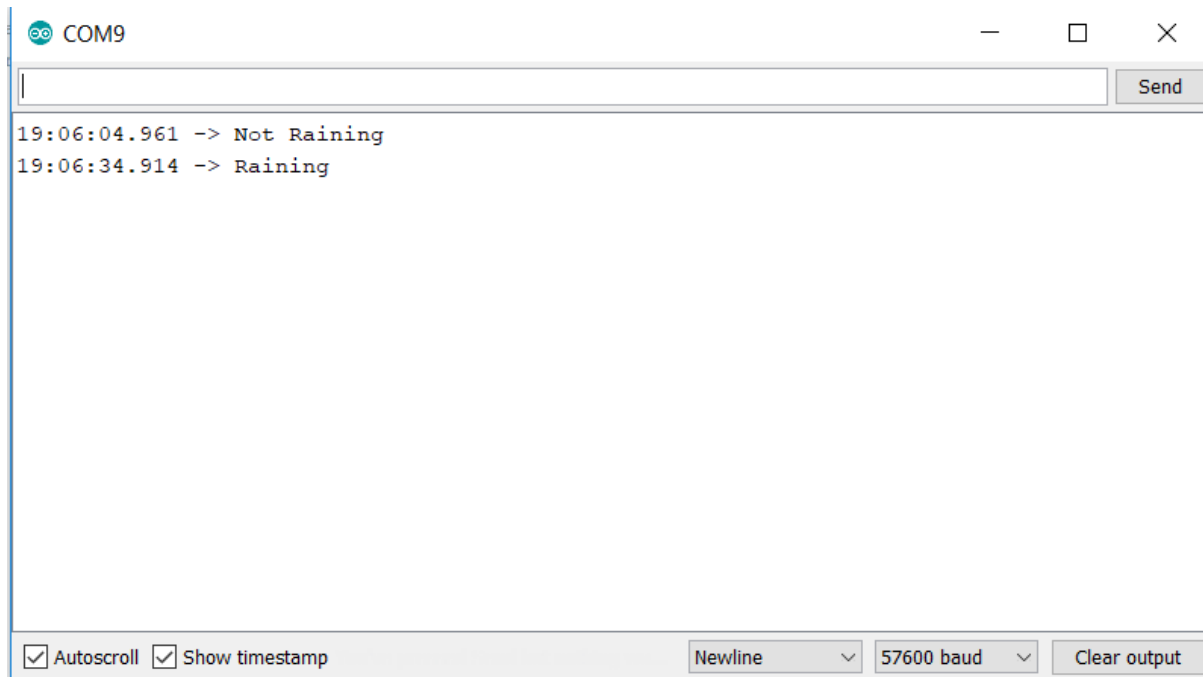


Figure 4.1.3.2

The above results display that the rain sensor is functioning as intended as it detected the water droplets.

4.1.4 PIR sensor

The PIR sensor is used to detect movement and flags the data back to the Arduino. This will work by the sensor lying dormant until the infrared sensor actually picks up movement, once it picks up movement my code outputs "Motion detected!", this will not update again until the input signal to the Arduino is high therefore showing no movement and will output "Motion ended!". Figure 4.1.4.1 below shows how this is implemented in the IDE.


```
val = digitalRead(inputPin); // read input value
if (val == HIGH) {           // check if the input is HIGH
    if (pirState == LOW) {
        // Motion detected
        Serial.println("Motion detected!");
        // Prints on output change
        pirState = HIGH;
    }
} else {
    if (pirState == HIGH){
        // Motion ended
        Serial.println("Motion ended!");
        // Prints on output change
        pirState = LOW;
    }
}
```

Figure 4.1.4.1

The above figure displays how the test code is implemented within the IDE, now to show the results in serial monitor as depicted in figure 4.1.4.2.

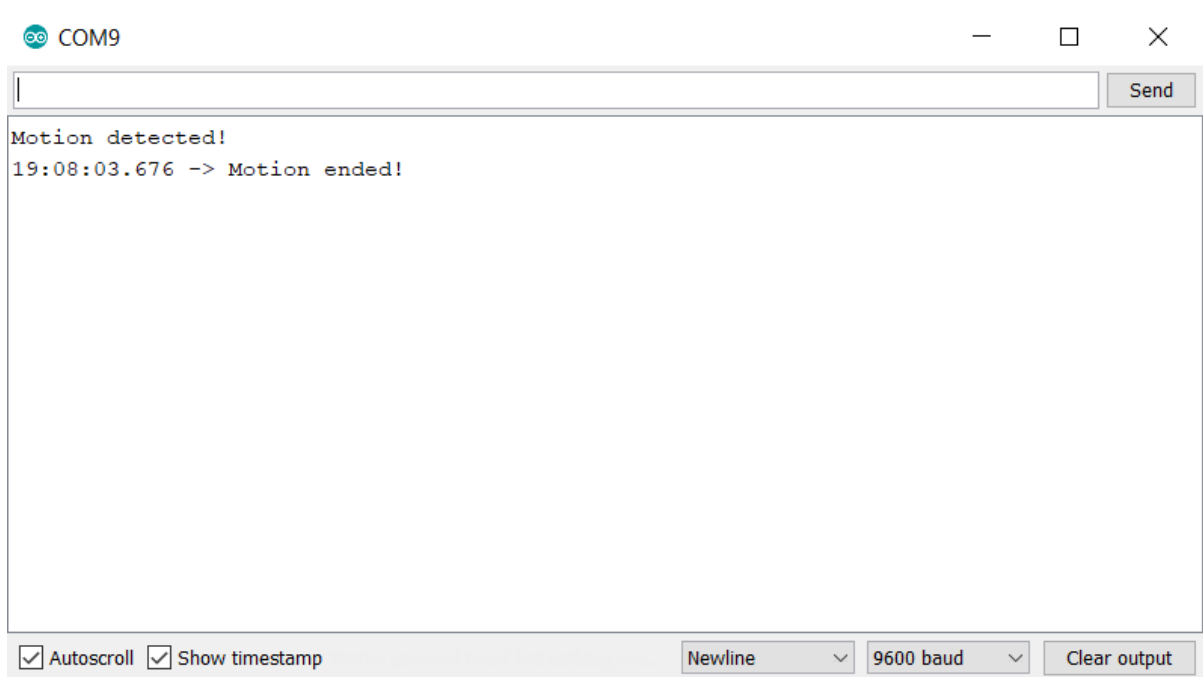


Figure 4.1.4.2

The PIR sensor was tested by moving an object in front of it to check if it would detect the movement and print to the serial monitor. The above figure shows the output to the serial monitor and proved that the PIR sensor was working correctly.

4.1.5 DSB1820 temperature probes

The temperature probes are used to collect live temperature data of the surroundings. They are waterproof so can withstand the elements if one of them is to be used outside and could be used for such a case if required to be placed in a fish tank or something of the likes to collect the temperature data from here. The temperature probes were the most complex components to get configured correctly and connected to the

Arduino as all other components were only connected via the breadboard to get the 5v power from the Arduino whereas the temperature probes themselves are connected onto the breadboard. This is because the probes needed a 4.7k ohm resistor to function correctly - otherwise they will read a consistently incorrect reading. The resistor was placed between the signal cable and the 5v power on the breadboard to correct this issue. Figure 4.1.5.1 shows how these probes are connected on the breadboard with green outlining the signal cable, red as 5v power and blue as the ground.

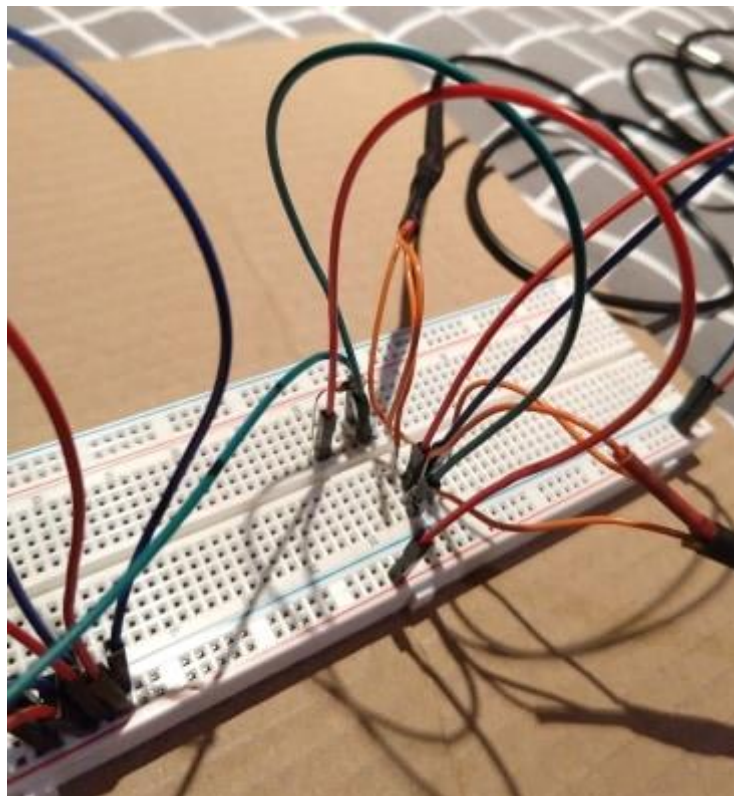


Figure 4.1.5.1

As figure 4.1.5.1 shows, the connections must be bridged on the breadboard to allow these temperature probes to display the correct readings as the 4.7k ohm resistor needs to be in place between the signal cable (green) and the 5v power (red) with the signal cable then connecting from the breadboard to the Arduino on digital port 2.

These sensors also had to be tested to ensure they were working correctly. The below figure 4.1.5.2 shows the testing results.

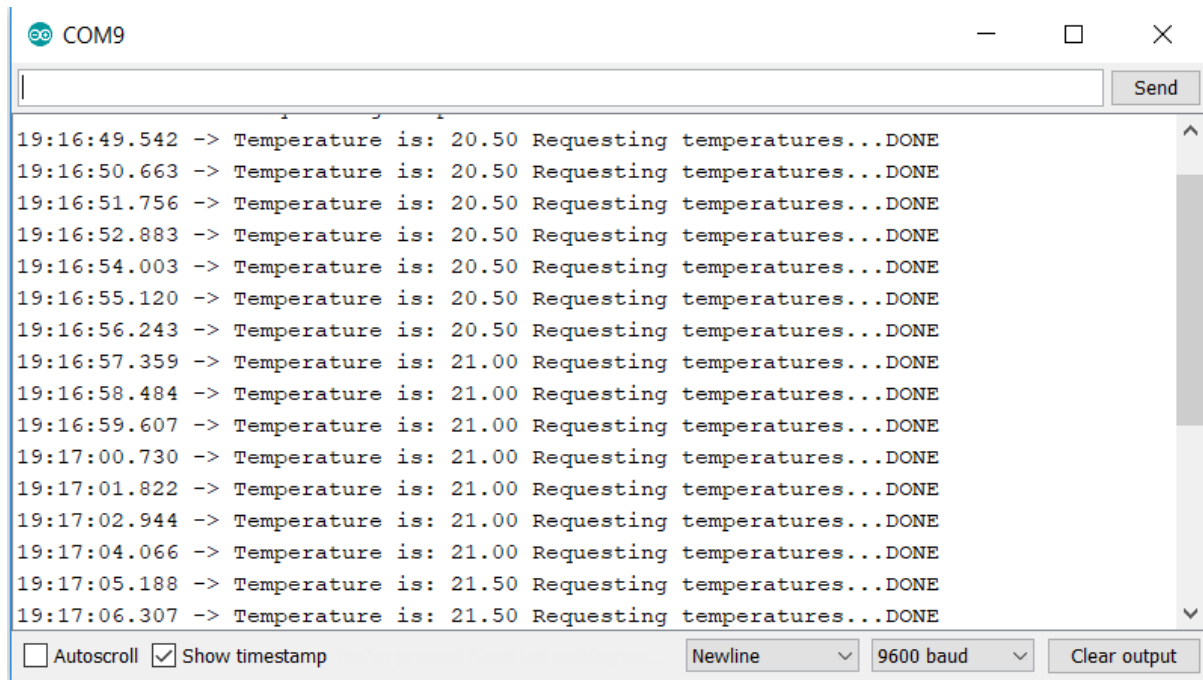


Figure 4.1.5.2

These sensors were first tested and had an issue where they were reading -270 degrees Celsius, this was in the first implementation where a resistor was not installed between the 5V power and signal cables. After this was rectified the sensors were then placed near a warm cup of water to test if they would detect the heat change. The above figure shows the sensors detected the change in heat and outputted this to the serial monitor.

4.1.6 Reed Switch

The reed switch is used in the system to detect if the likes of a door or window is open or closed. The reed switch works via 2 electromagnetic rods encased in a glass tube. When it detects a magnetic force nearby the rods will move together and touch, thus completing the circuit and allowing the Arduino to output the data due to the code behind it. Figure 4.1.6.1 below outlines the code used for the reed switch.

```

{
  int proximity = digitalRead(REED_PIN); // Read the state of the switch
  if (proximity == LOW) // If the pin reads low, the switch is closed.
  {
    Serial.println("Door closed");
  }
  else
  {
    Serial.println("Door open");
  }
}

```

Figure 4.1.6.1

The output of this test code is shown below in figure 4.1.6.2.

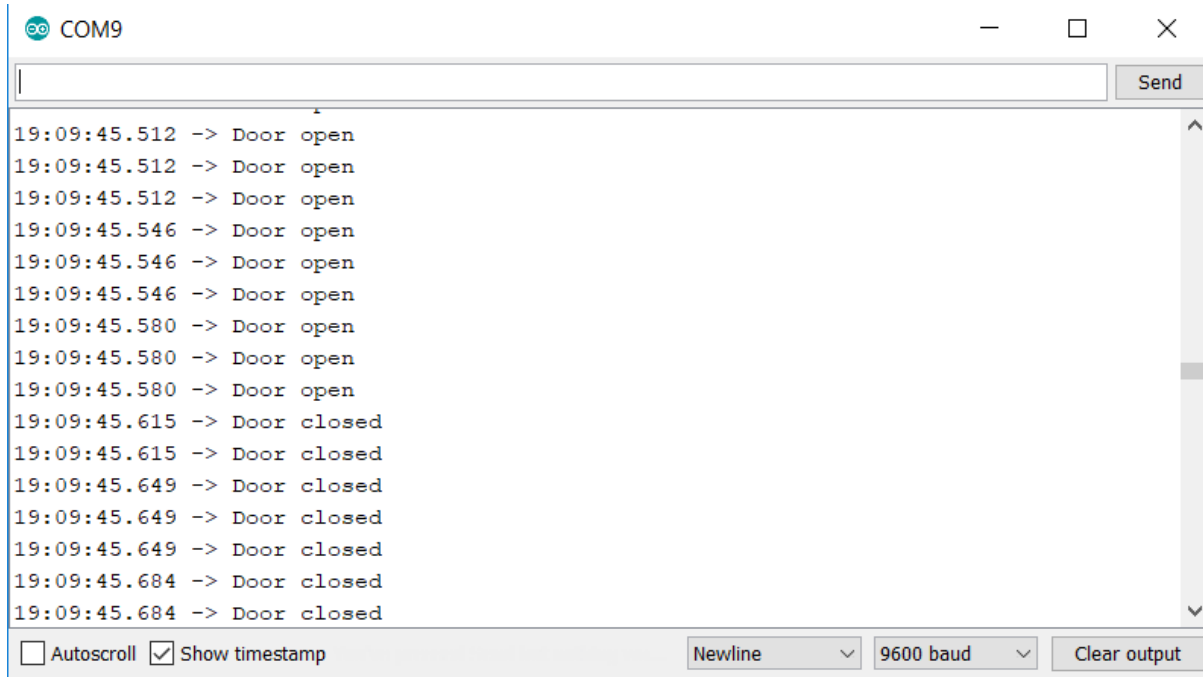


Figure 4.1.6.2

Here the serial monitor is updating correctly when a magnet is held close to the reed switch. This sensor is working correctly.

4.2 Testing

The testing was completed for each individual hardware component before being implemented as a whole system to establish that they were connected to the Arduino / breadboard correctly. Table 4.2.1 displays the hardware testing whereas Table 4.2.2 shows the tests carried out on the software.

Table 4.2.1

Test performed	Expected result	Actual result	Comments
Temperature readings	To display the correct temperature readings	The sensors detected the change in temperature and outputted correctly to the serial monitor	Successful
Rain sensor readings	To display the changes that the sensor reads when it detects water	The system output and monitored the correct changes	Successful

PIR sensor reading	To display when motion is detected	The PIR sensor detected the movement and outputted to serial monitor correctly	Successful
Reed switch reading	To display if the circuit was open or closed	The sensor output correctly	Successful
Test to confirm ethernet module can have an internet connection.	To be able to ping google.	Could ping google.	Successful

Table 4.2.2

Test performed	Expected result	Actual result	Comments
Testing the Arduino could communicate with the PHP files and populate the Realtime firebase	To populate the Realtime firebase	Database was being populated	Successful
Testing a new user could register via firebase authentication on the app	New user account to be registered	User account was registered	Successful
Testing users can log in to the app	Log in successful	Log in was successful	Successful
Testing the navigation within the app	User can switch between app pages smoothly	App navigation functions as intended	Successful
Testing the app could display the firebase data	Sensor readings update within the app	Sensor readings functioned correctly	Successful
Testing the user could log out of the app	App returns to log in screen	App returned to log in screen	Successful

4.3 Summary

Each component was tested and known to be working correctly before they were combined to create the smart home system. After the components were implemented together, they were then checked to ensure no bare cables could touch to cause a short in the system or any inconsistent data produced. After this was checked the same tests were carried out on the implemented system to ensure all sensors were still reading correctly. The next chapter of the report will carry out the evaluation of this project.

5 Evaluation

This chapter evaluates the overall system that has been created. The main purpose of this system was to be a low-cost system that would be affordable for all, if not the majority of users. It is important to evaluate the user and system requirements, both functional and non-functional previously outlined in chapters 2.3.1 and 2.3.2 respectively. Below are tables 5.0.1 and 5.0.2 outlining the functional and non-functional user requirements.

Table 5.0.1 functional user requirements

Functional Requirements	Outcome	Comment
The user can control the system through the mobile app	Needs to be improved	Throughout the development of the system it became clear that there weren't that many features that the user could control via the mobile application, with the alarm feature being the only feature that the user could interact with from their mobile by enabling or disabling the alarm option that would then display a notification on their app if the option 'enable' was selected and the sensor detected movement.
The user can view the data collected through the mobile app	Complete	The data recorded from the system is correctly uploaded to firebase and the mobile app downloads the correct data from the database to display to the user.

The user can enable the alarm through the mobile app	Complete	This feature functioned as intended by giving the user the option to enable or disable the alarm through the mobile app which would then popup a notification to their device when movement was detected from the PIR sensor while the 'enable' option was selected.
The user can force an update of data	Incomplete	This feature was not implemented into the system as it was deemed unnecessary. It would be more beneficial to change the system to upload data to the database more often rather than giving the user an option to force an upload or download of data.
The user will only be able to view their own data	Incomplete	As of yet there is only one database and one system, the app will only display the data collected from both of these.

Table 5.0.2 Non-functional user requirements

Non-functional requirements	Outcome	Comment
The mobile app will be easy to view / use	Complete	The mobile app has a very simplistic design and lay out, making it intuitive for users.
The mobile app will alert the user when the PIR sensor detects movement if it is enabled	Complete	This feature works as intended.
The system will update the data every 30 seconds	Complete	The Firebase database was repopulated every 30 seconds with new data uploaded from the Arduino and then downloaded to the app.
The system won't need to go offline	Mostly complete	The system, while working as intended will not need to go offline, however if any updates are needed to the code, it will need to be reuploaded to the Arduino temporarily knocking it offline for ~30 seconds.
Only the user will be able to view the data	Complete	Only if the user is signed into the mobile application via

		firebase authentication the data cannot be viewed.
--	--	--

As can be seen from the above tables, the majority of requirements have been met, however there is still a lot of room for improvement within the system. These will be highlighted after the functional and non-functional system requirements have been evaluated in the below tables 5.0.3 and 5.0.4 in chapter 5.1.

Table 5.0.3 functional system requirements

Functional requirements	Outcome	Comment
The user should have no reason to interact with the hardware of the system	Complete	The user will not need to touch the hardware once it has been plugged into a power source.
The system must collect new data every 30 seconds	Complete	The Arduino uploads new data collected from the sensors every 30 seconds to Firebase.
The system must be able to communicate with the Firebase database	Complete	The Arduino can communicate with the database and populate it with the relevant data.
The system must be secure to not allow any intruders to access the data	Partially complete	This can be improved upon, there has been no pen testing performed. However, the Arduino uploads to Firebase via HTTPS on port 443 which is much more secure than HTTP on port 80 due to SSL.
The mobile app must be compatible with Android	Complete	This requirement has been met with the app being developed within Android Studio.
The system must always have a network connection	Mostly complete	The system, while working as intended will not need to go offline, however if any updates are needed to the code, it will need to be reuploaded to the Arduino temporarily knocking it offline for ~30 seconds.
The system must be easy to install	Complete	With the code uploaded to the Arduino all that is required is a power source for the smart home system to run as intended.

Table 5.0.4 non-functional system requirements

Non-functional requirements	Outcome	Comment
The system must be cheap to construct	Complete	This requirement was met, the entire system costs only £18.03 to construct at the time of writing.
The system shouldn't stand out aesthetically	Incomplete	This requirement was not met. The system definitely stands out aesthetically.
The system must be robust	Incomplete	This requirement was not met, the system would possibly cease to function if there was a spillage of liquid over the microcontroller and or it was picked up or moved aggressively as there is a possibility of shorting and cables being disconnected.
The mobile app must be user friendly	Complete	This requirement was met, the app is user friendly with a simplistic design allowing for intuitive use.
The mobile app's UI should be simplistic	Complete	This was requirement was completed as the app is simplistic in design and use.

From the above tables it shows that the majority of requirements have been met once again for the system requirements however there is room for improvement and features could have been implemented more efficiently with more functionality. There is also changes that could be made to the system to provide the user with a more interactive and functional system, these will be highlighted in the below chapter.

6 Reflection and Conclusion

6.1 Reflection

Looking back on the development of the system and having learnt a vast amount about both Arduino and PHP if the development of the project would be restarted it would have been tackled differently.

Prior to starting the development of the system more research time would be assigned to Arduino to understand both the hardware and software of this microcontroller and to gain some experience with it before starting to develop a project so minor issues don't cause as much of a delay of development as they did.

More research time would also be assigned to figuring out how it is possible to upload the data from the Arduino to Firebase and exploring these options to avoid more delays in the development of the system.

If the project were to restart development a more solid plan of the systems architecture would be drawn out in order to easily build the physical system instead of having to perform a trial and error approach which can cause a lot of stress and consumes large amounts of time.

A good feature of the system would be that it produces the results that were intended - to put it simply, the user of the app can view the information that is collected by the Arduino on their android app. The way the project was designed, it allows for the user to not have to be on the same local network as the system due to Firebase hosting the database, allowing for connections from anywhere as long as the user is authenticated through Firebase authentication.

A bad feature of the system is that it is not wireless – to be discussed more in chapter 6.2 – Improvements. By having the system communicating over data cables doesn't allow for the most practical use as there is simply too many cables taking up too much physical space. There is also the issue of the sensors not being able to necessarily reach where a user would like to place them, such as the rain sensor being under the sink to detect any leaks – with the cabled system this is not possible but instead shows the prototype as being functional and proving it's validity.

The most difficult feature of this project was the connection to Firebase from the Arduino. This is because not enough research time was spent to figure out how it would be possible to connect it instead of assuming it could go directly from the Arduino to the database. Prior to this project, the knowledge of Arduino and PHP was non-existent as it is not something that I have been educated in so it was very much trial and error with many, many hours spent researching these languages to find out how it would be possible to complete this key feature of the project.

The easiest feature of the project to implement was the development of the android application, this is due to it being quite a simple application and prior to this project starting there was previous experience in developing apps with Java and connecting them with Firebase authentication and database. It is also quite a simplistic app to develop as there are not too many advanced features. The alarm notification and being able to enable / disable the PIR sensor from the app seemed like an advanced feature so the main components of the application were developed first and this alarm feature was not implemented due to time constraints, therefore allowing it to be developed for a future prototype of this system.

Overall the project is deemed successful as it has delivered a smart home system at a low cost of £18.03 for the components. If this same project was to be developed again in summary there are a number of things that would be tackled differently such as building it with an Arduino compatible with Wi-Fi and Wi-Fi enabled sensors to make it completely wireless, more research time would be assigned to the key features of the system to avoid long developmental delays and research into the Arduino and PHP languages.

This project was challenging to complete as I had no knowledge of Arduino or PHP before development of it began, making it very much a trial and error-based approach to completing it with many hours spent researching how these languages work. However, as it was challenging and not straightforward it has taught me a lot about Arduino and PHP specifically and how it is possible to develop with these languages. Building the physical system was also something I had no experience with which took quite a long time with the majority of it spent researching how to connect each individual sensor and how the breadboard should be engineered to distribute the power and signal cables. The project seemed to be a mix of computer science with

the android app (Java) and the PHP development to allow for the Firebase connection and computer and electrical engineering with the Arduino building, development and testing.

6.2 Improvements

As can be seen in tables 5.0.1 through 5.0.4 the majority of requirements have been completed with the main focus to keeping the system as low cost as possible. That being said there are many areas that could be improved within this project to make it more functional and effective.

The first improvement and the one that would provide the most functionality would be to use an Arduino that has built in WIFI such as an Arduino WIFI Rev 2, which would be mildly more expensive therefore it may be a better option to keep the cost down to use a WIFI module instead of an ethernet module. This would allow for the Arduino to be placed anywhere within the users' home instead of needing access to the router allowing it to be much more discrete and would complete the requirement that the system mustn't stand out aesthetically. Using a WIFI module would also complete the requirement of the system needing to be robust as it could provide a wireless system that would not be as easily breakable.

Another improvement that would allow for a much more futureproof and scalable project would be to use wireless sensors alongside the WIFI module. This would allow the sensors to report back to the Arduino via WIFI, making the system wireless. This improvement would be massive over the current implementation of the system, it would create a much more effective and functional system that would give the user much more freedom over how they would like the system to be implemented. An example of this would be to use a wireless rain sensor and place it under the kitchen sink or in the loft that would then detect any leaks within the users' house before any serious damage could occur. This would be the case for any other wireless sensors, the user would be free to place them wherever they wished, allowing for a scalable "create your own" smart home system with the components being sourced at a low cost.

Another improvement to the system that would benefit the user would be to add more functionality into the app with an example of this being to have an option for how often

they would like the data downloaded from the Firebase database and they could choose say every 2 seconds. This could be implemented by having the Arduino upload the sensors' data in Realtime to Firebase and then having the mobile app download it to whatever timeframe the user selects. This would resolve and exceed the incomplete functionality of the user being able to force an update of the data. Another improvement to the android app would be to redesign the UI to make it more aesthetically pleasing.

While the system displays area of large improvement to make it much more scalable and functional it shows that a low cost smart home system can be created and function as intended, with the potential of future developments.

6.3 Conclusion

The aim of this project was to develop a low-cost smart home system that could collect data from a user's home and display it to them via an Android app. This was achieved using Arduino as the microcontroller of the system, the sensors indexed in chapter 4.1, Firebase Realtime Database and an Android application which was developed. The project required a mix of technologies to make this possible, such as Arduino IDE, Android Studio, PHP, Firebase, computer science, computer engineering and electrical engineering.

While the system is functional it has shown large areas of improvement that could make it much more beneficial to end users and would provide a much better base for futureproofing and scalability.

A different approach would be taken if this project was to be repeated, specifically in the early stages of data collection and research. The design stage would also be taken in a different approach to reach the end goal of a system that is fully scalable.

7 References

Cambridge University Press. (2019). Meaning of feasibility study in English. Available: <https://dictionary.cambridge.org/dictionary/english/feasibility-study>. Last accessed 23/10/2019.

James Gill. (2018). Amazon Prime Video guide: what to watch and how much it costs. Available: <https://www.radiotimes.com/news/on-demand/2019-04-05/amazon-prime-video-cost-guide-what-to-watch-how-to-sign-up/>. Last accessed 23/10/2019.

Martyn Casserly. (2019). iPhone vs Android market share. Available: <https://www.macworld.co.uk/feature/iphone/iphone-vs-android-market-share-3691861/>. Last accessed 22nd October 2019.

Martyn Williams. (2018). Amazon Echo Dot vs. Google Home Mini. Available: <https://www.techhive.com/article/3289344/amazon-echo-dot-vs-google-home-mini.html>. Last accessed 21st October 2019.

PingWave. (2020). Arduino Uno Pinout Microcontroller General-purpose input/output, others PNG. Available: <https://www.pngwave.com/png-clip-art-vggqs>. Last accessed 20/03/2019.

Ryan Lawson. (2012). Normalisation (UNF, 1NF, 2NF & 3NF). Available: <https://ryanhmfc.wordpress.com/2012/09/13/normalisation-unf-1nf-2nf-3nf/>. Last accessed 15/11/2019.

Shanhong Liu. (2019). iPhones installed base in the United States, China and ROW 2017. Available: <https://www.statista.com/statistics/755625/iphones-in-use-in-us-china-and-rest-of-the-world/>. Last accessed 22nd October 2019.

SmartDraw. (2019). What is an Entity Relationship Diagram (ERD)?. Available: <https://www.smartdraw.com/entity-relationship-diagram/>. Last accessed 15/11/2019.

Sonny Ali, Zia Yusuf. (2018). Mapping the Smart-Home Market. Available: <https://www.bcg.com/en-gb/publications/2018/mapping-smart-home-market.aspx>. Last accessed 17/10/2019.

Zion Market Research. (2017). Forecast market size of the global smart home market from 2016 to 2022 (in billion U.S. dollars)*. Statista. Statista Inc. Accessed: October 16, 2019. <https://www.statista.com/statistics/682204/global-smart-home-market-size/>

8 Appendices

8.1 Appendix A Code

Arduino code:

```
SmartHomeSystem

//library required by the ENC28J60 Ethernet Module
#include <EtherCard.h>

//libraries required by the DS18B20 sensors
#include <OneWire.h>
#include <DallasTemperature.h>

//TEMPERATURE
#define number_of_sensors 2
#define one_wire_bus 2 //data pin
#define temperature_precision 9 // Lower resolution
//Sets up a onewire instance to communicate with other onewire devices
OneWire oneWire(one_wire_bus);
//Passes oneWire reference to dallas temperature
DallasTemperature sensors(&oneWire);
DeviceAddress tempDeviceAddress; // Variable used to store a found devices address

//RAIN
const int analog_min = 0; // sensor minimum
const int analog_max = 1024; // sensor maximum
#define rain_pin A0 //the pin used by the rain sensor

//REED
#define reed_pin 7 // Pin connected to reed switch

//PIR
#define pir_pin 4 // choose the input pin (for PIR sensor)
bool pir_state = LOW; // we start, assuming no motion detected

//PHP
#define PATH "firebaseUpload.php" //the path of the php file responsible to upload to firebase
const char website[] PROGMEM = "www.matthew-webster.com";
```

```
//ETHERNET MODULE
const byte ethernet_mac[] = { 0x74, 0x69, 0x69, 0x2D, 0x30, 0x31 }; //mac address of the ethernet module
byte Ethernet::buffer[500];
Stash stash;

void setup_temperature_sensors() {
    Serial.println("Setting up temperature sensors");
    sensors.begin();
    int error_code = 0;
    //check if both sensors are connected to the onewire bus.
    int number_of_sensors_found = sensors.getDeviceCount();
    if (number_of_sensors_found != number_of_sensors) {
        Serial.println("Unable to found the exact number of sensors");
        Serial.print("Expected: "); Serial.print(number_of_sensors);
        Serial.print(" Found: "); Serial.println(number_of_sensors_found);
        error_code = 1;
    }

    Serial.print("Setting temperature resolution to "); Serial.println(temperature_precision);
    for (int i = 0; i < number_of_sensors; i++) {
        if (sensors.getAddress(tempDeviceAddress, i)) {
            sensors.setResolution(tempDeviceAddress, temperature_precision);
        } else {
            error_code = 2;
        }
    }

    Serial.print("Temperature sensors set up.");
    if (error_code != 0) {
        Serial.print(" Error code = "); Serial.println(error_code);
    }
    Serial.println();
}
```

SmartHomeSystem

```
void setup_ethernet_module() {
    Serial.println("Setting up ethernet module.");
    int error_code = 0;

    //initialize a new instance for the ethernet module with the
    //the specified buffer (make sure that the buffer isn't too big) and
    //the mac address
    if (ether.begin(sizeof Ethernet::buffer, ethernet_mac) == 0) {
        error_code = 1;
        Serial.println("Unable to connect to the Ethernet module.");
    }

    //ask the router for an IP address.
    if (!ether.dhcpSetup()) {
        error_code = 2;
        Serial.println("DHCP failed");
    }

    ether.printIp("IP: ", ether.myip);
    ether.printIp("GW: ", ether.gwip);
    ether.printIp("DNS: ", ether.dnsip);

    if (!ether.dnsLookup(website)) {
        error_code = 3;
        Serial.println("Unable to find the IP of the host. DNS failed.");
    }
    else {
        ether.printIp("SRV: ", ether.hisip);
    }

    Serial.print("Ethernet module set up.");
    if (error_code != 0) {
        Serial.print(" Error code = "); Serial.println(error_code);
    }
    Serial.println();
}
```

```
void setup_reed_switch() {  
  
    pinMode(reed_pin, INPUT_PULLUP);  
}  
  
void setup_pir_sensor() {  
    pinMode(pir_pin, INPUT);  
}  
void setup() {  
  
    Serial.begin(57600);  
    setup_temperature_sensors();  
    setup_ethernet_module();  
    setup_reed_switch();  
    setup_pir_sensor();  
  
}  
  
long lastTimeUploaded = millis();  
struct info {  
    double tempOne;  
    double tempTwo;  
    String rainStatus;  
    String reedStatus;  
    String pirStatus;  
} uploadData;  
  
void refresh_temperatures() {  
    //Send the command to the sensors to send the value of temperature  
    sensors.requestTemperatures();  
  
    //Get the temperature from the first sensor
```

```
if (sensors.getAddress(tempDeviceAddress, 0)) {
  /*store the temperature for the first sensor in the struct
  uploadData, field tempOne. */
  uploadData.tempOne = sensors.getTempC(tempDeviceAddress);
} else {
  /*if the address is invalid, it means it is a ghost device,
  so we store a value that indicates that there is an error */
  uploadData.tempOne = -9999;
}

//Get the temperature from the second sensor
if (sensors.getAddress(tempDeviceAddress, 1)) {
  /*store the temperature for the first sensor in the struct
  uploadData, field tempTwo. */
  uploadData.tempTwo = sensors.getTempC(tempDeviceAddress);
} else {
  /*if the address is invalid, it means it is a ghost device,
  so we store a value that indicates that there is an error */
  uploadData.tempTwo = -9999;
}
delay(5);
}

void refresh_rain() {
  /*Read the voltage on the analog pin rain_pin

  */
  int sensorReading = analogRead(rain_pin);
  /*The map function re-maps a number from one range (analog_min - analog_max)
  to another range (0,3).
  */
  int range = map(sensorReading, analog_min, analog_max, 0, 3);
```

```
int range = map(sensorReading, analog_min, analog_max, 0, 3);

switch (range) {
  case 0:    // Sensor getting wet -> Raining
    uploadData.rainStatus = "Raining";
    break;
  case 1:    // Sensor getting wet -> Rain Warning
    uploadData.rainStatus = "Rain Warning";
    break;
  case 2:    // Sensor dry -> "Not Raining";
    uploadData.rainStatus = "Not Raining";
    break;
  default:
    uploadData.rainStatus = "Undefined state";
    break;
}
delay(5);

oid refresh_reed() {
  /*A reed switch is a switch activated by a magnet.
  */
  bool proximity = digitalRead(reed_pin);
  if (proximity == LOW) {
    /* if the reed switch is activated, it means that the door is closed,
    so we store in the uploadData struct, field reedStatus the status
    "Door closed".
    */
    uploadData.reedStatus = "Door closed";
  }
  else {
    /* if the reed switch is deactivated, it means that the door is open,
    so we store in the uploadData struct, field reedStatus the status
```

```
    /* if the reed switch is deactivated, it means that the door is open,
       so we store in the uploadData struct, field reedStatus the status
       "Door open".
    */
    uploadData.reedStatus = "Door open";
}

}

void refresh_pir() {

    bool pir_reading = digitalRead(pir_pin); // read input value
    if (pir_reading == HIGH) {               // check if the input is HIGH
        if (pir_state == LOW) {
            // we have just turned on
            uploadData.pirStatus = "Motion detected";
            //Serial.println("Motion detected!");
            // We only want to print on the output change, not state
            pir_state = HIGH;
        }
    } else {
        if (pir_state == HIGH) {
            // we have just turned off
            uploadData.pirStatus = "Motion ended";
            //Serial.println("Motion ended!");
            // We only want to print on the output change, not state
            pir_state = LOW;
        }
    }
}
```

```
void loop() {  
  ether.packetLoop(ether.packetReceive());  
  
  refresh_temperatures();  
  refresh_rain();  
  refresh_reed();  
  refresh_pir();  
  
  long currentTime = millis();  
  if (currentTime - lastTimeUploaded > 5000) {  
    Serial.println("Display data:");  
    Serial.print("TempOne = "); Serial.print(uploadData.tempOne);  
    Serial.print("  TempTwo = "); Serial.println(uploadData.tempTwo);  
    Serial.print("Reed status = "); Serial.println(uploadData.reedStatus);  
    Serial.print("Pir status = "); Serial.println(uploadData.pirStatus);  
    Serial.print("Rain status = "); Serial.println(uploadData.rainStatus);  
    uploadDataToFirebase();  
    lastTimeUploaded = millis();  
  }  
  
  delay(10);  
}
```

```
void uploadDataToFirebase() {
    Serial.println("Sending new data to server");

    byte sd = stash.create();

    stash.print("tempOne="); stash.print(uploadData.tempOne);
    stash.print("&tempTwo="); stash.print(uploadData.tempTwo);
    stash.print("&reed="); stash.print(uploadData.reedStatus);
    stash.print("&rain="); stash.print(uploadData.rainStatus);
    stash.print("&pir="); stash.print(uploadData.pirStatus);

    stash.save();

    // generate the header with payload - note that the stash size is used,
    // and that a "stash descriptor" is passed in as argument using "$H"
    Stash::prepare(PSTR("POST http://$F/$F HTTP/1.1" "\r\n"
        "Host: $F" "\r\n"
        "Content-Length: $D" "\r\n"
        "Content-Type: application/x-www-form-urlencoded" "\r\n"
        "\r\n"
        "$H"),
        website, PSTR(PATH), website, stash.size(), sd);

    // send the packet - this also releases all stash buffers once done
    ether.tcpSend();
}
```

firebaseLib.php:

```
firebaseLib.php
1 <?php
2
3
4 class Firebase
5 {
6
7     private $_baseURI;
8     private $_timeout;
9     private $_token;
10
11
12     //Constructor
13     //@param String $baseURI Base URI
14
15     function __construct($baseURI = '', $token = '')
16     {
17         if (!extension_loaded('curl')) {
18             trigger_error('Extension CURL is not loaded.', E_USER_ERROR);
19         }
20
21         $this->setBaseURI($baseURI);
22         $this->setTimeout(10);
23         $this->setToken($token);
24     }
25
26     //Sets Token
27
28     public function setToken($token)
29     {
30         $this->_token = $token;
31     }
32
33
34     //Sets Base URI
35
36     public function setBaseURI($baseURI)
37     {
38         $baseURI .= (substr($baseURI, -1) == '/' ? '' : '/');
39         $this->_baseURI = $baseURI;
40     }
41
42
43     //Returns with the normalized JSON absolute path
44
45     private function _getJSONPath($path)
46     {
47         {
48             $url = $this->_baseURI;
49             $path = ltrim($path, '/');
50             $auth = ($this->_token == '') ? '' : '?auth=' . $this->_token;
51             return $url . $path . '.json' . $auth;
52         }
53     }
54
55     //Sets REST call timeout in seconds
56
57     public function setTimeout($seconds)
58     {
59         $this->_timeout = $seconds;
60     }
61
62
63     //Writing data into Firebase with a PUT request
64
65     public function set($path, $data)
66     {
67         return $this->_writeData($path, $data, 'PUT');
68     }
69
70
71     //Pushing data into Firebase with a POST request
72
73     public function push($path, $data)
74     {
75         return $this->_writeData($path, $data, 'POST');
76     }
77 }
```



```

79 //Updating data into Firebase with a PATH request
80
81 public function update($path, $data)
82 {
83     return $this->_writeData($path, $data, 'PATCH');
84 }
85
86 //Reading data from Firebase
87
88 public function get($path)
89 {
90     try {
91         $ch = $this->_getCurlHandler($path, 'GET');
92         $return = curl_exec($ch);
93         curl_close($ch);
94     } catch (Exception $e) {
95         $return = null;
96     }
97     return $return;
98 }
99
100 //Deletes data from Firebase
101
102 public function delete($path)
103 {
104     try {
105         $ch = $this->_getCurlHandler($path, 'DELETE');
106         $return = curl_exec($ch);
107         curl_close($ch);
108     } catch (Exception $e) {
109         $return = null;
110     }
111     return $return;
112 }
113
114 //Returns with Initialized CURL Handler
115
116 private function _getCurlHandler($path, $mode)
117 {
118     $url = $this->_getJSONPath($path);
119     $ch = curl_init();
120     curl_setopt($ch, CURLOPT_URL, $url);
121     curl_setopt($ch, CURLOPT_TIMEOUT, $this->_timeout);
122     curl_setopt($ch, CURLOPT_CONNECTTIMEOUT, $this->_timeout);
123     curl_setopt($ch, CURLOPT_SSL_VERIFYPEER, false);
124     curl_setopt($ch, CURLOPT_RETURNTRANSFER, true);
125     curl_setopt($ch, CURLOPT_CUSTOMREQUEST, $mode);
126     return $ch;
127 }
128
129
130 private function _writeData($path, $data, $method = 'PUT')
131 {
132     $jsonData = $data;
133
134     $header = array(
135         'Content-Type: application/json',
136         'Content-Length: ' . strlen($jsonData)
137     );
138
139
140     try {
141         $ch = $this->_getCurlHandler($path, $method);
142         curl_setopt($ch, CURLOPT_HTTPHEADER, $header);
143         curl_setopt($ch, CURLOPT_POSTFIELDS, $jsonData);
144
145         $return = curl_exec($ch);
146         curl_close($ch);
147     } catch (Exception $e) {
148         $return = null;
149     }
150
151     return $return;
152
153 }
154

```

firebaseUpload.php:

```
firebaseUpload.php ●
1 <html>
2 <?php
3
4 //import the firebaseLib, which is responsible to upload data
5 //to the firebase database.
6 require_once 'firebaseLib.php';
7
8 //credentials for the firebase.
9 $firebase_url = 'https://smart-home-system-3fc1a.firebaseio.com/';
10 $firebase_token = 'cPKLxS3tXs01L0xxBpRbGjvTXITauzKEIpsVYhSe';
11
12
13
14 //store the variables passed in the POST method
15 //$_POST['key'] return the value of the variable with the matching 'key'
16 $tempOne = $_POST['tempOne'];
17 $tempTwo = $_POST['tempTwo'];
18 $reedStatus = $_POST['reed'];
19 $rainStatus = $_POST['rain'];
20 $pirStatus = $_POST['pir'];
21
22
23 //as the json_encode is missing from the host, we have to manually create
24 //a json with the data received from POST method.
25 $json_object = '{"TempOne": "'. $tempOne.'",
26                 "TempTwo": "'. $tempTwo.'",
27                 "Reed": "'. $reedStatus.'",
28                 "Rain": "'. $rainStatus.'",
29                 "PIR": "'. $pirStatus.'"
30                 }';
31
32
33
34 //specify the path where you want to store the new json.
35 $firebasePath = '/System';
36
37 //Create a new instance for the firebase, then update in the specified path.
38 $fb = new firebase($firebase_url, $firebase_token);
39
40 //Upload the data to the desired path, in our case "System"
41 $response = $fb->update($firebasePath, $json_object);
42
43 echo $response;
44
45 ?>
46 </html>
```