



Galway-Mayo Institute of Technology
Bachelor (Hons) of Software & Electronic Engineering

House Hestia

IoT Smart Home Automation System



Paulina Osikoya
G00348898

Supervised by Brian O'Shea

Submitted to the Galway-Mayo Institute of Technology
May 2020

Declaration

This project is presented in partial fulfilment of the requirements for the degree of Bachelor of Engineering in Software & Electronic Engineering at Galway-Mayo Institute of Technology.

This project is my own work, except where otherwise accredited. Where the work of others has been used or incorporated during this project, this is acknowledged and referenced.

Abstract

Automation is becoming an increasingly massive section and hot topic in today's IoT sector. From the automation of arduous long complex installations using basic to automating our day-to-day tasks by simply implementing a mechanism that notifies you of your daily tasks to be executed during your set-out schedule, it is pertinent to state that the significance of automated activities in post-modern society cannot be disregarded. The term automation is very broad and encapsulates a wider spectrum of categories, one of which is tagged home automation. Home automation is a prevailing subject in automation and IoT and generates billions of dollars in revenue each year. During my Internship, my role was centralized around automation and design of software to achieve and reach levels of performance that the world has never witnessed before. Premised on this fact, I wished to implement similar into my Final Year Project (FYP). My Final year project is an internet of things (IoT) home automation system. The name of my Home Automation System is called Hestia. The name is derived from ancient Greek religion. She was the goddess of the ordering of domesticity, the family, and the home, hence the adoption of the name which I thought would be befitting for my FYP.

The objective of my project was to allow the homeowner to utilize and leverage the power of IoT to control and monitor their house remotely via a website while they executed their day to day tasks. As the project progressed on, I incorporated other technologies such as container technology, AWS cloud framework and Google-Assistants APIs. After this revelation, the goal of my project shifted towards being able to enable my Home Automation System to be fully accessible from any IoT device and to be optimizable and efficient. The IoT home automation that I built was based on the Home Assistant – Hassio that uses microcontrollers and development platforms to host its technologies responsible for monitoring the house. Currently, the IoT home automation system is working, and all sensors data can be monitored live. Most of the goals in this project have been achieved, resulting in a stable system.

Acknowledgements

I would firstly like to express sincere gratitude to all the mentors, supervisors and technicians in the Galway-Mayo Institute. They helped and guided me through the various aspects of my project, gave me the constructive criticism, the to the sites and sources that made it entirely possible for me to comprehend the factors that attributed to my project. I would also like to express my appreciation for their tireless efforts in making sure the module ran smoothly throughout the Semester.

Being in final year can be a very stressful time for all students, but I felt as a fellow final year student the unwavering support I received from all staff was incredible.

I would like to give a special thanks to Brian O'Shea, Michelle Lynch and Paul Lennon for their supervision and support throughout the year. Throughout my project, I received fantastic advice, knowledge and guidance that motivated and enlightened my mind and pushed me to advance my project to new levels.

Also, I would also like to give thanks to my colleagues for making the project labs an enjoyable learning and informative environment, which took the imminent stress of the project.

Lastly, I would not have been able to complete this project without the support and continued support from my family and friends.

1 Relevant Project Documentation and Links

GitHub page Link to my Final Year Project (FYP) :

https://github.com/PaulinaOsikoya99/FYP_Project

Trello page Link to my Final Year Project (FYP) :

<https://trello.com/b/wMabNsvC/fyp>

YouTube Link to FYP:

<https://www.youtube.com/watch?v=QF8W7Al0L70>

Table of Contents

1	Important Project Documentation and Links	5
2	Summary	8
3	Introduction	10
4	Chapter breakdown	13
5	Project Architecture	14
5.1	MQTT	15
5.2	ESP32	16
5.2.1	Sensors on the ESP32	17
5.3	The Raspberry Pi 4	18
5.3.1	The Raspberry Pi Camera	19
5.3.2	Fingerprint sensor	23
6	Development Platform, IDEs, and Tools	23
6.1	Arduino IDE	23
6.2	PyCharm	24
6.3	Atom	25
7	Agile Project Management	26
8	AWS	28
8.1	AWS EC2	29
8.2	AWS IOT Core	30
8.3	Aws kinesis	31
8.4	Docker + AWS	33
8.4.1	What does it have to do with home automation	33
8.4.2	What is docker	34
8.4.3	What is docker compose	35

9	The important of UX design for website interface	37
9.1	Designing applications.....	37
10	Nodejs	39
10.1	Why MEAN ?	40
10.2	Express – routing	40
10.3	Mongodb Mongoose	41
10.4	User Authentication – Bcrypt	42
10.5	Handle bars HBS.....	44
10.6	HTML, CSS, JavaScript, jQuery and Flexbox.....	44
11	Voice recognition for IOT house System.....	46
11.1	Alexa	46
11.2	Google Assistant	47
12	IFTTT.....	48
13	Webhooks	49
14	System Integration.....	50
14.1	Overall Power consumption of System	51
14.2	Problems I encountered with integration and power	51
15	3D House Models and Technologies.....	52
16	[ISSUES] throughout project lifecycle	53
16.1	Pandemic	54
16.2	Raspberry Pi 4 Model B	54
16.3	My Laptop.....	55
17	Conclusion.....	55
18	Final Words	56
19	References	57
20	Images References	59

2 Summary

The rationale underpinning my decision to choose home automation as my FYP project is based on several factors: my current interest in home automation, my recent internship at Intel and my hobbies which included design structures and models, but most importantly I think it has to do with an article that I read in the Sunday Irish Times.

Initially, the objective of my project was to have a website hosted on a server where the user could access and control their house remotely which would, in turn, automate their day-to-day tasks. However, during the regular meetings that I had with my supervisor regarding my project, my perspective on my project changed, so much so, that the goal of the project was now geared towards Container technology, the use of Amazon's AWS to have my Home automation system accessible from anywhere in the world from any IoT device.

My system is composed of two development platforms: the ESP32 microcontroller and the Raspberry pi 4. The platforms host the following sensors which hold the responsibility of monitoring and controlling the house: Pi Camera, Fingerprint Sensor, multiple LEDs, and Servo motors, PIR ultrasonic sensor, smoke sensor, fingerprint sensor, piezo buzzer, motion sensor. Also, incorporated in my project is the use of AWS Kinesis which is used to stream facial recognition camera surveillance. In addition to my project, I have a website that is hosted inside an AWS EC2 instance which shows the operations of the house. Also, I implemented Docker container technology into my project as an experimental aspect. I will revisit this aspect further in my report.

The leading technologies I used include MQTT, HTTP, IFTTT, Webhooks, Web-sockets, JS, jQuery, AWS, Docker, Docker compose, the MEAN Stack and HTML/CSS. The MQTT Broker is hosted on AWS IoT Core. The micro-controller esp32 and the pi4 connect to the MQTT broker to allow live data communication which is visualized on the AWS IoT Core. This data is then showcased on the EC2 using NodeJS, Express and the AWS SDK.

The scope of my project encompasses the implementation of Agile tactics intending to make the execution of project possible, and this includes multiple trials and tests to ensure that my project works the way that I envisioned.

Finally, the data that I collected from the multiple trials and tests that I carried out was satisfactory because it showed that my project was fully functional and could be implemented correctly as I intended.

3 Introduction

As mentioned previously, the main inspiration for my project emanated from reading the weekly instalment of the Sunday Irish Times. Whilst reading the newspaper, I stumbled across an article entitled ‘House Hacks’ written by columnist India knight. The article detailed how the average homeowner can spend a minimum of two hours to a maximum of seven hours enslaved to house chores. When I read this, I was puzzled by the dimension of this figure and the reason why it was very high. The article further highlighted a solution that homeowners could reduce the time spent of domestic chores by almost 60 % when House automation strategies were implemented.

While doing further research into the topic, I came across these very interesting statistics from a paper entitled “Energy conservation through smart homes in a smart city” by Abhishek Bhati[4]. In the Publication, the author carried out a survey on the benefits of smart homes with varied responses from the interviewees shown in the graph below.

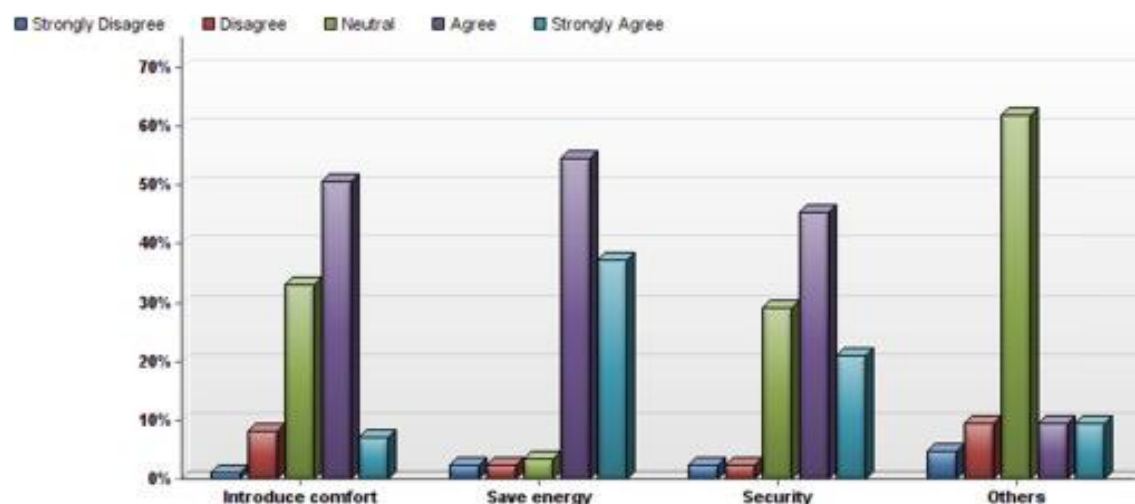


Figure 3-1 Population graph of benefits of Smart Homes 1

As can be seen from the graph, the respondents stated that home security and energy conservation had increased while that had implemented home automation strategies. This can also be seen here as well in another study that was taken. Pie-chart below shows the advantages of E-homes. E-homes are comfortable, convenient, and most importantly has various ecological benefits.[5]

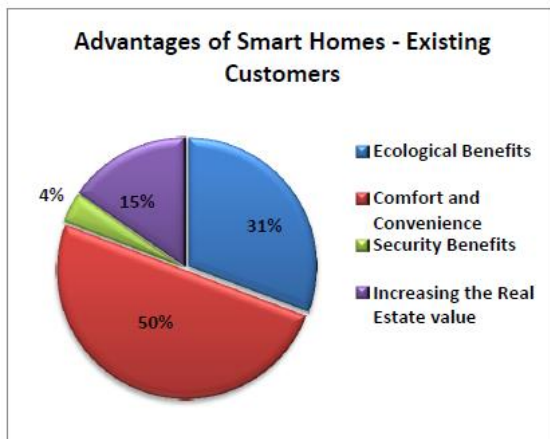


Figure 3-2 Population graph of benefits of Smart Homes 2

Also, It can be seen how the incredible energy conversation benefits E-homes can save power from 25% of a standard household's electricity consumption to about 50% in the case of a commercial building. The Record below gives a brief idea of annual savings.

Average annual savings on regular halogen bulbs			
Dimming	Electricity Savings	Average life of Bulbs	Annual Savings+
10%	10%	3 years	200
25%	22%	5 years	450
50%	35%	8-10 years	600

Figure 3-3 Population graph of benefits of Smart Homes 3

In recent years, Home automation has become an increasingly popular topic and has been used to perform tasks around the house that are either too arduous or tedious. Popular examples of home automation systems are Amazons's 3rd gen Alexa, Google Home, Kasa Smart WiFi Plug by TP-Link etc. These systems serve to integrate all the different IoT appliances in your house into one dedicated IoT hub.

The Amazon Echo (3rd Gen)- Smart speaker with Alexa is a voice-controlled speaker that works to connect your home's smart devices and can create a home that's fully automated.

The Echo works with Alexa Voice service to answers queries and reads audiobooks, news reports, and sports schedules.[1]

Premised on all these findings, I decided to take on my FYP as an IOT Smart Home automation system as I wanted to see what I could bring as my contribution to this sector of prevailing technology.

Although, I initially planned to implement something similar to my FYP, I decided to use the following microcontrollers available to me: ESP32 and Raspberry pi 4. I also opted to base my project on the House Automation system Hassio. Hassio utilizes Home Assistant the development platform Raspberry Pi (or another device) and turns it into a home automation hubs.

Hassio possesses many features like its optimization for embedded devices like Raspberry Pi, it is easy and seamless in terms of installation. It also incorporates the management of web interface integration into Home Assistant, Installation of popular add-ons such as Google Assistant, encryption and dynamic DNS.

For my FYP, instead of doing the bulk of the development work on the raspberry pi 4, I resorted to placing all my main functionalities on the ESP32, and the rest of the functionalities which was facial recognition camera surveillance and the fingerprint authentication of the homeowner was developed on the pi 4.

The objective of this project was then to enhance my knowledge and engage myself in the home automation topic and to become familiar and comfortable with new technologies that I would implement into my project. In order to do this, I researched the best protocols and technologies to use for my IoT Smart Home, drafted house model, drafted architectural diagrams and drafted flowchart diagrams.

I also wanted to challenge myself by figuring out how I could make my innate possible with a view to building my own device in a way that would help future homeowners in optimizing their day. I wanted to make sure after the completion of my project that the device to be constructed would be available for people to utilize and easy to understand. The scope of my project was to create a house model from scratch, which placed all of the technologies and sensor that I coded into the house. The user would then have access to the website which

allowed them to monitor and control their house and also allow them to view a real-time facial recognition camera surveillance all from the comfort of their IoT device and mobile.

Finally, the data that I collected from the multiple trials and tests that I carried out was satisfactory because it showed that my project was fully functional and could be implemented as I intended.

4 Chapter breakdown

The design and the development of my project are explained in detail in the subsequent sections. The following sections give greater insight into the methodologies and technologies used in this project.

Section 1 introduces project architecture with all the components associated with the development of the Home Automation System.

Section 2 serves as an informative section regarding the different Integrated Development platforms that were used during the project lifecycle, while section 3 delves into further details concerning the development platforms discusses how the project uses Agile tactics to managed the project.

Section 7 demonstrates the power of AWS in Home automation and the uses in my FYP.

Section 9 and 10 discusses User experience, UI and UX design, the MEAN Stack framework and the implementation of these features in my system.

Vocal commands and speech synthesis is covered in section 11 and section 12 and 13 advances the theme of technology, paying special attention to IFTTT and webhooks utilization in my FYP.

Section 14 captures the usefulness of the system integration, while section 15 discusses the methodology used for the development of the house models.

Section 16 details the issues experience during the project lifecycle, section 17 concludes the entire project.

5 Project Architecture

The architecture diagram for my final project consists of two major development boards. The ESP32 is the microcontroller that is the heartbeat of the system as it is responsible for hosting all the primary sensors that communicate with the other technologies features in my system. The Raspberry pi 4 holds the responsibility of Camera surveillance around the house and incorporates a security feature of my system, which is finger authentication. The MQTT broker which is hosted on AWS which is named AWS IoT Core is the glue that binds the two development boards together in terms of communication and ensures all data is sent and received by the publishers and subscribers of data.

The broker delivers the messages received from the development boards to a webpage that is hosted and encapsulated inside an AWS EC2 instance. The website includes user authentication and utilises the full mean stack. In addition to that, another device which I create was voice recognition using Google assistant API to control features within the ESP32. The esp32 that is connected to the AWS IOT Core MQTT broker, so it acts as a gateway for communication between the google assistant and the IoT broker. Finally, additional feature that I included was purely experimental as pictured in the diagram is using docker containers to encapsulate all the features I developed into separate containers and have the still communicate flawlessly using docker-compose.

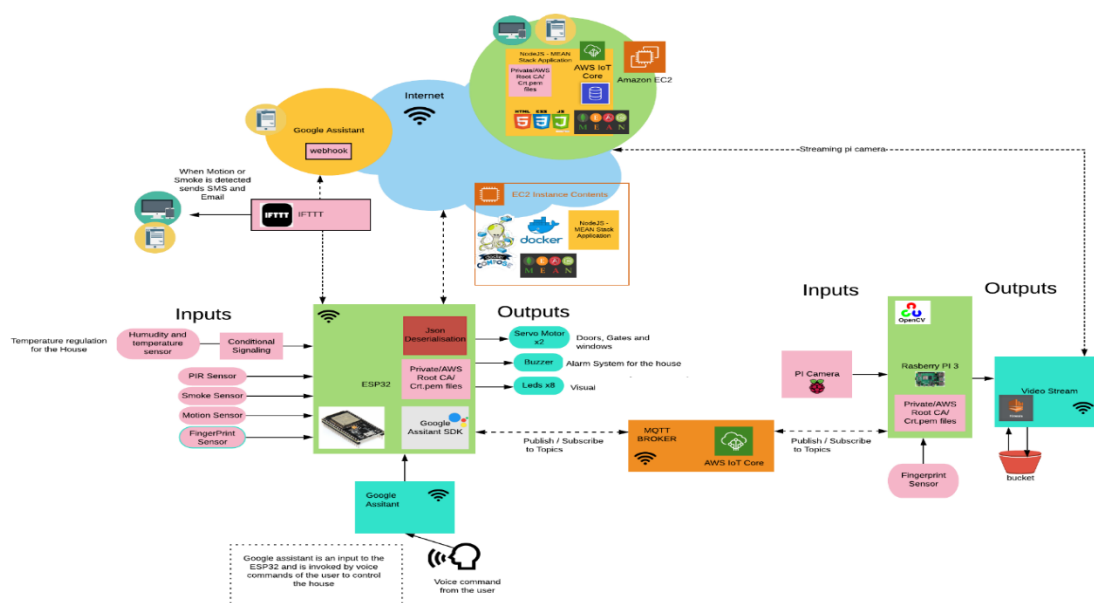


Figure 5 Architectural diagram

5.1 MQTT

MQTT (Message Queuing Telemetry Transport) is a machine-to-machine/"Internet of Things" connectivity protocol. It was designed as an extremely lightweight publish/subscribe messaging transport between multiple clients and a dedicated broker. [17]

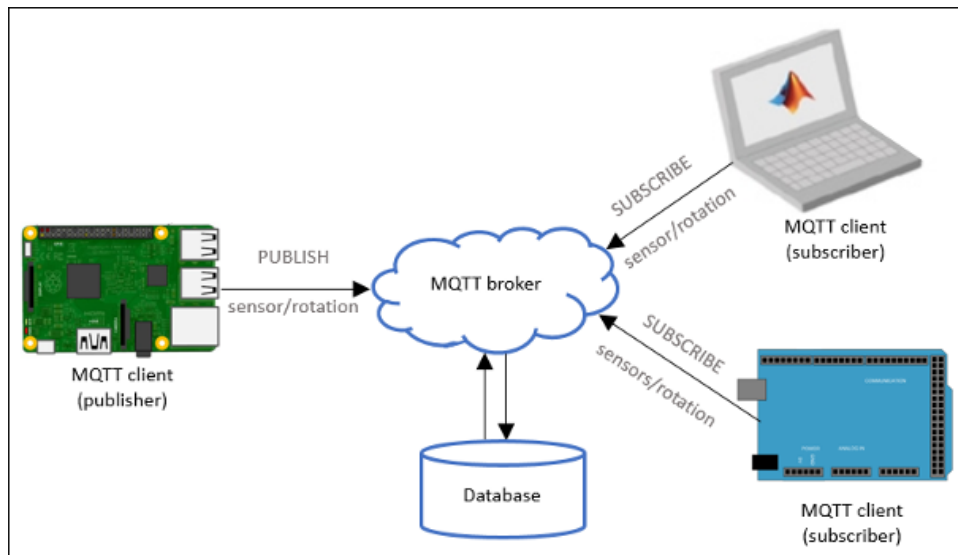


Figure 5-1 MQTT communication

An MQTT client can be any device from a small microcontroller to a powerful server inside a datacentre. The only requirement that is desired from a client is the ability to have the MQTT library running and connected to the specified MQTT broker over any network that is being used for the data communication transfer.

The MQTT broker is primarily responsible for receiving all messages, filtering them and deciding which client or entity available is interested in it. Then upon this clarification process, the broker then proceeds to send the messages to all subscribed clients. In addition to the broker's primary responsibility, another responsibility it holds is the authentication and authorisation of clients.

The MQTT protocol is based on top of TCP/IP, and both client and broker need to have a TCP

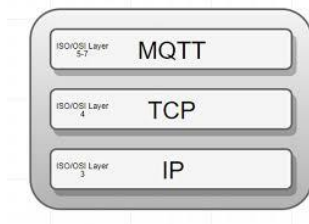


Figure 5-1 MQTT communication1

As MQTT is data inclined the use of IoT development is more suitable for its operation.

5.2 ESP32

As mentioned earlier in my architectural diagram, the ESP32 is a fundamental part of my project premised on the fact that it hosts all the sensors responsible for communication with the website interface via the MQTT broker.

ESP32-WROOM-32 is a powerful, generic Wi-Fi+BT+BLE MCU module that targets a wide variety of applications, ranging from low-power sensor networks to the most demanding tasks, such as voice encoding, music streaming and MP3 decoding which is capture below. [15]

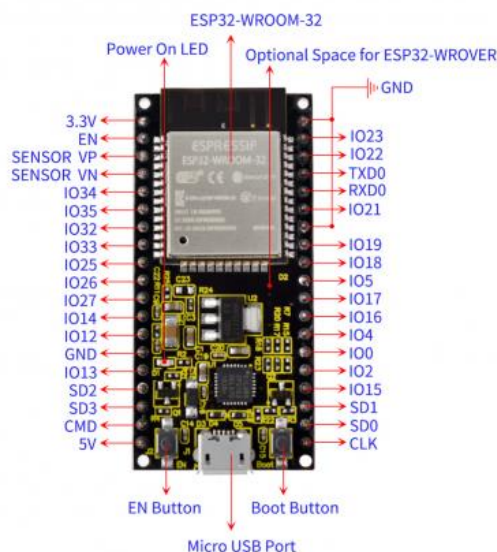


Figure 5-2 ESP32 pinout

The core of this module is the ESP32-D0WDQ6 chip*. The chip embedded is designed to be scalable and adaptive. Two CPU cores can be individually controlled, and the CPU clock frequency is adjustable from 80 MHz to 240 MHz.

The ESP32 runs on an ESP32-D0WDQ6 chip. Espressif System ESP32-D0WDQ6 chip Dual-Core Wi-Fi® Dual-Mode BLUETOOTH® SoC (System-on-Chip) is a single 2.4GHz Wi-Fi and Bluetooth combo chip designed for mobile, wearable electronics, and Internet-of-Things (IoT)

applications. [18] The ESP32 is powered through a USB power supply or the USB cable. The power supply provides between 2.5V and 5V and provides at least 500mA current. During testing, it is recommended to use either 5V. Battery packs with this specification work as well.

I refer to the home automation system; the development board runs the master script, which I created and begins communication with the AWS IOT core MQTT Broker. The development board accepts requests by continually polling the main loop. When a request is received; it is processed. Subsequently, the ESP32 responds telling the user what action has been performed.

5.2.1 Sensors on the ESP32

As alluded to, the ESP32 is the pulse of the home automation system because it hosts the majority of the sensors on the boards, which are outlined as follows:

- Smoke sensor – used a security feature to alert the user via text message when the smoke has been detected in the house and also sends a message on the specific topic alerting the MQTT broker which then supplies this data to the website, providing an update for the user
- Motion sensor – sends text and email notifications to the user when an intruder has entered the house. This is also updated on the webpage via MQTT
- Servo motor – is used for the door, windows functionality to simulate a real-life house and sends integer values to update the website via AWS IoT core.
- DHT11 temperature sensor – used to track the temperature of the house and send the values to the MQTT broker to update the webpage
- LEDs – are used as a visual source to display the internal actions of the sensor and also used as lights in the house.

The connection of how the ESP32 connects to the sensor is captured below.

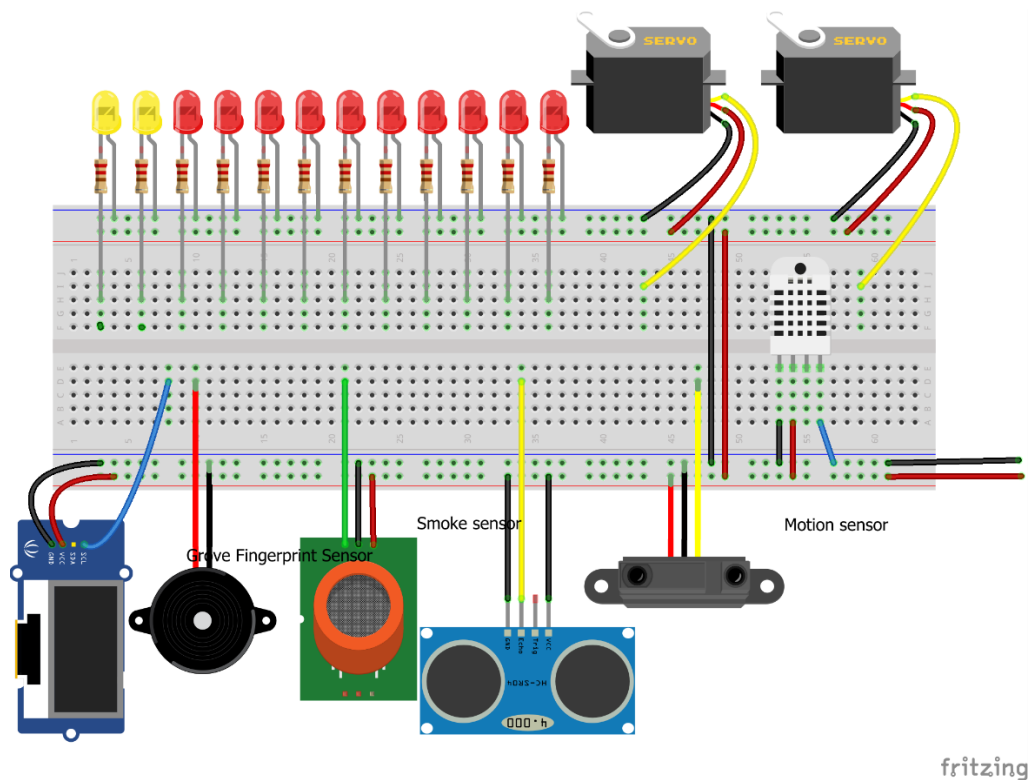


Figure 5-2 esp32 connections

5.3 The Raspberry Pi 4

The Raspberry Pi is a series of small single-board computers developed in the United Kingdom by the Raspberry Pi Foundation. 16

Raspberry Pi 4 Model B processor is a 1.5 GHz 64-bit quad-core ARM Cortex-A72 processor, equipped with on-board 802.11ac Wi-Fi, Bluetooth 5, full gigabit Ethernet, two USB 2.0 ports, two USB 3.0 ports, and dual-monitor support via a pair of micro HDMI (HDMI Type D) ports for up to 4K resolution.

The Pi 4 is also powered via a USB-C port, enabling additional power to be provided to downstream peripherals,

I used the raspberry pi 4 in my project for fingerprint authentication of the house, and also I used it to implement the feature of the facial recognition surveillance camera.

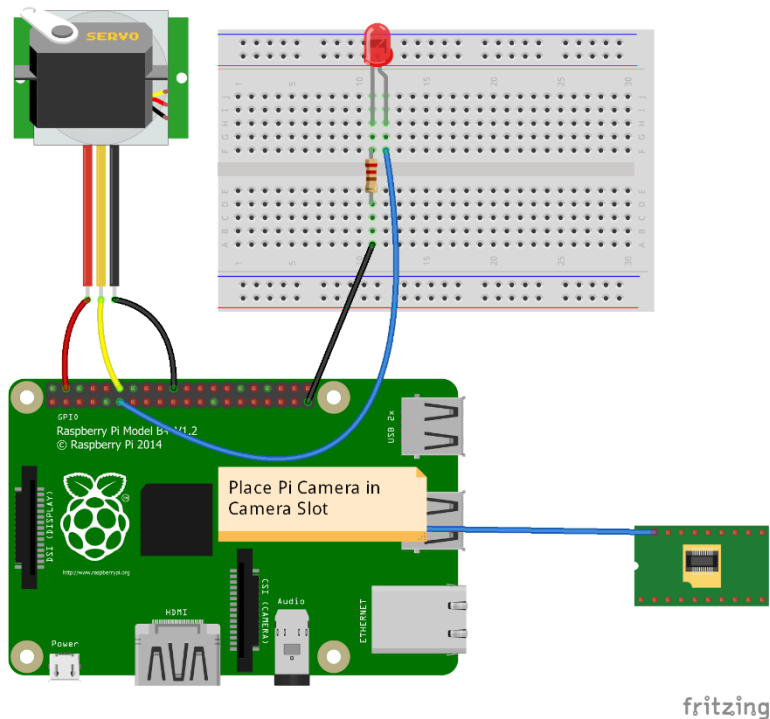


Figure 5-2 Raspberry pi 4 connections

5.3.1 The Raspberry Pi Camera

The Raspberry Pi Camera v2 is a high quality 8 megapixel Sony IMX219 image sensor custom designed add-on board for Raspberry Pi, featuring a fixed focus lens. It's capable of 3280 x 2464 pixel static images, and also supports video. It attaches to Pi by way of one of the small sockets on the board upper surface and uses the dedicated CSI interface, designed especially for interfacing to cameras. It connects to Raspberry Pi by way of a short ribbon cable. In terms of still images, the camera is capable of 3280 x 2464 pixel static images, and also supports 1080p30, 720p60 and 640x480p90 video.[19]



Figure 5-2 Pi Camera

The raspberry pi camera was used in the module to control the door as a security feature. This feature used facial recognition which uses the user's face as entrance into the house by comparing to a trained data set of the user's face using the machine learning algorithm the Haar Cascade. The Haar Cascade was implemented using OpenCV for the images processing and import gpio into the raspberry pi to control the feature in the house.

A facial recognition system is a technology capable of identifying or verifying a person from a digital image or a video frame. It is described as a Biometric Artificial Intelligence based application that can uniquely identify a person by analysing patterns based on the person's facial textures and shape.

Some face recognition algorithms identify facial features by extracting landmarks, or features, from an image of the subject's face.

Other algorithms normalise a gallery of face images and then compress the face data, only saving the data in the image that is useful for face recognition. A probe image is then compared with the face data.

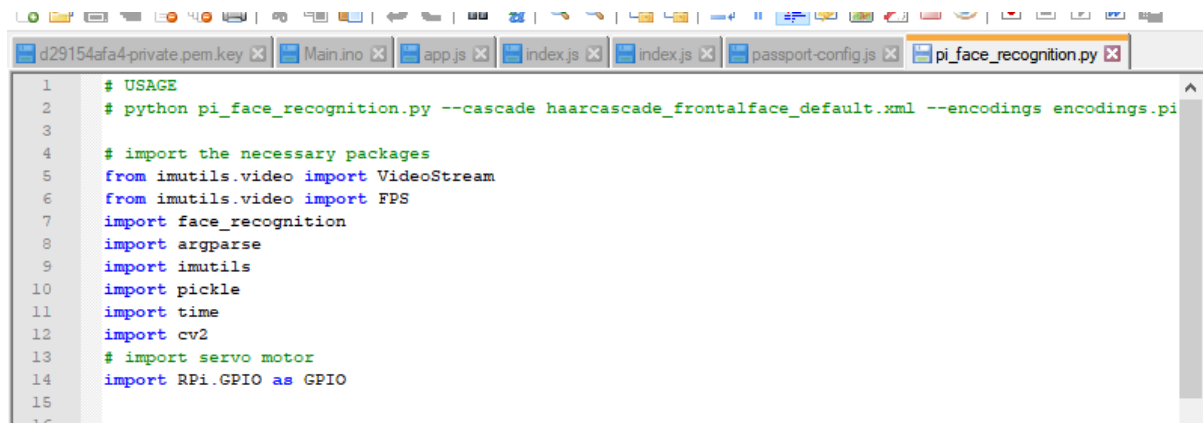
The Haar Cascade is a machine learning object detection algorithm used to categorise objects it has been trained for in a video or image.

That is because unlike humans, computers are machines, and they are designed to interpret an image as a matrix. In the matrix, the colour in each matrix element corresponds to a number. Depending on the number that is associated with the element gives it its colour. The elements in the matrix can be thought of blocks as displayed below in the diagram.

It consists of an array containing each pixel intensity value. Because facial recognition deals are working with grayscale images, the values range from 0 (White) to 255 (Black). The method of using haar cascades for facial detection uses only the black and white version of the image for detection.

When training a Haar Cascade classifier, one needs a set of positive images; these images are one that contains the item or person you are trying to detect. A text file that contains the path to the image and the x,y,w,h positions where the item you want to detect is found in the image is also required. If more than one item is present you must also add its x,y,w,h values. For facial recognition to be successful, I then needed to collect double the number of negative images as positives. A negative image is an image that does not contain the item you want to detect. And then the training of images is done with the Haar Cascade. This is done by using the open cv function `opencv_traincascade` with the needed parameters for that cascade.

I developed a facial recognition script which is captured below in the code extract.



```

1  # USAGE
2  # python pi_face_recognition.py --cascade haarcascade_frontalface_default.xml --encodings encodings.pickle
3
4  # import the necessary packages
5  from imutils.video import VideoStream
6  from imutils.video import FPS
7  import face_recognition
8  import argparse
9  import imutils
10 import pickle
11 import time
12 import cv2
13 # import servo motor
14 import RPi.GPIO as GPIO
15
16

```

The code imports the packages necessary for facial recognition.

```

73
74 # loop over frames from the video file stream
75 while True:
76     # grab the frame from the threaded video stream and resize it
77     # to 500px (to speedup processing)
78     frame = vs.read()
79     frame = imutils.resize(frame, width=500)
80
81     # convert the input frame from (1) BGR to grayscale (for face
82     # detection) and (2) from BGR to RGB (for face recognition)
83     gray = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)
84     rgb = cv2.cvtColor(frame, cv2.COLOR_BGR2RGB)
85
86     # detect faces in the grayscale frame
87     rects = detector.detectMultiScale(gray, scaleFactor=1.1,
88     minNeighbors=5, minSize=(30, 30),
89     flags=cv2.CASCADE_SCALE_IMAGE)
90
91     # OpenCV returns bounding box coordinates in (x, y, w, h) order
92     # but we need them in (top, right, bottom, left) order, so we
93     # need to do a bit of reordering
94     boxes = [(y, x + w, y + h, x) for (x, y, w, h) in rects]
95
96     # compute the facial embeddings for each face bounding box
97     encodings = face_recognition.face_encodings(rgb, boxes)
98     names = []
99
100    # loop over the facial embeddings
101    for encoding in encodings:
102        # attempt to match each face in the input image to our known
103        # encodings
104        matches = face_recognition.compare_faces(data["encodings"],
105        encoding)
106        name = "Unknown"
107
108        # check to see if we have found a match
109        # inside do the servo motor action
110        if True in matches:
111            # find the indexes of all matched faces then initialize a
112            # dictionary to count the total number of times each face
113            # was matched
114            matchedIdxs = [i for (i, b) in enumerate(matches) if b]
115            counts = {}
116
117            # loop over the matched indexes and maintain a count for

```

Figure 5-2 Facial recognition code

The true loop is where all the main work is done. The image begins the process by the frames per second count from the pi camera stream. The size of the stream is resized to 500px to speed up the processing time, and then the image is converted to grayscale. It then loads up the Haar Cascade file into a variable called detector. The parameters we pass on to `detector.detectMultiScale` are (gray, scaleFactor, minNeighbors, Flag, minSize). `scaleFactor` refers to how much the image is scaled up while carrying out detection. `minNeighbors` is the amount of space (neighbourhood) there is between each detection area of the given images. The Flag variables representing the usage of the version of haar. `minSize` is the minimum size possible for the given object size, which is the image if the Objects are smaller than the variable they are disregarded.

After these processes have been complete, the output looks like the following that was a screen capture from my stream.

Unfortunately, due to an unprecedented even, my board fried, and I was no longer able to continue the development and real-life testing on my board utilising these new technologies. I explain in greater detail in the section.

I further developed this feature by implementing it on the cloud using technologies such as AWS Kinesis, and this would implement live stream facial recognition camera stream on AWS which would stream to my website which was also hosted on an AWS EC2 instance. I go into further detail regarding this topic in the section entitled AWS.

5.3.2 Fingerprint sensor

The fingerprint sensor was initially developed on the esp 32, and It was quite a tedious process, instead, I decided to opt to do the development work on the raspberry pi. However, I began to experience various issues such as the facial recognition script taking up complete dominance of the raspberry pi, thus not allowing the fingerprint sensor to run, so I decided to shift the development of the fingerprint sensor to the esp32. I used a voltage converter to get rid of the voltage issue that initially erased with developing on the esp 32 fingerprint feature on the raspberry pi. since the esp 32 microcontroller hardware responses were instantaneous and not simulated like pi 4, it made more sense to place the fingerprint sensor on the esp 32

6 Development Platform, IDEs, and Tools

Here in this chapter, I delve into the IDEs that I used for the development of my final year project concerning what each IDEs serves to do in my project

6.1 Arduino IDE

The Arduino Integrated Development Environment - or Arduino Software (IDE) - contains a text editor for writing code, a message area, a text console, a toolbar with buttons for standard functions and a series of menus. It connects to compatible hardware to upload programs and communicate with them. This IDE was used to program the functionality of the Home automation system. This included building the web server, setting up the MQTT communication using the AWS IoT core SDK and libraries, Coding integrating all the sensors, setting up HTTP communication for IFTTT and webhooks used. [2]

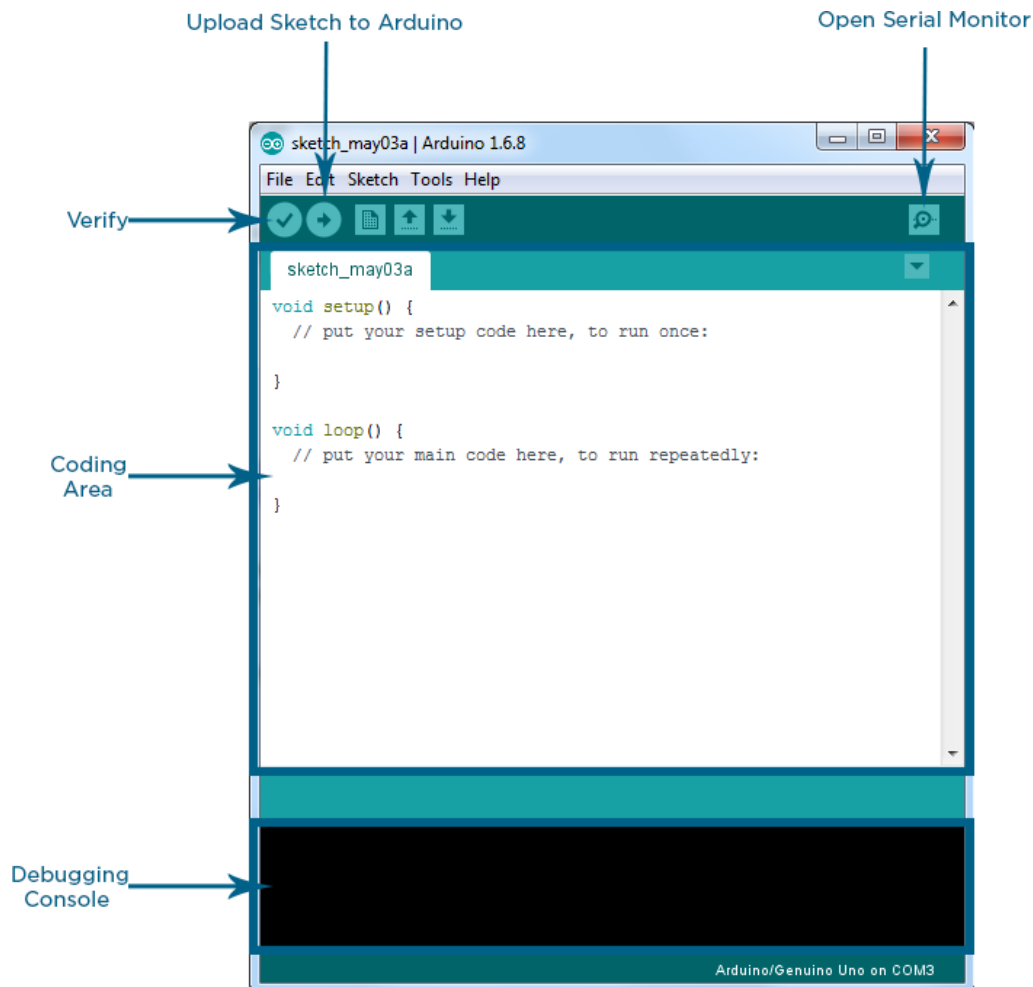


Figure 6-1 Arduino IDE

6.2 PyCharm

PyCharm is an integrated development environment (IDE) used in computer programming, specifically for the Python language, developed by the Czech company JetBrains. It provides code analysis, a graphical debugger, an integrated unit tester, integration with version control systems, and supports web development with Django as well as Data Science with Anaconda.[3] PyCharm was used as a development platform for the raspberry pi 4. On this platform, code development for the elements inside the home automation system, which include the facial recognition pi camera and the fingerprint system was established. Also, the MQTT communication for the raspberry pi was established in this IDE using the AWS IoT Core SDK.

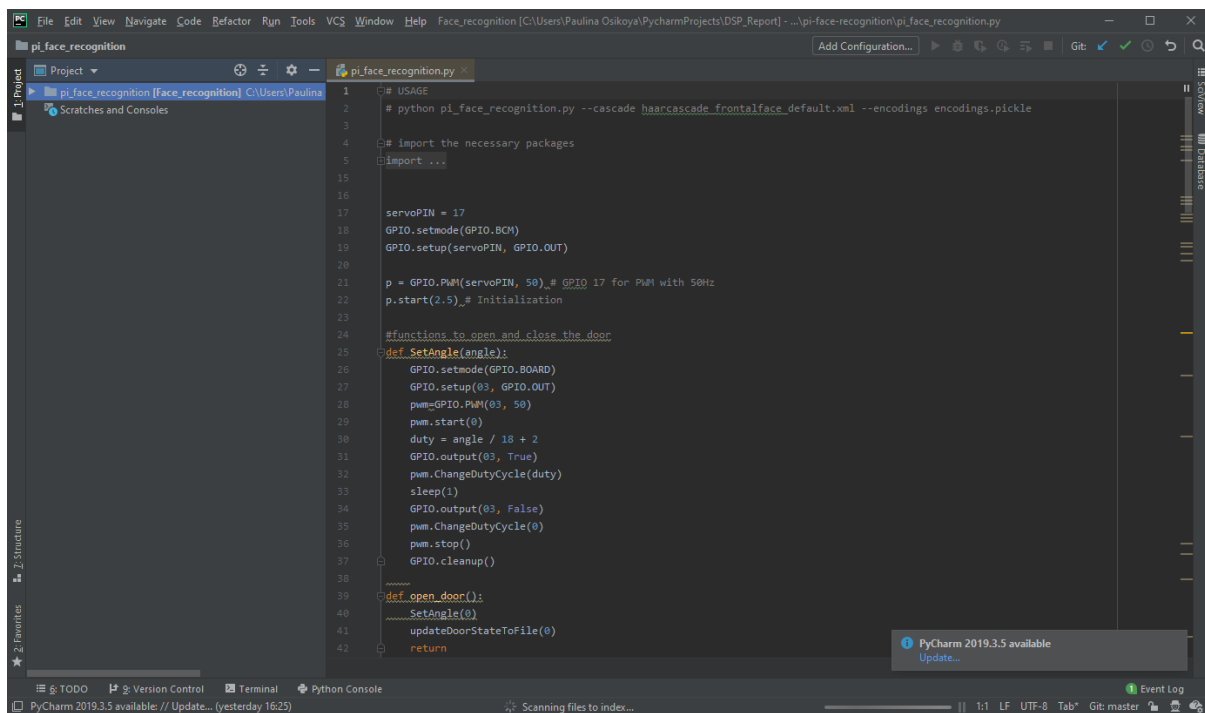


Figure 6-2 Pycharm IDE

6.3 Atom

Atom is a free and open-source powerful IDE for modern JavaScript development. Atom enables a text editor text and source code editor for macOS, Linux, and Microsoft Windows with support for plug-ins written in Node.js, and embedded Git Control, developed by GitHub. Atom is based on Electron (formerly known as Atom Shell), a framework that enables cross-platform desktop applications using Chromium and Node.js. [21] Atom was used mainly for all the JS, HTML & CSS files hosted on the HTTP web server hosted by the AWS EC2 instance. Inside the IDE, the software was built in such a way where I could automatically upload the web development code that I wrote to the HTTP server (via sftp) over Wi-Fi.

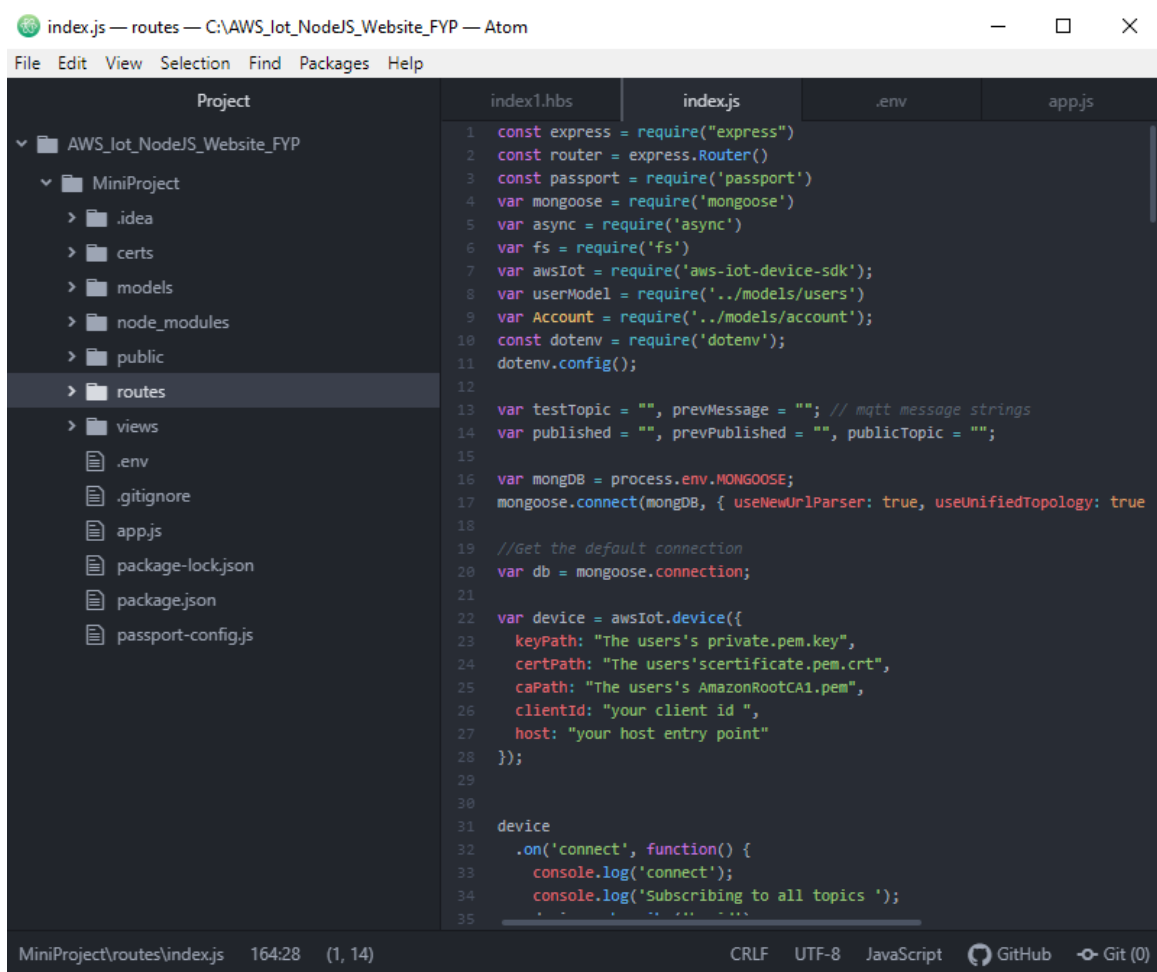


Figure 6-2 Atom IDE

7 Agile Project Management

For my project, I evaluated my successes and progression using Agile Project Management (APM).

APM focuses on outputting quality products by monitoring the development process and carrying trials and tests continuously throughout the life of the project. APM implements scrum as a framework and is typically carried out in teams of people, who have various roles in the product development process. These people include scrum master, team member and product owner. Doing an individual project, I had to take on these various roles, which were both exciting and challenging.

I chose to use this approach for my project because I knew it would be the best way to examine my project's development, and it would also be a useful tool for monitoring success and failure.

During the formation of my project, I kept and made many product backlogs which were essential to-do lists in agile terminology, of objectives I had to meet in my project.

Trello was the online Kanban board that was used to organize my different tasks into “cards” and the into lists which are capture below. In Trello, those cards can be tasks, notes, projects, shared files, or anything else that helps your team work together.

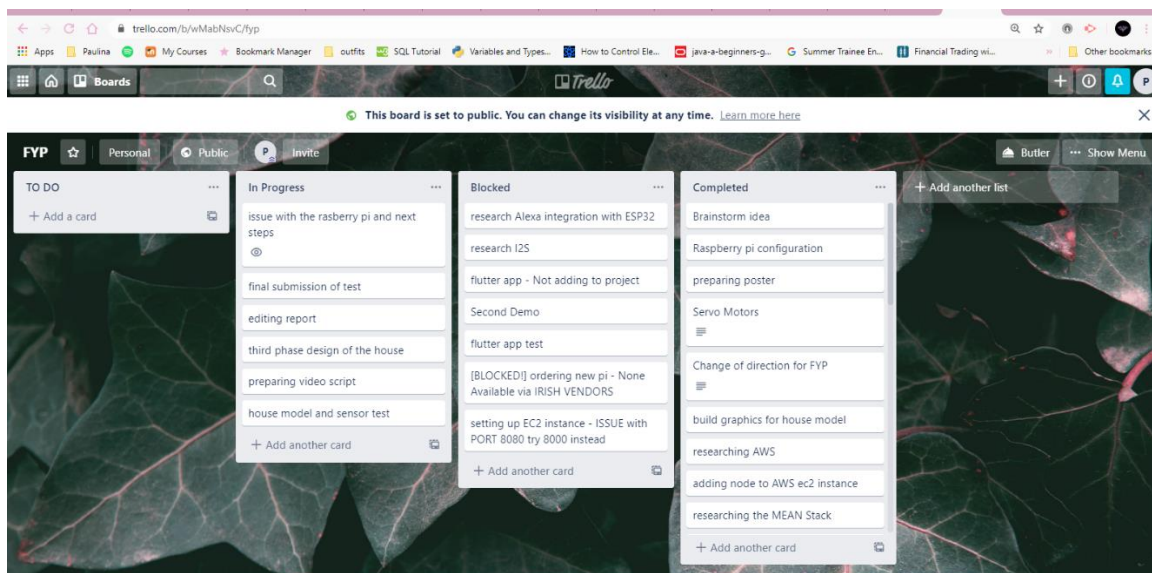


Figure 7-2 Trello Backlog

GitHub was the online Git repository hosting service where all of my code development and diary logs was stored. GitHub provides a Web-based graphical interface. It also provides access control and several collaboration features, such as wikis and basic task management tools for every project.[6]

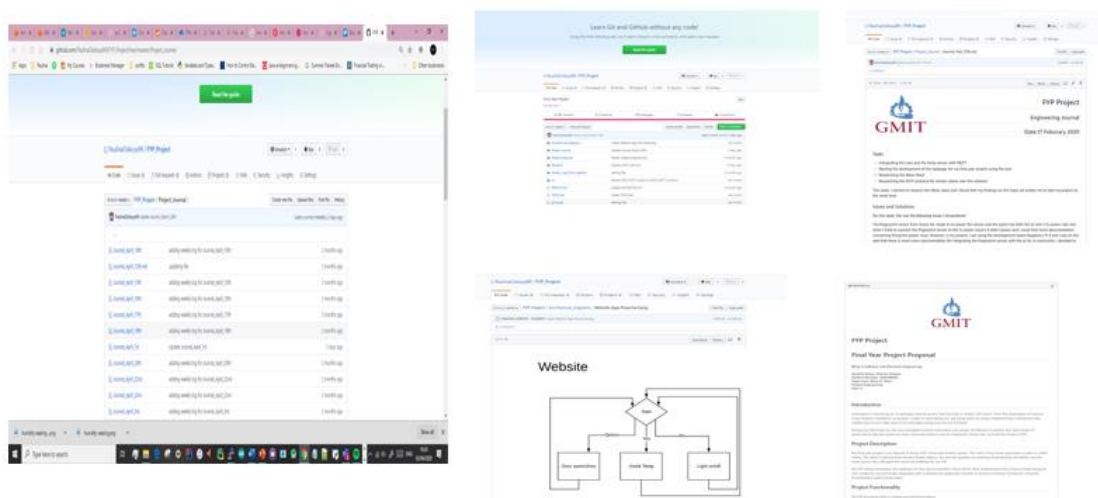


Figure 7-2 Github

In my FYP, I split the workload into epics and story points to ensure the completion of my project. Stories, also called “user stories,” are short requirements or requests written from the perspective of an end-user. Epics are large bodies of work that can be broken down into a number of smaller tasks (called stories). Initiatives are collections of epics that drive toward a common goal.[7]

I believe that using APM was indeed very useful because It made me stay on track with my project objectives as I was able to see my project develop in incremental steps by implementing multiple trials and tests.

8 AWS

One of the main aspects of my project was using the AWS technologies to carried actions in my project. Before the decision of hosting my entire on the cloud, I decide to opt for more native operation and protocols for microcontrollers and development boards. These operations included :

Hosting the MQTT and HTTP server on the raspberry pi 4 on localhost.

Using I2C for the communication between the ESP32 and raspberry pi 4 that shared sensors.

Hosting my website interface on localhost port 8000

Using and creating a separate server for my streaming functionality for camera surveillance.

Having formalised my plan and the sprint regarding how I was going to pull off my project, after the first demonstration of my final year project, I had several meetings with my supervisor. During these sessions, my perspective on my entire project began to shift. Also, during that time, I was undertaking a module called Cloud Computing which I took a liking to, and this could be seen in the assignment that I carried out in that module. Combined with the meetings with my supervisor and my new interest in cloud computing, I decided to take a leap of faith and completely redesign my proposal for my project. The functionality of my project would still stand, but the architecture of technologies was now different. Now, I utilised technologies such as AWS, Docker, IFTTT, webhooks, Restful APIs and Stacks and frameworks such as the MEAN Stack.

In regard to AWS, the services I used for my project were the following:

- AWS IoT Core
- AWS EC2
- AWS Kinesis
- AWS recognition
- AWS SNS

Below, I expand on my experience with AWS and the mention features of AWS that I incorporate

8.1 AWS EC2

Amazon Elastic Compute Cloud (Amazon EC2) is a web service that provides secure, resizable compute capacity in the cloud. It is designed to make web-scale cloud computing easier for developers. Amazon EC2's simple web service interface allows you to obtain and configure capacity with minimal friction. It provides you with complete control of your computing resources and lets you run on Amazon's proven computing environment.[22]

Mention it is like a virtual machine

The EC2 instance was utilized as a virtual machine for my Home Automation System. Inside my system, I hosted the MEAN website that I created as pictured below.

To set up the EC2, I had to create an appk file from a pem key that I obtained when creating a key pair for the instance. This file allowed me to connect to putty the ssh terminal to do development work.

After configuring the instance, I had to install the required environment variable for the install and to do a fresh install of nodeJs and the MongoDB.

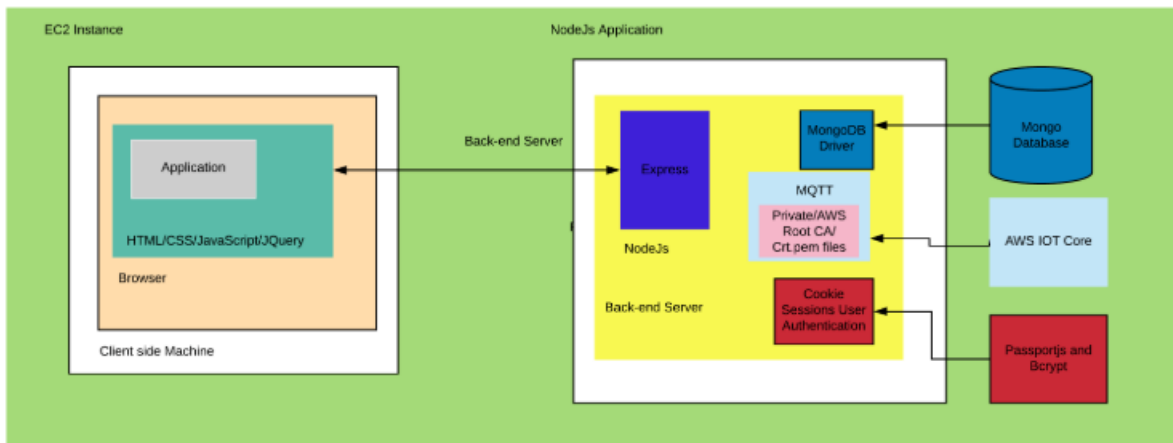


Figure 8-1 AWS EC2 Instance

In configuring the ec2 instance, I had to set up the networking for the virtual machine which involves setting up the inbound and outbound rules to make sure it's internal Ip address was mapped to port 8000 which is the port of which, the website was mapped to.

The setup and deployment of the EC2 was seamless.

8.2 AWS IoT Core

AWS IoT Core is a managed cloud service that lets connected devices easily and securely interact with cloud applications and other devices. AWS IoT Core can support billions of devices and trillions of messages and can process and route those messages to AWS endpoints and to other devices reliably and securely.[23]

The AWS IoT message broker implementation is based on MQTT version 3.1.1, but it differs from the specification in these ways:

AWS IoT Core supports MQTT Quality of Service (QoS) levels 0 and 1 only. AWS IoT Core does not support publishing or subscribing with QoS level 2. When QoS level 2 is requested, the AWS IoT message broker does not send a PUBACK or SUBACK.

In AWS IoT Core, subscribing to a topic with QoS level 0 means that a message is delivered zero or more times. A message might be delivered more than once. Messages delivered more than once might be sent with a different packet ID. In these cases, the DUP flag is not set when responding to a connection request, the message broker sends a CONNACK message.

This message contains a flag to indicate if the connection is resuming a previous session. The message broker uses the client ID to identify each client. The client ID is passed in from the client to the message broker as part of the MQTT payload.

AWS was an element that allowed the communication from website hosted inside the EC2 to both the raspberry pi 4 and the ESsp32.

In order to establish this communication, I had to configure the AWS IoT Core, which includes creating an IoT thing on IoT core. I then created and activated policies and certificates to enable the broker to be usable by the development platforms.

This can be seen here where I manually configure the connection to the MQTT broker on the esp32.

```
#include <AWS_IOT.h>
#include "ArduinoJson.h"
AWS_IOT aws;

char HOST_ADDRESS[]="a3v4cms69ln0e7-ats.iot.us-east-1.amazonaws.com";
char CLIENT_ID[]="add your client id here";
char TOPIC_NAME[]="Add your topics here";
```

Figure 8-2 AWS IoT core connect

8.3 Aws kinesis

Amazon Kinesis is an Amazon Web Service (AWS) for processing big data in real time. Kinesis is capable of processing hundreds of terabytes per hour from high volumes of streaming data from sources such as operating logs, financial transactions and social media feeds.[24]

This was a feature that I took upon myself to develop, which was suggested to me by my supervisor. At the time when I was developing the Kinesis feature, there was not and still is not all of the documentation on how to build such a thing, so I had to primarily get my knowledge from AWS keynote video on AWS Kinesis and the AWS documentation. During the beginning of the feature, I was developing this on my raspberry pi and intended to add it to the array of elements I had already implemented into my home automation system but unfortunately, my raspberry pi fried and this is further detailed in the section ISSUES.

Developing I exposed to different AWS services such as SNS Kinesis Recognition, and Lambda,

First, I had to set up the AWS CLI which is the AWS command-line interface this involves me acquiring the secret keys and creditable given by AWS

During this process, I set up the pi camera configure it to work as a module with the raspberry pi 4

The next aspect that I set out to do was set up creating the kinesis video stream. I found this aspect rather challenging to implement as during this project was the first time I dealt with AWS, and I was not accustomed to the way one calls and instantiate and calls functions and flags within the AWS CLI.

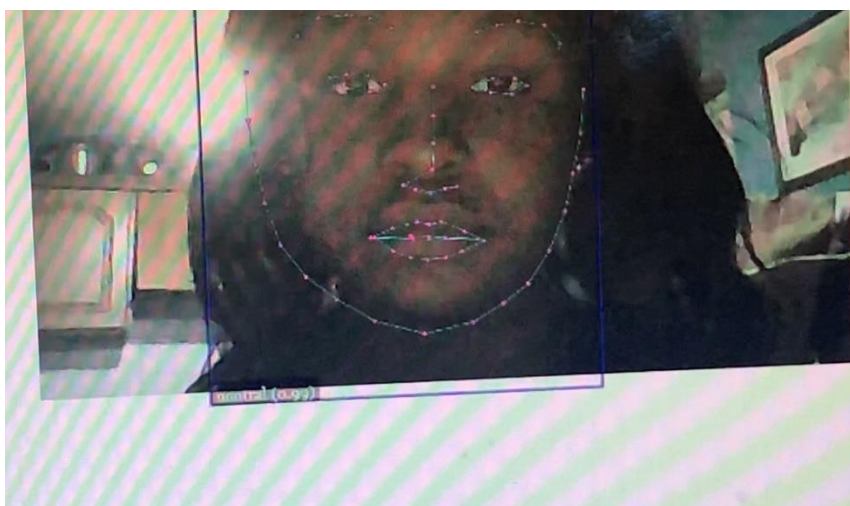
Using this command:

```
aws kinesisanalytics create-stream --stream-name PaulinasStream --data-retention-in-hours 2
```

Figure 8-3 Creating Kinesis Stream

The next aspect was streaming to the AWS Kinesis, which I did using a Producer which is used to create and send data to the stream. And then I

cloned the C++ Producer SDK onto my raspberry pi to allow the stream to begin the stream to AWS rekogiton which I was able to achieve as captured below;



AWS Rekognition is a service that lets developers working with Amazon Web Services add image analysis to their applications. With AWS Rekognition, your apps can detect, remember and recognise objects, scenes, and faces in images.

This was the last feature I was able to implement before my raspberry pi fried. But afterwards, I researched how I could create collections which are datasets that used to train the stream using AWS rekogintion. And I also looked in using MQTT to have the camera stream subscribe to a topic on the broker, and when a specific condition was met, it would invoke a lambda function.

AWS Lambda is Amazon's answer to Serverless Infrastructure or Functions as a Service (FaaS). It allows you to run code in the cloud, without having to manage servers and other infrastructure.

The function which would make a subscription to the SNS topic. A subscription can be an HTTP request (webhook), SMS, Lambda function, or via email. Anything that is published to this SNS Topic will be sent to the provided email address. Which is configured in the AWS CLI

8.4 Docker + AWS

Another aspect of my project that I worked tirelessly was integrating Docker into my home automation system whilst utilising AWS.

How I achieved this feature was by AWS EC2 as a machine to host my development work, and Docker compose to spin up the container and create the network between the containers

8.4.1 What does it have to do with home automation

Containers may not sound like the place where Home Automation will live, but Docker Containers house applications in an isolated environment. For my IoT smart home, I separate my Website interface, the database, the MQTT broker into their own isolated environment,

which is pictured below.

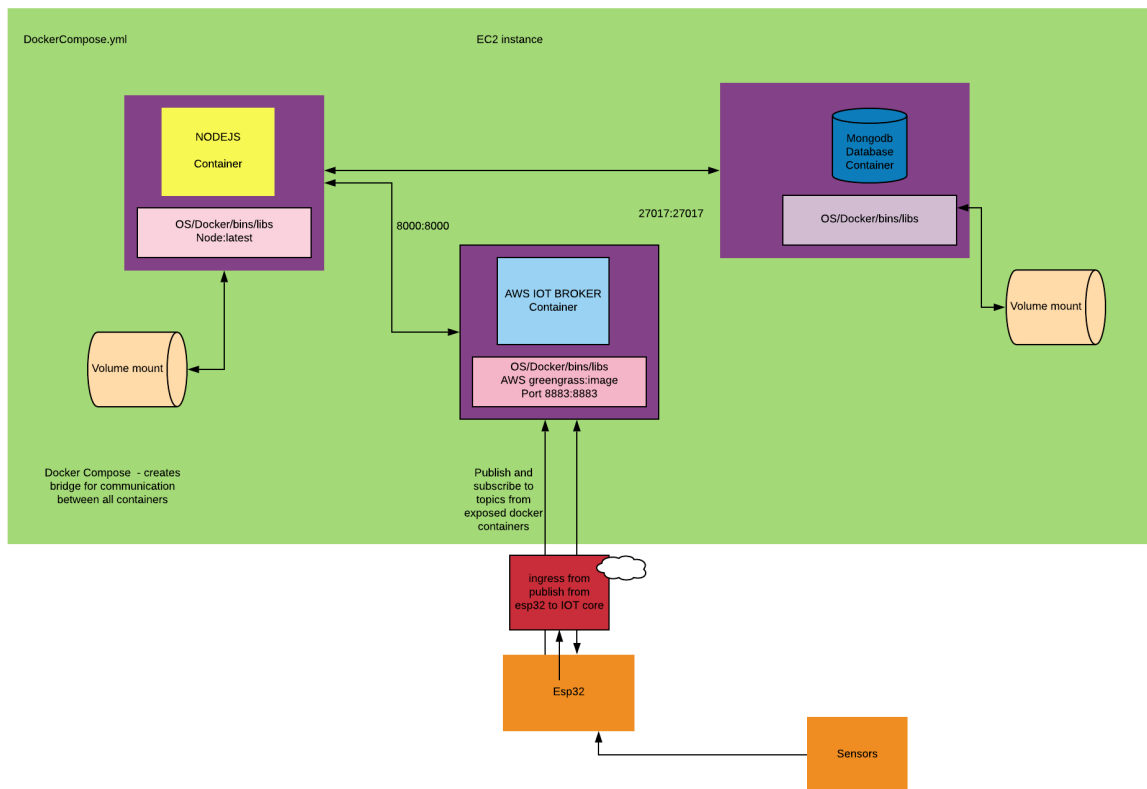


Figure 8-4 Docker Container diagram

8.4.2 What is docker

What Docker aims to simplify is the acceleration of workflows for applications. In addition to this it allows the developer creates their applications and then deploy and ship their applications them with ease. Docker utilises containerization of application that It host to manage the application. The containers are lightweight entity and allow for portability enable them to run on any host. Docker features automatic releases a new builds of the project every time a new commit is pushed to the master branch, hence why Docker allows for continuous integration of new components for the applications I had used Docker previously before during my internship but I chose to use it for this project as it is rapidly growing in popularity in this present day and it is instrumental in today's industry as companies focus on releasing patches and updates to the user as soon as they are ready to their systems and application. Based on this fact, I thought it was appropriate to incorporate this feature as if I decide to add more features to my project. I could do so without disrupting the current project that the user

is accessing. Also, the added advantage of using Kubernetes, the orchestration of Docker containers, which replica set feature which will spin the docker container back to its desired state if it were to crash.

```
ubuntu@ip-172-31-94-16:~$ docker build -t home-automation-image .
Sending build context to Docker daemon 140.4MB
Step 1/6 : FROM ubuntu:18.04
18.04: Pulling from library/ubuntu
23884877105a: Pull complete
0c38caa0f5b9: Pull complete
2910811b6c42: Pull complete
86505266dcc6: Pull complete
Digest: sha256:3235326357dfb65f1781dbc4df3b834546d8bf914e82cce58e6e6b676e23ce8f
Status: Downloaded newer image for ubuntu:18.04
--> c3c304cb4f22
Step 2/6 : RUN apt-get update && apt-get -y install apache2
--> Running in 52882bef152d
Get:1 http://security.ubuntu.com/ubuntu bionic-security InRelease [88.7 kB]
Get:2 http://security.ubuntu.com/ubuntu bionic-security/universe amd64 Packages [839 kB]
Get:3 http://archive.ubuntu.com/ubuntu bionic InRelease [242 kB]
Get:4 http://security.ubuntu.com/ubuntu bionic-security/restricted amd64 Packages [44.6 kB]
Get:5 http://security.ubuntu.com/ubuntu bionic-security/multiverse amd64 Packages [8213 B]
Get:6 http://security.ubuntu.com/ubuntu bionic-security/main amd64 Packages [88.7 kB]
Get:7 http://archive.ubuntu.com/ubuntu bionic-updates InRelease [88.7 kB]
Get:8 http://archive.ubuntu.com/ubuntu bionic-backports InRelease [74.6 kB]
Get:9 http://archive.ubuntu.com/ubuntu bionic/main amd64 Packages [1344 kB]
Get:10 http://archive.ubuntu.com/ubuntu bionic/multiverse amd64 Packages [186 kB]
Get:11 http://archive.ubuntu.com/ubuntu bionic/universe amd64 Packages [11.3 MB]
Get:12 http://archive.ubuntu.com/ubuntu bionic/restricted amd64 Packages [13.5 kB]
Get:13 http://archive.ubuntu.com/ubuntu bionic-updates/restricted amd64 Packages [59.0 kB]
Get:14 http://archive.ubuntu.com/ubuntu bionic-updates/universe amd64 Packages [1372 kB]
Get:15 http://archive.ubuntu.com/ubuntu bionic-updates/multiverse amd64 Packages [12.6 kB]
Get:16 http://archive.ubuntu.com/ubuntu bionic-updates/main amd64 Packages [118 kB]
Get:17 http://archive.ubuntu.com/ubuntu bionic-backports/universe amd64 Packages [7671 B]
Get:18 http://archive.ubuntu.com/ubuntu bionic-backports/main amd64 Packages [886 B]
Fetched 17.8 MB in 3s (6790 kB/s)
Reading package lists...
```

Figure 8-4 Deploying Docker image

8.4.3 What is docker compose

Compose is a tool for defining and running multi-container Docker applications. With Compose, you use a YAML file to configure your application's services. Then, with a single command, you create and start all the services from your configuration. Compose works in all environments: production, staging, development, testing, as well as CI workflows[25]

Compose has a three-step process which involves defining your app's environment with a Dockerfile so it can be reproduced anywhere and sequentially. Outlining the services that make up your app in a file called

docker-compose.yml so they can be run together in an isolated environment. And finally Running the command docker-compose up and Compose starts and runs your entire app.

Docker Compose sets up a single network for your application(s) by default, adding each container for service to the default network[26]. Containers on a single network can reach and discover every other container on the network which is captured below.

Examples

List all networks

```
$ sudo docker network ls
NETWORK ID          NAME                DRIVER              SCOPE
7fca4eb8c647        bridge              bridge              local
9f904ee27bf5        none                null                local
cf03ee007fb4        host                host                local
78b03ee04fc4        multi-host          overlay             swarm
```

Figure 8-4 Docker compose

```

1618 drwxr-xr-x 3 root root 4.0K Apr 27 18:47 ..
256441 -rw----- 1 ubuntu ubuntu 2.1K Apr 28 19:43 .bash_history
256079 -rw-r--r-- 1 ubuntu ubuntu 220 Apr 4 2018 .bash_logout
256077 -rw-r--r-- 1 ubuntu ubuntu 3.9K Apr 27 19:13 .bashrc
256097 drwx----- 3 ubuntu ubuntu 4.0K Apr 28 18:38 .cache
256782 drwx----- 3 ubuntu ubuntu 4.0K Apr 27 19:14 .config
256361 -rw-rw-r-- 1 ubuntu ubuntu 28 May 6 01:02 .dockerignore
256094 drwx----- 3 ubuntu ubuntu 4.0K Apr 27 19:09 .gnupg
527715 drwxrwxr-x 5 ubuntu ubuntu 4.0K Apr 28 18:51 .npm
256237 drwxrwxr-x 8 ubuntu ubuntu 4.0K Apr 27 19:14 .nvm
256078 -rw-r--r-- 1 ubuntu ubuntu 807 Apr 4 2018 .profile
256081 drwx----- 2 ubuntu ubuntu 4.0K Apr 27 18:47 .ssh
256221 -rw-r--r-- 1 ubuntu ubuntu 0 Apr 27 19:10 .sudo_as_admin_successful
256367 -rw----- 1 ubuntu ubuntu 9.4K May 6 01:02 .viminfo
256366 -rw-rw-r-- 1 ubuntu ubuntu 859 May 6 01:02 Dockerfile
262360 drwxrwxr-x 11 ubuntu ubuntu 4.0K Apr 28 18:38 MiniProject
265487 drwxrwxr-x 3 ubuntu ubuntu 4.0K May 6 00:52 composetest
ubuntu@ip-172-31-94-16:~$ ls
Dockerfile MiniProject composetest
ubuntu@ip-172-31-94-16:~$ cd composetest/
ubuntu@ip-172-31-94-16:~/composetest$ ls
Dockerfile MiniProject docker-compose.yml requirements.txt
ubuntu@ip-172-31-94-16:~/composetest$ cat requirements.txt
flask
redis
ubuntu@ip-172-31-94-16:~/composetest$ vi requirements.txt
ubuntu@ip-172-31-94-16:~/composetest$ rm -rf requirements.txt
ubuntu@ip-172-31-94-16:~/composetest$ docker images
REPOSITORY          TAG                 IMAGE ID            CREATED             SIZE
aws-iot-image        latest              2ded9f22e882        32 minutes ago     220MB
composetest_web      latest              b783f39f842f        8 days ago          220MB
home-automation-image latest              01e8864f1b7c        8 days ago          189MB
python               3.7-alpine         16f919b9ecd5        11 days ago         95.8MB
ubuntu               18.04              c3c304cb4f22        12 days ago         64.2MB
hello-world          latest              bf756fblae65        4 months ago        13.3kB
ubuntu@ip-172-31-94-16:~/composetest$ ls
Dockerfile MiniProject docker-compose.yml
ubuntu@ip-172-31-94-16:~/composetest$ vi docker-compose.yml
ubuntu@ip-172-31-94-16:~/composetest$ ls
Dockerfile MiniProject docker-compose.yml
ubuntu@ip-172-31-94-16:~/composetest$ cd ..
ubuntu@ip-172-31-94-16:~$ ls
Dockerfile MiniProject composetest
ubuntu@ip-172-31-94-16:~$ docker images
REPOSITORY          TAG                 IMAGE ID            CREATED             SIZE
aws-iot-image        latest              2ded9f22e882        52 minutes ago     220MB
composetest_web      latest              b783f39f842f        8 days ago          220MB
home-automation-image latest              01e8864f1b7c        8 days ago          189MB
python               3.7-alpine         16f919b9ecd5        11 days ago         95.8MB
ubuntu               18.04              c3c304cb4f22        12 days ago         64.2MB
hello-world          latest              bf756fblae65        4 months ago        13.3kB
ubuntu@ip-172-31-94-16:~$

```

Figure 8-4 Docker images

9 The important of UX design for website interface

For my website, I knew that while functionality was paramount and another aspect that needed special attention was UX design. UX is vital for an application as the absence of it can warn people off using the application as the application will not look as appealing to the eye. In this section, I will be delving into the methodology I took when designing the Front-End for my website to make sure it added to the user experience of my device.

9.1 Designing applications

During the course of my project, I took the aspect of UI in regards to user experience into serious consideration, So in order to implement this, I went online to research different web

layout to get inspiration for my website design. From my research, I saw that the sites were minimalistic. This was done intentionally to not devastate the user with a busy website that is hard to navigate through. Sites that I regularly use such as TikTok, Pinterest and Instagram employ this design choice with the purpose to hide the complex code execution and procedure that take place behind aesthetically pleasing front end. In my website, I have also decided to implement similar strategies. The user experience is extremely important for a home automation application in particular, as the user should feel comfortable and at ease when looking at the different features in their own homes.

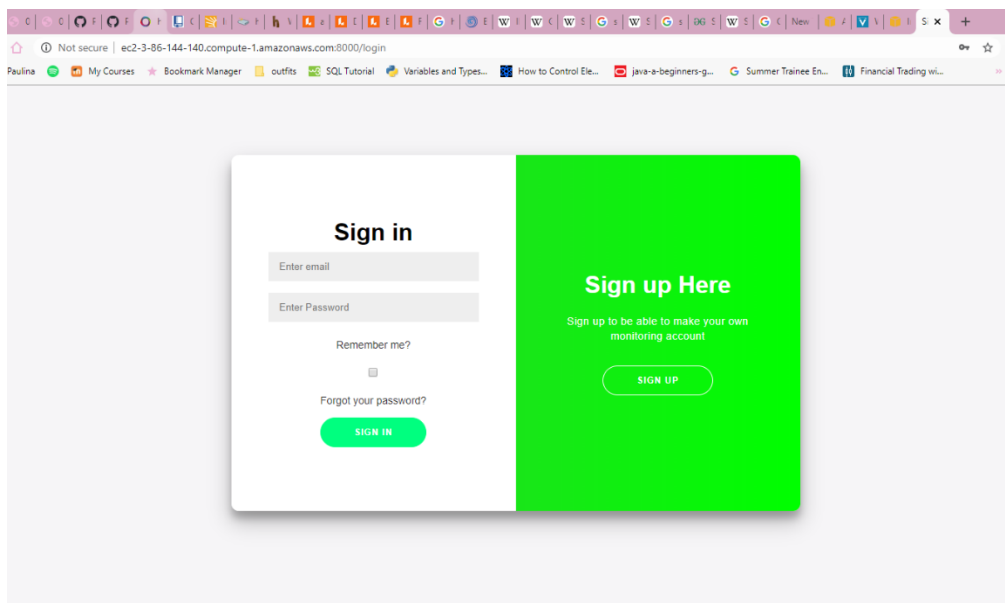


Figure 9-1 sign in page

This website interface is designed to allow the user to sign in and out using dynamic, secure user login authentication together with Passport js and Bcrypt. By implementing these technologies, I was able to successfully set user sessions, password hashing encryption, error messages using flash connect, secrets and user authentication. The entire website was styled using HTML, CSS and JavaScript. The seamless navigation through the website is handled using the engine Hbs(HandleBars). To add an extra element of style to my website interface, I advanced my JavaScript knowledge by creating line animations as seen in the login page which incorporated a smooth finish to the design of the login page. AJAX was the technique that I also used to asynchronously update my webpage with the data coming from the esp32 by exchanging the data with the server, which resulted in our webpage being both fast and dynamic.

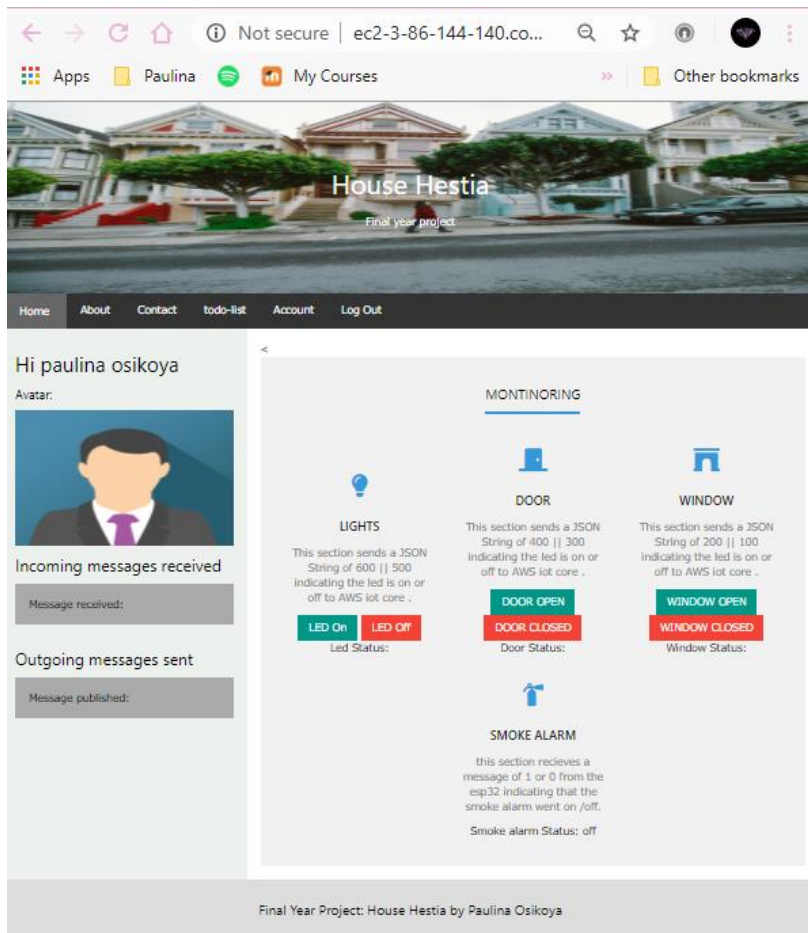


Figure 9-1 main page

10 Nodejs

The reason I chose Node JS was because it offers a great benefit of the utilisation of the FullStack and it is an open-source technology and the fact that it is well documented and backed with a large active community. Node is very useful when developing full-stack applications Node can also set up a running server within minutes and is incredible at handling HTTP requests such as GET,PUT,POST,DELETE etc very well.

NodeJS is an asynchronous event-driven JavaScript run-time environment which is designed to be scalable and can be utilised to build network applications [23]

NodeJS provided a framework to build the server-side architecture and made creating and designing the Back-End for this application both easy and enjoyable. During the lifecycle of my project, NodeJS taught me a lot about the Back-End design of the application. I have

utilised NodeJS on a previous project for an assignment in my coursework, but my knowledge of NodeJS was nowhere near what it is now thanks to building this aspect of my house automation system.

10.1 Why MEAN ?

The MEAN stack is a technology which is used to make full-stack web applications. Each component in a MEAN stack has a specific function to allow for full-stack development. The MEAN stack is widely used in industry and is composed of the following components:

- MongoDB - For storing items in a database
- Express - Provides a web application framework
- Angular – For Handling the front end
- NodeJS - For server architecture and Back-End

I used the MEAN stack to craft my entire website interface. MongoDB Atlas was responsible for hosting my database collection of user details in my cluster, which was stored on the cloud.

Express was responsible for the communication between the front end and back end of my website

Angular did not feature in my website application because I decided to design my layouts from scratch using Flexbox HTML, CSS and JavaScript.

And finally, NodeJS provided a framework to build the server-side architecture to my website

10.2 Express – routing

I chose Express because of its ability to easily create and design a running server. When being used pulling data from the server and displaying it to the client, it is proven to be very helpful, and it is also very scalable. With Express, I was to set-up a fully functional web application I also utilised Express to provide routing to multiple paths which were defined in the files route.js and app.js to retrieve, delete, create & update user details information stored on my Mongo database. Examples of routing and paths in my project are captured below.

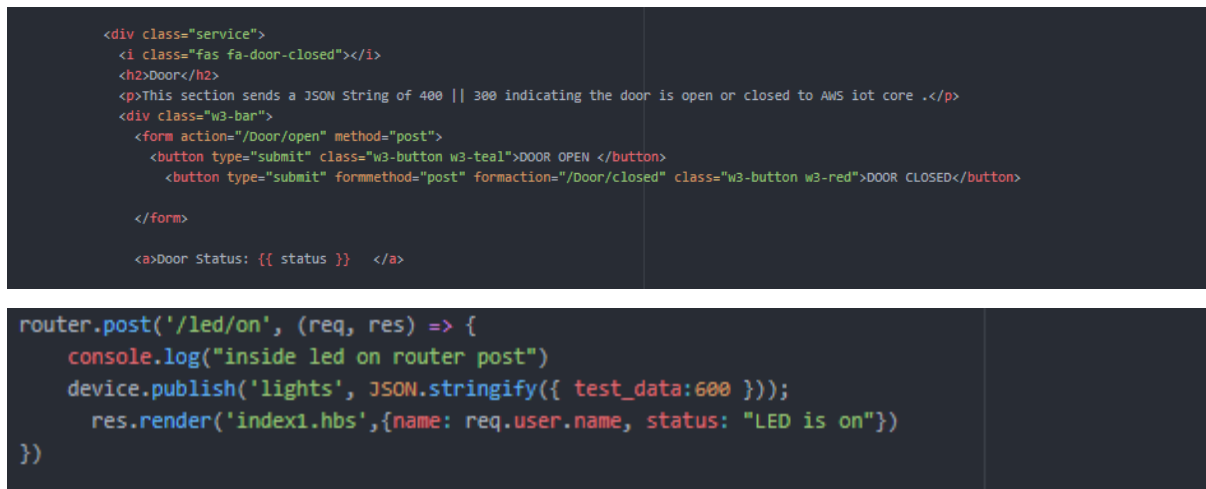


Figure 10-1 express routing examples from FYP

Express was able to send data I needed in JSON(JavaScript Object Notation) format to be set to the Esp32 via the MQTT which would then go through JSON deserialisation to be utilised by the microcontroller to execute the specific actions such as: changing the states of different sensors inside my IoT smart home. I could also pass through parameters in the path and find information on a specific user or piece of data. This was very important for features such as logging the user into the system as I had to compare the username and password and ensure both were correct.

Also, I deployed routing on the server-side so could be navigation effectively handled as this is how the navigation through static pages and handlebar pages is executed. HTTP is also featured in my project. If my project was in production, I would have opted for HTTPS which is more secure as it offers the security of an SSL Certificate which usually cost money, but for demonstration, I found the latter enough.

10.3 MongoDB | Mongoose

Initially, at the beginning of my project, I used mongo hosted on localhost, but I decide to opt for MongoDB Atlas, and it is stored on the cloud and is more secure.

MongoDB is a document database which provides the user with functions to perform actions on data stored in the database. MongoDB stores data in JSON-like documents which means the fields can vary between documents and that the structure of the data can morph over time. The use of JSON-like objects makes the data very easy to access and to manipulate.

Mongoose is a modelling tool for MongoDB, and I utilised it on the BackEnd to create schemas, create objects, read data, update data and delete data which captured below. Mongoose was very useful in communicating with the database and allowed me to create custom features which could be utilised to search for various objects using their parameters. All requests made to the database were made using Mongoose to achieve a full CRUD(Create, Read, Update & delete) application.

```
var mongoose = require('mongoose');
var Schema = mongoose.Schema;

var Account = new Schema({
  entry: { type: Date, default: Date.now() },
  name: {type: String, default: "None given"},
  email: String,
  password: String
});

module.exports = mongoose.model('Account', Account);
```

Figure 10-3 MongoDB – Schema creation

10.4 User Authentication – Bcrypt

Bcrypt is an adaptive hash function based on the Blowfish symmetric block cypher cryptographic algorithm and introduces a work factor (also known as security factor)[26],

In my system, I used SHA256(Secure Hashing Algorithm) to hash and encrypt the user's passwords are encrypted with SHA256 as well. The application complies with the most up to date cyber-security principles and offers the user protection from hackers and implements security features.

SHA256 stands for Secure Hashing Algorithm 256, which is used as a one-way function which means that data which is hashed using this algorithm is impossible to revert into its original form. I also featured user sessions in my system.

A session key is an encryption and decryption key that is randomly generated to ensure the security of a communications session between a user and another computer or between two computers. Session keys are sometimes called symmetric keys because the same key is used for both encryption and decryption.[28]

Below is a code snippet of how I achieve the password hashing and salting of the user's password and if the user password were entered incorrectly, it would direct the user to the home page

```
app.post('/register', checkNotAuthenticated, async (req, res) => {
  try {
    const hashedPassword = await bcrypt.hash(req.body.password, 10)
    var entry = Date.now().toString();
    await Account.create({
      id: entry,
      name: req.body.name,
      email: req.body.email,
      password: hashedPassword},
    function(err){
      if (err) {
        console.log(err);
        return;
      }
      req.flash('info', 'Successful Registration');
      res.redirect('/login')
    });
    users.push({
      entry: entry,
      name: req.body.name,
      email: req.body.email,
      password: hashedPassword
    })
  } catch {
    req.flash('info', 'Unsuccessful Registration');
    res.redirect('/login')
  }
})

function checkNotAuthenticated(req, res, next) {
  if (req.isAuthenticated()) {
    return res.redirect('/')
  }
  next()
}
```

Figure 10-4 Hashing of password

10.5 Handlebars | HBS

Handlebars. Js is a popular templating engine that is powerful, simple to use and has a large community. It is based on the Mustache template language but improves it in several important ways.[29] With Handlebars, you can separate the generation of HTML from the rest of your JavaScript and write cleaner code used the handlebars to store things like the user's name and status of different sensors on my website as capture in the code extract below.

```
app.set('port', 8000);
app.use(fileUpload({
  createParentPath: true
}));
// view engine setup
app.engine('hbs', hbs({extname: 'hbs'}));
app.set('views', path.join(__dirname, 'views'));
app.set('view engine', 'hbs');
```

```
<div class="service">
  <i class="fas fa-door-closed"></i>
  <h2>Door</h2>
  <p>This section sends a JSON String of 400 || 300 indicating the door is open or closed to AWS iot core .</p>
  <div class="w3-bar">
    <form action="/Door/open" method="post">
      <button type="submit" class="w3-button w3-teal">DOOR OPEN </button>
      <button type="submit" formmethod="post" formaction="/Door/closed" class="w3-button w3-red">DOOR CLOSED</button>
    </form>

    <a>Door Status: {{ status }} </a>
```

Figure 10-5 handlebars examples from FYP

10.6 HTML, CSS, JavaScript, jQuery and Flexbox

CSS allowed me to style the HTML pages and to make them look how I wanted them to look. CSS describes how the HTML is supposed to look when the browser loads a web page

Hyper-Text Markup Language or HTML is a markup language and is the standard for creating web pages[30]. I used HTML to structure my web page and insert various elements into the web pages such as images, headers, paragraphs etc.

JavaScript is a lightweight programming language which is interpreted and is compiled just in time[31]. I used JavaScript for performing various functions within my website which added to the functionality of my system

jQuery is a lightweight, "write less, do more", JavaScript library. The purpose of jQuery is to make it much easier to use JavaScript on your website. jQuery takes a lot of everyday tasks that require many lines of JavaScript code to accomplish and wraps them into methods that you can call with a single line of code. [32] I used JQuery in my todo list application which is part of my website to invoke an alert on the HTML notify the user to add something to their list

Flexbox is a layout model that allows elements to align and distribute space within a container. Using flexible widths and heights, elements can be aligned to fill a space or distribute space between elements, which makes it a great tool to use for responsive design systems.[33]

Flexbox was featured in the monitor control panel of my IoT smart home which is captured here

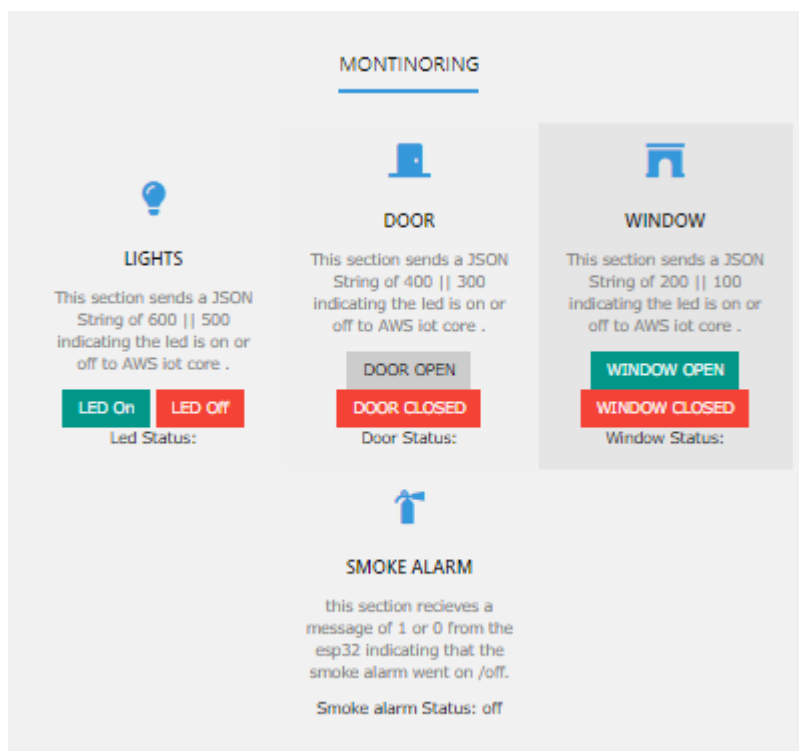


Figure 10-6 Control monitoring panel

11 Voice recognition for IOT house System

11.1 Alexa

For my FYP, I decided that I wanted to add a vocal recognition to my system. The reason behind this idea originated from the fact that in IoT, Control elements with one's voice is becoming the new norm this can be seen in appliances and application such as Amazon's Alexa, google home etc., hence I knew if I could pull off this feature, it would be a significant addition to my project.

Initially, the technology that I intended to use was Amazon's Alexa because of its diverse application in home automation.

The Amazon Alexa, also known simply as Alexa, is a virtual assistant AI technology developed by Amazon, first used in the Amazon Echo smart speakers developed by Amazon Lab126. It is capable of voice interaction, music playback, making to-do lists, setting alarms, streaming podcasts, playing audiobooks, and providing weather, traffic, sports, and other real-time information, such as news. Alexa can also control several smart devices using itself as a home automation system.[34]

How I intended to develop the vocal feature was by integrating an ESP32 with the amazon Alexa and the Alexa would control the sensors hosted on the ESP32. However, trying to implement this technology feature was next to an impossible task. A week or so to my first Project demonstration, I worked tirelessly to find a solution to integrate the device with the microcontroller but to no avail and then I experienced the painful truth of not having read the most recent documentation for the Amazon Alexa which stated that the security for the Alexa device third generation(the device which I possessed) had been changed and the Fauxmo library had been deprecated which meant that the task of implement in this feature was not possible.

The Fauxmo library was the library was developed by Xose Pérez as a hack/loophole to bypass Amazon's Alexa security to use the Alexa to control microcontroller which masked their identity as Belkin Wemo devices; which are renowned for their seamless integration with the Amazon Alexa. Since the start of the year, 2020, Perez abdicated the role of maintaining the repository since the new Amazon update to technology.

However, all hope was not lost as I was still determined to implement this feature, and I was aware that there are many devices like the Amazon Alexa. After going back to the drawing board and doing more research, I came across the Google assistant and the protocol IFTTT and decided that I would opt for this route to implement my voice recognition to my system

11.2 Google Assistant

Google Assistant is an artificial intelligence-powered virtual assistant developed by Google that is primarily available on mobile and smart home devices. The Google Assistant can engage in two-way conversations. The Assistant has been further extended to support a large variety of devices, including cars and third-party smart home appliances.

Users primarily interact with the Google Assistant through natural voice, though keyboard input is also supported. In the same nature and manner as Google Now, the Assistant is able to search the Internet, schedule events and alarms, adjust hardware settings on the user's device, and show information from the user's Google account. Google has also announced that the Assistant will be able to identify objects and gather visual information through the device's camera, and support purchasing products and sending money, as well as identifying songs.[11]

While developing the voice recognition feature, I felt that the process was seamless as setting up the device to work with my device took little to no effort and wasn't as tedious as setting up the Amazon Alexa.

The core work of controlling the sensors on the esp32 was executed and completed by me in the Arduino IDE. Below I have created a diagram which captures the functionality of this feature.

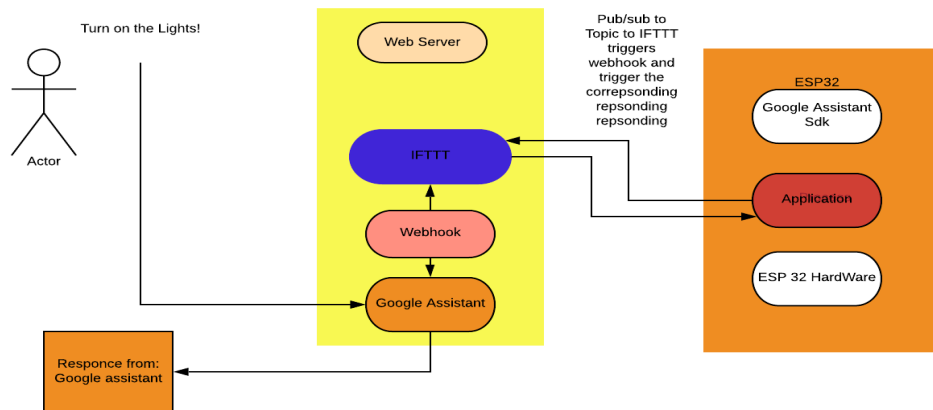


Figure 11-2 Google Assistant

How the feature works is by the user saying a voice command. Once the voice command has been processed inside the google assistant application on tier mobile device, this then fires a webhook or HTTP push API which sends a request to IFTTT and invokes the IFTTT protocol by checking if the words that the user said matches with any of the function instances inside the google assistant.h file that is linked to a topic that I created. When it has found a match a message is printed to the terminal details the request it got from IFTTT, and the ESP32 then responds by carrying out the action and returning the status of the sensor by sending an https post request. This request is then received by IFTTT which then fires off another webhook which is responsible for contacting the google assistant API which will display the result on the app.

12 IFTTT

If This ,Then That, also known as IFTTT, is a freeware web-based service that creates chains of simple conditional statements, called applets.[35]

An applet is triggered by changes that occur within other web services such as Gmail, Facebook, Telegram, Instagram, or Pinterest. For example, an applet may send an e-mail message if the user tweets using a hashtag or copy a photo on Facebook to a user's archive if someone tags a user in a photo.

IFTTT was used in voice recognition feature and was also showcased in the Smoke, Fingerprint and Motion sensor feature. These features also included the use of the SMS and SMTP protocols when certain conditions happened in my home system.

The Simple Mail Transfer Protocol (SMTP) is a TCP/IP communication protocol for electronic mail transmission. SMTP is limited in its ability to queue messages at the receiving end. It is usually used with one of two other protocols, POP3 or IMAP, that let the user save messages in a server mailbox and download them periodically from the server.[36]

The protocol used to send the home of my system an e-mail notifying them that an intruder had entered their house in their absence. Mail servers and other message transfer agents use SMTP to send and receive mail messages which picture below in this diagram. SMTP servers commonly use the Transmission Control Protocol on port number 25.

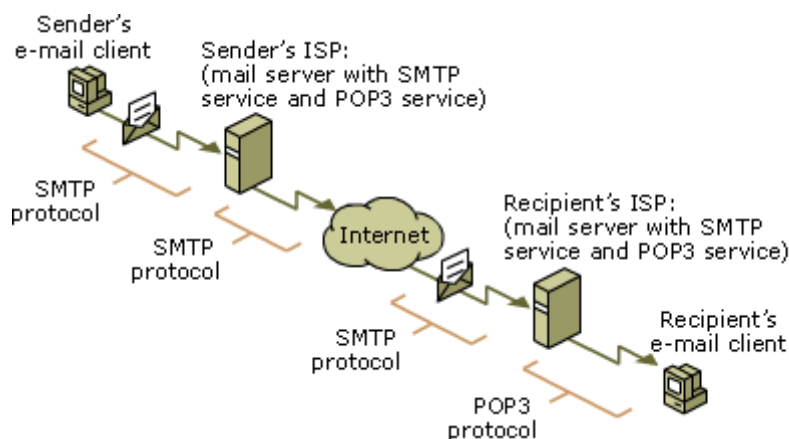


Figure 12 SMTP Protocol

SMS (Short Message Service) is a text messaging service component of most telephone, Internet, and mobile device systems.[37] It was used in my project to notify the user when a particular event occurred in the home automation system. It uses standardized communication protocols to enable mobile devices to exchange short text messages.

The protocols allowed users to send and receive messages of up to 160 characters (when entirely alpha-numeric) to and from GSM mobiles.

13 Webhooks

With IFTTT comes Webhooks, in my House automation, Webhooks were essential for the particular features in my house system that utilized it

Webhooks are basically user-defined HTTP call-backs (or small code snippets linked to a web application) which are triggered by specific events. Whenever that trigger event occurs in the

source site, the webhook sees the event, collects the data, and sends it to the URL specified by you in the form of an HTTP request which is display in the diagram below.[13]

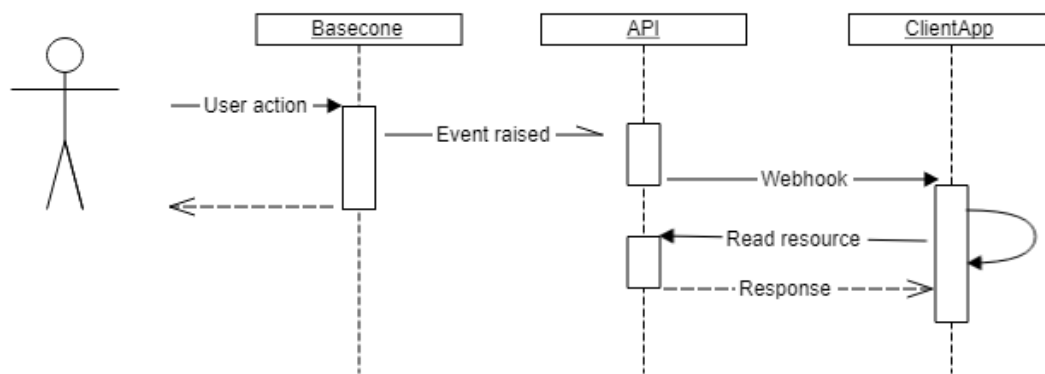


Figure 8-4 Webhook Diagram

If the webhook destination responds with any HTTP status code other than 200 (Success), we will consider it a failure. The request is also considered a failure.

14 System Integration

System integration was one of the parts of my project that I worked tirelessly on. In the end, I was thrilled that I was able to combine all the parts of my project into one fully-functional unit.

Integration in my final year project took two forms: one begins the system integration utilising a master script that I created from all my element, which used MQTT to communicate with AWS EC2 containing the website interface.

The other form being docker containers and separating each functionality into its own individual container but using docker-compose to enable the containers to still maintain communication over a dedicated network.

14.1 Overall Power consumption of System

Altogether, my project takes 1.2A of current to make it function this is solely because of the parts in my project like the motors which to 500ma of current each.

When not connected to a power source, my Home automation can lastly approximately one hour and thirty minutes before the battery dies.

That is why, in the event of commercial deployment of my system, a power source would be required to have consistent communication between the website interface to the user's house

14.2 Problems I encountered with integration and power

- Initially, while developing my system I came across a few errors when it came to integrate all the functionality for my system. This was due to the fact that I was developing a master script on the esp32 for all the various sensors that I coded and I placed all the code into separate header files and include them in the main file. The problem I encountered with integrating my project was an error on my side which was the absence of external header files like AWS-IOT.h and DHT.h. but after viewing errors I received the terminal I was able to resolve the issue
- Another problem I experienced was power problems with the pi 4. When connecting the fingerprint sensor and pi camera, I started to see that one application would run which was the facial recognition script for the pi and the script completely take over the system and this meant the other functionality on the fingerprint would not get executed because the former would constantly be polling for the recognition of a face in the pi camera. Initially, my first thoughts to resolve this system was placing the two functionalities inside two different threads and to have them run in parallel to one another. But instead, to solve this problem, I utilized the power of MQTT and placed the fingerprint sensor instead on the ESP32 microcontroller and have the two deliver their messages of successful completion of their tasks to the MQTT broker.

15 3D House Models and Technologies

By summing up all the exercises together, I built the house model shown below from scratch. Prior to building the house, I had zero experience building house models and no knowledge about apparatus or technology used to design house models.



Figure 13 House model

In order to build the house models, I had to gain an understanding on how a house is built and how the electronics and technology are wired throughout the house. Also, it was crucial to become equipped and accustomed to the jargon used in the architectural world to succeed in its construction.

For my virtual building environment, I used Revit. Revit is a software product and comprehensive workflow from Autodesk that helps architects, designers, builders, and construction professionals work together. The software is a sophisticated way to create models of real-world buildings and structures. It is primarily used in BIM, or building information modelling.[8]

To facilitate the housing project, I paid for an Autodesk Revit licence to be able to utilise the environment and also purchased Udemy course entitled: Autodesk Revit - beginner to an intermediate level, to educate myself on how to go about building my house model. In this course, I learn the fundamentals of how to construct a house model in a CAD environment as captured below.

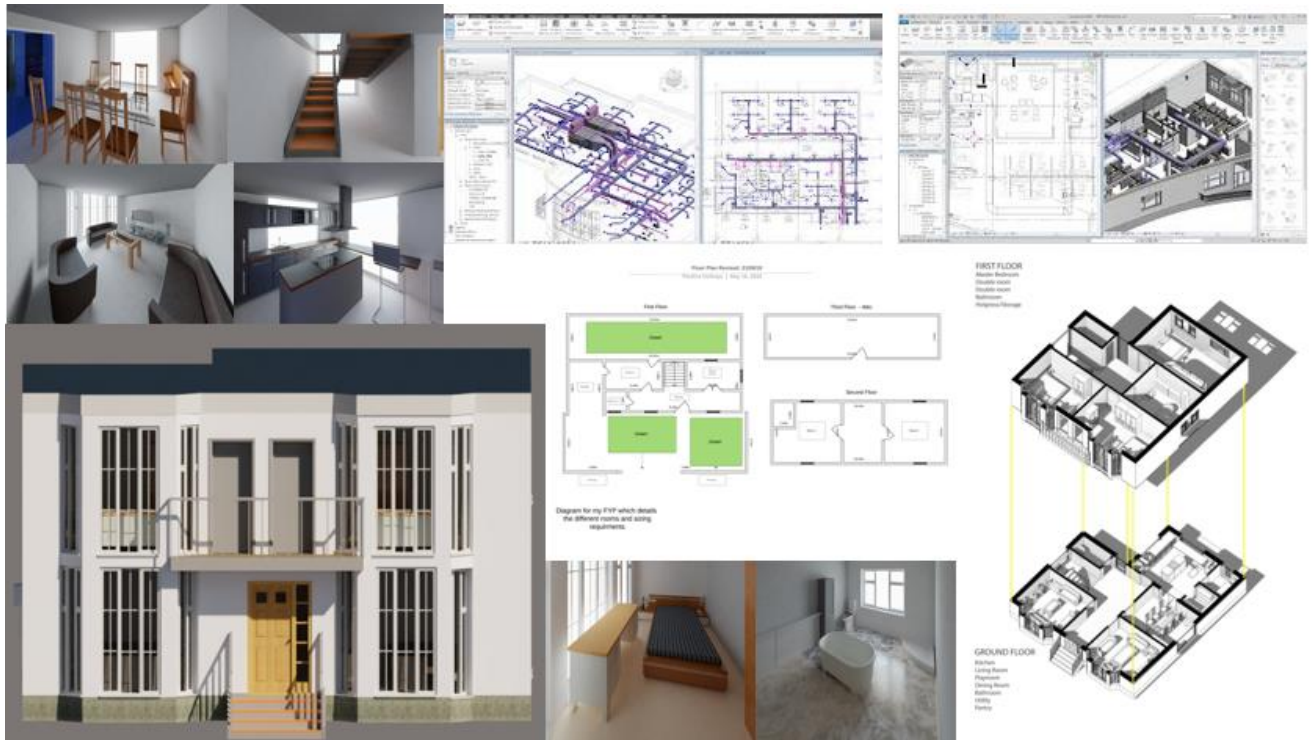


Figure 13-1 Application of models built by me

Throughout the project lifestyle, my construction style and model developed immensely, The physical representation of my House Model was made possible through the usage of the following equipment and materials: handles, glues gun, scaling rules, razor saws and planes, cardboard, perfboard, foam board, polystyrene, and balsa wood.

16 [ISSUES] throughout project lifecycle

After the first project demonstration for my final year project, I was determined to improve on my project with reference to the constructive criticism and suggestions I received from

mentors, supervisor and lecturers with the mindset that I would be showcasing something that I was proud of during my second and final project demonstration.

At the development phases of my project, I was quite happy with the progression of my project and the items I was completing off my backlogs. However, I did feel that these following factors may have impacted my project progression slightly.

16.1 COVID 19 Pandemic

Firstly, I felt that the unprecedented worldwide outbreak of the COVID-19 definitely affected my Final project for a number of reasons stated as follows:

1. Since the commencement of my project, I had used the computers in the college lab to code and process my development work. This is self-explanatory primarily because the computers in the college labs are 10x times faster and powerful than my laptop.
2. I did not have the amenities and resources available to me for project support as I did while I was still attending college
3. There was an unfortunate event which saw my development boards crashing and frying, leaving them unresponsive and not suitable for development. This situation forced me to remove one of the core features from my Home Automation System. In addition to this, while, I was making frantic effort to purchase a new development board, I realised that delivery from Irish vendors was either limited or had been suspended, thus resulting in lags in delivery time. Therefore, if I opted to order a development board it would be arriving after the date I would demonstrating my project.

16.2 Raspberry Pi 4 Model B

As stated earlier, the development board that I was working on was the Raspberry pi 4 and the feature that I was implementing was a facial recognition camera surveillance which is detailed further in this report. Although, my development board had crashed, I was still determined to implement this feature in a hypothetical sense in the event that I was questioned on this feature during my demonstration because as regards my final project

proposal this was a feature I intended to implement in my home automation system . Doing this meant, I could still talk and discuss the work and time I took researching and implementing this feature. Furthermore, the code and documentation are available on my GitHub.

16.3 My Laptop

The absence of the development environment that I usually access in my college was definitely felt when I began developing my house automaton system on my personal PC. Prior to this situation, I had tried to do development on my PC, but personally, I was somewhat more comfortable with using the ones available to me in the labs in my college. Due to this change, I had to work late each night at home to complete the development work for my final year project. The major issues that I faced with my PC were the upload times from IDEs to test my work and the processing power of my laptop. On average, when I was to test functionality that I added to my development platform the esp32 microcontroller, to verify the code inside the Arduino IDE, the process could take anywhere from six minutes to a total of thirteen minutes to verify. Following the verification of code process, one is then required to upload code to the microcontroller in order to test it. Normally, to put everything into perspective, the time it takes the college computer to process work is less than 30 seconds while my PC takes longer to execute. This brings the entire process to nearly 30 minutes to just test a single feature that I wanted to integrate with the rest of my system. I strongly believe that this unfortunate situation negatively affected the project as it halted the development of my project. However, I persevered and put in many late nights to ensure that my project would be completed by the given due date.

17 Conclusion

The outcome of my project was the full functionality of the device, meaning that my Home automation system could be accessed from any IoT device in the world via the website interface that I successfully created. The system was capable of being controlled through Google Assistant API and the system still worked despite the fact that I placed each individual service inside its individual container.

I was very proud of the fact that I delivered a project from an idea on a piece of paper to a fully functioning Home Automation System inside the house model I designed from scratch.

In the future, I would like to further expand on my project by making it commercial and give more focus on AI as this is what the world of technology is currently shifting towards. In addition, I would like to implement the idea into a life-size house

18 Final Words

Firstly I would like to thank you so much for taking time to look at my Final Year Project, I believe it has been a long four years at GMIT, but despite that, I feel that over the years, I have learned so much in a diverse range of topics from the various modules I elected and definitely as a software engineer. Initially, before undertaking my major in software and electronic engineering, I used to see software engineering and computer science with a crowded view as just the ability to crack code and hack into systems at instantaneous speeds. Approaching the culmination of my final year, I now know that it is so much more than that, to be a software engineer one must possess the initiate skill to be a problem solver which involves having the ability to be able to tackle various sets of problems and being able to design elegant applications.

Software engineering over the years has definitely become a passion of mine, and I feel that these four years have exposed me to technologies and system that I never have seen before and has definitely given me different lenses to view the world through. I would like to once again express my highest appreciation to my supervisor, Brian O'shea, for mentoring and guiding me during the development of this project and for imparting suggestions which I would not have thought of otherwise, for example, the integration of AWS Kinesis to stream my raspberry pi camera and the advice of shifting my project towards cloud technologies and incorporating new stacks like the MEAN stack that utilize for the website interface that I created. I would also like to extend my appreciation and gratefulness to all of the lecturers at GMIT. They have helped me in various ways throughout these long, tedious, arduous four years. It is due to the brilliant and talented staff at GMIT that I was able to transfer an idea that I had on paper to a fully functional system.

Further, I wish to extend my thanks to the external examiners who will be grading this project, I would like to thank you for looking through my code, and I hope that you find the documentation I provided and the codes are up to the academic standards.

Finally, I wish to leave you with a quote, which I find beautifully depicts my experience at GMIT, by the highly esteemed Oprah Winfrey "A mentor is someone who allows you to see the hope inside yourself." [20] I think this quote is a fitting end to this report and my time at GMIT as in a few words this quote shows how much I have learned from some of the most exceptional lecturers in the field of Software and Electronic engineering. Thank you for reading this report. It is my sincere hope that I present you with a detailed and extensive overview of the development process of my Home automation system.

19 References

- [1]"The Best Home Automation Systems", Watchdogreviews.com, 2020. [Online]. Available: <https://watchdogreviews.com/best-home-automation-systems/>. [Accessed: 15- May- 2020].
- [2]"Arduino - Environment0018", Arduino.cc, 2020. [Online]. Available: <https://www.arduino.cc/en/Guide/Environment0018>. [Accessed: 18- May- 2020].
- [3]"Energy conservation through smart homes in a smart city", ScienceDirect, 2020. [Online]. Available: <https://www.sciencedirect.com/science/article/pii/S0301421517300393>. [Accessed: 16- May- 2020].
- [4]H. Assistant, "Home Assistant", Home Assistant, 2020. [Online]. Available: <https://www.home-assistant.io/hassio/>. [Accessed: 16- May- 2020].
- [5]G. Blogger, "E-HOMES- A SMART WAY TOWARDS G-HOME - Follow Green Living", Follow Green Living, 2020. [Online]. Available: <https://followgreenliving.com/e-homes-smart-way-towards-g-home/>. [Accessed: 16- May- 2020].
- [6]"Using Trello for Project Management: An Easy, Step-by-Step Guide", 2020. [Online]. Available: <https://blog.hubstaff.com/trello-project-management/>. [Accessed: 16- May- 2020].
- [7]"Epics, Stories, Themes, and Initiatives | Atlassian", Atlassian, 2020. [Online]. Available: <https://www.atlassian.com/agile/project-management/epics-stories-themes>. [Accessed: 16- May- 2020].
- [8]"Why You Should Be Using Revit Architecture Software", Designblendz.com, 2020. [Online]. Available: <https://www.designblendz.com/blog/top-3-reasons-to-consider-using-revit-architecture-software>. [Accessed: 16- May- 2020].
- [9]Haagsman, Ernst (4 April 2019). "Collaboration with Anaconda, Inc". PyCharm Blog. Retrieved 26 May 2019.
- [10]"Amazon Alexa", En.wikipedia.org, 2020. [Online]. Available: https://en.wikipedia.org/wiki/Amazon_Alexa. [Accessed: 16- May- 2020].
- [11]"Google Assistant", En.wikipedia.org, 2020. [Online]. Available: https://en.wikipedia.org/wiki/Google_Assistant. [Accessed: 16- May- 2020].
- [12]"IFTTT", En.wikipedia.org, 2020. [Online]. Available: <https://en.wikipedia.org/wiki/IFTTT>. [Accessed: 16- May- 2020].
- [13]S. Balaji, "What are Webhooks and Why You Can't Afford to Ignore Them - Chargebee's SaaS Dispatch", Chargebee's SaaS Dispatch, 2020. [Online]. Available: <https://www.chargebee.com/blog/what-are-webhooks-explained/>. [Accessed: 16- May- 2020].
- [14]"Basecone Developers", Developers.basecone.com, 2020. [Online]. Available: <https://developers.basecone.com/ApiReference/Webhooks>. [Accessed: 17- May- 2020].all animated circuit diagrams in this were produced and can be located art: (2018) , Available at: fritzing.org (Accessed: 20th April 2018).
- [15] ESP32 WiFi-BT-BLE MCU Module / ESP-WROOM-32 ID: 3320 <https://www.adafruit.com/product/3320>
- [16] Raspberry Pi - Wikipedia. https://en.wikipedia.org/wiki/Raspberry_Pi_4

- [17] Data API (MQTT) | The Things Network. <https://www.thethingsnetwork.org/docs/applications/mqtt/>
- [18] ESP32 WROVER module PCB and IPEX version - שרע arduino <https://www.arduinothai.com/product/2065/esp32-wrover-module-pcb-and-ipex-version>
- [19]Raspberry Pi NoIR Camera Module V2. <https://www.pishop.us/product/raspberry-pi-noir-camera-module-v2/>
- [20]"Oprah Winfrey", Who Mentored You, 2020. [Online]. Available: <https://sites.sph.harvard.edu/wmy/celebrities/oprah-winfrey/>. [Accessed: 16- May- 2020]
- [21] How to Install Atom Text Editor on Linux - Hack The Sec <https://www.hackthesecc.co.in/2016/06/how-to-install-atom-text-editor-on-linux.html>
- [22] Amazon EC2. <https://aws.amazon.com/ec2/videos/>, 2019
- [23] AWS IoT Core Overview - Amazon Web Services. <https://aws.amazon.com/iot-core/>
- [24] What is Amazon Kinesis? - Definition from WhatIs.com. <https://searchaws.techtarget.com/definition/Amazon-Kinesis>
- [25] Overview of Docker Compose | Docker Documentation. <https://docs.docker.com/compose/>
- [26]Docker Compose Networking | Runnable Docker Guides. <https://runnable.com/docker/docker-compose-networking>
- [27]Why You Should Use Bcrypt to Hash Stored Passwords. <https://www.sitepoint.com/why-you-should-use-bcrypt-to-hash-stored-passwords/>
- [28]What is session key? - Definition from WhatIs.com. <https://searchsecurity.techtarget.com/definition/session-key>
- [29] Learn Handlebars in 10 Minutes or Less - Tutorialzine. <https://tutorialzine.com/2015/01/learn-handlebars-in-10-minutes>
- [30]"Hypertext Markup Language (HTML)", W3.org, 2020. [Online]. Available: <https://www.w3.org/MarkUp/1995-archive/html-spec.html>. [Accessed: 18- May- 2020].
- [31]J. Azzopardi, J. Kotlinski, D. Dormann and D. Dickinson, "What's a good lightweight programming language that compiles to native windows code?", Stack Overflow, 2020. [Online]. Available: <https://stackoverflow.com/questions/933129/whats-a-good-lightweight-programming-language-that-compiles-to-native-windows-c>. [Accessed: 18- May- 2020].
- [32] jQuery Introduction - W3Schools. https://www.w3schools.com/jquery/jquery_intro.asp
- [33] When to use Flexbox - The Brolik Blog. <https://brolik.com/blog/when-to-use-flexbox/>
- [34] Amazon Alexa - Wikipedia. https://en.wikipedia.org/wiki/Alexa_Voice_Service
- [35] What does IFTTT stand for? - Abbreviations.com. <https://www.abbreviations.com/IFTTT>
- [36] What is SMTP (Simple Mail Transfer Protocol)? - Definition <https://whatis.techtarget.com/definition/SMTP-Simple-Mail-Transfer-Protocol>
- [37] SMS - Wikipedia. https://en.wikipedia.org/wiki/Short_Message_Service

20 Images References

Arduino

- [1]"Our Arduino IDE Tutorial", Core Electronics, 2020. [Online]. Available: <https://core-electronics.com.au/tutorials/arduino-ide-tutorial.html>. [Accessed: 16- May- 2020].
Mail serve image
- [2]"[Tut] Types of server Computers - Printable Version", Widane.com, 2020. [Online]. Available: <https://widane.com/printthread.php?tid=47>. [Accessed: 17- May- 2020].
Webhook diagram
Webhook image
- [3]"Basecone Developers", Developers.basecone.com, 2020. [Online]. Available: <https://developers.basecone.com/ApiReference/Webhooks>. [Accessed: 17- May- 2020].
MQTT communication
- [4]M. Publish and M. Subscribe, "Publish MQTT Messages and Subscribe to Message Topics- MATLAB & Simulink", Mathworks.com, 2020. [Online]. Available: <https://www.mathworks.com/help/supportpkg/raspberrypi/ref/publish-and-subscribe-to-mqtt-messages.html>. [Accessed: 18- May- 2020].
- MQTT Tcp/ip picture
- [5]T. Team, "Client, Broker / Server and Connection Establishment - MQTT Essentials: Part 3", Hivemq.com, 2020. [Online]. Available: <https://www.hivemq.com/blog/mqtt-essentials-part-3-client-broker-connection-establishment/>. [Accessed: 18- May- 2020].
Docker command
- [6]"docker network ls", Docker Documentation, 2020. [Online]. Available: https://docs.docker.com/engine/reference/commandline/network_ls/. [Accessed: 18- May- 2020].
Raspberry pi camera
- [7]Projects.raspberrypi.org, 2020. [Online]. Available: <https://projects.raspberrypi.org/en/projects/getting-started-with-picamera>. [Accessed: 18- May- 2020].
ESP32 Pinout
- [9]"ESP-WROOM-32 DEVKIT V4 pinout", Flickr, 2020. [Online]. Available: <https://www.flickr.com/photos/jgustavoam/40089095211>. [Accessed: 18- May- 2020].