Diploma Thesis

SMART HOME AND INTERNET OF THINGS
BUILD USING A RASPBERRY PI

ANDREAS PARASKEVA

UNIVERSITY OF CYPRUS

DEPARTMENT OF COMPUTER SCIENCE

January 2021

UNIVERSITY OF CYPRUS

COMPUTER SCIENCE DEPARTMENT

Smart Home and Internet of Things build using a Raspberry Pi

Andreas Paraskeva

Supervising Teacher

Dr. Andreas Pitsillides

This Diploma Project is submitted for partial fulfillment of the requirements for acquiring a degree in Computer Science from the University of Cyprus

January 2021

Acknowledgements

I would like to thank my supervising professor, Dr. Andreas Pitsillides for entrusting and choosing me in order to complete the following dissertation project, and providing guidance and feedback throughout.

Furthermore, I would like to thank all my professors at the University of Cyprus, for providing the knowledge and resources, that acted as the building blocks of my project, and had a huge impact into shaping me into the person that I am today.

Additionally, I would like to thank friends and colleagues, which evaluated the system developed, and their feedback was valuable for finalizing the design of the system, as well as expanding some aspects of it, with the goal of achieving a good user experience.

Finally, I would like to thank my family and friends for their understanding, continuous support, guidance, and their role in the never-ending process of character-development.

Abstract

The term Internet of Things is becoming increasingly more familiar to our society and the benefits of platforms integrating such devices are more evident. There are many fields of applications, that have embraced the usage of Internet of Things devices, that exhibited great improvements compared to the traditional, and previously employed method for achieving certain goals. Due to the abundance of network appliances in our daily lives and our domestic environment, the idea of developing a smart home is popular and attainable. The development of a smart home will enhance the quality of our lives, by automating and optimizing tasks that previously required manual effort or physical presence to be completed. Optimization of these tasks can be accomplished, by collecting data from these smart appliances and with analysis and manipulation, manage to operate them with greater utilization. Therefore, in an attempt take full advantage of household appliances and improve our lives, whilst providing an environmentally friendly solution, a smart home projects need to be built.

This dissertation designs and implements such a smart home project that can be accessed remotely through a computer or a smartphone device. The user is given the ability to monitor and control their smart appliances and the data collected from them at any given time, and with the usage of a User Interface, provide these features in a visual and comprehensive format. Automations provided, aim to improve the experience of the user both during the presence or absence from the household. This includes automated lights based on motion detection, and geo-location tracking of the user in order to automate task based on their location, according to their departure or arrival from and to their home zone. Light automations are also implemented in accordance with the status of a video stream, to enhance their video watching experience.

Table of Contents

Chapter 1

1. Introduction

1.1 General Introduction

The impact of technological advancements in our lives is undeniable. Technology is a part of many scientific fields and has a tremendous effect in many important aspects of our life. As the technology becomes readily available with a more affordable pricing, industries as well as individuals tend to switch to and rely on such equipment. Most of our technological advancements, especially the ones that we rely on in our everyday lives, emerged through the development of the Internet. The Internet has allowed humans to break many formerly existing boundaries. Several things that we take for granted today, as simple as talking with another human being through a service that supports video call, were unimaginable a couple of decades ago. One of the most important breakthroughs of the Internet is the collection and availability of vast information from the comfort of our own homes. Thus, amongst others, the **I**nternet **o**f **T**hings (IoT) devices emerged.

The Internet provides the user platform to control these smart devices and to collect data from them. The collection of such data and their analysis, based on big data technology, will in turn allow the users to enhance the efficiency of their devices, maximizing automations and minimizing their own involvement in mundane, previously manually executed tasks. Based on today's pricing and availability, IoT devices are domestically used and the idea of developing a smart home is gaining popularity.

1.2 Problem Statement and Motivation

Our daily lives have become more complex and hectic. Thus, efforts to release stress or human involvement, have a positive reception. Hence, the motivation and interest in this specific project, arose from this upcoming desire for simplicity and automations in our lives. The implementation of smart devices in our environment, and the development of a smart home, is a growing idea with great prospects in the aforesaid direction.

After conducting research on pricing, availability, and feedback from previous similar projects, it was clear that the market had to offer numerous IoT capable devices, at affordable prices that matched such demands and necessities. These independent devices require a centralized hub, which would allow intra-communication, to achieve a smart home system. My experience with the Raspberry Pi in various small projects in the past gave me the opportunity to explore and discover the vast capabilities and prospective applications of this device. This, in conjunction with its low cost, makes it ideal for "Smart Home" development projects, for personal as well as commercial use. Conclusively, the idea of a smart home system that could move from a conceptual level to a fully developed system.

1.3 Definitions, Acronyms, and Abbreviations

Acronyms:

1) IoT: Internet of Things
2) SQL: Standard Query Language
3) VCC: Voltage Common Collector
4) GPIO: General Purpose Input/Output
5) USB: Universal Serial Bus
6) IP: Internet Protocol
7) LTE: Long-Term Evolution
8) GSM: Global System for Mobile Communication
9) LAN: Local Area Network

10) WAN: Wide Area Network
11) URL: Uniform Resource Locator
12) DNS: Domain Name System
13) PIR: Passive Infrared
14) RTSP: Real-Time Streaming Protocol
15) HACS: Home Assistant Community Store
16) UI: User Interface
17) DHCP: Dynamic Host Configuration Protocol
18) SSID: Service Set Identifier
19) YAML: YAML Ain't Markup Language
20) OS: Operating System
21) AI: Artificial Intelligence

Definitions:

1) Zigbee is an IEEE 802.15.4-based specification for a suite of high-level communication protocols used to create personal area networks
2) DHCP is a network management protocol used on IP networks for IP address assignment
3) FFMPEG: command line toolbox to manipulate, convert and stream multimedia content

1.4 Structure of Dissertation

The rest of this dissertation is organized and follows the structure explained below:

Chapter 2:

This chapter revolves around the background research that was conducted prior to the development of the dissertation, as well as historical and future importance of major aspects of this field. It includes details about IoT and smart homes, their dependence and coexistence in the development of a modern smart home hub as well as their importance

in several aspects of our lives. Chapter 2 also includes some of the bibliography and/or forum posts that were helpful in the development of this dissertation. These consist of information gathering and analysis of similar projects.

Chapter 3:

This chapter is related to the methodology followed. It presents information about both hardware and software, and their importance in this project. It emphasizes on the appealing features of the choices in these two areas. The conclusion towards these choices was a result of the research conducted, analyzed in **Chapter 2**, but during the development phase and the process of problem solving, adjustments were made.

Chapter 4:

Chapter 4 features details regarding the actual development of the dissertation project, such as certain guidelines that were followed, the approach towards the implementation of each of the features in this "Smart Home" project, and the implementation itself. The final product is displayed, in order to allow the user to comprehend the interaction process with the system as well as its capabilities.

Chapter 5:

Chapter 5 focuses on problems faced during the development phase and explains in detail solutions. It aims to serve as a collection of information for future guidance in similar projects, in order to avoid the occurrence of the same problems as well as offer explanation on how to resolve them.

Chapter 6:

The main focus of chapter 6 is the evaluation of the project after its completion. This was divided in two methods, through personal analysis which involved monitoring the system under extensive usage to ensure that it behaved as intended and based on feedback from third party members who interacted with the final system.

Chapter 7:

This chapter elaborates the conclusions from this dissertation and analyzes the expandability options of this project, with the inclusion of some of the future plans. It aims to present ideas for improving or expanding this dissertation project.

Chapter 2

2. Background

2.1 Internet of Things

The concept of smart devices, and consequently their interaction/communication in a network, arose through a discussion regarding a vending machine and its ability to be an internet-connected device. This has allowed the machine to report its inventory stock and provide information regarding the temperature of its refreshments [1]. Even though the inception of the idea of smart devices was introduced in the 1980's, which precedes the

Internet era, the core objective of such devices has remained essentially the same. The main goal is to minimize the effort required towards achieving a specific goal, and optimizing it by implementing automations, whilst in the meantime rendering the need for human involvement to a bare minimum. This can be as simple as tracking inventory stock in a vending machine, or as complex as a weather reporting system.

The main, and quite substantial difference between the time of the initial introduction of smart devices, back in early 1980's and the time that we live in now, is that due to the advancement of the Internet and its commercialization, most of the devices that we interact with on a daily basis have the ability to connect to the Internet. This enables for communication between those devices, as well as data collection, which in turn can be used to achieve a specific goal. This has enabled us to create more complex and personalized systems. The benefits of usage of IoT and smart devices, are not limited to just improving our lives by minimizing human involvement in the completion of one task. The data collected from such devices can be used to achieve a more environmentally friendly solution, to an existing process and even achieve better resource management by improving their utilization. Applications of IoT can range, and the number of Internet enabled devices is growing at a high rate. This allows more applications to arise as well as improve the already existing ones, such as Smart Homes, Smart Cities, Agricultural applications, medicinal applications, and many more.

2.2 Smart Cities

One of the most useful applications of smart devices and IoT, is the implementation of such devices to achieve a Smart City. The IoT devices are used to collect data and with proper data manipulation and analysis we can extract information that will be used to achieve a more environmentally friendly and efficient city. There are a great variety of application areas such as mobility, healthcare, water, and many more. Dubai is a great example that is in the process of approaching the idea of a smart city. They have plans to digitalize all the government services and improve transportation systems by implementing artificial intelligence. The possibilities of the applications and benefits from Smart Cities are vast.

The rising trend of IoT devices and their inclusions in our everyday lives will boost greatly the idea of Smart Cities around the globe.

2.3 Smart Home

The continuous economic growth urges residents to pursue a higher quality of life. The idea of a smart home aims to satisfy some of our needs and expectations. The abundance of network appliances and devices, both in our daily lives and domestic environment, has emphasized the need for the development of a customized software to control these devices. In addition, the benefits of implementing a smart home are evident in many areas. These include the provision of a more environmentally friendly solution to the currently employed methods for achieving certain tasks and the release of stress and minimization of efforts exerted by humans for completing these tasks. By analyzing the data collected from many of the smart appliances, we can determine when there is the need for usage of certain devices and utilize them only when required. An example would be the heater of the water tank being enabled at a specific time of the day, based on the user's schedule and only if the water temperature is below the user's preferred value. This ensures full utilization of the device and maximizes the user experience since there is no requirement for exertion of manual effort by the user in order to turn on the heater and check the water temperature. There is a vast number of automations that can be implemented that would fit the needs of a specific user, and this is limited only by the hardware available and the effort exerted for achieving a smart home system that revolves around the user's demands. However, assuming that these are limitless, a fully personalized system can be built that will satisfy the user and maximize the utilization of the devices in the network.

2.4 Home Automation Open-Source Systems

In order to conclude on the choice of open-source home automation software that was used in this project, it was important to research the numerous options in the market that are compatible with a Raspberry Pi device. Some of the main candidates that were researched prior to the implementation phase, are covered below. The examples of open-

source systems that were further explored are the ones that were deemed ideal for the development of this project, based on the available information and personal opinion. These include:

- Home Assistant
- openHAB
- ioBroker
- Jeedom
- Domoticz
- OpenMotics
- MisterHouse

2.4.1 Main Candidates

Some of the main candidates are explained and analyzed in more detail below. Their benefits and drawbacks that were apparent through the research are mentioned and these were taken into account for the decision making in terms of the system that was used for the development of this smart home project.

2.4.1.1 Home Assistant

Home Assistant is an opensource Home Automation software which can be utilized as a fully localized system, build around the user needs and that enables them to achieve a personalized Home Automation System. It can be run and setup using a Raspberry Pi device, which essentially enables the user to achieve their goal with relatively inexpensive hardware, that is also readily available. Its installation is simple and the guide available at the HomeAssistant webpage [2] explains the requirements as well as steps that need to be followed, in order to complete the initial installation on advised hardware. Since HomeAssistant is an opensource project, the code is available to be viewed on GitHub, and any potential user can get familiar with the history of the code or research code snippets. This project is backed by an active community which aims to improve the software both in terms of its core functionality but also by increasing support and integration of third-party

software. Hence, the usage of HomeAssistant has been proven beneficial in two major ways. The active community page [3] has been a good source of information, since it allows potential users to view experiences of others with specific use case scenarios, as well as examine potential alternatives to their proposed solutions or implementations. Furthermore, the HomeAssistant page provides documentation for various integrations as well as automation guidelines that provide the necessary knowledge for even an inexperienced user to embark on their smart home project.

2.4.1.2 **openHAB**

open **H**ome **A**utomation **B**us (openHAB) is an open-source home automation software that is coded in JAVA. It aims to provide control over devices from different vendors and provide a **U**ser **I**nterface (UI), which will be best suited for each individual. The device that can act as a central hub and that openHAB will be installed on is a Raspberry Pi. Guidelines for the installation process and initial setup can be found on their website [4], as well as a beginner's tutorial section that is useful for familiarization. OpenHAB receives regular updates that improve the system and expand on its capabilities and features and has an active community which can be helpful for any individual regardless their level of experience. There is a supporting community and an active community page [5]. It also provides different UI options that will best fit the user needs and experience level. In each of the UI options some elements are features are restricted in order to provide a more personalized experience. These include:

- Paper UI
- HABmin
- Basic UI
- HABPanel
- Classic UI

Another great benefit of this specific software is that it provides native apps for Android and iOS, in order to be usable with a smartphone, as well as native app for Windows 10.

With research of the documentation provided as well as help from the community page the necessary knowledge can be provided and act as building grounds for developing a smart home system.

2.4.1.3 **Jeedom**

Jeedom is another open-source software that allows the user to create a personalized and autonomous home automation system without the need of external servers. It compatible with many protocols such as Z-wave, EnOcean, RTS and many more. It can provide the means and allow the user to expand on their idea of their smart home project and consequently build a finalized system with the usage of a Raspberry Pi or any other Linux based system. It allows access to the UI with a native application on both major platforms for the smartphone industry, Android and iOS. Jeedom is a France based company and even though its website as well as community guidelines provide an English translation, as well as documentation in many other languages, unfortunately the Jeedom community is predominantly French.

2.4.2 Choice of Home Automation Open-Source Systems

There is vast number of options available in the market, for home automation open-source software that can be installed and run on a Raspberry. This list is not indicative for the best use scenario of each of the systems, since personal experience with each of the choices is lacking. However, through personal research, it was concluded that the system of choice would be Home Assistant since it has a more substantial and active community, which can be ideal especially for early stages of the development of a smart home project. Furthermore, it offers the ability to develop a more personalized and user-friendly UI with the usage of its lovelace UI, which allows an inexperienced user to create their dashboard with relative ease, and additionally enables an experienced user to take advantage of the YAML configuration file for the dashboard and customize it fully. These are subject to change due to the rapid emergence of more open-source systems as well as the development and expansion of the current ones.

2.5 IoT Devices

An IoT device is any type of hardware that is designed for a specific task, and through its internet connection capabilities, it can transmit data over the internet or different networks [6]. The uprising trend of IoT devices has materialized the need for other fields, such as **A**rtificial **I**ntelligence (AI) and Machine Learning to be integrated with such devices to achieve a more "intelligent" and autonomous processing. Their applications vary from "simple" home automation, to sophisticated medical equipment. One of their main appealing features is their small size, low cost and low power consumption. This field is an emerging field of study, and since it revolves around data transmission and manipulation it is interconnected and inter-dependent with other fields such as the cloud, data security and privacy. There are many concerns regarding IoT devices and their extensive usage on greater network systems such as "Smart Cities", that are constantly studied, in order to optimize many of current processes. Many technologies were founded or expanded upon, with the aim to meet the needs of such large-scale data collection and analysis systems as well as counter and solve these concerns. Some of the technologies involved are Low-Power Wide-Area Networks (LPWANs), wireless broadband communication (GSM, LTE, 5G) and Wi-Fi and Zigbee for LAN/WAN intra-network data transmissions. This dissertation mainly relies on Wi-Fi based technologies, and the collection and analysis of data from devices on the same network. However, some automations, such as the geo-location tracking, require LTE technology in order to exchange data between the mobile device of the user and the instance of HomeAssistant (the Raspberry Pi).

2.5.1 Sensors

Collection of data is a huge aspect of designing a smart home. Sensors are usually used to collect data, and the data collected can be used in various ways. Data could act as a trigger, which will enable or disable certain automation. An example of this can be the use of a motion sensor. The detection of motion can trigger an automation, such as turning on the lights. Another, great use of data collection can be pure visualization of a set of data, which will be used to inform the user on the current state of their home. For instance, in this

dissertation project, data of temperature and humidity is collected and is then represented to the user in a graphical way so that they can visualize it properly and be informed of said attributes. Furthermore, this data can also be analyzed and manipulated, to fit the user needs or even provide new relevant information. The data of the relative humidity and the temperature can be used in an algorithm in order to provide the heat index. The importance of heat index is further analyzed and explained on **Section 2.6 Heat Index**.

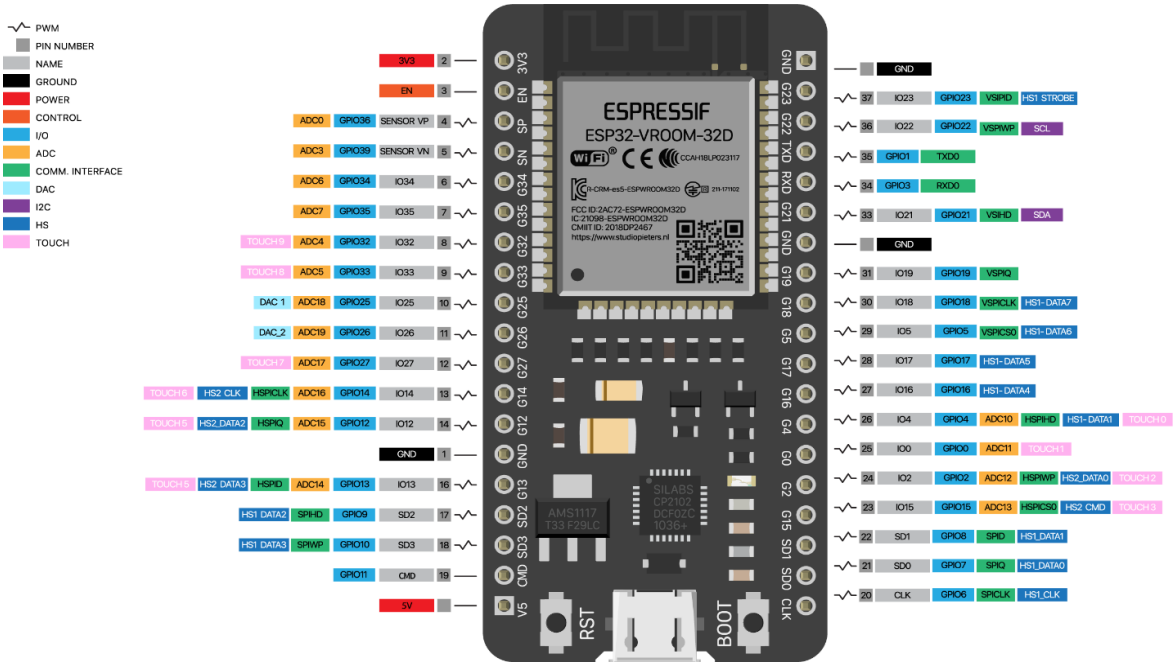
It is also important to note, that the term "sensors" does not necessarily mean that it is an object constrained by the physical world (such as the temperature sensor). A great example of a different kind of sensory equipment can be the collection of data from a server. In this dissertation project the Raspberry Pi, which acts as the hub will collect information from the Plex Server (a media streaming service setup on the Personal Computer), and when a specific device starts streaming videos, a corresponding automation will be triggered. Another Raspberry Pi is setup as a plex client (a device which will stream video/data from the server) and when a video playback is initialized, the lighting in the room is dimmed. When the video streaming is paused, the lights return to their original state. Sensors have a huge role during the setup of a "Smart Home" both towards achieving a system with great usability, as well as being able to setup the necessary or desired automations. In addition, their usage can help achieve a more optimized and energy efficient house since many of the electronic devices will only be used when needed.

2.5.2 ESP32 and ESP8266

The ESP32 and ESP8266 are Wi-Fi development boards from the company Espressif. ESP32 is the model that was chosen to be used in this dissertation project, since it has a higher CPU speed than ESP8266 and it also has Bluetooth capabilities. Both boards rely on the GPIO pins in order to connect the sensory equipment to them, and thus another great advantage that is observed on the ESP32 device is the fact that it has more than double the number of pins (36 pins on ESP32 and 17 pins on ESP8266) [7]. This allows for more sensors to be connected on the same ESP module at once, if that is required. An example where this might be advantageous is when we need to use multiple sensors in the same room/area

(such as the combination of a motion sensor and a temperature sensor). However, the ESP8266 is cheaper than the ESP32, therefore by considering the needs and budget for a project we can decide which would be more suitable.

The ESP32 module needs relatively low power to function and with the utilization of its Deep Sleep Operating capabilities it is ideal for even portable IoT devices and projects. It can be reprogrammed to be used by different sensors or even multiple sensors at a time, therefore it can be a valid candidate for prototyping and testing different hardware. Its relatively low cost makes it ideal for “Smart Home” projects and people that aim to experiment with similar technologies. It can even be deemed ideal for experienced users or more advanced projects.



Source: https://raw.githubusercontent.com/AchimPieters/esp32-homekit-camera/master/Images/ESP32-38PIN-DEVBOARD.png

Figure 2.1: ESP32 Module

2.6 Heat Index

Aside from the actual temperature of the room that can be recorded and presented to the user, it is important to inform them on the “feel like” temperature. This is the temperature

that the human body will feel, and it depends on the actual temperature as well as the relative humidity of the area. Regulation of body temperature relies on perspiration, which the bodily function also known as sweating. Sweating is used to decrease our body temperature when needed. However, if the surrounding air is humid, sweat will take more time to evaporate from the skin and as a result our body temperature is not properly regulated. This will in turn affect the body's perception of the temperature of the room.

Relative Humidity %	Air temperature °C										
	21	24	27	29	32	35	38	41	43	46	49
0	18	21	23	26	28	31	33	35	37	39	42
10	18	21	24	27	29	32	35	38	41	44	47
20	19	22	25	28	31	34	37	41	44	49	54
30	19	23	26	29	32	36	40	45	51	57	64
40	20	23	26	30	34	38	43	51	58	66	
50	21	24	27	31	36	42	49	57	66		
60	21	24	28	32	38	46	56	65			
70	21	25	29	34	41	51	62				
80	22	26	30	36	45	58					
90	22	26	31	39	50						
100	22	27	33	42							

Serious risk to health - heatstroke imminent

Prolonged exposure and activity could lead to heatstroke

Prolonged exposure and activity may lead to fatigue

Source:
https://www.researchgate.net/publication/325471848/figure/fig2/AS:632342498537474@1527773608374/Apparent-temperature-heat-index-in-degrees-Celsius-according-to-air-temperature-and.png

Figure 2.2: Heat Index

2.6.1 Algorithm for Heat Index Calculation

The algorithm for finding the heat index [8] follows these steps:

- Initially we need to convert the temperature from our sensor to the imperial system (if originally in metric)

  T(°F) = T(°C) × 1.8 + 32

- According to the regression equation of Rothfusz the Heat Index (HI) is:

$$HI = -42.379 + 2.04901523 * T + 10.14333127 * RH - .22475541 * T * RH - .00683783 * T * T - .05481717 * RH * RH + .00122874 * T * T * RH + .00085282 * T * RH * RH - .00000199 * T * T * RH * RH$$

  Where Relative Humidity in percentage (%) is RH and T is the current temperature in degrees Fahrenheit.

- Then we have 3 scenarios:

  1. if the relative humidity (RH) is less than 13% and the temperature is between 80 and 112 degrees Fahrenheit we perform the following adjustment:

$$ADJUSTMENT = [(13 - RH)/4] * SQRT[17 - ABS(T - 95.)]/17$$

     Where ABS in the absolute value and SQRT is the square root function.

     HI will be:

$$HI = HI - ADJUSTMENT$$

  2. If the relative humidity is above 85% and the temperature is between 80 and 97 degrees Fahrenheit, the following adjustment is performed

$$ADJUSTMENT = [(RH - 85)/10] * [(87 - T)/5]$$

     and HI will be:

$$HI = HI + ADJUSTMENT$$

3. If the temperature is below 80 degrees Fahrenheit then the Rothfusz

$$HI = 0.5 * [T + 61.0 + [(T - 68.0) * 1.2] + (RH * 0.094)]$$

regression algorithm is deemed not appropriate. Steadman's results will be used:

- Finally, we need to convert the Heat Index (HI) back to the user's original unit system. If the imperial system was used the result is displayed as is. If the metric system was originally used, then we convert back to degrees Celsius with the following formula:

$$HI_C = (HI - 32)/1.8$$

Where HI_C is the heat index values in degrees Celsius and HI is the result received earlier for the heat index in degrees Fahrenheit.

Chapter 3

3. Methodology

3.1 Hardware

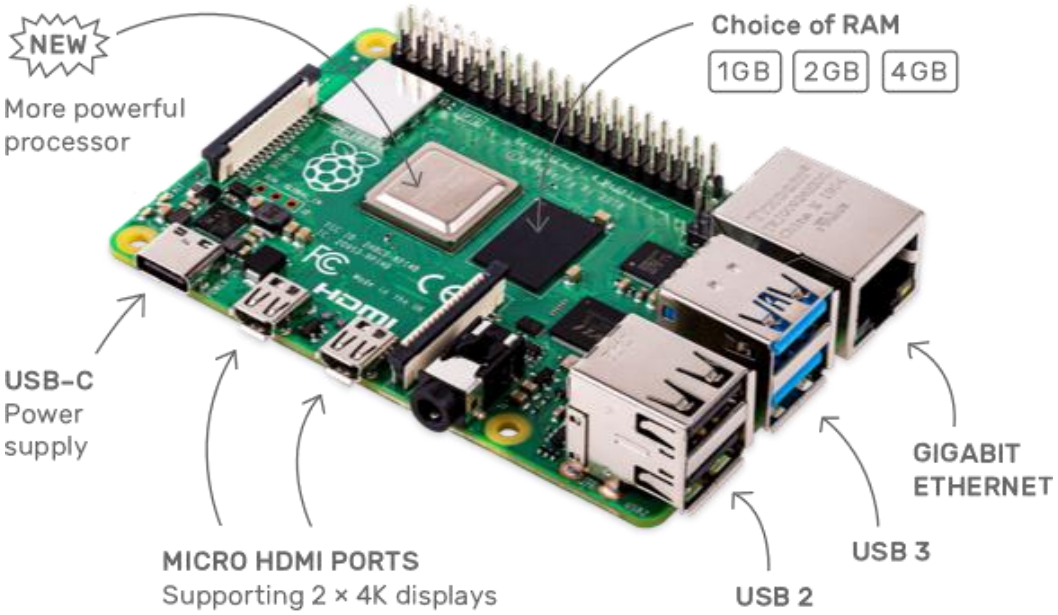
To implement this dissertation project and be able to have a functional smart home system, which would perform specific operations and automations, some hardware components were necessary. Through the research conducted, further explained, and outlined in the **Chapter 2: Background**, a list of essential components was composed whilst managing to keep the expenses to a minimum. These hardware choices are easily accessible in the

market, and in combination with the exceptionally large community supporting the Raspberry Pi, made these ideal for a smart home project. Furthermore, the vast number of forums surrounding these hardware choices has provided more information on the field of smart home projects and automations. After the extensive research was performed, it was concluded that the necessary products required during the development stage are the ones explained in more detail below.

3.1.1 Raspberry Pi

The Raspberry Pi (Figure 3.1) has been introduced in February of 2012 and has seen a series of iterations, improving its hardware capabilities with each successor model. Aside from the main series, there is the raspberry pi zero series which focuses on less demanding tasks. This device was developed by the Raspberry Pi Foundation to promote teaching of computer science studies, especially in developing countries. The appealing price has allowed it to be used for the purpose intended but it has also encouraged a community of individuals interested in applying such portable hardware to various small projects. Hence, these single-board computers have sparked the interest of many people and are constantly been tested to their limits by the active community backing it. Its operating system is arguably one of the main reasons of its expanding nature and wide acceptance, since it is Linux based (Raspbian). In addition to this, its ARM based architecture allows for low-level programming (Assembly) with an architecture that is used in smart phone devices.

In the field of smart home projects, and especially data collection from sensors, the Raspberry Pi will act as a central hub, collecting all the information from various of sensors and storing them locally. This allows for ease of access to the data collected as well as a form of security.



Source: https://www.raspberrypi.org/homepage-9df4b/static/pi4-labelled@2x-0894491e6de97a282dde5a5010cc8b61.png

Figure 3.1: Raspberry Pi 4 Model B

3.1.2 Philips Hue Starter kit E27 (White)

Philips is a well-known brand and one of the pioneers in the smart lighting department. Their Philips Hue line of LED bulbs has an impressive life span but what makes it a good applicant for a smart home system is the fact that it has a lot of third-party software support, hence its integration with other software used will be more seamless. In addition to this, the starter kit includes two white smart lamps, which is sufficient for the needs of testing and development, and the included Hue bridge guarantees an easier integration of more of hue products in the future with a relatively low programming cost.

Source: https://images-na.ssl-images-amazon.com/images/I/71rH6GRE-1L._AC_SL1500_.jpg

Figure 3.2: Philips Hue Starter Kit

3.1.3 Elegoo Electronic Kit

An electronic kit is essential in two main stages of the development. Firstly, the usage of sensors as well as breadboards was an unexplored area for myself, hence this kit will be proven useful in the experimentation stage, so that I can get a chance to familiarize myself with this area and be more knowledgeable and confident prior to moving onto later stages of this dissertation development. Also, it was dominant in the development stage, where sensors had to be connected to the Raspberry Pi. Elegoo is one of the known brands in this area, hence forum posts for help as well as their own documentation is be a great source of information regarding electronic kits and sensors' applications.

More information regarding the kit as well as the manuals can be found on their website [9].

Source: https://www.elegoo.com/wp-content/uploads/2017/01/E2-1.jpg

Figure 3.3: Elegoo Starter kit

3.1.4 Sensors and Resistor kit

For the needs of this dissertation and project outline explained previously, a list of specific sensors was determined to be necessary. Through research it was decided that the best choice of sensory equipment was some widely used sensors for each of the specific tasks that are to be developed and implemented. This means that an abundance of information will be available through forum posts as well as manuals provided by the manufacturers. For the motion sensors that will be implemented to areas of the house, in order to track motion and essentially determine whether the room is occupied, I have chosen the HC-SR501 sensors. The DHT-11 is the sensor of choice for tracking the temperature and the humidity levels of the room that it will be installed to. The information collected by the sensor will then be processed by the Raspberry Pi and saved locally. On a later stage the data will be presented in a visually comprehensive way on the UI that the client will use to control IoT device and view useful data and information. For recording and tracking the

temperature of the water (water tank) the DS18B20 is an ideal candidate due to its waterproof capabilities but also due to its widespread use for this type of applications. This means that information will be more readily available, regarding any issues that might occur or even guidelines for its usage. Similarly, to the DHT-11, information from the DS18B20 will be collected and used for a visual representation as well as presenting live information regarding the water temperature. A resistor kit was also purchased for usage in both the experimentation phase and familiarization with sensor connections and breadboards, as well as at later stages in the development phase.

3.1.5 ESP32 devices

ESP32 is a series of low-power system on a chip microcontrollers that were utilized to connect the sensors to them through the GPIO pins on the board. Since the ESP modules provided integrated Wi-Fi and dual-mode Bluetooth, data could be sent to the hub (Raspberry Pi) without the need of the sensor to be physically connected to it. The Wi-Fi capabilities were used for the purposes of this project. This allowed communication with the Raspberry Pi which has Home Assistant installed on it.

Source: https://images-na.ssl-images-amazon.com/images/I/51Dzi2l4azL._AC_.jpg

Figure 3.4: ESP32 device

3.1.6 Digoo BB M1X

To provide security and monitoring abilities for the smart home developed, an IP camera was used. This allowed for a live camera feed to be streamed to the user, through the UI developed, as well as provide the ability to rotate the camera and monitor a wider area. The camera that was used was a Digoo BB M1X since it is available at an affordable price whilst covering the needs for this project. It supports an FFmpeg stream using RTSP, as well as rotation abilities in all directions (up, down, left, and right). More details regarding the drawbacks of this selection as well as guidance for a better alternative (according to the budget available) are provided in **Chapter 5: Experience and Problem Solving.**

Source: https://img.mydigoo.com/mydigoo/images/FB/0B/670c1159-2bf3-4a0f-a1d2-ba1f6fb476ff.jpg

Figure 3.5: Diggo BB M1X

3.2 Software

Aside from the choice in hardware, another important aspect of the development of a smart home system, is the choice in software, such as existing platforms, Operating Systems, and programming languages of choice. These decisions are very important for two

major reasons. Firstly, the final product should be user-friendly whilst being able to support the features of the desired smart home system, with minimal to no effort exerted by the user. Secondly, the choice in software is critical in terms of expandability of this project. Future efforts to expand this smart home system, by myself or other interested parties, should be straight forward and aid the process.

3.2.1 HomeAssistant

HomeAssistant is an open-source home automation OS, for the Raspberry Pi, primarily focused on control and privacy. It has a huge team supporting its development, since it is open source, and also many third-party software have smooth integration. Third-party software will be deemed useful, both in terms of integration and control, in the development of features in the smart home project. One of the main third-party software used is ESPHome, which is explained in further detail below. The Raspberry Pi which will be setup with HomeAssistant and will act as an HBMQTT broker. The publisher will be YAML scripts which will be used to alter the behavior of other devices based on the output of the sensors (subscribers).

Source: https://2.bp.blogspot.com/-4jo-zhdUzvQ/WBq17bboVsI/AAAAAAAAm-k/hVIq6fONlfcxFee8zM6A0HnaGh9kuGiYgCPcB/s640/mqtt_diagram.PNG

Figure 3.6: HBMQTT setup

3.2.2 ESPHome

ESPHome can be integrated into HomeAssistant, and it serves as a system to control ESP modules, such as ESP32 and ESP8266. Through the third part integration, you can inject and

run code that will be send onto the ESP device. The initial setup requires connection to the ESP module over USB and then once the code is injected with the necessary Wi-Fi credentials in the configuration, the module will receive an IP address and connect to the network. You can take the necessary measures to ensure static IP allocation, to avoid future reference problems to the device, in case the IP changes. This can be achieved through configuration setting in the ESPHome system, or by altering the settings of the router on the personal network. Future configuration changes and addressing of the ESP module, no longer requires a USB connection. The HomeAssistant will collect data from the sensors connected to the ESP module, assuming that the code injected is correct and the right IP is addressed. ESPHome, in conjunction with ESP32 modules enabled the conversion of a sensor with no Internet connectivity capabilities to an IoT device.

3.2.3 OwnTracks

OwnTracks is a mobile app (available both for Android and IOS), which tracks the current location of the user, based on the geolocation of their smartphone device (with OwnTracks installed on it). Even though for the purposes of this thesis project the location tracking was achieved by using the HomeAssistant application and a smartphone of choice, I concluded that the usage of this specific application will be a good alternative since it is available for both major platforms in the smartphone industry (iOS and Android), hence no limitations on the user. Another important reason for this choice as a good alternative is its integration with HomeAssistant. Using webhooks (**A**pplication **P**rogramming **I**nterface (API) used to implement event reaction methodology), installed on HomeAssistant, the broker (in this case the Raspberry PI) can be informed of the user's specific geolocation. Hence, this can be setup to detect key locations in the routes used more frequently by the user on their way home or identify when the user enters or leaves a predetermined zone, in order to trigger an event when then leave the house or prior to their arrival.

More information regarding its integration can be found in the official Home Assistant website [10].

3.2.4 YAML

YAML is a human-readable data serialization standard. Its most common usage is to write configuration files, and this is precisely its application in the Home Assistant system. It relies heavily on indentation for structure and its lack of the need for special characters such as escape characters, make it easier to define and comprehend. It is a data representation language which can be inferred with ease by humans, without extensive knowledge required in programming languages. Despite this it can be used to express complex configurations and does not limit the abilities of the system in any way. More information is available in the Home Assistant official website, which offers an introductory explanation on its usage in the system [11].

3.2.5 Lovelace

Lovelace is integrated into Home Assistant by the development team and it is the Home Assistant dashboard that the user will interact with. It is responsive since it can be accessed on web browsers as well as the mobile applications on the respective application stores on iOS and Android. The dashboard editor allows for users that have limited technical knowledge in programming to develop their dashboard, but a YAML code editor is also available, for experienced users to take advantage of, and allow them to customize their dashboard to the limit. It is a card-based interface, meaning that users will have card like boxes on their interface to control and access information from their smart devices. These cards can be arranged according to the user's preference to achieve a personalized dashboard which will enhance the user experience. There are 29 card types available to the user upon the initial setup, however, more can be installed from HACS, and these are created and supported by the active community.

3.2.6 HACS

HACS is the Home Assistant Community Store which allows users to install and integrate third-party addons, which would be otherwise unavailable to them. This expands on the already officially supported add-on list and allows for more personalization of the system.

However, since these add-ons are not officially supported, it would be advised to backup the system prior to any add-on installation, in an effort to avoid any unexpected errors which might render the system unusable.

Chapter 4

4. Design and Implementation

This chapter will cover the guidelines and requirements that are to be met, in order to acquire a functioning and secure system that the user will be pleased to use. It also covers

an overview of the steps that were taken to achieve the finalized system of this thesis, so that the reader will be able to follow these and alter or expand the system wherever deemed necessary or desirable.

4.1 Requirements and Guidelines

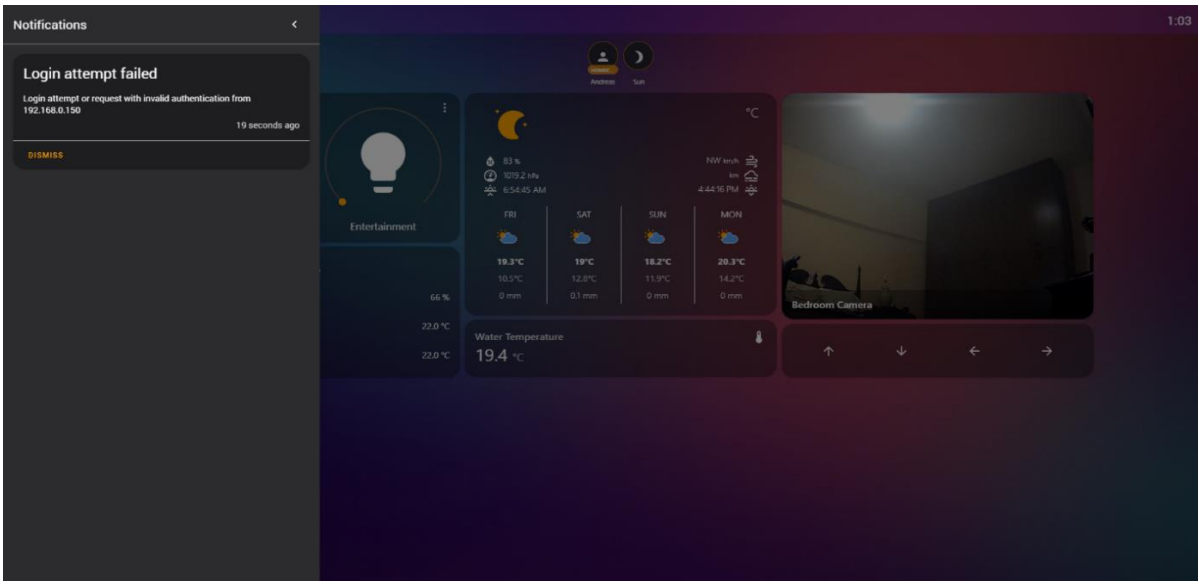
Following the decision of the open-source software, which was to be used, I had to setup some guidelines and requirements regarding the project that I would embark upon, so that a smart home system can be designed that will provide usability and offer a good user experience. The following guidelines and requirements are to be followed:

- Connection to the HomeAssistant system: the sensors and devices should be able to connect to the central hub, the Raspberry Pi. In general, there should not be limitations for this factor since any device which has internet connection capabilities can be assigned an IP in the network and should be able to be integrated into the system. Research that was done prior to the purchase of the products was conclusive towards their ability to be integrated.
- Low cost: another important factor for designing this smart home system is to keep the cost to a minimum. This is a key aspect since it is important to be able to design a system that can be achievable and affordable so that it can be replicated and expanded upon in future work.
- Flexibility in terms of hardware: the HomeAssistant system is flexible in terms of sensors of choice and the project can be altered or expanded upon by using different sensors and equipment. The Raspberry Pi that the system is installed on is readily available, however if there is need for more powerful hardware the system can be installed on a Docker environment.
- Expandability: it is a key factor for this project that it can be expanded upon. This is achieved since with the choice of hardware and software used, there is no restriction that would limit the hardware of choice for expansion of this project. The HomeAssistant system will support the majority of available sensors and

implementation of new sensors or equipment will not affect or have conflict with the current implementation. Each new equipment is considered and treated as its own specific entity which allows for direct reference to the hardware, and avoids invalid referencing issues or conflicts.

- Accessibility: the system should be accessible by the user with an interactive frontend. This is achieved through the lovelace UI which is supported by HomeAssistant. It is also important to have remote access to the system even when away from the local network. The usage of “Nabu Casa”, a cloud service by HomeAssistant, which provided remote access to the HomeAssistant system.
- Usability: the system should be a follow a design that aids the user and provides a pleasant experience during their interaction with the system. The lovelace UI which the user will use to interact with the system should be user-friendly and aim to provide a good user experience. This should improve their subjective satisfaction and prompt them to use the system in the future again. The user should be able to achieve their goals with ease and by automating certain function their need for manual work is limited. When the user wants to manually achieve a goal the system should guide them with the usage of the appropriate interface design.
- Privacy and Security: the system can be accessed by the IP address assigned to the Raspberry Pi and a specified port number. This provides an extra security measure since only people with access to the network, and consequently the network settings should be able to have this information. In addition, it is also required to provide the username and password that was assigned on the initial setup. The username and password are required only on the first time that a device tries to connect to the system, hence this does not limit the usability and at the mean time provides an extra step to ensure security. Additionally, regarding the remote access to the system, the interested party that wants to connect to the system, should have a URL generated by “Nabu Casa” the first time that remote access is enabled. The complexity of the URL renders it extremely difficult to guess it, and the requirement for authorization with a username and password is still in place. Furthermore, all

data that is exchanged between the instance of Home Assistant and the Raspberry Pi is fully encrypted. In case there is an attempt to get into the system, the user will be notified on their “Notifications” tab (Figure 4.1). They can then proceed to take the necessary measures, such as changing their passcode.



Source: Personal Screenshot

Figure 4.1: Failed login attempt notification

4.2 Implementation Guidelines

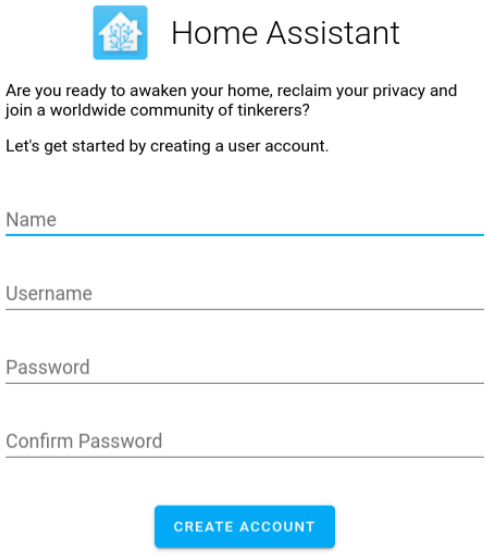
The following section will provide a general overview of the steps followed in order to achieve this finalized system. It will also explain steps followed to achieve certain features and their validation process.

4.2.1 Setup of Home Assistant

To install HomeAssistant you will need to firstly download the respective image according to the Raspberry Pi that will be used. The requirements for the installation and recommended setup are stated on the HomeAssistant getting started page [2]. For the installation, the interested party needs to have a program which will be used to flash the

image file onto the SD card. The recommended program is balenaEtcher but there are alternative options available. Following the flash of the image file, the user will need to have an internet connection provided to the Raspberry Pi. It is advised to use an ethernet connection since it is the most reliable connection, however if the user decides to use a Wi-Fi connection or a static IP address the guide on the website explains the process and provides examples of the necessary files that need to be adjusted accordingly. The user will then need to plug in the Raspberry Pi to a power supply and insert the micro sd card into the Raspberry Pi. The first boot will require some time to download the latest version of Home Assistant. The time required will depend on the user's network speeds, but in general this will take about 20 minutes. The user can then proceed to visit the URL for accessing the HomeAssistant system, which will be the IP address assigned to the Raspberry Pi with the port number 8123. The link should be as follows: http://X.X.X.X:8123 with the "X.X.X.X" replaced with the Raspberry Pi's IP address.

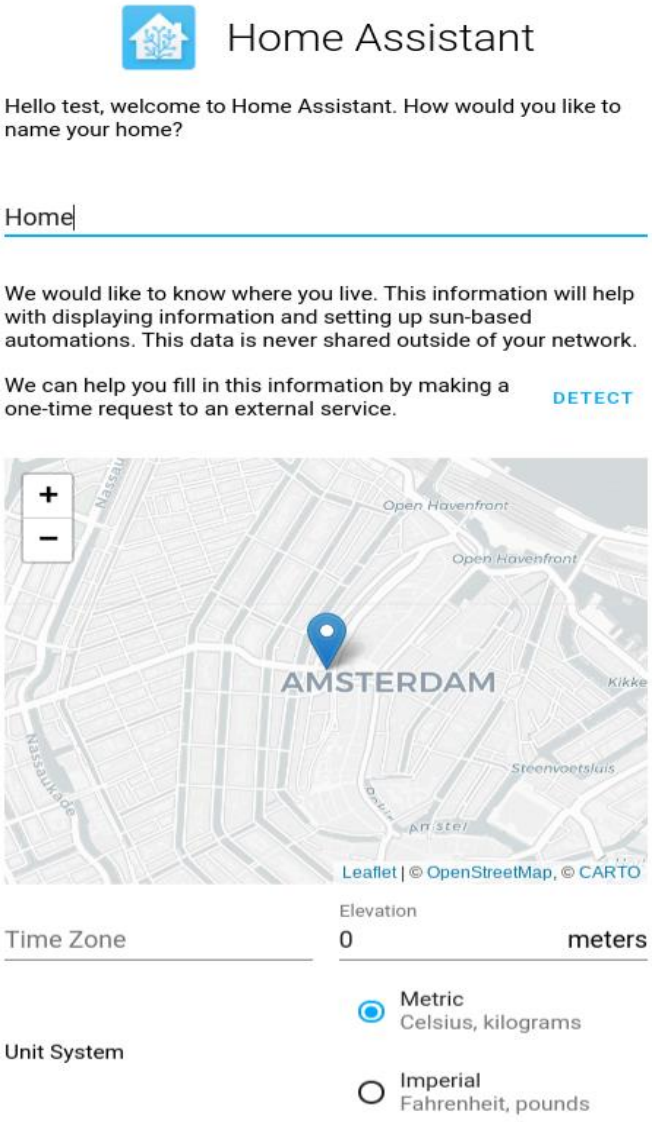
When the download is finished, the user will be greeted with the page on Figure 4.2 to create an account.



Source: https://www.home-assistant.io/images/getting-started/username.png

Figure 4.2: Create an Account

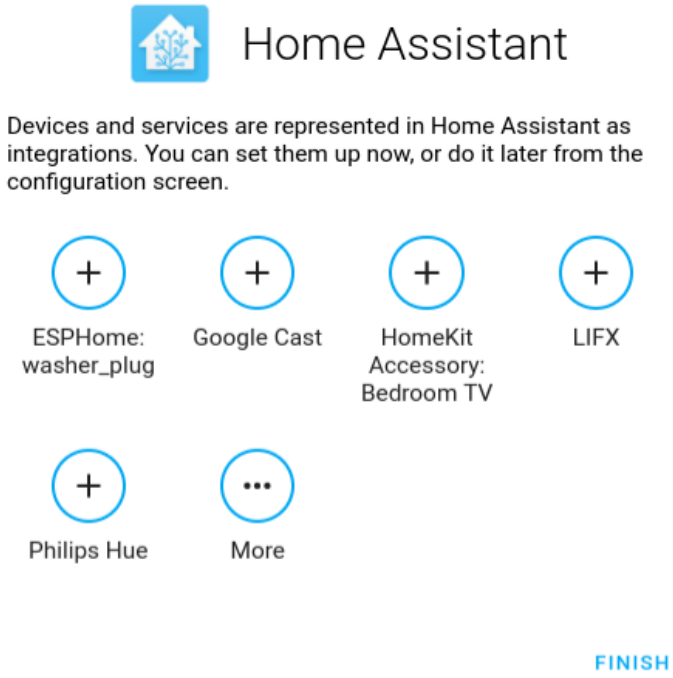
Following the user's credentials insertion and the creation of the account, the user will be asked to name their home and by making a request to detect his location, the geolocation and time zone of the user will be automatically filled. The user will also be asked to choose their unit system of choice. The webpage for these requests is presented on Figure 4.3.



Source: https://www.home-assistant.io/images/getting-started/location.png

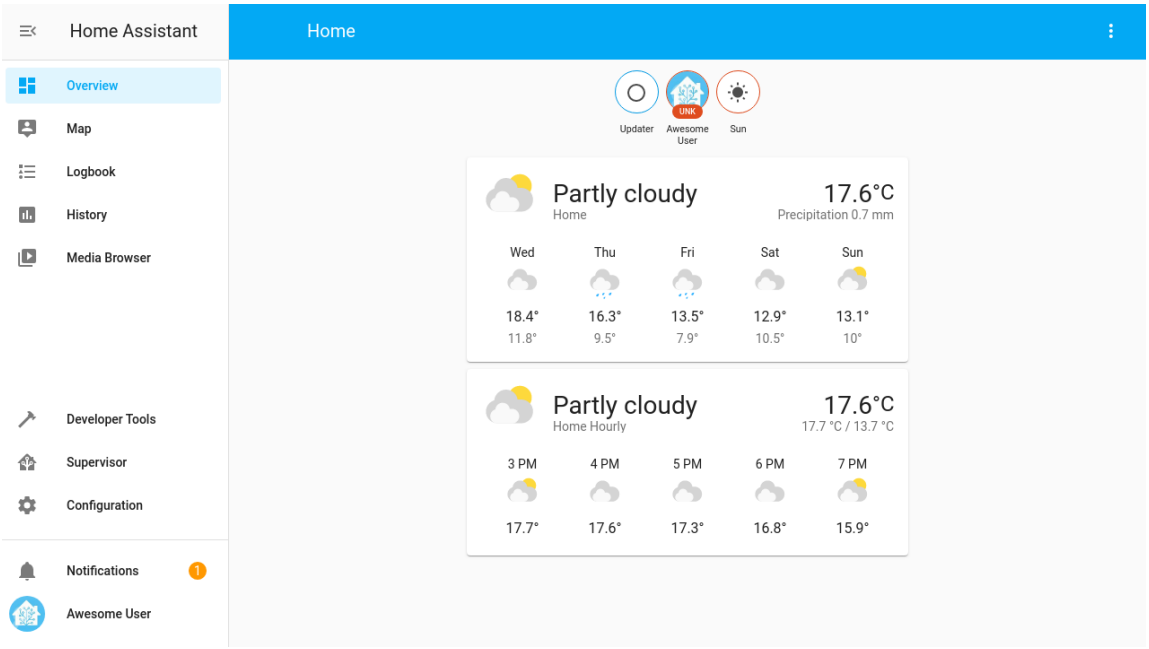
Figure 4.3: Set location, time zone and unit system

The system will then present any discovered devices in the network and prompt the user to integrate them. Devices that are not presented during this stage, can be added manually by the user at a later stage. An example of this page is represented in Figure 4.4. Once the user clicks on finish, they will be redirected to the Home Assistant web interface (Figure 4.5).



Source: https://www.home-assistant.io/images/getting-started/devices.png

Figure 4.4: Discovered devices



Source: https://www.home-assistant.io/images/getting-started/lovelace.png

Figure 4.5: Home Assistant Web Interface

The initial setup has now been completed the user can proceed to integrate their devices of choice and implement the desired automations.

For the purposes of this thesis project, the following addons were installed from the Supervisor tab:

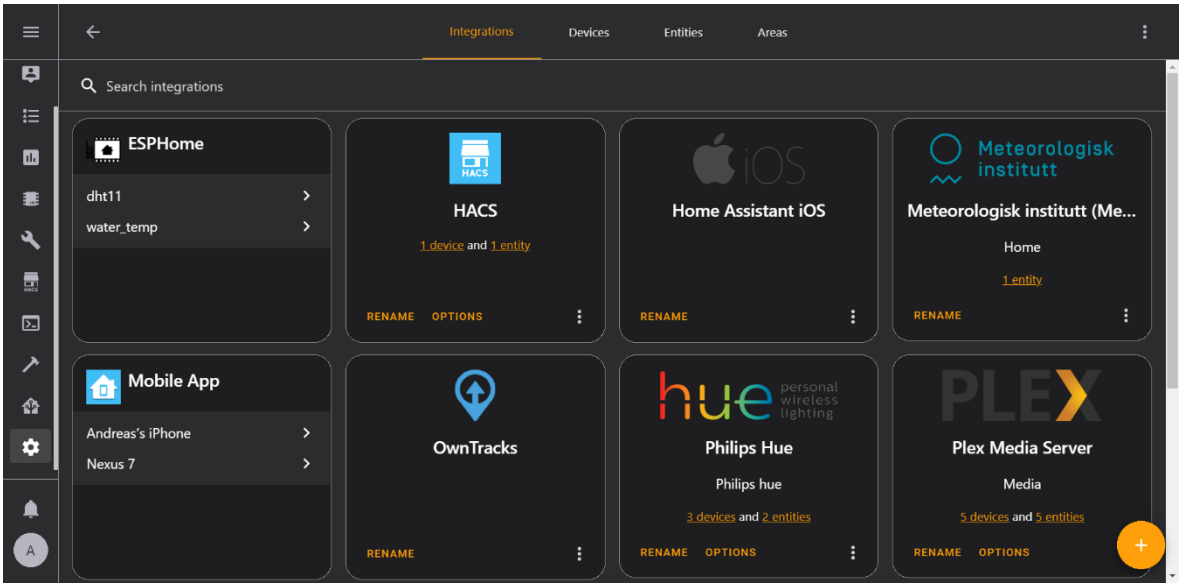
- ESPHome
- File editor
- Plex Media Server
- Samba share

HACS was also installed on the system to use third party addons that are not officially supported by the Home Assistant development team. By following the installation guide on the official HACS website [12], HACS can be integrated onto the Home Assistant system.

4.2.2 Connect devices to the network

After the initial installation of Home Assistant on our Raspberry Pi, the devices that are of interest had to be connected to the network. As explained in **Section 4.2.1** some of the devices that are connected to the local network, will be discovered during the initial setup of the system. If the devices were not discovered automatically the user can proceed to connect them manually, through the integration tab in the Home Assistant user interface.

The “Integrations” tab can be found under the “Configurations”, which is visible on the sidebar of the UI. In the “Integrations” tab you can view the already integrated devices, as well as proceed to add new ones with the “+” button in the bottom right hand-side (Figure 4.6).



Source: Personal Screenshot

Figure 4.6: Integrations Tab

For the purposes of this diploma project, the following integrations were used:

- Philips Hue light bulbs were used, therefore, to connect them to the Home Assistant system, the “Philips Hue” integration, was installed to the system.
- The “Home Assistant iOS” integration was used to connect and monitor the smartphone that was used in this project, which was an iPhone. This was needed for monitoring the phone status, if the user desires, but the main functionality that depended on it, was the geo-location tracking feature, used for the implementation of any automations that were based on the location of the user.
- “HACS” was integrated since it provided a variety of third-party software which was created by the active community, and not officially supported by Home Assistant.
- “Plex Media Server” provided access and monitoring tools for the Plex Media Server set up on the **P**ersonal **C**omputer (PC). An access token was needed for the authorization of Home Assistant and allowing access to the media server.
- “ Meteorologisk institutt” provided live information of the weather in the “home area” of the user [13]. It also informed the user on the weather forecast.
- “Speedtest” integration was used to implement the internet connection speed test functionality provided by Ookla [14], from where live information of the download and upload speeds were fetched, every thirty minutes. The update frequency was considered to be good for the use case, since it provided the necessary information at good intervals of time, without causing too much load in the network.
- “ESPHome” allowed for communication with the ESP modules, that had sensors connected to them.

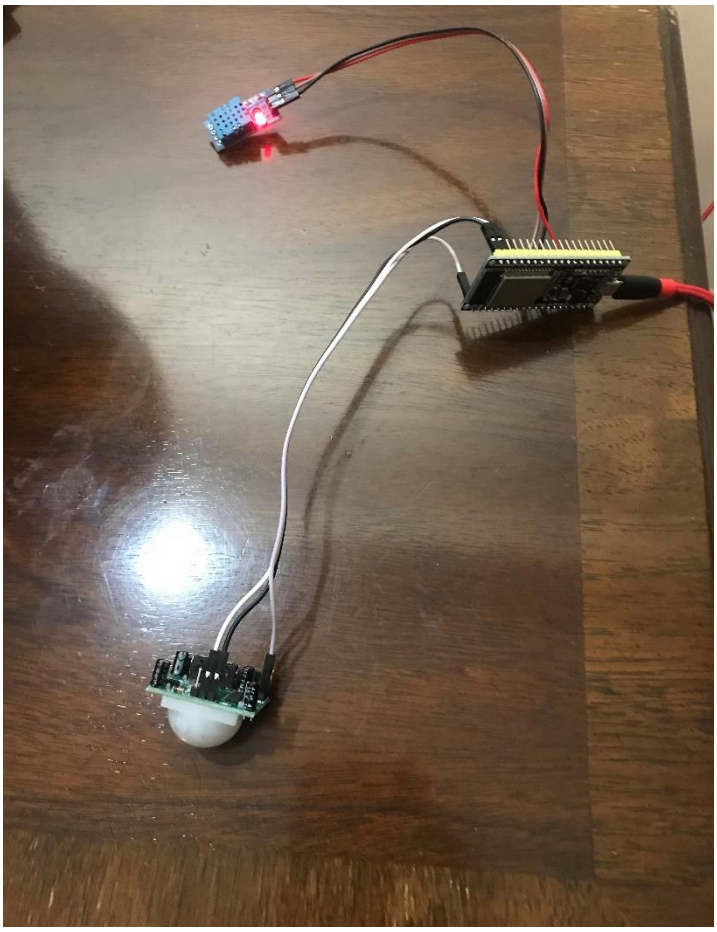
Following the installation of the integrations mentioned above, we can proceed with setting up the sensors and the automations that will used the collected data.

4.2.3 Connect sensors to the network

The sensors that were used had to be connected to the ESP modules. There were two ESP modules used for this project. However, there is no limitation regarding the number of ESP

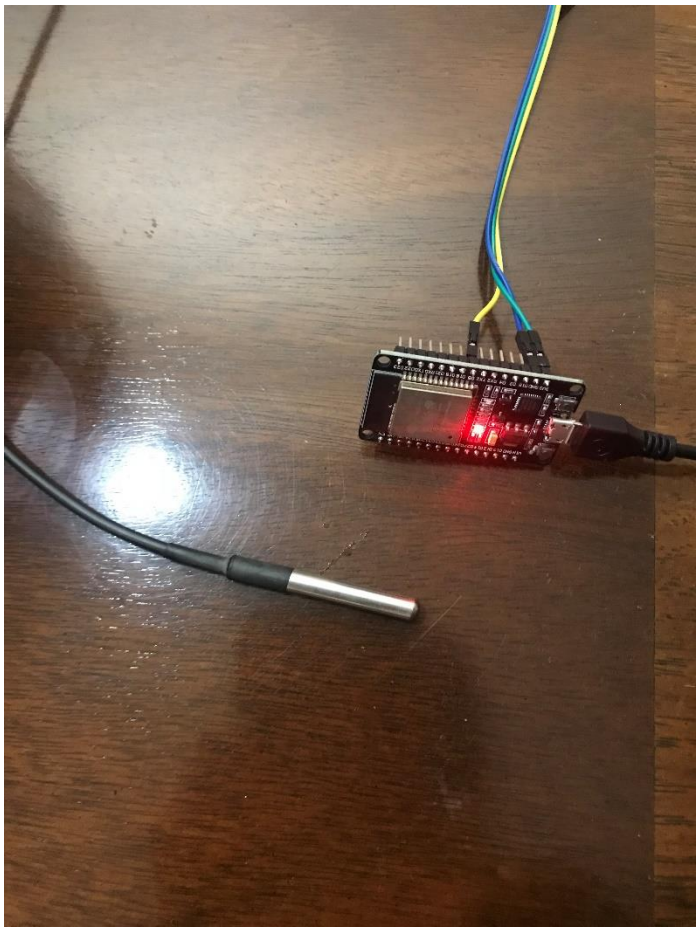
modules that can be used for such a project if there is a need for more sensors to be integrated. The ESP modules had the following sensors connected to them:

- 1st ESP32 module (Figure 4.7):
    - DHT11 temperature and humidity sensor.
    - HC-SR501 PIR motion sensor
- 2nd ESP module (Figure 4.8):
    - DS18B20 water temperature sensor



Source: Personal Screenshot

Figure 4.7: ESP module with DHT11 and HC-SR501



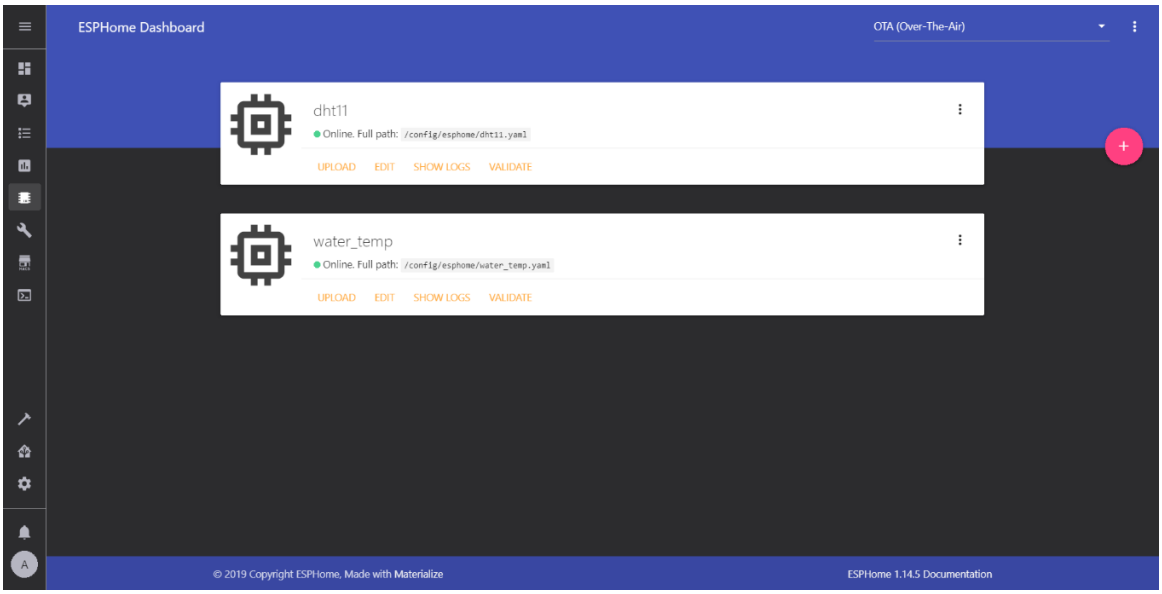
Source: Personal Screenshot

Figure 4.8: ESP32 with water temperature sensor

The 1st ESP32 module had two sensors connected since the two sensory equipment were both placed in the bedroom area of the house, and there was no need to separate the two sensors with the usage of a second ESP module. The two sensors were both connected with 3 pins (voltage, ground, and data) and the sensors were placed with a considerable distance apart to avoid interference.

The 2nd ESP32 module was placed in a different room of the house. To simulate the water temperature monitoring of a water tank for a house, a small bucket of water was used, and it was filled with hot water in order to evaluate whether the temperature values retrieved from the sensor are valid.

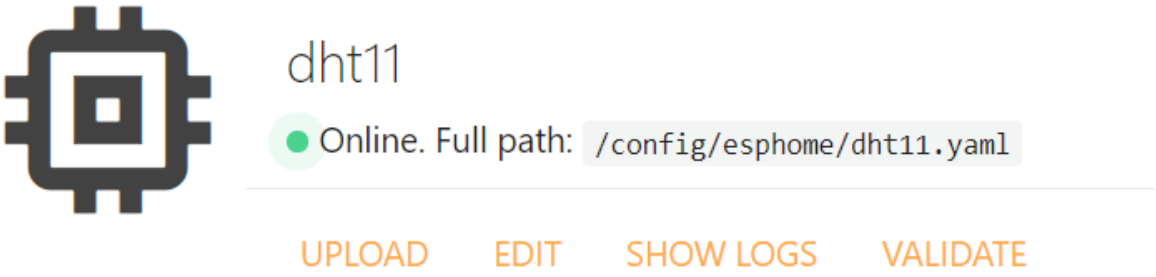
The two ESP32 devices were setup using ESPHome, and the first uploaded code was via a micro-USB cable in order to provide a physical connection to the Raspberry Pi. Each of the ESP32 modules had to have a code snippet uploaded to them that would provide the credentials of the Wi-Fi network (2.4Ghz connection only), more specifically the SSID and password of the network. This allows the ESP32 modules to be connected to the network and using DHCP network protocol, be assigned a unique IP address which they can be referenced by. After they were assigned an IP address, and appeared “online” in the ESPHome interface (Figure 4.9), their IP address were made into static IP addresses. This is important in order to avoid a change in their IP address in the future, which would terminate the communication with Home Assistant due to wrong referencing. In the case that this happens the IP for referencing the two devices can be changed in the “Integrations” tab.



Source: Personal Screenshot

Figure 4.9: ESPHome

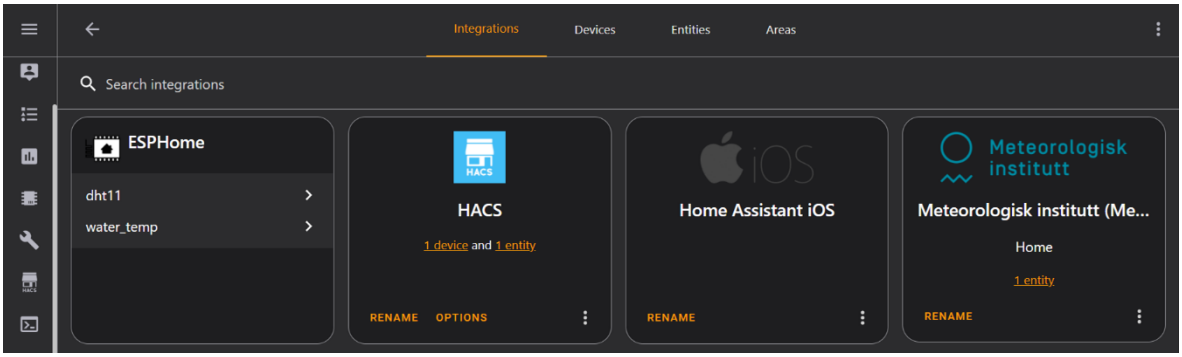
After they were online and set a static IP address, the code which is used to collect data from the sensors is also inserted in the YAML file that is used to upload code to the ESP modules. This can be accessed through the path of the file or clicking on the edit button (Figure 4.10).



Source: Personal Screenshot

Figure 4.10: ESP32 YAML file path

After the code for data collection has been successfully uploaded to the ESP modules, the sensors connected to the respective ESP32 devices were established in the system as entities. This allows for referencing them in the UI, where needed. This is achieved through the "Integrations" tab of the system (Figure 4.11), where the user will have to add the ESP devices with their respective IP address.



Source: Personal Screenshot

Figure 4.11: ESPHome Integrations

4.2.4 Camera implementation

In addition to the sensory equipment setup, an IP camera was used to stream live video feed onto the UI and provide control of it in order to monitor the area using its rotating

capabilities. The camera used was the “Digoo BB M1X” which by using RTSP the muxer sends a stream of the video feed to a server which supports it. The live video feed of the camera can be accessed and viewed using a link which the RTSP URL which the video is streamed to. The link follows this format:

rtsp://{username}:{password}@{camera’s IP}/onvif1

The username and password of the camera depend on the manufacturer and model number of the camera and the respective information will replace the “{username}” and “{password}” areas in the URL example above. Additionally, once the camera is setup and connected to the network, the “{camera’s IP}” potion of the link should be replaced by the IP address assigned to the camera when connected to the local network. By using this link, the live video feed of the camera is streamed directly to the system, using the appropriate code as explained in the documentation of Home Assistant.

For the functionality of rotating the camera, the code is used in XML format and is based heavily on the work that was already available by a member of the community of Home Assistant [15]. It was altered to fit the needs of the project and be able to move the camera in increments that would suffice the needs of the user.

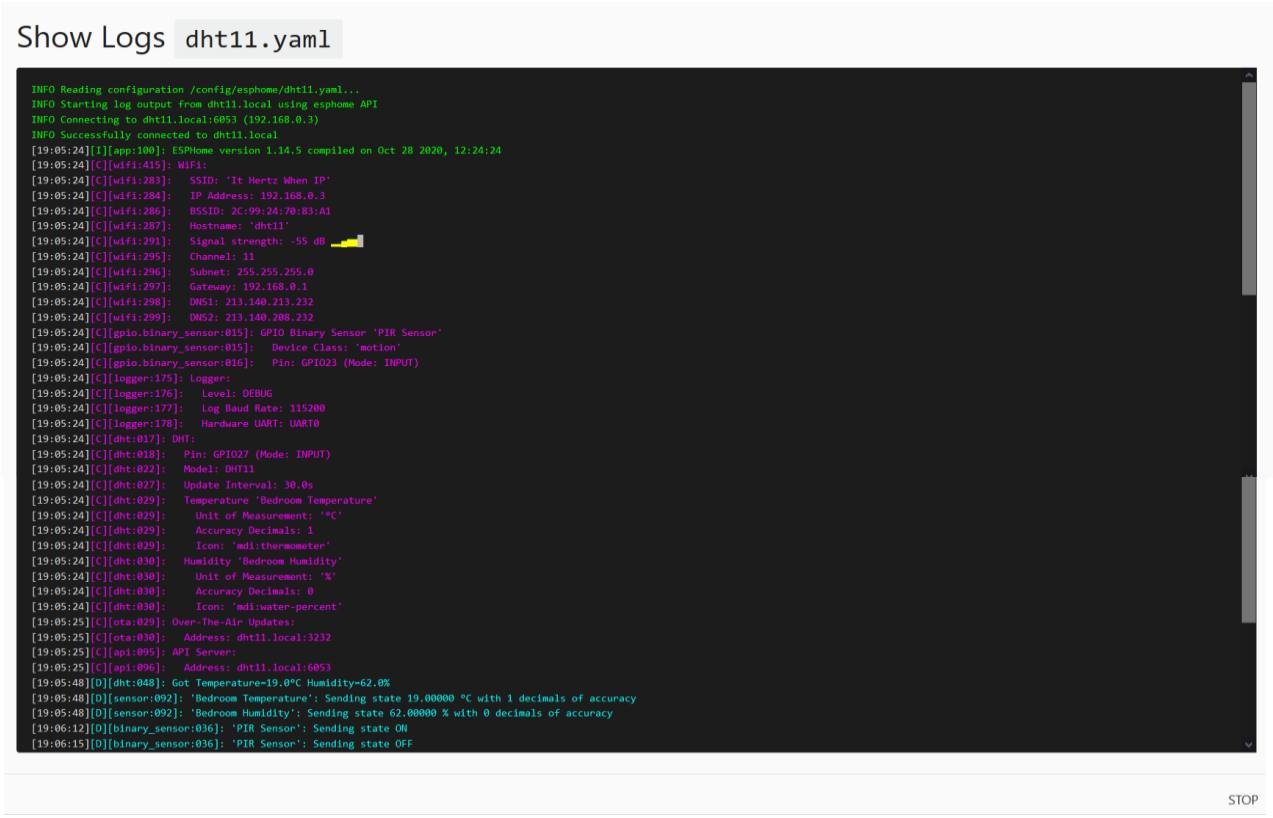
There was a delay in the camera feed that is shown in Home Assistant as well as poor frame rate. This issue is explained thoroughly in **Chapter 5: Experience and Problem Solving**, which also provides indicative information as to why this problem occurs as well as suggestions for alternative options in order to avoid this in future projects.

4.2.5 Code validation

Following the setup of the sensors and their ESP32 modules, it was necessary to validate the code snippet and the collected data. To validate the code in terms of formatting, the “VALIDATE” button from Figure 4.10 was used. This will check whether the code snippet uploaded to the ESP module is correct and follows valid YAML format. It is important to note that this validation check will not ensure whether the code behaves as expected and

if the intended operation is executed. This only acts as a compiler that checks if the statements written in the particular language can be processed.

In order to ensure that the code uploaded to the ESP32 module achieves the intended goal, we can press the “SHOW LOGS” button in Figure 4.12. This will proceed to present to the user, logs of the connection and data collected. The user can then manually assess the collected data, to ensure whether their goal was achieved. An example of the log file is shown in Figure 4.12.



Source: Personal Screenshot

Figure 4.12: ESPHome Logs

The example above demonstrates that both the DHT11 sensor and HC-SR501 PIR sensor can communicate with Home Assistant and that data is collected from them.

The same concept is followed with the DS18B20 sensor (water temperature sensor) for validating the format and intended functionality.

4.2.6 Automations

The Home Assistant system that was build had some automations based on data collection from sensory equipment, to enhance the experience of the user and remove the necessity for manual efforts needed by the user for the completion of a specified task.

4.2.6.1 Motion Detection Automations

The data from the motion sensor was used to achieve automations for turning the bedroom light on and off. When movement is detected by the motion sensor placed in the bedroom, the light turns on. The trigger for this automation is the detection of movement and a precondition would be that the light was turned off. A Boolean type variable was also initialized in the system, and its importance is significant since it is detrimental for the correct operation of the automations whilst ensuring that in case the user manually turns on the light these automations will not be triggered. When the automation for turning on the light, after motion detection, is executed, the Boolean which was initially set to false, is now set to a true state. Checking the light if was already turned off is essential since we must avoid carrying out this automation if the user already manually turned-on light. The Boolean is used as condition for the automation for turning off light. If there is no motion detected for one minute, then the light will turn off. The Boolean is important to be checked if it is set to true, since we must not turn off the light if it was turned on manually. The Boolean can only be set true when the automation for turning on the lights is executed, hence the only way that the automation for turning off lights is executed, is if the lights were turned on by the automation and not by the user. When this automation is executed, the Boolean is set to false again to allow for consecutive executions of these two automations without causing problems. A diagram for these two automations is shown in Figure 4.13.

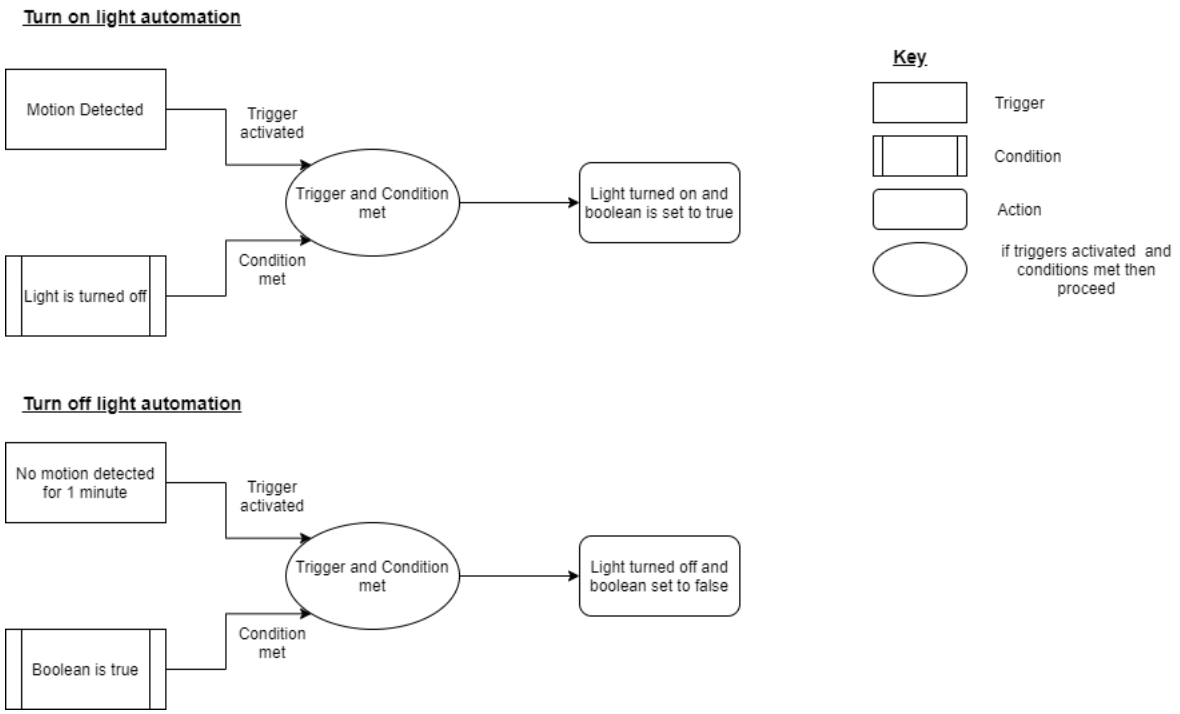


Figure 4.13: Light Automations based on motion detection

4.2.6.2 **Plex Server State Automations**

When video is streamed from the authorized Plex Server, light automations are executed accordingly. More specifically, when the Raspberry Pi, in the living room, that is setup as a Plex client called Rasplex(for streaming content from the Plex server), starts playing a video from the server, lights will be dimmed to 20% brightness. The trigger for this automation is that a video starts playing and the condition is that the lights were turned on and above 20% brightness. If the trigger is activated and the condition is met the current state of the lights is saved as a scene and the lights will be dimmed to 20% brightness. The scene created will contain the current state of the lights prior to deeming them in order to be able to return them to their original state at a later time. When the video playback is paused the lights will return to their original state using the created scene. In case the lights were originally turned off or dimmed under 20% brightness the automation for playing a video stream will not be executed, and hence no scene exists. As a result, the state of the lights will not be affected in this case. Additionally, to avoid unnecessary execution of this

automation, there is a condition that the lights are turned on. The lights that are affected by this automation are only the ones that are placed in the living room and they are referenced by their entity IDs, hence conflicts or wrong referencing is avoided. A diagram for these two automations is shown in Figure 4.14.

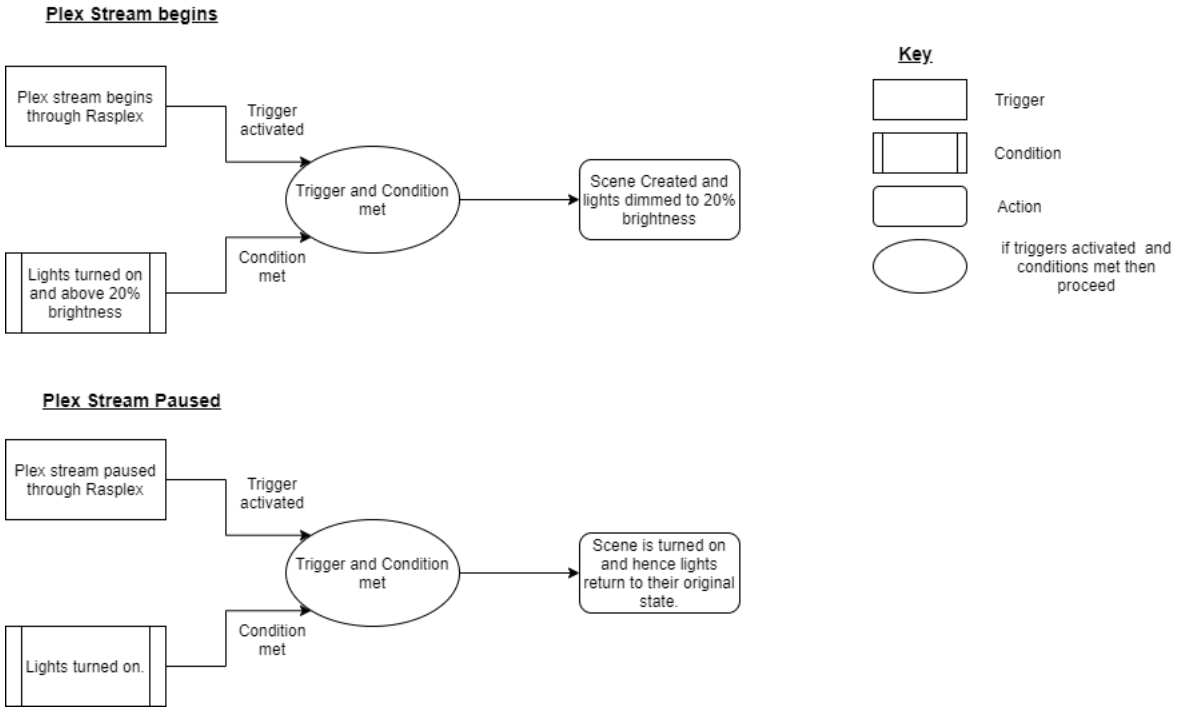


Figure 4.14: Light Automations based Plex streaming

4.2.6.3 **Geo-location Tracking Automations**

The location of the user is monitored using the geo-location tracker of the smartphone in their possession. This information is needed for the geo-location tracking automations implemented in the system. There is a zone setup, with the home of the user in the center. The radius of the zone can be changed according the needs of the user. When the user, that is identified as an entity in the system, leaves the zone an automation is executed. The triggers as stated earlier, is that the user leaves the “home zone”. There are no conditions required for this automation, and the action taken is that the user is notified on the smart phone that they have left the zone, and information regarding the temperature of the house is provided to them. Additionally, if any of the lights are turned on, they will be

turned off. When the user re-enters the zone, the automation for entering the zone is executed. The trigger is that the user, enters the zone and that he was previously not in the zone. This is used to avoid triggering this automation if the user is inside the zone beforehand. There are no conditions for this automation and the action taken is providing the same message to the user as the one he received when he left the zone. These automations are not considered essential for a good user experience but are the building blocks for achieving more useful automations based on geo-location tracking, such as turning a smart air conditioning system on or off. Hence, they were implemented and tested, with the intension of future expansion with the required hardware. A diagram for these two automations is shown in Figure 4.15.

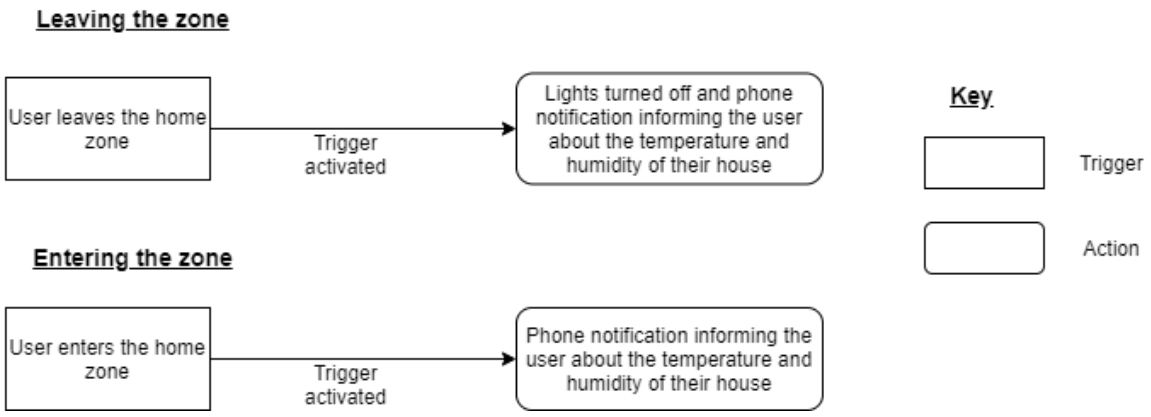
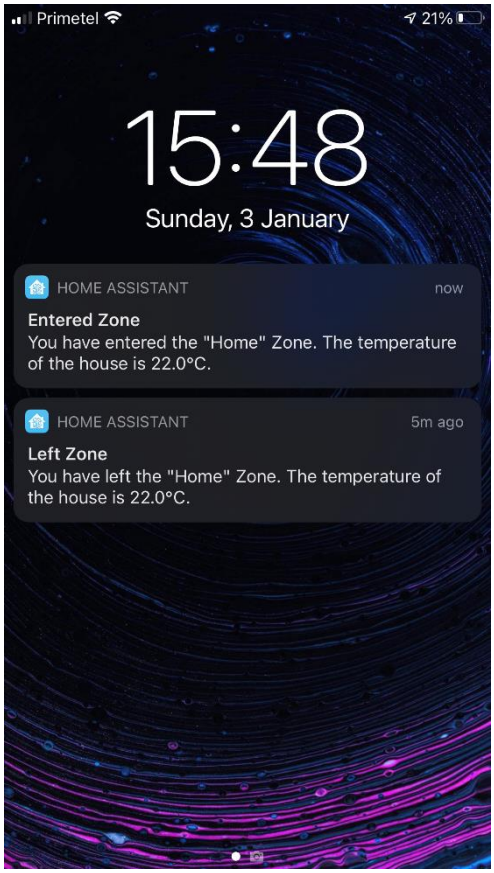


Figure 4.15: Geo-location tracking automations

An example of the information that is displayed to the user, through the notification received, is shown in Figure 4.16.



Source: Personal Screenshot

Figure 4.16: Notification using the geo-location tracking based automation

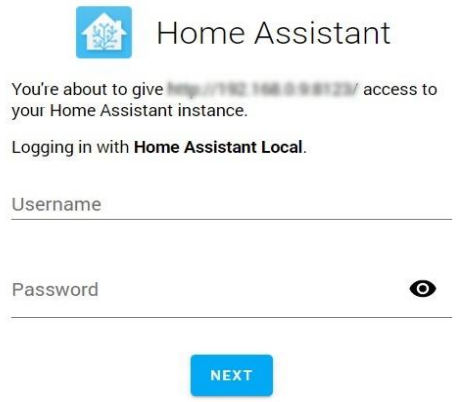
4.3 User Interface

The user interface of the setup system can be accessed by a web browser on any device that has a web browser application. However, for smartphone devices it is advised to use the native application available on the respective platform (IOS and Android). Therefore, by using a web browser on the Personal Computer device and/or laptop, and the respective mobile application on the smartphone of choice the user can have the recommended and desired experience through their interaction with the Home Assistant system.

4.3.1 Web Interface

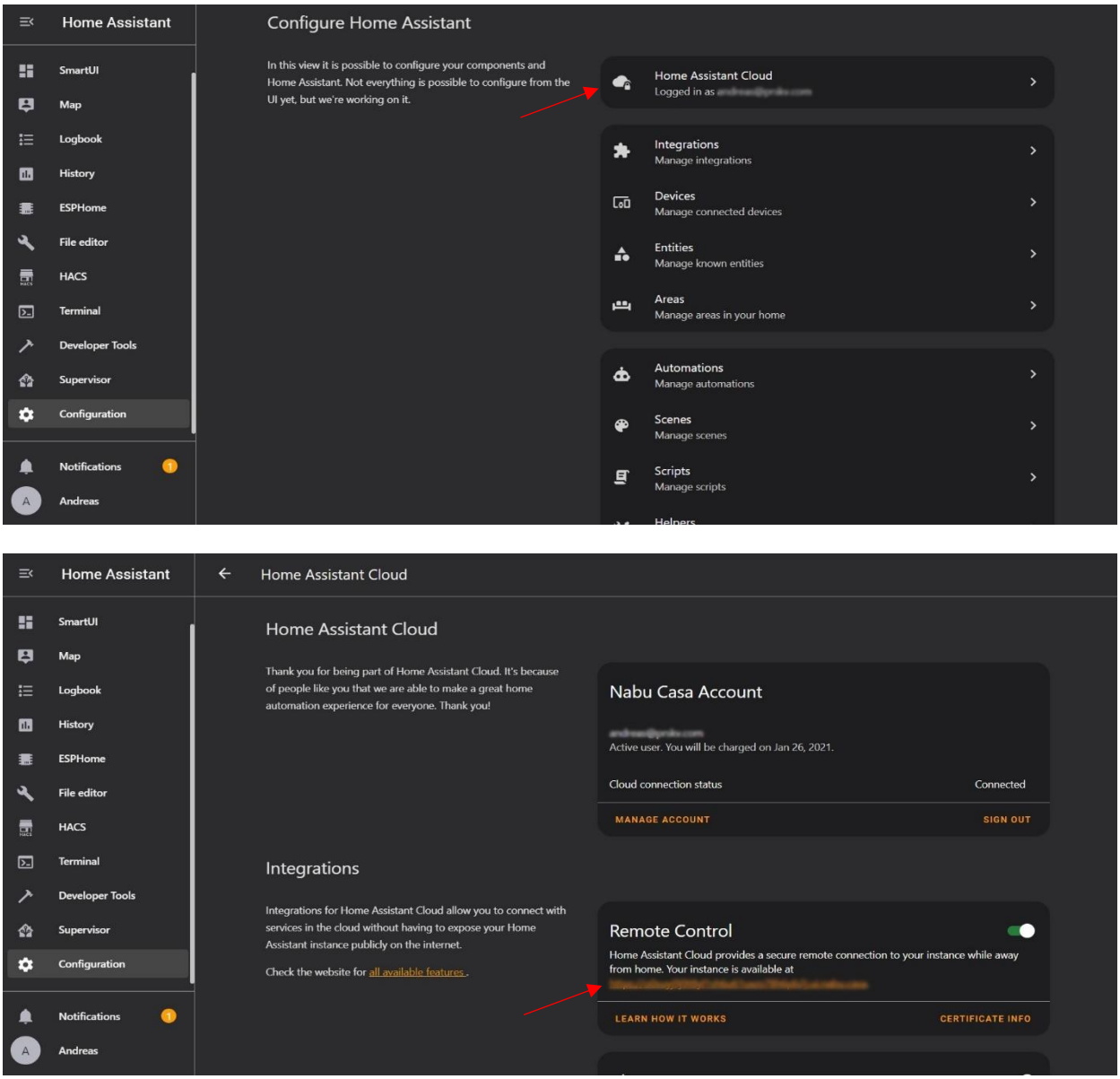
When the user visits the URL for accessing the Home Assistant system (explained in the initial setup guide**, Section 4.2.1**), he will be asked to enter his credentials to authorize the

device and login to the system (Figure 4.17). Alternatively, the Remote URL could be used, and this can be found in the “Home Assistant Cloud” selection, under the “Configuration” tab (Figure 4.18).



Source: Personal Screenshot

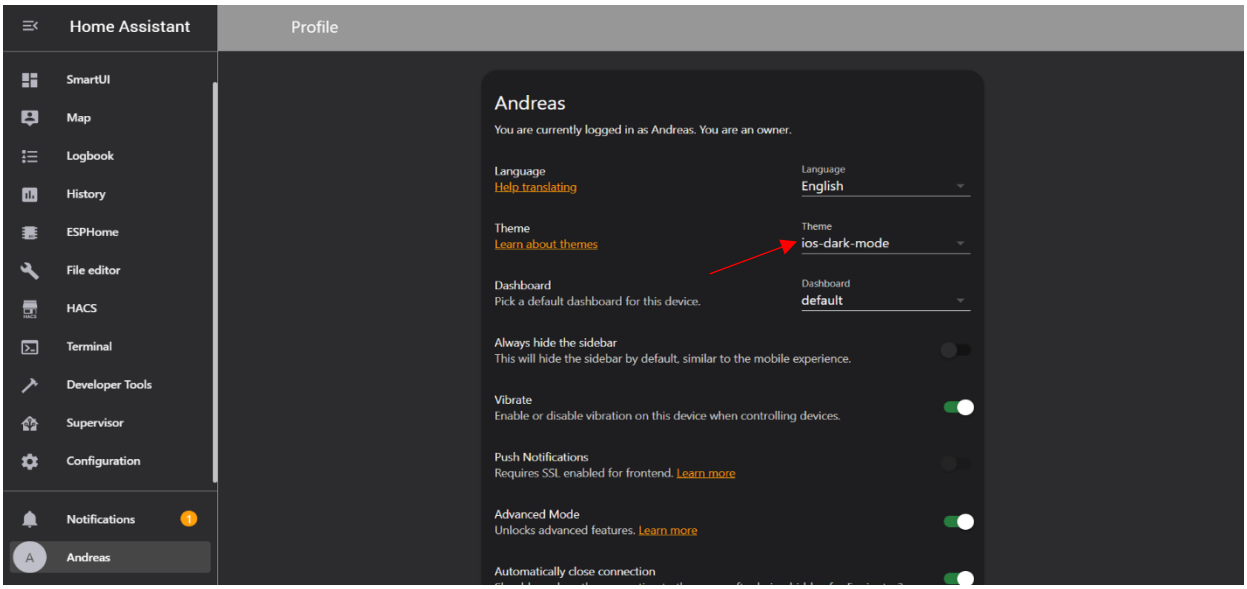
Figure 4.17: Authorization Page



Source: Personal Screenshots

Figure 4.18: Remote URL

There was a theme installed from HACS and the user will have to select it the first time that he visits the system with a new device. To make the theme selection, the user will need to navigate to the “User” tab on the bottom left-hand side of their page and click on their username. There they will need to choose their theme of choice, that they must have previously installed. In this thesis project the “ios-dark-mode” theme was used (Figure 4.19).



Source: Personal Screenshots

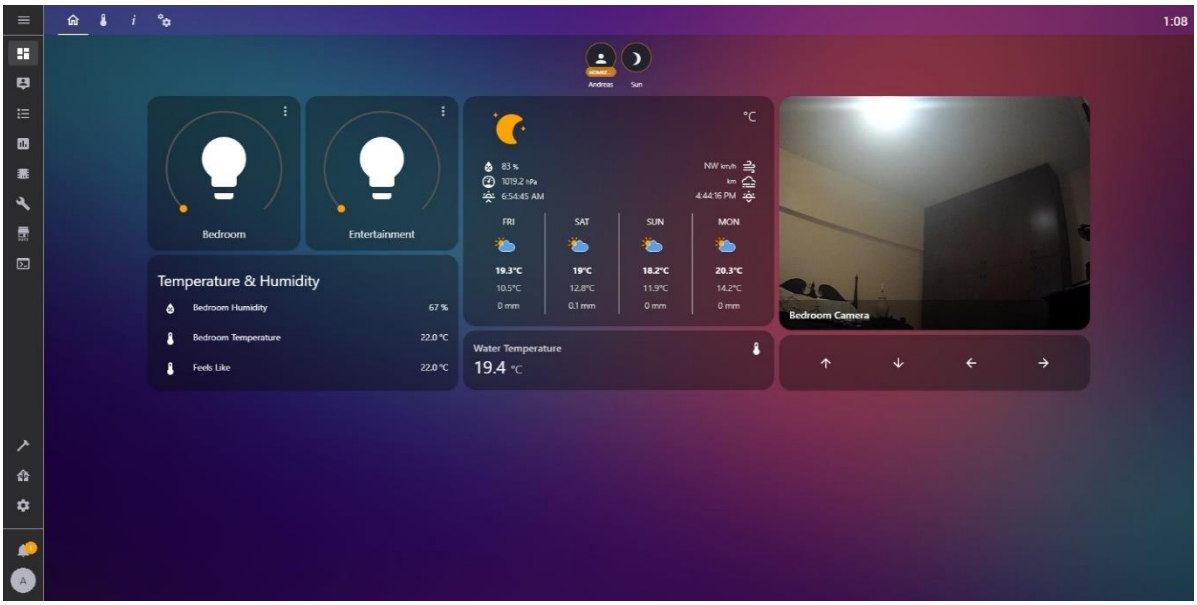
Figure 4.19: Theme selection

The dashboard is divided in four pages that each provide different functionality and information to the user. The information was grouped and categorized according to their relevance, with the first page being the most frequently used by the user and with the outmost importance. Information was also grouped together based on relation with the rest of the content in the page. These four pages are explained and demonstrated on the following sections.

4.3.1.1 **Main Page of Dashboard**

The user is greeted by the dashboard of the device and more specifically they are on the main page of the dashboard (Figure 4.20). Here they will be able to view information regarding the weather forecast of the next 4 days (the card in the middle). They will be able to switch on or off the light bulbs in the bedroom and entertainment room (named accordingly) as well as change their brightness with the usage of the circular slider around them. The user is also presented with information regarding the current bedroom temperature and humidity as well as the calculated heat index with the algorithm explained

in **Section 2.6**. The heat index is presented with the label “Feels Like” since this term will be more familiar to the user and hence easier to identify what information is presented to them. Additionally, the user can view the current water temperature as well as the camera feed streamed to them. The camera feed is streamed live from the IP camera and can be viewed on the right-hand side of the page. The arrows underneath the video feed are used in order to remotely control the camera and be able to monitor the whole area that can be covered by the camera’s rotating capabilities. The press of an arrow will rotate the camera towards the selected direction.



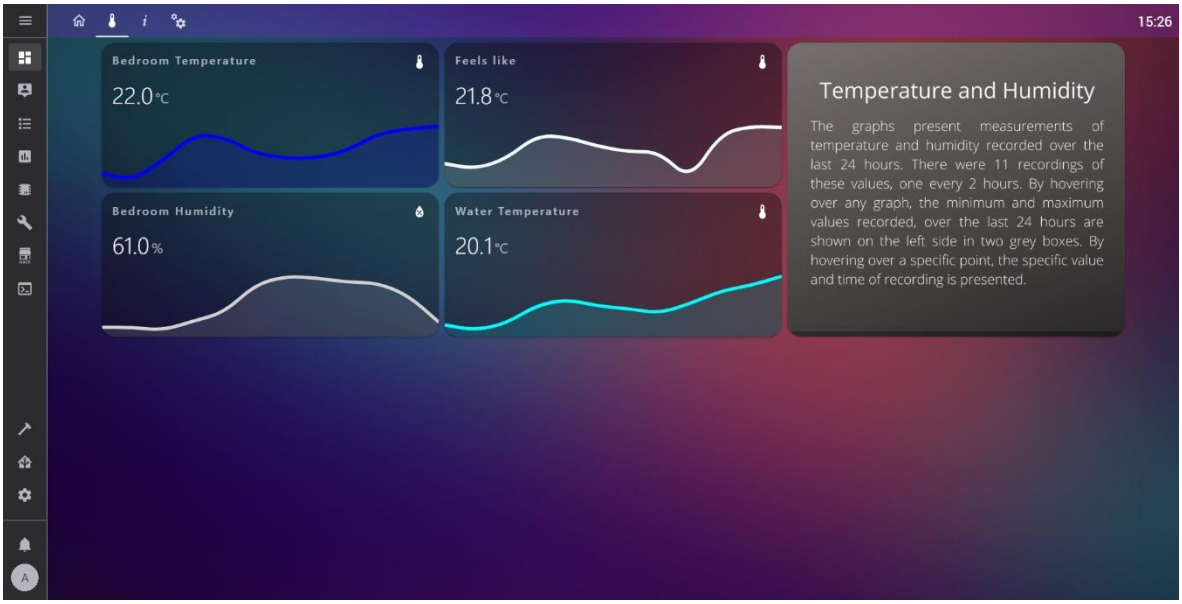
Source: Personal Screenshots

Figure 4.20: Dashboard main page

This main page contains the most frequently used functionalities that the user will utilize, as well as the regularly accessed information such as the sensory retrieved values of water temperature, room temperature and humidity.

4.3.1.2 **Temperature and Humidity Page**

The next page, as indicated by the thermometer icon, contains information regarding the water temperature and ambient temperature and humidity. In this page the user will have the ability not only to look at the current values, but also the values for these attributes in the last twenty-four hours. Each graph (shown in Figure 4.21) contains 11 points which on hover presents the temperature as well as the time of recording with a range of two hours for each point. Information regarding the scaling of the graphs as well as details such as the presentation of minimum and maximum recordings in the last 24 hours are explained in the text box on the right-hand side of the graphs (Figure 4.21).



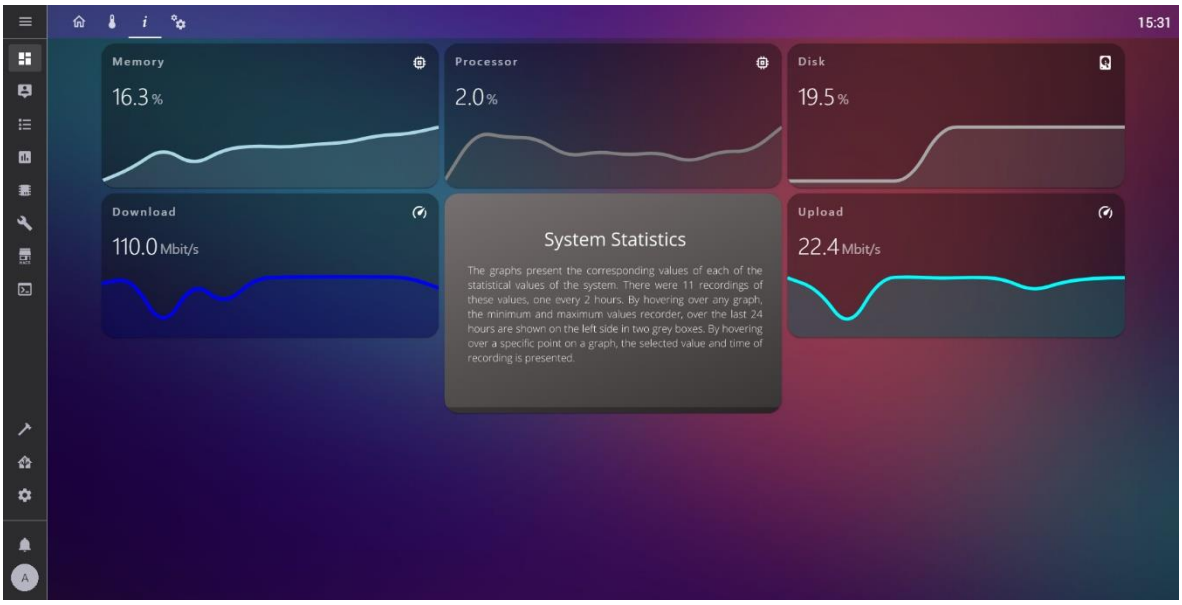
Source: Personal Screenshots

Figure 4.21: Temperature and humidity history

4.3.1.3 **System Information Page**

In this page, the user will be able to view information regarding the hardware of the system, in this case the Raspberry Pi which has Home Assistant installed on it. This information will likely not be accessed as frequently but it will be useful if the user wants to monitor the state of its hardware and troubleshoot if there is a perceivable hinderance in performance.

Similarly, to the previous page, the graphs follow the same scaling and variable presentation, but the information presented is the memory, processor, and disk usage, as well as upload and download speeds of the internet connection established by the system. This information is visualized in the respective graphs with self-explanatory titles for each graph (Figure 4.22).

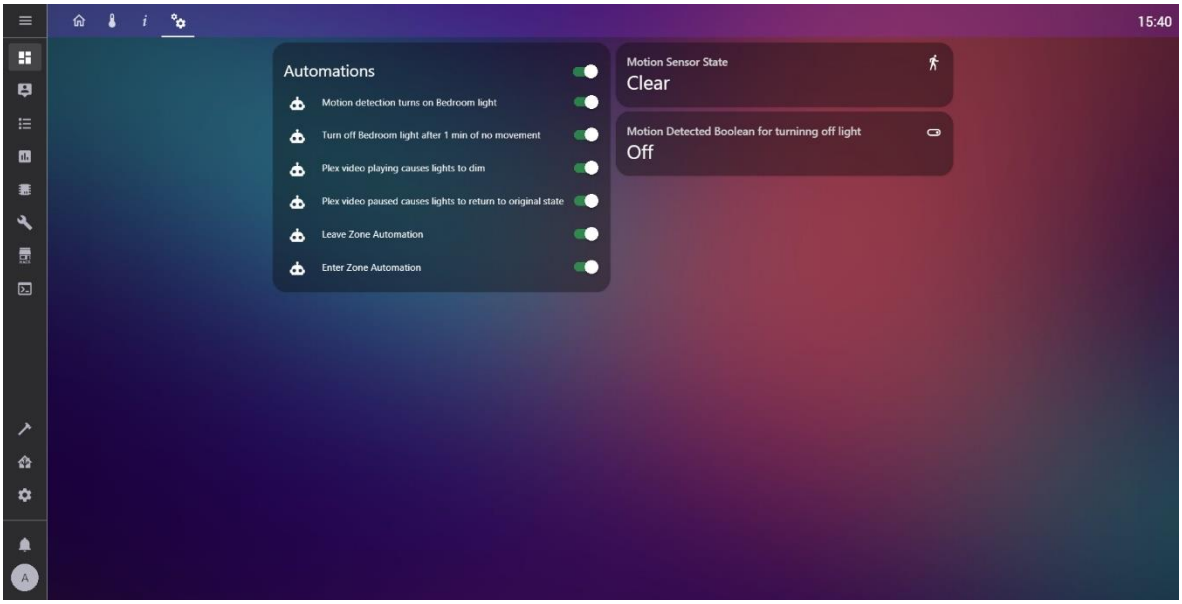


Source: Personal Screenshots

Figure 4.22: Raspberry Pi statistical performance

4.3.1.4 **Setting and Automations Page**

The final page is accessed by clicking on the settings/grind icon (Figure 4.20), and here the user will be able to view their automations, as well as turn them on and off if they desire to do so. This can be useful if they want to temporarily turn off a functionality such as turning on and off the lights based on the motion detection. They are also able to view the state of the boolean used for the light automations, explained in **Section 4.2.6.1**, as well as the motion sensor and examine if there is motion detection. This was very useful for troubleshooting and identifying the occurrence of false positives. The solution to this issue is explained in detail in **Chapter 5: Experience and Problem Solving**. It is a relatively common issue with these type of motion sensors. The page is shown in Figure 4.23.

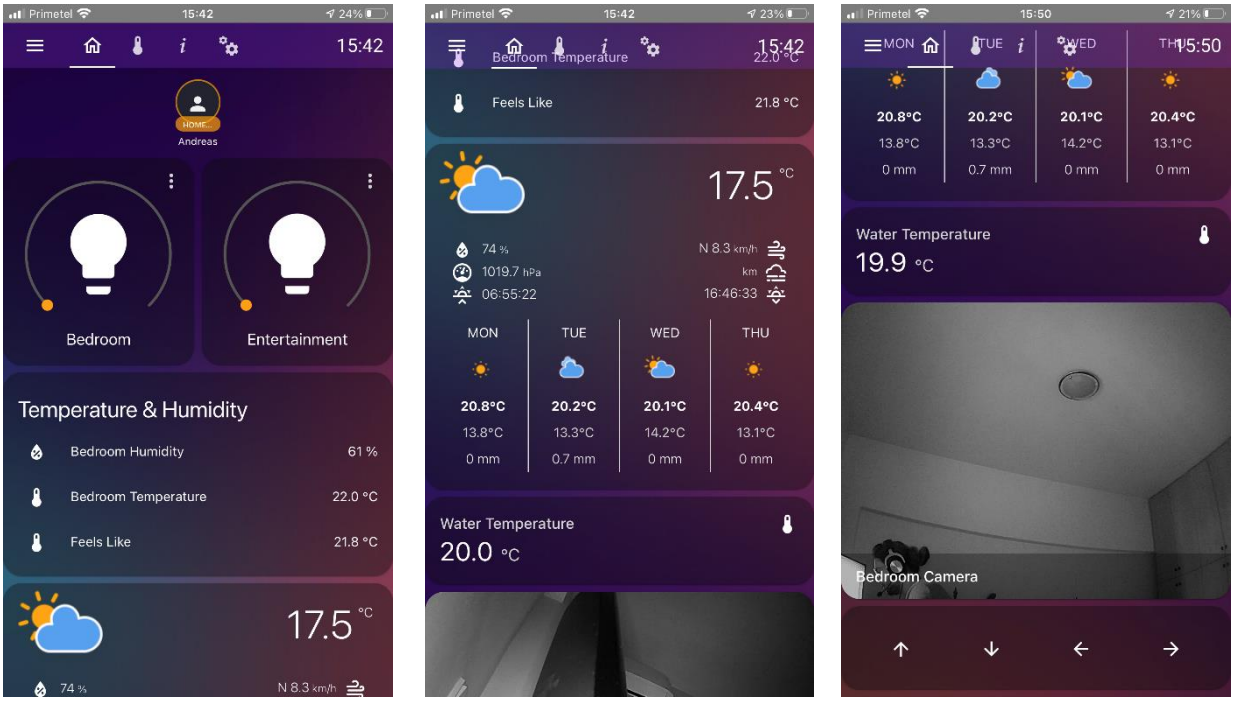


Source: Personal Screenshots

Figure 4.23: Automations and settings page.

4.3.2 Smartphone and tablet Interface

The design of the UI is responsive and is presented without issues on mobile devices. The functionality is the same as the one provided in the web browser. The main page is shown in Figure 4.24. The page displaying the history of collected values for temperatures and humidity is shown in Figure 4.25. The information of the state of system based on current performance of the hardware is shown in Figure 4.26. The final page with the automation and settings is shown in Figure 4.27.



Source: Personal Screenshots

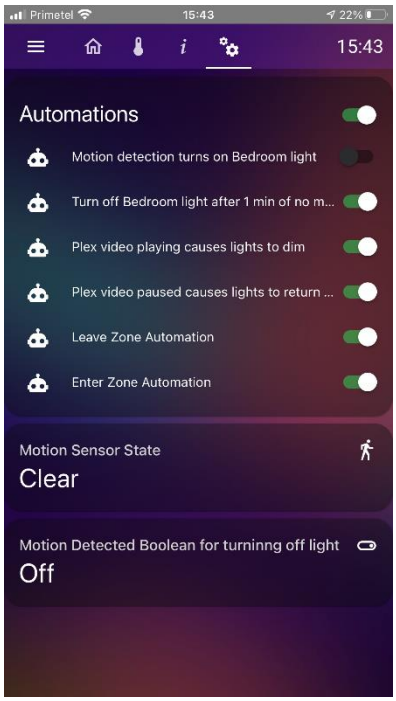
Figure 4.24: Main page on smartphone

Source: Personal Screenshots

Figure 4.25: Temperature and humidity history page on smartphone

Source: Personal Screenshots

Figure 4.26: State of machine page on smartphone



Source: Personal Screenshots

Figure 4.27: Automations and setting page on smartphone

Chapter 5

5. Experience and Problem Solving

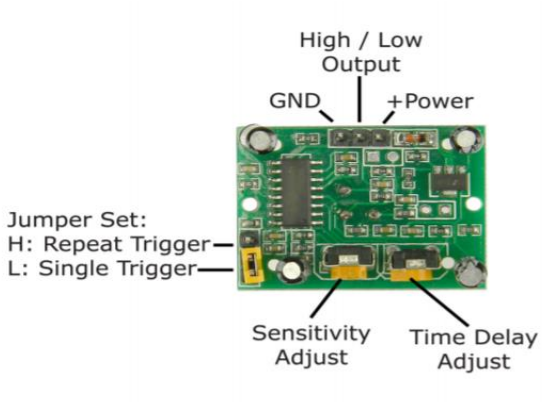
Through the process of developing this project, different problems were faced, that I decided to note down in order to inform anyone interested in developing a HomeAssistant based smart home project with the hardware and software that was used. These can help the reader decide if they want to use the same hardware and/or software or whether they will decide to look for and test alternative options.

One of the first difficulties encountered was configuring the DHT11, temperature and humidity sensor, with the ESP32 module, in order to be able to retrieve data from the sensor over the network, with the usage of Wi-Fi. Even though the necessary code for the data retrieval as well as the initial Wi-Fi setup for the ESP32 device was correct, after it was done uploading onto the ESP32 module, the connection with the sensor was never accomplished. This was evident when the log history was analyzed. To verify that this issue was not related to a hardware problem of the sensor nor the ESP32 module, the process of the coding and uploading was carried out again with different hardware of the same models. Since the problem persisted, and the code that was implemented was valid both in terms of syntax and functionality (the documentation was followed to achieve the desired code), the only logical source of the problem would be a software related issue. After extensive research, it was concluded that the core of the problem was an acknowledged bug in the version of ESPHome, which was the software used to code and control the ESP modules. The version installed at the time was 1.14.4, which affected one wire sensors connectivity, such as the DHT11, DHT11 and Dallas sensors [16].

After, installing a newer version of ESPHome, 1.14.5, the problem was no longer present, and the sensor equipment connected to the ESP332 module was able to transmit the necessary data without any issues. Hence, it would be advised to avoid using version 1.14.4 of ESPHome and conduct prior research regarding known bugs of the version that it is intended to be installed on the system.

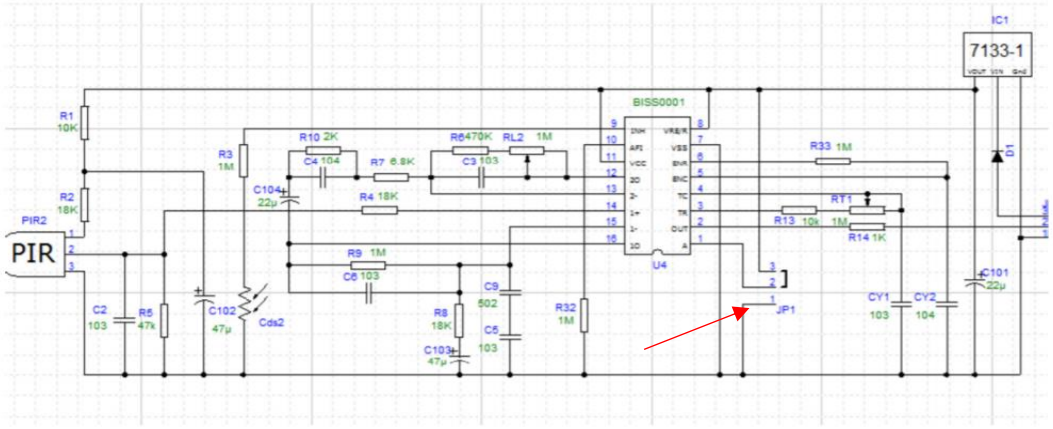
Another problem that was faced, was with the motion sensor of choice, the HC-SR501 PIR sensor. After setting up the motion sensor with an ESP32 module, so that motion detection data would be sent over the network, false positives were experienced. This issue is quite common with PIR sensors due to their sensitive circuitry and one of the most obvious explanations would be interference by the Wi-Fi signals of the ESP32, which can cause a false trigger. Using longer wiring to separate further the ESP module from the sensor, and thus avoid interference did not solve the problem, but it is a common fix for this issue. Upon further investigation, it was determined the jumper set would select between a single or a multiple trigger. By connecting the voltage cable next to jumper set (Figure 5.1), it allowed the usage of 3.3V due to the open circuit on line 3 from the jumper set (Figure 5.2). Normally the 3-pin connector would take a 5V input. However, through research it was determined that since a single trigger is enough for the needs of this project and the 3V input would suffice. The source of the issue was the 5V input and when the input voltage was switched to line 3 of the jumper set, the problem was fixed. Therefore, it would be advised to try this method if false positives are an issue on a project with this specific sensor. The wiring is presented in Figure 5.3.

There are more solutions to this issue available, hence in case these two proposed methods are not able to solve the occurrence of false positives, alternative solutions can be found in various forums. This method has been tested with another sensor exhibiting this issue, and the same results were yielded . This enhances the idea that the suggested method would be able to solve the occurrence of false positives.



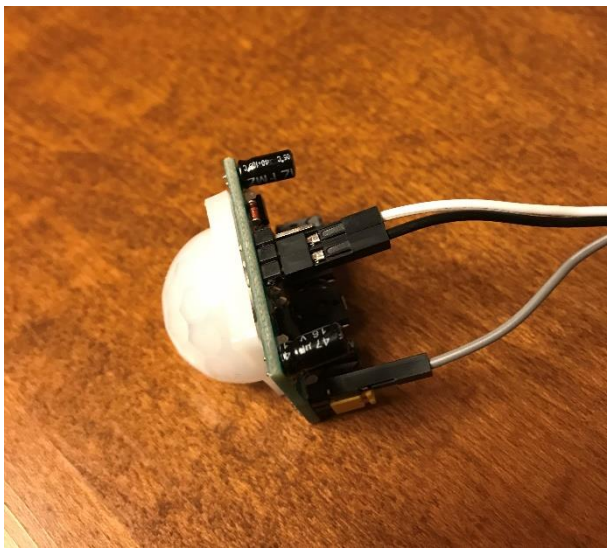
Source: https://www.mpja.com/download/31227sc.pdf

Figure 5.1: HC-SR501 picture



Source: https://www.mpja.com/download/31227sc.pdf

Figure 5.2: HC-SR501 schematic



Wire Color	Pin Purpose
White	Data
Black	Ground
Grey	3.3V

Source: Personal photograph

Figure 5.3: Wiring Guide

In the development of this project, the IP camera that was used was the “Digoo BB M1X”. The camera feed is streamed, using RTSP so that the muxer can be used to send a stream to a server supporting it. There is an RTSP URL that is using FFmpeg software in order to access a live video feed of the camera. FFmpeg was integrated on Home Assistant and the video was streamed using the RTSP URL with the correct credentials and IP reference. However, the stream had an apparent delay and a low framerate. After conducting research, it was concluded that FFmpeg is resource intense and the hardware of the Raspberry Pi is the most likely explanation for this delay and low framerate, since the transcoding of the stream was handled on the device itself. With reference to a post in the community homepage of HomeAssistant, it was concluded that the best stream for a similar project would be **M**otion **J**oint **P**hotographic **E**xperts **G**roup (MJPEG). Since this camera only supports an FFmpeg stream, it is advised to look for an alternative option that supports MJPEG stream. The post in the Home Assistant community was an experiment that elaborated and evaluated different streaming services provided by IP cameras [17] and it was a major factor in the conclusion explained above. Since there was no personal experience with alternative options, the suggestion for a camera with MJPEG support, is solely based on the results and evaluations covered by this post.

Chapter 6

6. Evaluation

6.1 Corona Virus effects

Due to the global pandemic that we are facing at the time of the development of this project, it was rendered difficult to have people physically interact with the system and have a hands-on experience. Hence, for the safety of both parties (myself and the testers of the system), it was concluded that the best alternative solution for an evaluation would be to host online meetings with the testers of this project, and present them the finalized product. It is certain that this solution limits the experience of the testing group, since they have to interact with the system in an unorthodox way. Additionally, the testing group was limited in size due to the difficulties that occurred through the evaluation method. The method of evaluation and retrieval of feedback is further explained in **Section 6.2: Evaluation Method**.

6.2 Evaluation Method

To overcome the difficulties that were faced due to the global pandemic, as explained earlier, it was decided that a suitable alternative method for evaluation would be presenting the system and its capabilities in video call meetings, with a small number of participants. This involved people from a variety of ages and professions, to ensure that the best possible sample of users was used, and the feedback given would be representative of a real-world usage.

A sample of ten people was used to receive feedback and have a conclusive evaluation that would be considered creditable. Each of the participants had a personal online meeting,

where they had the chance to view the final system through a live presentation and a combination of prerecorded videos that were used to demonstrate the automations that are in effect. Both throughout and at the end of the presentation, the test group had the chance to ask questions as well as request further information or elaboration on a specific aspect of the system, if that was required. An interview style approach was implemented hence detailed opinions and criteria was extracted from the test group.

The feedback was over-all positive. Each of the test group members when asked to provide the positive and negative aspects that they have recognized throughout the presentation, have unanimously found the following positive aspects to be recognizable:

- The system was considered to have a minimal and clean design that does not overwhelm the user.
- It provided usability since the users managed to achieve specified goals throughout the interaction with the system and the following five factors of usability were met:
    - Learnability: the simplicity of the system and the use of metaphors and visual affordance, helped the user realize the intended usage and effect of the aspects of the UI.
    - Efficiency: each of the task that the UI provided was completed quickly.
    - Memorability: the users that examined the system after the initial presentation were able to recall how they would interact with it since they recognized aspects of the UI.
    - Errors and Error Frequency: no errors occurred during the presentation or personal evaluation of the system.
    - Satisfaction: as indicated by other comments of the users, they have found the system pleasant to use and would be interested in further interaction with it.
- The system provided clear categorization and the users were able to identify/recognize the contents of the pages by the icons in the navigation bar. They have commented that the main page indeed contained the most relevant

information and functionality, that they would typically use and the secondary information on the remaining pages was also deemed useful, but they would not use or require access to it as regularly. Hence the order of the pages was also considered convenient based on frequency of usage.

- The functionality provided by the system and the automations were characterized as essential and useful. Furthermore, they were able to identify and list beneficial scenarios of the implementation of this system.
- The functionality of each of the aspects of the UI was clear and the effect of their interaction with them could be predicted accurately (self-explanatory).
- Beautiful UI with a nice background and color palette.

The negative comments and/or suggestions for improving the system are collectively the following:

1. Graphs could be the same color, for example all white.
2. Download and upload speeds could also be on the home page of the system
3. Titles and explanation for the graphical representation, such the scaling can be useful for the users to understand what is shown to them.

From the suggestions received above the first two points (points 1. and 2.) were considered personal preference and are subjective to each user hence they were left unchanged. To further solidify that these two opinions were purely subjective and would not change the experience of most of the users, the testers were asked whether they agree with these comments. Since the majority disagreed or found the change, if in effect, irrelevant, these changes were not implemented. However, the third suggestion was deemed crucial for further enhancing the experience of the user and the comprehensibility of the system. The final product presented in **Chapter 4: Design and Implementation** has this change already in effect, as portrayed in the screenshots provided.

In addition, to the evaluation based on feedback received by a test group, the manual testing that ensued was another fundamental aspect for assessing the system. The system

was operational for consecutive days to ensure that real world application would be feasible. It was also tested repeatedly, for each of the function and automations that it provided, to ensure that were all operating efficiently and as expected. Its application in my personal house also further validated its usefulness, since its operation verified the expectation, that such a system would remove the need for manual interference of the user into achieving specific goals (e.g., dimming the lights when a movie is playing). Also, the usage of an interactive UI that provided useful information, such as water temperature, room humidity and room temperature, was very handy, and the control and monitor of lights and cameras was practical as well as beneficial both in the presence and absence from the house.

Feedback from the users was a crucial part of finalizing the system as well as ensuring that it can be integrated and used with relative ease at the house of interested people. It was concluded the user's mental model was simple and that people with limited experience or knowledge in a more technical or technology-oriented field would be able to interact with the system and identify the functionality provided to them without guidance.

Chapter 7

7. Conclusions and Future Work

7.1 Conclusions

The personal evaluation of the dissertation project, in conjunction with the testing phase with a sample user group, helped infer several conclusions related both to the system developed and the potential improvements or expandability options, which would further enhance the user experience. Firstly, it was determined that the smart home project provided useful automations, that would be used frequently throughout the daily lives of the potential user. This validates the need for these automations since their absence would otherwise require the user to execute these tasks manually. In addition, users in the test group were able to identify the prospect of expanding these automations, with the acquirement of more smart devices. The automation based on the geo-location tracking of the user is the most obvious automation that would benefit greatly from the integration of more smart devices in the network. For example, the addition of a smart air-conditioning system would allow for better temperature control of the house and efficient power usage, whilst maintaining a pleasant domestic environment for the user. It is also clear that many of the automations can be replicated with more hardware and be implemented in more areas of the household.

The camera used for this project is considered sufficient for the needs of monitoring a specified area of the house and the provision of security. Despite the low frame rate of the video feed, the user can execute the rotation of the camera and inspect the state of the area that the camera is placed at. However, it is concluded that in efforts to maximize the security it would be ideal to minimize the delay of the video feed. Reduced frame rates

would also be needed if the camera would be used for reason other than security, such as monitoring a baby. Hence an alternative hardware would be more suitable in most use case scenarios.

The motion sensor placement is crucial, and it is advised to place it away from any other hardware that has Wi-Fi capabilities since the interference might cause false positives. Although, the problem with false positives was countered with the method explained in **Chapter 5: Experience and Problem Solving**, it is concluded that alternative motion sensors should be considered. This will allow interested parties to compare their experience with other sensors and conclude on a finalized choice, with all factors examined.

The DHT11 sensor was a good choice for monitoring the temperature and humidity, with regards to an indoor usage. A typical household will not exceed the range of temperature and humidity that this sensor can record. If the need for an outdoor monitoring of these variable arises, then this sensor is no longer applicable due to its limitations in range and accuracy.

Monitoring the Raspberry Pi with extensive usage, provided conclusive evidence that there is a need for a casing with support of a cooling system. There is heavy load on the Raspberry Pi since it is constantly communicating with the sensors of the network and controls and monitors various devices. Running the system without a cooling fan caused the Raspberry Pi to overheat and become warm to the touch. This issue was eliminated when the Raspberry Pi was enclosed in a case with a fan, and running the project for consecutive days showed no signs of thermal issues.

It was also observed that under-voltage of the Raspberry Pi will cause it to underperform. This is to be expected and it is advised to use the official power supply of the Raspberry Pi as to ensure that the voltage supplied is not dropping below the operational value. The cause of under-voltage can also be the cable that supplies power.

The requirements that were set for the successful implementation of this project were fulfilled. The list of the requirements as well as their fulfillment during the development of the system is explained in **Section 4.1 Requirements and Guidelines**.

Finally, the research and development phase for this diploma project indicated that the choice of hardware for the sensory equipment is likely not to be limited by the usage of Home Assistant. However, the community forum would be ideal for validating the compatibility of a specific hardware. The limited restrictions of the software and the active community as well as development team of Home Assistant are substantial factors in the recommendation of this open-source software for a smart home project.

7.2 Future Work

The possibilities for future work and expansion on this thesis are limited only by the person who will aim to expand upon the already done work and their budget. One of the first and important aspects that I would advise to further optimize this project is engage in statistical research regarding the geo-location tracking. More specifically, it would be beneficial to track the overhead of the current geo-location tracking, which is in place, and analyze both the power consumption as well as CPU usage. Additionally, other methods for geo-location tracking could be analyzed and compared to the integration of the phone as sensory equipment (the current method in place). OwnTracks is another possible alternative which I would suggest. This third-party application can be integrated onto HomeAssistant and the smartphone of choice (Android and IOS). It then proceeds to exchange information with the HomeAssistant instance using webhooks, and the remote URL. Regarding the remote access of the Raspberry Pi (HomeAssistant hub), the current method in place is using “Nabu Casa”, which as described in **Chapter 3: Methodology**, is a premium service that is provided by HomeAssistant which provides cloud service for remote access (outside of the local network) to the HomeAssistant. An alternative solution to this, can be setting up DuckDNS, a third-party software available at the community add-on store. Since there is a supporting community, it is easy to find guides on how to install and run this, however it requires technical knowledge, such as how to change the router settings in order to setup the DNS.

The major benefit is that this is a free third-party add-on, which can be ideal if we have a limited budget. Nabu Casa provides further functionality that could be considered essential according to the needs of the user, therefore these factors should be taken into account for the decision of the best suited method.

Alternative Sensors:

- Temperature and Humidity Sensor:

  The temperature and humidity sensor which was used in this project was the DHT11. This sensor has a limited range of temperature and humidity tracking which might not necessarily be of much importance. Its range for relative humidity readings is between 20-90% and for temperature tracking is between 0-50°C. These ranges can be considered good for Smart Home usage and tracking of temperature and humidity of an indoors environment, since values for temperature and humidity would not typically fall outside of this range. However, the major drawback for this sensor is the accuracy of these two readings. It has a ±5% accuracy for the humidity tracking and ±2°C accuracy for the temperature reading. A good alternative option for this sensor would be its successor model, the DHT22. It is slightly more expensive, but it offers a great accuracy which can be ideal for such a project and the calculation of heat index. The accuracy and range of each sensor is presented on the table below.

	DHT11	DHT22
Temperature Range(°C)	0 to 50	-40 to 80
Humidity Range (%)	20 to 80	0 -100
Temperature Accuracy (°C)	±2	±0.5
Humidity Accuracy (%)	±5	±1

Table 7.1: Accuracy and range of DHT11 and DHT22

It is also important to note that there are many more options in the market, which can provide an even better accuracy, or be a better solution due to the lack of the need of IO pins. However, these two sensors are ones of the highly recommended by the community due to their pricing and since they are suited for a smart home project or prototyping phases.

- Motion Sensor:

The motion sensor used for this project was the HC-SR501 PIR Sensor. As explained in **Chapter 5: Experience and Problem Solving**, a lot of false positives were encountered upon the initial setup of the sensor. However, through some research a solution to the problem was discovered. Considering the cover distance of about 120° and 7 meters, the HC-SR501 PIR sensor is more than suited for this specific system. It was able to discover accurately when there was motion, and hence the automations for turning on the lights and keeping them turned on for a set time after the last motion detection was implemented. However, since this is an IO connected device which requires connection to an ESP module to transmit data over

a Wi-Fi connection, there was a need of constant power supply. A suggestion in order to improve the setup for a smart home would be either to supply power through a power bank or an even more efficient and convenient approach would be to take a device with an in-built battery. Since motion sensors require insignificant amount of power to function, both options stated would be convenient. There is also the aspect of aesthetics since this is a motion sensor with exposed hardware. This could be solved by 3D-printing a housing for the sensor or using an alternative sensor. We have to take into account the budget for the Smart Home project as well as our needs in order to decide on the most suited sensors.

- Camera:

  The camera that was used for this project is the "Digoo BB MIX". It is an IP camera and the stream of the camera's video feed is rendered on the HomeAssistant frontend. As explained in **Chapter 5: Experience and Problem Solving** the camera feed that is streamed on the frontend, is using RTSP so that the muxer can be used to send a stream to a server supporting it. Then the feed is streamed from the RTSP URL with FFmpeg, which is a free and open-source software project for handling multimedia files. Since the camera used only supports an FFmpeg streaming service, there were no alternatives options that could be tested by myself. Through research I have concluded that FFmpeg is resource intense and will require a lot of power in terms of CPU to achieve a good streaming experience. Some of the variables that are of essence is framerate and the delay of the stream. Even though it seemed like the camera handled the streaming without a problem, the CPU of the Raspberry Pi which needed to do the necessary transcoding could have been the cause for the low framerate and delay. In the community page of HomeAssistant there was an experiment that covered many of the popular streaming services provided by IP cameras. This can be referenced in order to conclude on a more suitable IP camera and streaming service but more research in this area would be also helpful. Based on the evidence from the community page [17] I have concluded that a good

alternative for further experimentation would be a camera that supports MJPEG streaming since it was handled better by the HomeAssistant frontend.

Additionally, since this is a smart home project, there can always be improvement upon it by implementing more sensors and equipment. With a more flexible budget there can be implementation of more smart bulbs around the house with the combination of motion sensors in order to achieve a more seamless experience while navigating around the rooms of the household. More automations could also be integrated in order to improve the experience. For example, if there are different machines available in rooms that are capable of streaming media content from the Plex Server or other streaming services, light automations could be specified for different scenarios. A great addition could also be mapping the house to provide a more interactive and comprehensive visualization of the house and the smart devices in each of the areas of the house, so that the user can interact with the front end more easily. The HomeAssistant website provides information and guides for achieving this [18] but more information can also be found through the community HomeAssistant page. A smart air conditioning system or heaters can also be used in order to set the temperature of the house at a desired value prior to the user's arrival. If a smart air conditioning system or heaters are in place, the data from the temperature and humidity sensor can also be used in order to maintain room temperature, which is considered ideal for a living environment. This can be done by using the geolocation tracking method that is already integrated. Instead of informative notifications that are currently displayed on the user's phone upon entering or leaving a specified area, the user can alter the automation so that the air conditioning system or heaters are turned on or off. Another option would be setting different areas of the map as a specified location with a radius around them indicating a different zone. This can have significant advantages, and an example of this could be setting the location that the user attends for their occupation. Specifying more automations that revolve around the user's schedule, whether those are time specified automation or location based, can provide a more

economically and environmentally friendly solution since there would be optimized usage of electricity in order to achieve the user's desired environment.

Furthermore, the data collected from the different sensors can be used and analyzed through machine learning. This can optimize certain scenarios and will provide a more personalized experience for the user, hence improving the user experience. This will require data collection for a prolonged period of time and the necessary knowledge in the field of machine learning, but it will undeniably improve the system and the experience of interaction with it.

Chapter 8

8. Bibliography

[1] J. Teicher, “The little-known story of the first IoT device - Industrious.” https://www.ibm.com/blogs/industries/little-known-story-first-iot-device/ (accessed Nov. 06, 2020).

[2] Home Assistant, “Install Home Assistant.” https://www.home-assistant.io/getting-started/.

[3] Home Assistant, “Home Assistant Community.” https://community.home-assistant.io/ (accessed Dec. 24, 2020).

[4] openHAB, “Raspberry Pi | openHAB.” https://www.openhab.org/docs/installation/rasppi.html (accessed Jan. 15, 2021).

[5] openHAB, “openHAB Community.” https://community.openhab.org/ (accessed Jan. 15, 2021).

[6] ARM, “What are IoT Devices – Arm.” https://www.arm.com/glossary/iot-devices (accessed Dec. 26, 2020).

[7] “ESP32 DevKitC Pinout, Overview, Features & Datasheet.” https://components101.com/microcontrollers/esp32-devkitc (accessed Dec. 26, 2020).

[8] Weather Prediction Center, “Heat Index Equation.” https://www.wpc.ncep.noaa.gov/html/heatindex_equation.shtml (accessed Nov. 06, 2020).

[9] Elegoo, “ELEGOO Upgraded Electronics Fun Kits (4 Versions) for Arduino, Raspber – ELEGOO Official.” https://www.elegoo.com/collections/electronic-component-kits/products/elegoo-electronics-fun-kits-4-versions?variant=32365666336816 (accessed Jan. 03, 2021).

[10] Home Assistant, “OwnTracks - Home Assistant.” https://www.home-assistant.io/integrations/owntracks/ (accessed Jan. 03, 2021).

[11] Home Assistant, “YAML - Home Assistant.” https://www.home-assistant.io/docs/configuration/yaml/ (accessed Jan. 03, 2021).

[12] “Installation | HACS.” https://hacs.xyz/docs/installation/manual (accessed Dec. 30, 2020).

[13] “Norwegian Meteorological Institute.” https://www.met.no/en (accessed Jan. 03, 2021).

[14] “Speedtest by Ookla - The Global Broadband Speed Test.” https://www.speedtest.net/ (accessed Jan. 03, 2021).

[15] Felixsteghofer, “GitHub - felixsteghofer/digoo-m1x_hacks.” https://github.com/felixsteghofer/digoo-m1x_hacks (accessed Nov. 06, 2020).

[16] Home Assistant, “ESP32 DHT11 Sensor Problem - Third party integrations / ESPHome - Home Assistant Community.” https://community.home-assistant.io/t/esp32-dht11-sensor-problem/204494/8 (accessed Nov. 06, 2020).

[17] Home Assistant, “I tried all the camera platforms so you don’t have to - Configuration - Home Assistant Community.” https://community.home-assistant.io/t/i-tried-all-the-camera-platforms-so-you-dont-have-to/222999 (accessed Dec. 29, 2020).

[18] Home Assistant, “Picture Elements Card - Home Assistant.” https://www.home-assistant.io/lovelace/picture-elements/ (accessed Dec. 29, 2020).