DISSERTATION
SMART HOME OFFICE HEALTH SYSTEM

Maryam Gadiali

School of Computing and Mathematical Sciences, University of Leicester
CO3204 Software Engineering Project

May 2025

Declaration

All sentences or passages quoted in this report, or computer code of any form whatsoever used and/or submitted at any stages, which are taken from other people's work have been specifically acknowledged by clear citation of the source, specifying author, work, date and page(s).

Any part of my own written work, or software coding, which is substantially based upon other people's work, is duly accompanied by clear citation of the source, specifying author, work, date and page(s).

I understand that failure to do this amounts to plagiarism and will be considered grounds for failure in this module and the degree examination as a whole.

Name: Maryam Gadiali

Signed: mgadiali

Abstract

Rise in remote working since the COVID-19 pandemic has led to an increase in sedentary lifestyles. This paper presents the development and implementation of a low cost smart home office health system solution that aims to provide the user insight into their working conditions and mitigate their health risks.

The system consists of an IoT network featuring a central hub, occupancy sensor, light sensor, temperature sensor, heart rate sensor, lamp, and a fan, where each device uses an ESP32 microcontroller programmed in C++. The system also contains a user-friendly Java web application including a REST API which allows the user to view health metrics and control features such as notification and automation rules.

This paper explores similar existing IoT systems, the justification, implementation, and quality evaluation of the included hardware and software components, as well as an in depth look at the process flows involved in the operation of the system, and a security overview. A critical appraisal of the project is performed reflecting on challenges, personal lessons, and improvements to be made.

The project has been tested in a home working space with improvements made. The system has potential for further devices to be added owing to its scalable design.

Contents

Introduction

Motivation

Ever since the COVID-19 pandemic, the rise of remote and home working has increased which has caused remote workers to experience a more sedentary lifestyle which has been associated with increased health risks such as “overweightness, obesity, cancer and chronic illnesses such as cardiovascular diseases, metabolic syndrome, type 2 diabetes and low back pain”[1][2].

In the UK there are regulations such as the “Health and Safety at Work Act (1973)” and the “Workplace Health Safety and Welfare Regulations (1992)” where employers have to legally ensure the health and safety of their workers[3][4][5][6][7]. However, this is typically followed in regards to traditional workspaces where employers have easy access to risk assessing the environment they own, as opposed to a home where it is the employee’s domain. There is lack of specific regulation regarding home working, and so this unmanaged environment does not guarantee an optimal workplace for the user and invites increased health risks[8].

Aims

The aim of this project is to develop a scalable solution to promote health and wellbeing by helping people be more conscious of their physical state and environment, and to encourage a more active lifestyle to help mitigate some of the health challenges associated with remote work.

This is achieved by creating an IoT network of electronic devices that is placed in the user’s home office, and whilst they are working, readings are gathered describing their user and office health metrics such as temperature, light levels, and heart rate, where the data is displayed in a user friendly web application. Users can choose to receive web notifications, and to automate actuator devices, such as a lamp and fan, based on the readings received. If enabled, the notifications aim to encourage the user to take an action or just to provide as a warning alert.

Objectives

In order to achieve the aims, the following objectives need to be achieved:

1. Learn how to set up an IoT system
2. Improve electronic proficiency by exploring and learning how to implement sensor and actuator circuits
3. Build an IoT system of sensors and actuators
4. Set up and maintain a database that stores user, sensors, and actuator related data
5. Develop a backend API

6. Develop a web application using Spring Boot MVC framework
7. Receive data from the IoT system to display in the web application
8. Send commands from the web application to the IoT system
9. Automate actuator actions based on threshold values
10. Automate notifications based on when metric thresholds are exceeded
11. Convert received data into informative graphs

Overview of technologies used

The IoT network consists of circuits utilising ESP32 microcontrollers that are developed with C++. The web application utilises the Java Spring Boot framework following MVC methodology to provide an API and a user friendly interface.

Overview of solution

Figure 1 shows the architecture of the final system. The IoT network consists of the hub at the centre of a star topology with the nodes being the sensors and actuators.

The hub communicates with the API over Wi-Fi and can receive commands to relay back to the network.

Users can interact with the system via the website which can be used on any device. They can view readings, edit thresholds, enable/disable automation features as well as notification alerts, and manage their IoT network and account.

A database is used to store device details, user details, notifications, and readings.

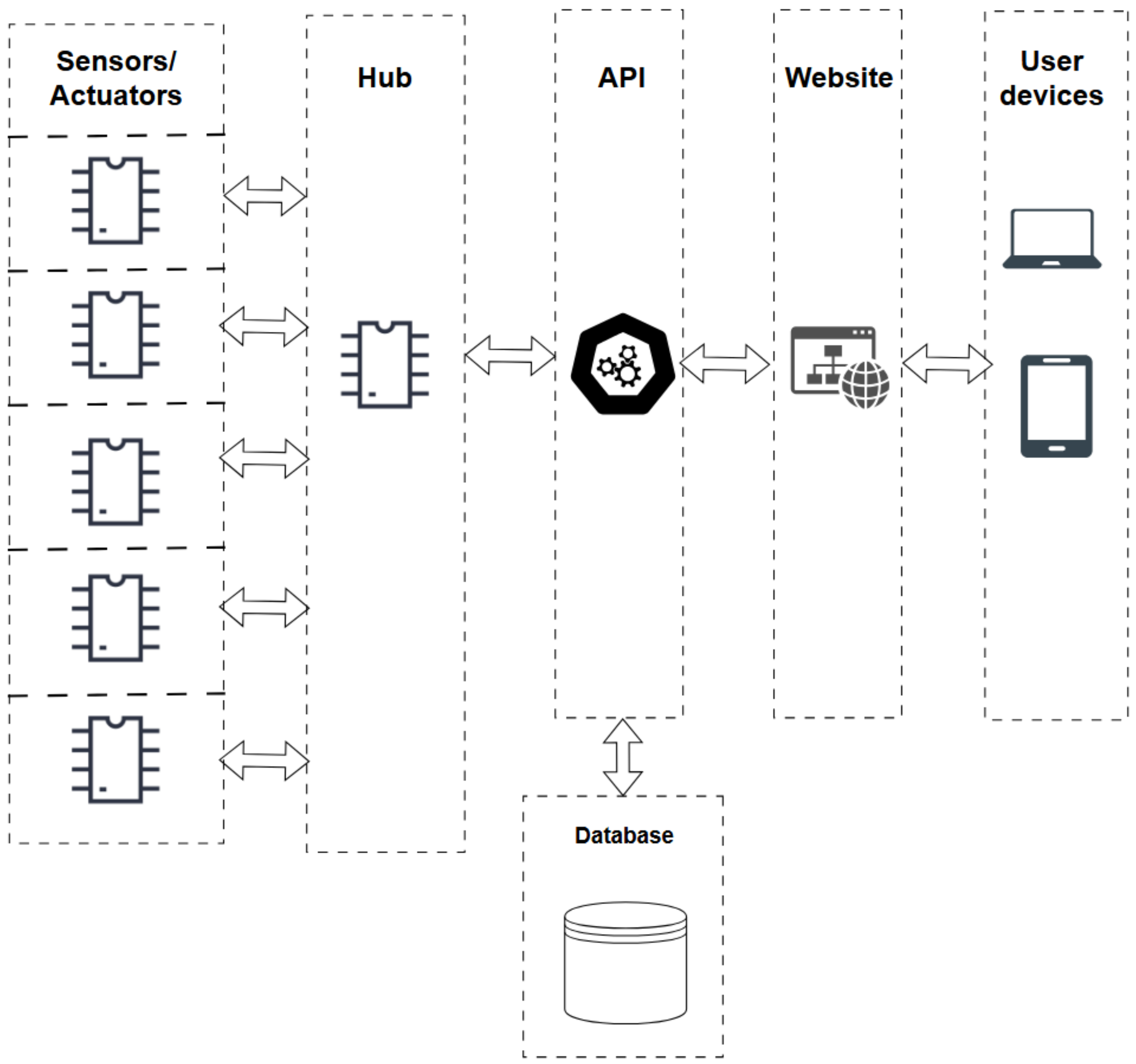


Figure 1

Development rules and constraints

The following rules were set at the start of the project to guide the design and implementation choices:

1. Each component must not cost over 5 pounds – As this project is self-funded, a "low cost" solution is being created. This also takes into consideration excess backup components needed, alongside extra development tools (that can go over this price limit).
2. The created devices need to be portable – As the final deliverables require travelling to University and passing the devices to testers to use, they need to be an appropriate size to handle.
3. Self-healing – The system needs to be user friendly, self-explanatory, and be able to handle any user cases and behaviours, such that the testers do not need input for help.

4. Scalable – There can be no hardcoded details (other than necessary database static set up details and the application's IP address for the hub to point to). The system is designed to be dynamic so that it can handle any future created devices with different MAC addresses. The project should also be developed to allow ease of scalability, with future new device type additions in mind.
5. Digital sensors only – The ESP32 lacks the ability to accurately process analogue to digital signals and is affected greatly by internal electrical noise and non-linearity[9][10]. Figure 2 and Figure 3 show the distinct difference in qualities between heart rate readings received from an analogue, and digital heart rate sensor connected to an ESP32 where the former does not show the heartbeat patterns.

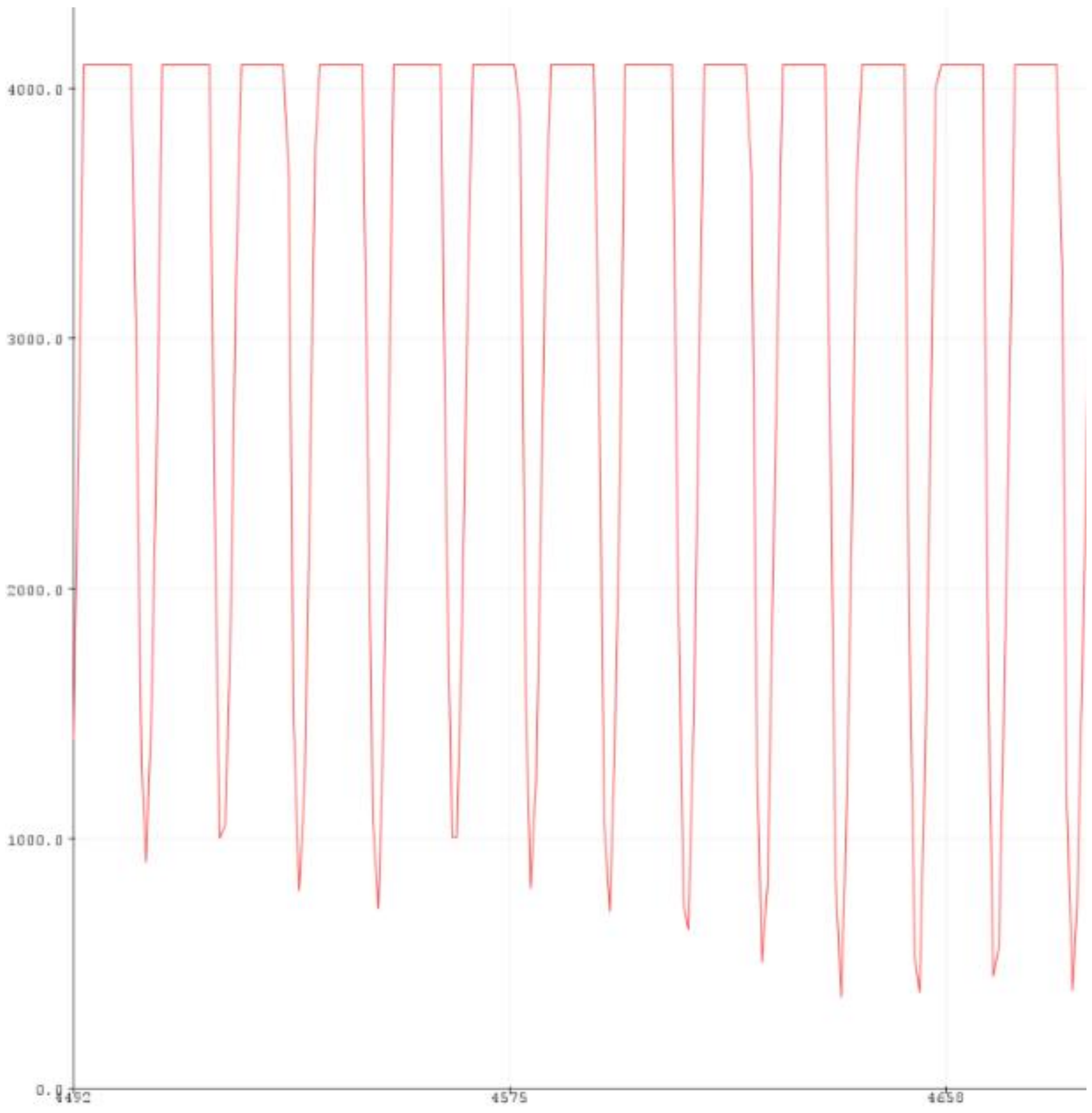


Figure 2 - The heart beat plotting of analogue signals received from the HW-827 pulse sensor on the ESP32

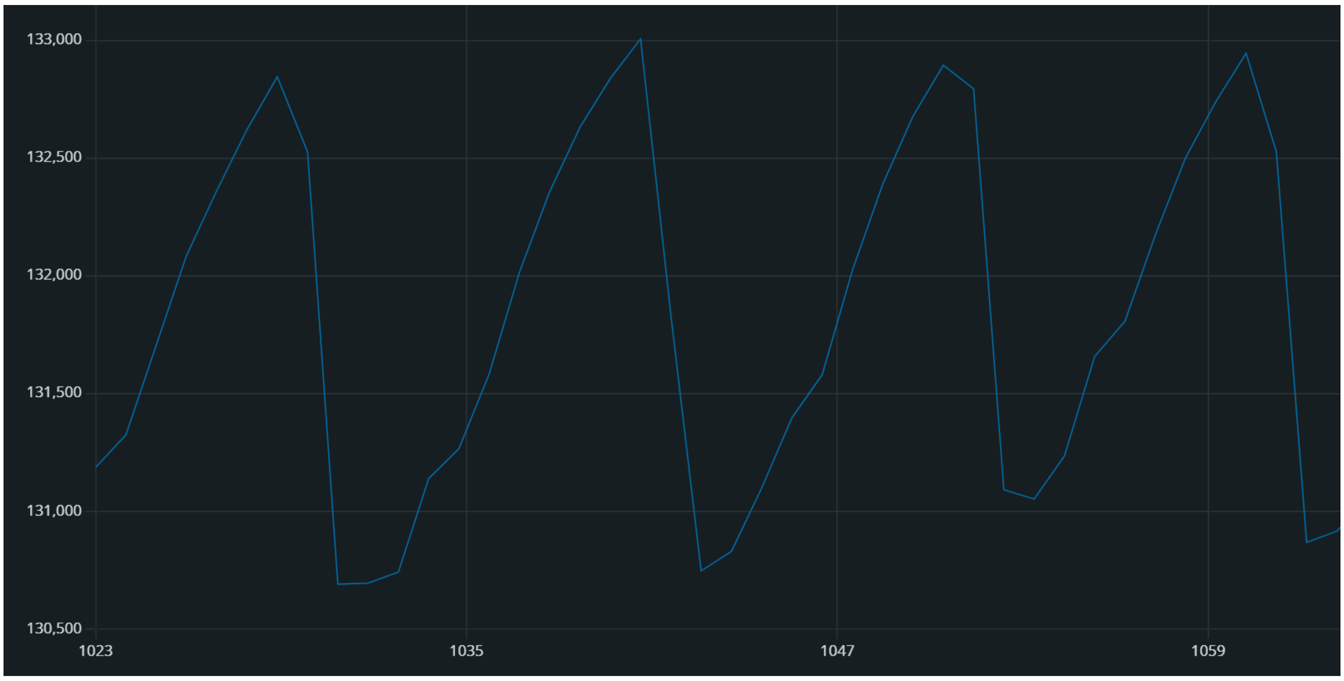


Figure 3 - The heart beat plotting of digital signals received from the MAX30102 on the ESP32

Dissertation overview

This paper explores existing solutions, the planning, design, implementation and security of the IoT network system, the hardware circuits, and the web application, along with a critical evaluation of the project including future improvements.

Existing similar solutions

As the project can be likened to developing a smart home system that is dedicated to only the office room, existing smart home solutions were researched to explore their architecture solutions and further the author's understanding of standard implementation styles.

Google Home and Amazon Echo

Google Home and Amazon Echo offer smart home solutions with the same methodology. Both utilise a voice assistant to a controller device such as a Hub or Echo device. They allow compatibility with external branded devices to join to the smart network, with Google home supporting over 50,000 devices and Amazon echo supporting over 100,000[11][12]. Alongside the voice interface, there is a partner app provided that allows the user to manage their devices and set up smart routines[13][14].

It is important to note that both the controller devices mentioned are not traditional IoT hubs as the other smart devices in the network are not aware of the controller's existence and there is no direct connection between them. This is because market IoT devices generally communicate with a cloud API for instructions and uploading data. When a user adds an external branded device to their network via the Google or Amazon app, they are required to sign into the respective cloud account for the device they want to link. In doing so, they grant the controller access to the external brand's cloud API in order to handle the user requested actions and automations.[15][16]

The controllers can vary in the types of visual interfaces they provide. For devices such as Amazon Echo Show and Google Nest Hub, they provide a touchscreen control panel dashboard[17][18]. Controllers such as the Amazon Echo Dot and Google Nest Mini instead provide LED light indicators to indicate the action they are performing to make up for the lack of a screen display[19][20][21][22].

Hive Home

Hive Home is a UK brand that offers smart home products mainly for energy and heating management and the IoT network is designed to only connect Hive products[23]. It utilises a phone app for user interface control, and the Hive Hub is the only device that connects to the Wi-Fi (or Ethernet) whilst the connected nodes communicate to the hub via Zigbee[23][25]. The system is generally reliant on Wi-Fi and so remote control functionality and metrics are unavailable when the hub is disconnected[26]. The hub utilises LED colours to indicate its actions and issues[27].

Home Assistant

Unlike the above solutions which are designed for the general non-technical consumer, the Home Assistant is an open-source automation tool that requires more technical experience to handle, and is designed to prioritise local control and privacy. It is based upon local non-cloud connections, so that the devices will still continue to function even in the absence of Wi-Fi. It is highly customisable and supports automation and scripting, companion apps, and setting up external IoT designed devices.[28]

Design and Implementation

Completion of initial requirements

At part of the Interim Report at the start of the project in November 2022, functional requirements were listed for the project and arranged into MoSCoW prioritisation. These were used as a guide throughout development. The following requirements have been achieved:

- The system will monitor heart rate, light levels and temperature data

- The system will display an overview of the user’s health and office metrics on a dashboard
- The system will provide alerts to take breaks when prolonged inactivity is detected
- The system will collect data from an IoT microcontroller network of sensors and actuators
- The system will only collect data whilst the user is in the office room.
- The system will allow users to create accounts
- The system will monitor light levels and automatically turn on a lamp when the light sensor detects that it is too dark.
- The system will monitor temperature and automatically turn on a fan when the temperature exceeds a set value.
- The system will allow users to securely log in to the web application
- The system will allow users to manage their own profile
- The system will issue alerts if any sensor “threshold” is passed (e.g. elevated heart rate)
- The system will allow users to add, remove and manage sensors and actuators
- The system will allow remote control automation of actuators
- The system will store and display historical metrics for users to view
- The system will display whether an actuator is currently on or off in the dashboard
- The system will allow users to set custom thresholds for different alerts

The only initial requirement not achieved was monitoring the user’s posture. There was an initial intention to design a wearable device on the back that could detect bad posture. Due to the limitations of not having equipment to provide suitable enclosures for the circuit, as well as choosing not to make the project utilise batteries, this feature was removed from the overall system due to the safety concerns of attaching the circuit to the user without visual confirmation of its safety.

Methodology choice

Due to the author’s lack of experience with C++ and hardware design, there was no formal system architecture in place beforehand. So, development was approached following the iterative and incremental method by allowing the project to be guided by the expected user journey, and was needs driven. Specific user process flows were developed at a time, in the sequence a user would interact with it.

It was not feasible to follow the Agile SCRUM methodology as that is centred around releasing usable versions to the users every sprint and being developed in a team along with ceremonies to reflect and plan. Due to the amount of time taken to establish an end-to-end working system (as foundations were changed frequently at the start), the project was incompatible with committing to delivering ready features every sprint. However, in future work, as the foundations are now fully established and coherent, it is suitable to follow an Agile methodology and gain benefit from feedback loops, reflections, and other practices.

The waterfall methodology was not followed due to its set linear sequence where each stage has to be completed before proceeding. This methodology assumes that the developer knows everything in advance, and this was incompatible with the author's lack of knowledge along with the needs driven approach.

Timeline

The project's software development timeline went as follows (note that bug fixing and edge testing was done very often throughout):

- Pre-December 28th (Initial planning and research)
    - Finalised project requirements, performed hardware research, completed the Interim report
- December 28th – January 2nd (Hub set up flow)
    - Set up Spring Security to handle login/registration and HTTPS, implemented pair code logic, established hub communication with the API, implemented the Wi-Fi Manager Captive portal, learnt how to use AJAX for dynamic pages
- January 2nd – January 20th (Occupancy sensor set up flow)
    - Established ESP-NOW communication (direct and broadcast), included dynamic handling of the MAC address, created the design for the message queue and sending/receiving of commands, configured the channel set up flow, handled LittleFS actions
- January 21st – February 2nd (Light sensor and Lamp set up flow)
    - Created reusable code for general adding of new devices to a network, defined templated code for future device types, handled sensor readings from nodes to database, configured RTC set up flow, learnt how to use Apache ECharts and created dashboard graphs
- February 2nd – February 28th (Front end usability)
    - Learnt how to use Bootstrap to create responsive pages, implemented input sanitation and error handling, learnt how to include IntroJS to create a user tour, worked on interactive elements and UX design, created a process for the hub to retrieve JSON-formatted commands from the application, learnt to use the Notifications API, created the thresholds and automation handling
- March 1st – March 5th (Temperature sensor and Fan set up flow)
    - Implemented device removal logic, created charts presenting historical data, implemented the Heartbeat protocol
- March 6th – 7th (Heart rate sensor set up flow)
- March 8th – 31st (Completing the project)

    - Added LED indicators to all the devices, added authentication requirement to Hub specific API endpoints, added Admin authentication requirement for registering new devices, implemented account deletion process, handled tracking of actuator states
- April 1st – May 2nd (Finalising deliverables)
    - Tested the system for a month whilst completing the dissertation writing, and continued with fixing bugs and implementing improvements

IoT Network

This section explores the planning, design, and implementation of the IoT network system and its structural components.

Background research and planning

Programming language

C++ is used to program the microcontrollers as it is the native language used for the ESP32. Micro-Python is also an option and provides an easier learning curve for beginners in programming, however it has smaller library support and slower performance.[29]

IDE

The Arduino IDE was used to compile and upload the code to the microcontroller, and was picked due to its familiarity, simple interface, and minimum configuration handling, but at a significant trade off to development ease due to its lack of IDE tools such as debugging features, code completion and more.

Other options such as PlatformIO and CLion are available, however they introduced a lot of complexity in handling which proved too overwhelming for a beginner in electronic development.

The author alternated between using Arduino IDE and Visual Studio Code for writing the code as the latter provided extra IDE support such as variable colouring and quick project searching.

Hardware foundation

ESP32-WROOM-32 development boards were chosen as the microcontroller used for all the devices due to its quality, affordability, and support for different communication methods including Wi-Fi, Bluetooth and ESP-NOW, and also supports different module interfaces including PWM, I2C, ADC, and UART[30].

The circuits were developed on breadboards to allow for prototyping and flexibility in experimentation of circuit designs. Specific module choices are explored in the section: “Hardware Circuits”.

Topology

A star topology was chosen instead of a mesh or lack of topology, as it would feature the hub at the centre of the network which means it would be the only device connected to the internet thus decreasing the security risk exposure of the entire hardware network.

A star topology encourages scalability so it is easy to add a new node onto the network, however this brings in a future limitation of the strain on the hub's performance with the more nodes added.

Communication protocol

The IoT devices should be able to carry 2-way communication with a short range protocol as all the devices are placed in a room and the nodes will not be network connected.

Research options

The following explores potential communication protocols and their suitability for this project.

Bluetooth Low Energy (BLE)

BLE is optimised for battery devices and operates as a client-server relationship that requires multiple configurations to handle such as services, characteristics, and descriptors. It struggles with high bandwidth and suffers from poor latency meaning it is not optimised for handling many commands and having fast performance.[31][32]

ESP-NOW

Released in 2016, ESP-NOW is built on top of the ESP's Wi-Fi protocol, and is designed specifically for ESP32s. It has fast performance due to operating at the data link layer of the OSI model and so doesn't perform a TCP handshake. It also supports encrypted communication with AES-128. It is not a popular form of IoT communication, so the documentation and support is more limited than other protocols.[33][34]

MQTT

This protocol requires all the devices to communicate with a broker over Wi-Fi or Ethernet (which would be the hub or an external service in our scenario). It supports message persistence and is designed for scalability. It requires specific configurations such as topics and subscriptions.[35][36]

Final choice

ESP-NOW was picked due to its simple design and fast performance which allows flexibility in how the communication architecture is designed without being bound by too many preset rules to abide by, and it complements the microcontroller choice the best.

Implementation

IoT network communication architecture

The following section provides an overview of the components forming the IoT network's communication structure, along with the defined commands used, and colour indicator meanings, which will be referenced throughout the rest of this paper.

Messaging

The devices form a star topology with the hub as the centre, and they exchange message objects that contain commands with data, which indicates an action for the receiver to perform. The messages can also contain sensor readings which are sent to the API to process and store, and are also evaluated by the hub against thresholds which may trigger an automation or another command.

The sensor data sent to the API is in JSON format, which is handled by the ArduinoJson library[37]. Likewise, the commands retrieved from the API is in JSON format.

Message queue

When the IoT network devices exchange messages, they each have a queue system to handle the incoming commands. The FreeRTOS library's Queue object was used for this[38]. The advantage of this object is that it is designed to handle interrupt processes, such as the ESP-NOW receive callback function, and grant them priority so that the queue handling does not run into any race conditions. This is a better alternative than initially using "std::queue" and a mutex lock which did not provide protection for interrupt conditions.

ESP-NOW receive callback function

When a message is received via ESP-NOW, it instantly gets handled by this callback method by adding them to the message queue or doing instant operations such as setting the RTC (Real-time clock). Commands cannot be handled in this callback function as the function was designed to be executed as quickly as possible due to blockage of Wi-Fi tasks[39].

IoT network commands

Table 1 shows all the commands used in ESP-NOW communication between the Hub and nodes. Each command can be sorted into the following categories:

Initialisation, Data upload, Threshold checking, Actuator actions, Factory reset, Heartbeat, and Continue/Pause.

Table 1

Command	From	To	Class	Instructions
1	Hub	Any nodes	Initialisation	Upon receiving the initialisation command, the nodes store the hub's mac address to their file and sends a corresponding initialisation confirmation command back (2 for Occupancy sensor, 3 for sensor nodes, 4 for actuator nodes)

2	Occupancy sensor	Hub	Initialisation	Upon receiving the initialisation confirmation command, the hub updates the relevant API endpoint to activate the device in the system which finishes the linking process.
3	Sensor nodes	Hub	Initialisation	Same as above
4	Actuator nodes	Hub	Initialisation	Same as above
5	Light sensor	Hub	Data upload	Upon receiving the JSON sensor data containing the Lux reading and timestamp, the hub sends the data directly to the API
8	Occupancy sensor	Hub	Data upload	Upon receiving the JSON sensor data containing the distance reading and timestamp, the hub sends the data directly to the API
11	Light sensor	Hub	Threshold checking	Upon receiving the Lux reading, if the automation for light is enabled, the hub evaluates it against the light threshold. If the reading is below the threshold, the hub sends command 12 to the lamp to switch on, and sends an update to the API regarding the lamp's state.
12	Hub	Any actuator	Actuator actions	Switches on
13	Occupancy sensor	Hub	Threshold checking	Upon receiving the distance reading, the hub evaluates it against the occupancy threshold for checking if a user is present or not. If the received distance exceeds the occupancy threshold, the user is marked as not being present anymore, and a pause command (20) is broadcasted to pause the sensor nodes from taking readings. Otherwise, the user is detected as present, and a continue command (22) is broadcasted.
14	Hub	Any actuator	Actuator actions	Switches off
17	Temperature sensor	Hub	Data upload	Upon receiving the JSON sensor data containing the Celsius reading and timestamp, the hub sends the data directly to the API
18	Temperature sensor	Hub	Threshold checking	Upon receiving the Celsius reading, if the automation for temperature is enabled, the hub evaluates it against the temperature threshold. If the reading is above the threshold, the hub sends command 12 to the fan to switch on, and sends an update to the API regarding the fan's state
19	Hub	Any node	Factory reset	Nodes delete their file containing the hub's MAC address, and restarts themselves
20	Hub	All sensor nodes	Continue/Pause	Causes affected sensors to pause and turn the LED red, whilst they wait for command 22 (Continue)

		(excluding Occupancy)		
21	Hub	All sensor nodes (including Occupancy)	Continue/Pause	Same as above, but affecting Occupancy sensor as well.
22	Hub	All sensor nodes	Continue/Pause	Is sent as an indicator for paused sensors to resume operation
23	Hub	All nodes	Heartbeat	Nodes return a command 24 back which indicates they are still active
24	Nodes	Hub	Heartbeat	Hub increments its number of active connections detected
25	Heart rate sensor	Hub	Data upload	Upon receiving the JSON sensor data containing the heart rate reading and timestamp, the hub sends the data directly to the API
26	Lamp	Hub	Data upload	Updates the API regarding the state of the lamp
27	Fan	Hub	Data upload	Updates the API regarding the state of the fan

Hub specific commands from API

Table 2 shows all the commands used for the Hub by the application.

Table 2

101	Turns off lamp
102	Turns on lamp
103	Sets thresholds
104	Removes a threshold
105	Adds/updates a threshold
106	Deletes a connected sensor
107	Deletes a connected actuator
108	Restarts the ESP32
109	Turns on fan
110	Turns off fan
111	Starts the adding new device process
112	Resets Wi-Fi settings

Hub colour indicators

Table 3 shows the scenarios that trigger a certain LED colour on the Hub.

Table 3

Colour	Meaning
Red	The hub was unable to connect to the previously stored home Wi-Fi details (if there was one), and has initiated the Wi-Fi manager captive portal

Purple	The hub is unable to reach the server, the server may be down or there is insufficient Wi-Fi signal
Green	When the hub is making requests to the API over Wi-Fi
Blue	Normal functioning (includes ESP-NOW operations)
Orange	The hub is broadcasting its local AP for nodes to connect to
Pink	The hub has received command 111, and will start its process for setting up a new device

Nodes' colour indicators

Table 4 shows the scenarios that trigger a certain LED colour on the nodes.

Table 4

Colour	Meaning
Red	The device has paused and stopped taking readings. Either because the hub has detected that the user is no longer present in the room, or it is handling the setup of a new device
Purple	The device is waiting for command 7 from the hub where it sets their RTC
Green	The device is currently taking readings
Blue	Normal functioning (includes ESP-NOW operations)
Orange	The device is trying to locate and connect to the Hub's AP

Hardware codebase

This following section outlines the file structure of the hardware and its reusability.

Code files

Table 5 presents the code files included to create a functioning IoT device, dependent on their types. The primary file is the "ino" suffix files, as this is required by the Arduino IDE in order to compile. The ".h" and ".cpp" suffix files are the header and implementation files that include different aspects of operation allowing modularity, reusability, and organisation.

Table 5

Hub	**Sensor**	**Actuator**
Hub.ino	[SensorName].ino	[ActuatorName].ino
FileHandling.(h/cpp)	FileHandling.(h/cpp)	FileHandling.(h/cpp)

Global.(h/cpp)	Global.(h/cpp)	Global.(h/cpp)
LEDChanger.(h/cpp)	LEDChanger.(h/cpp) – except for Heart	LEDChanger.(h/cpp)
Message.(h)	Message.(h)	Message.(h)
DeviceCommunication.(h/cpp)	HubCommunication.(h/cpp)	HubCommunication.(h/cpp)
APICommunication.(h/cpp)	SensorData.(h)	
InternalClock.(h/cpp)	SensorReading.(h/cpp)	
JsonHandling.(h/cpp)		
SetUp.(h/cpp)		
WifiSetup.(h/cpp)		

Templated code

Whilst the hub shares a lot of functionality with the nodes, it is considered separate here as there is only one hub type available due to the nature of its role without the need for scalability.

The sensors and actuators have been designed to prioritise reusability which has led to sections of code identified that is unique for the device, and those that can be used as a reusable template. The following template files can be copied and pasted to a new device type created (dependent on whether they are a sensor or actuator):

- Message.h
- HubCommunciation.(h/cpp)
- FileHandling.(h/cpp)
- LEDChanger.(h/cpp) – Except for heart sensor as there is no LED connected
- SensorData.h – Except for Actuators as they don't need it
- Set up method of [deviceName].ino – Actuators don't include an RTC (command 7) awaiting loop as they don't need to utilise timestamps, and the heart sensor omits LED method calls.
- Most of the [deviceName].ino files are the same for their respective device type - Except for certain aspects including: reading interval values, which commands trigger a pause, and commands sent for updating specific actuator states.
- Most of the Global.(h/cpp) files – Except for pin number configurations.

SensorReading.(h/cpp) is the main sensor dependent non-reusable file, as it is unique to the sensor module implemented.

Hardware Circuits

This section explores the planning, design, and implementation of the hardware devices created for this project. It also covers how the device readings and collected information is displayed to the user. Each device contains an ESP32 to handle the logic, and an LED indicator to provide a simple user interface on its actions (except the heart rate sensor). Figure 4 shows all the created devices and their pair codes.

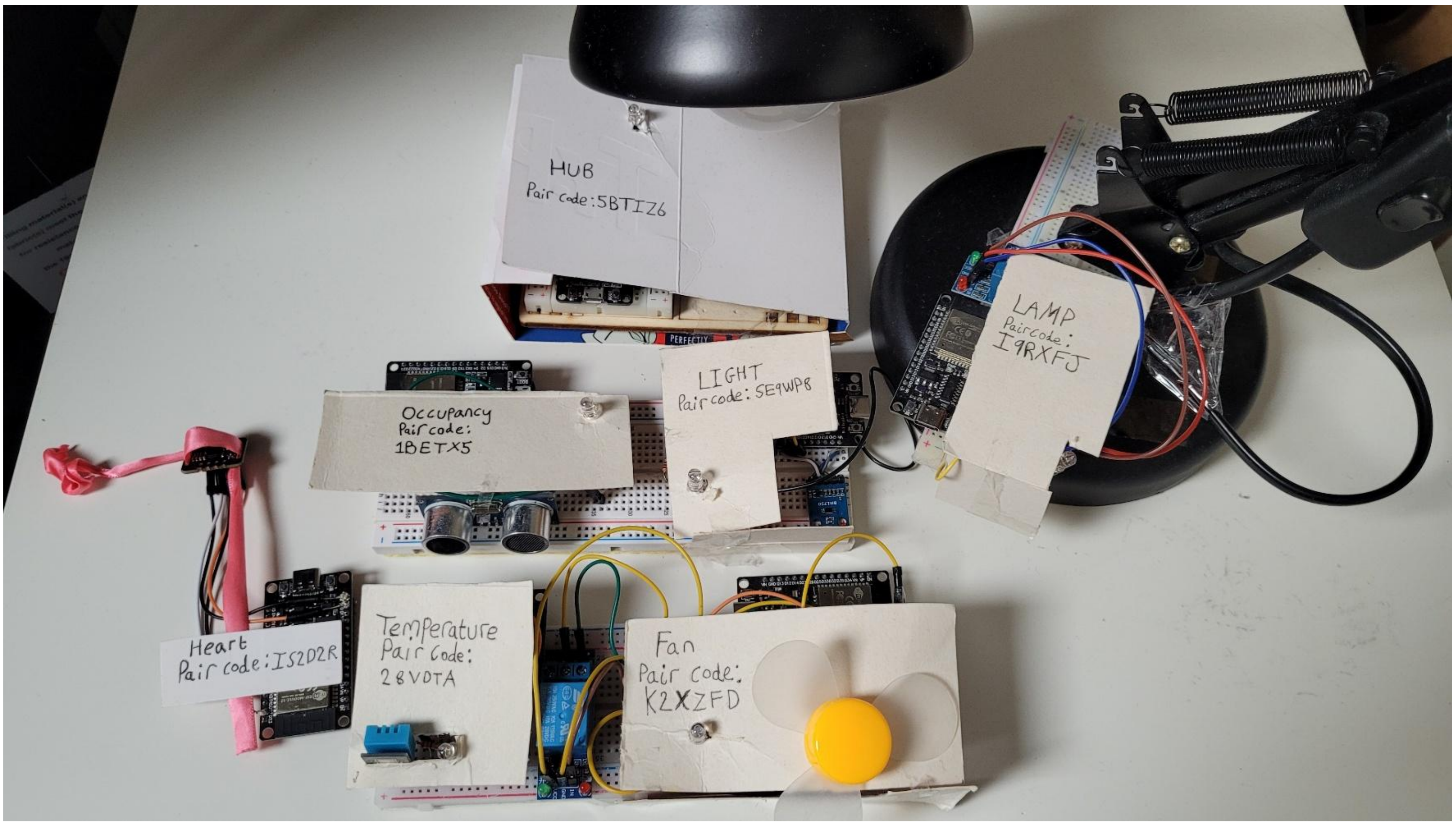


Figure 4

Hub

Purpose

The hub is at the centre of the IoT network and handles communications between the nodes and the API. It evaluates thresholds and triggers automations, if enabled.

Pinout table

The circuit's pinout table is shown in Table 6.

Table 6

ESP32 Pins	Component(s) Pins
18	LED Red
19	LED Green
21	LED Blue
3.3V	LED Power

Occupancy sensor

Purpose

The aim of the occupancy sensor is to detect if a person is sitting in their office chair which indicates that they are working and allows normal sensors operation. If the user is no longer detected, the other sensors are paused.

In the setting up process, guidance is provided to the user to place it in close distance to themselves, such as on the desk or the head of their chair. The accuracy and precision is not a strict priority here as we are not focusing on the reading itself, but rather the comparison of it against a boundary threshold value to reflect if the user is present or not.

The purpose for collecting this data is to make the user more actively aware of how much of a "sedentary" lifestyle they are leading, and to encourage them to reduce the amount sitting where possible. This is because prolonged sitting has been found to have increased risks in the development of obesity, cardiovascular disease, diabetes, and cancer[40][41]. There was research conducted into reducing the effects of prolonged sitting, and a resulting strategy included "standing and taking a break from the computer every 30 minutes"[41]. This strategy was also implemented in a separate research piece, where it found that there were some benefits to this interruption, such as lower blood pressure[42]. This has been incorporated into the project by having a notification alert sent when a user is detected for sitting for over 30 minutes, to take a break.

Research options

The following section explores different possible implementations of the occupancy sensor, and is guided by a comparative study that researched different low cost distance sensors[43].

Temperature

This requires a sensor that detects infrared radiation from a user and monitors the change to determine their presence, such as the Omron D6T Thermal Sensor or MLX90614[44][45]. They are both compatible with the ESP32 and non-contact. Some noticeable complexities to handle this type of sensor includes: filtering the heat given off from the potential computer the user will be working at, handling the different heats given off from different body parts, and handling different body temperature thresholds depending on the person which is quickly variable and can change depending on the location, season, and general weather. Due to this complexity and clash with the project budget rule 1, this is not a favourable option.

Ultrasonic

This requires using high frequency sonar to determine the distance to an object. An example application of this is how a submarine emits and receives the sound wave to calculate the distance to the ocean floor. Similarly, it can detect user presence by checking if they are within a specified distance or not. A limitation of an ultrasonic sensor is that it relies on the targeted object’s ability to reflect the sound back, and so may struggle with objects that instead absorb or scatter the sound wave, such as a sponge.[43]

LiDAR (Light Detection and Ranging)

LiDAR sensors (such as the TFMini) use laser pulses to detect the distance, in a similar methodology to the sonar practice, and are often used to scan an area and create detailed 3D maps. They are generally more expensive and goes over the budget, and they also introduce potential user risk of damage to eyes due to the powerful laser used.[46][47]

Final choice

The ultrasonic sensor was chosen as it was best suited for this project, and supported by the comparative study stating it had the highest accuracy compared to other methods[43].

Pinout table

The circuit’s pinout table is shown in Table 7.

Table 7

ESP32 Pins	Component(s) Pins
5V	Ultrasonic VCC
32	Ultrasonic Trig
34	Ultrasonic Echo
GND	Ultrasonic GND
27	LED Red
26	LED Green
25	LED Blue
3.3V	LED Power

Sensor reading mechanism

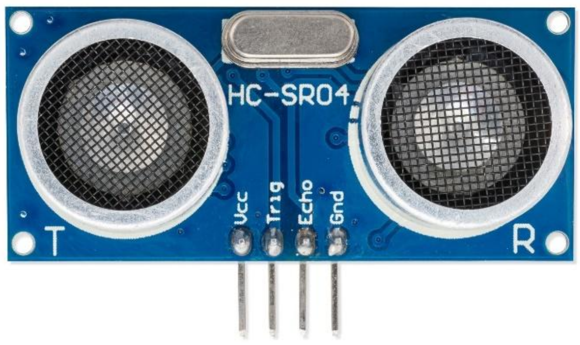


Figure 5

Figure 5 shows the HC-SR04 component. The left circle is the transmitter which sends out the sound wave and corresponds to the Trig pin. The right circle is the receiver that listens for the reflected echo and corresponds to the Echo pin[48].

For this system, the sensor reading is programmed as follows, and takes readings every 30 seconds:

1. The transmitter sends a short pulse for 10 microseconds.
2. Immediately after, the receiver starts listening for the echo. It only waits for 30,000 microseconds as this matches the maximum measurable range of the sensor, which is 4-5 metres[48][49]. If it doesn't receive an echo, it returns 0. The sensor will keep trying to take a reading until it receives an answer.
3. During testing, it has been observed that occasionally, the sensor will return large incorrect erroneous values (which incorrectly suggests the user is not present). To counter this, a check has been added, that if the distance value received is greater than 60, it will check 3 more times to see if there is a value under 60 and take that value as the final distance if true. If all 3 values exceeds 60, then it will take the first reading as the final distance. 60 has been used as the boundary guide here due to it being the author's general working distance.
4. It then sends the distance result in commands 8 and 13, and returns to normal queue handling operations.

Accuracy

The accuracy of the sensor was tested by observing the distance calculated when an object is placed 10cm in front of it (Figure 6). Figure 7 shows the result of the distance being calculated as 9cm, which is acceptable for our use case.

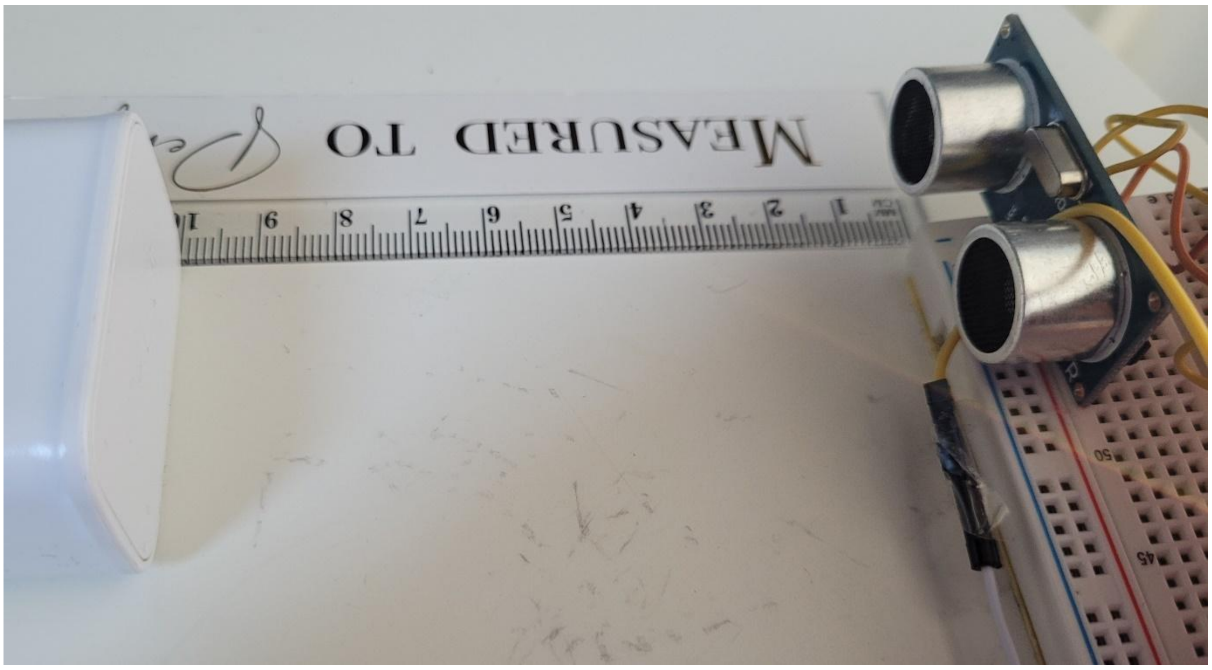
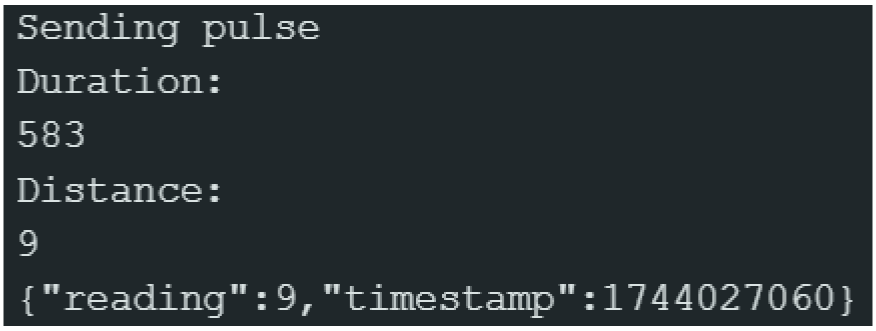


Figure 6



```
Sending pulse
Duration:
583
Distance:
9
{"reading":9,"timestamp":1744027060}
```

Figure 7

Today's data representation

Figure 8 shows the graph used to display the occupancy readings in the website's dashboard section for 'Today's readings'. The distance readings are used to record "sessions", where if a user is continuously detected, then a current session will continue running, and once they have stopped being detected, then the current session ends and is stored as a reading record with the time duration being the value. The sum of the stored time durations and the current running session (if there is one) makes up the value shown in the pie chart. The blue slice represents the remaining portion of today (out of 1440 minutes (24 hours)) that has not been spent seated in the office.

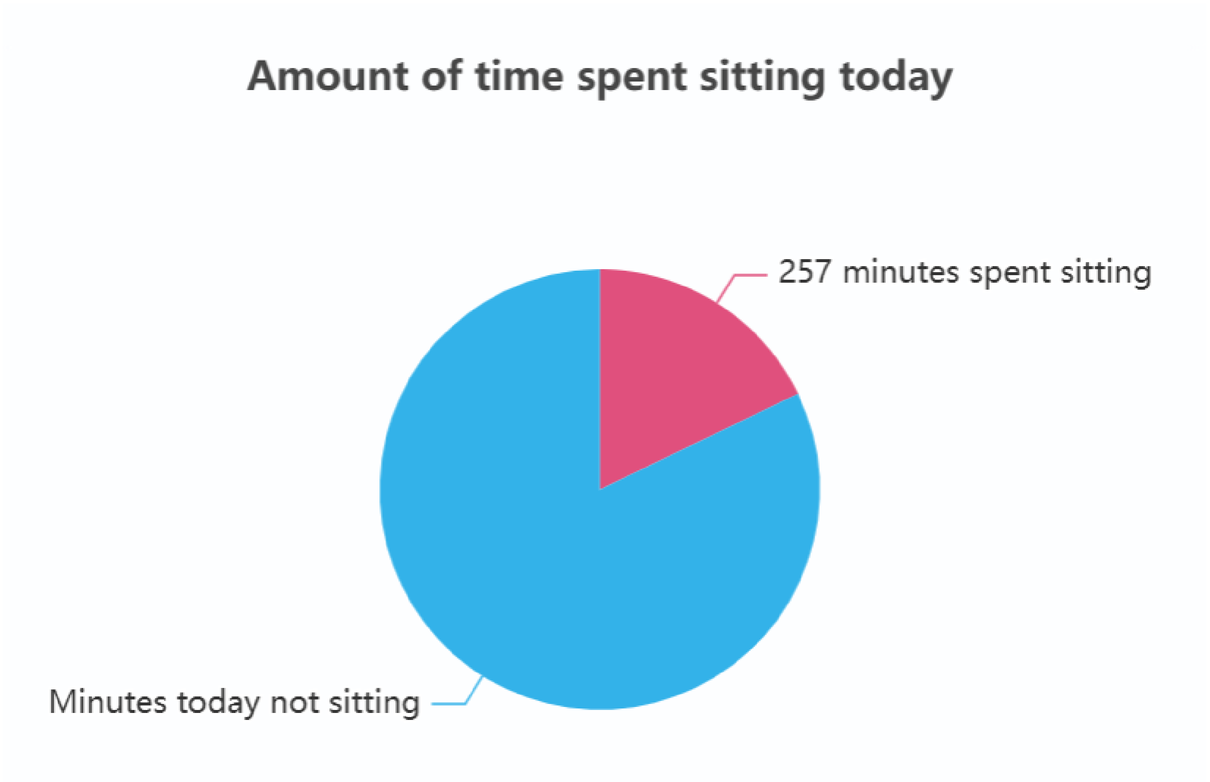


Figure 8

Historical data representation

Figure 9 shows the historical bar chart graph used to show the amount of time spent sitting per day, where the average time is also provided.

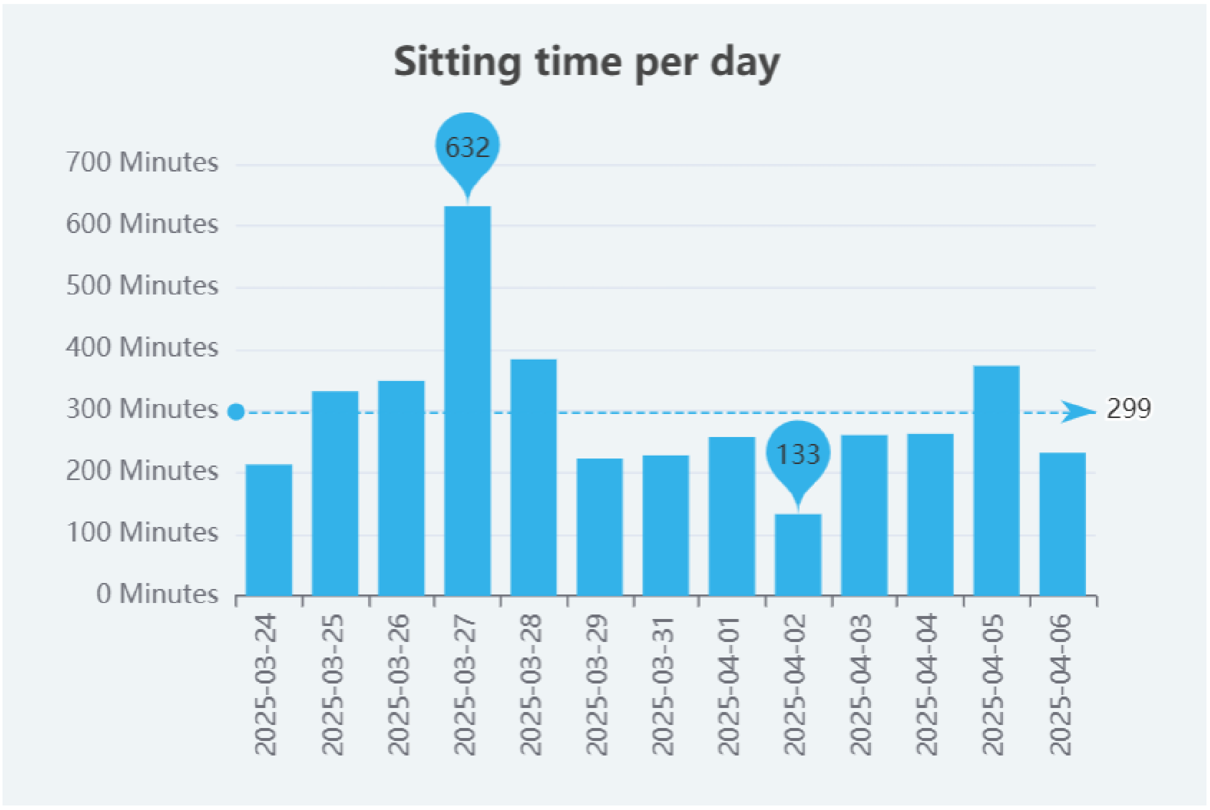


Figure 9

Light sensor

Purpose

The aim of the light sensor is to detect the light levels of the user's office room. CIBSE (Charted Institution of Building Services Engineers) is a professional body in the UK that states minimum lighting requirements based on the context the room is being used for[50][51][52]. For general office work, the minimum is 500 Lux.

There was research conducted that looked into the effects of light levels on home workers, and it was found that sufficiently bright light, such as bright sun in the daytime, ceases Melatonin production which causes more alertness[53]. As Melatonin is mainly produced in dark conditions (such as sleeping at night), this helps support the circadian rhythm, and working in dark conditions can confuse the rhythm and can lead to medical issues such as depression and sleep disturbances[54]. Working in good lighting also leads to better work performance and lower stress levels[55]. This highlights the importance of working in well-lit areas.

Including light analytics in the system allows the user to be more actively aware of their light conditions, and what times of the day they receive the most and least amount of light in the room, so they can manage their conditions better.

Research options

The following explores different light sensor modules to consider.

TEMT6000

This sensor provides an analogue voltage which doesn't meet criteria 5. and requires further calculations and calibrations in order to retrieve the Lux value measured.[56]

BH1750

This is a digital sensor that can provide direct Lux readings from the visible spectrum, between 1-65535 Lux.[57][58]

TSL2561

This is a digital sensor that uses measurements from the visible and infrared spectrum in order to return a Lux reading. This is important in environments with strong infrared radiation sources, such as in a greenhouse which is designed to trap sunlight heat, where infrared can interfere with the reading causing the Lux value to be overestimated.[59]

Final choice

There was a comparative study on low cost light sensors including the above mentioned 3, and it found that the BH1750 provided the best measurement accuracy[60]. As this light sensor is to be placed in a home office room, which is considered a low infrared environment, we do not need to account for the infrared light interference[61]. The BH1750 was picked after concluding it was the best for our use case and criteria.

Pinout table

The circuit's pinout table is shown in Table 8.

Table 8

ESP32 Pins	Component(s) Pins
5V	BH1750 VIN
25	BH1750 SDA
33	BH1750 SCL
GND	BH1750 GND
27	LED Red
26	LED Green
32	LED Blue
3.3V	LED Power

Sensor reading mechanism

The BH1750 sensor has 2 measurement modes: One-time and continuous. Each of these 2 modes have 3 different resolution modes: Low (4 Lux precision), High (1 Lux precision), High 2 (0.5 Lux precision). The one-time mode calculates the reading once and then powers down the sensor, which is the mode used for this project as readings are only taken once every 3 minutes. The standard High resolution was chosen as this precision is suitable for our use case where the values are presented in an integer format to the user. Also, using the High 2 resolution can increase the sensitivity of the sensor meaning noise has more interference with the final value.[57][62]

The reading process uses the BH1750 library[63] to help take the light readings and choose the modes without needing to handle the low level hardware module interface communications (such as I2C). It is programmed as follows:

1. The sensor is started up in the chosen measurement and performance mode.
2. It keeps taking a reading until a non-erroneous value is received (over 0).
3. It is then automatically powered off
4. Steps 1-3 is repeated 3 times and then the average of this is calculated and taken as the final light reading value.
5. It then sends the Lux reading in commands 5 and 11, and returns to normal queue handling operations.

Accuracy

As there was not a dedicated lux meter available to use, the accuracy of the Lux reading itself could not be checked. However, we can detect the accuracy in the rate of change of the light sensor by using a Samsung phone's flashlight which has 5 intensity modes. A big assumption here is that the Lux levels provided by the flashlight increases by the same amount per intensity level increase. In order to validate this assumption, we need to find existing light reading results for this tool.

There was a lack of information found about the Lux levels of Samsung phone flashlights, however, a forum commenter had experimented with finding this for the Samsung Note 8 which also has 5 flashlight intensity levels, and shared their results in Lumens unit[64].

The relationship between the 2 mentioned units is that one lux is equal to one lumen per square meter[65]. Lux and Lumens are both measures of light and so the unit difference does not matter here as we are analysing the trends of the increase in the rate of change of light.

Table 9 shows the results from the forum commentor and Figure 10 shows a corresponding plotted graph.

Table 9

Intensity level	Lumen value
Level 1	7.8
Level 2	15
Level 3	18.2
Level 4	27.6
Level 5	33.92

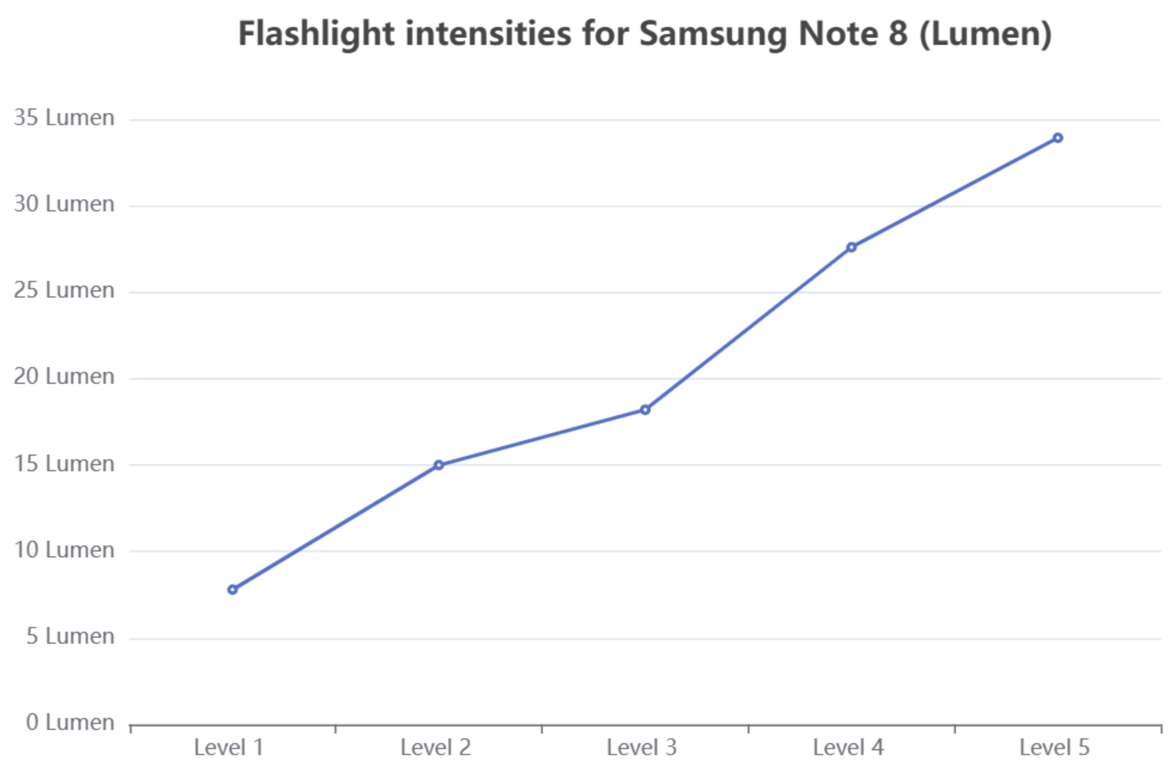


Figure 10

We can see that there is a strong linear relationship trend where the rate of change is nearly consistent, so we can conclude that the assumption is correct regarding the consistency in the rate of change of Samsung flashlight intensities.

Now we can test these 5 flashlight intensities against the chosen BH1750 sensor. Table 10 shows the results and Figure 11 shows the corresponding plotted graph.

Table 10

Intensity level	Lux value
Level 1	950
Level 2	1677
Level 3	2023
Level 4	2359
Level 5	3000

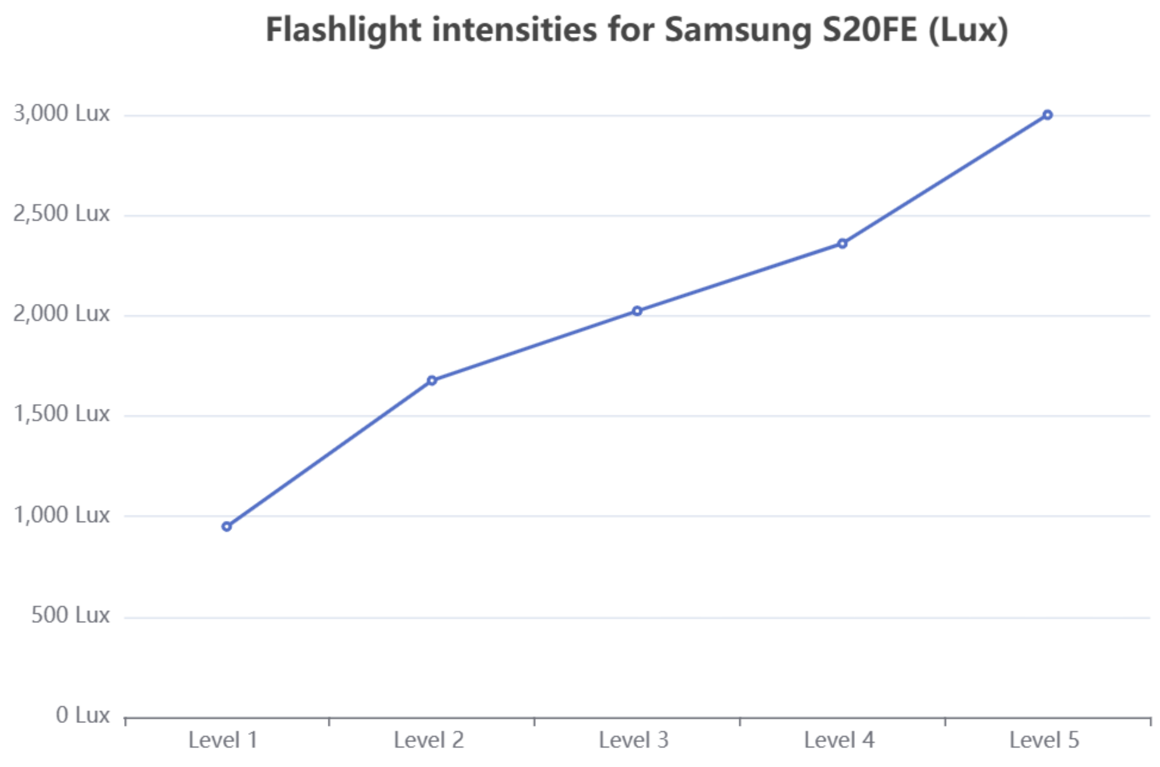


Figure 11

We can see Figure 11 shows a very similar strong linear relationship to that shown in Figure 10. We can conclude that the BH1750 sensor is sufficiently accurate in its rate of change as it presents the expected trend and so is fit for our use case.

The prevention of a perfect linear relationship for the sensor could be due to the lack of control of the room's general lighting from windows which could have introduced unwanted noise to the data. The slight mismatch between the lines plotted in Figure 10 and Figure 11 could be due to manufacturing differences between the 2 different Samsung phone types or lack of control over environment variables in the commentor's experiment.

Today’s data representation

Figure 12 shows the dashboard display of the readings received for the “Today’s readings” section. It is plotted directly to a line graph and allows us to observe general trends about the light levels in the room against the time of day.

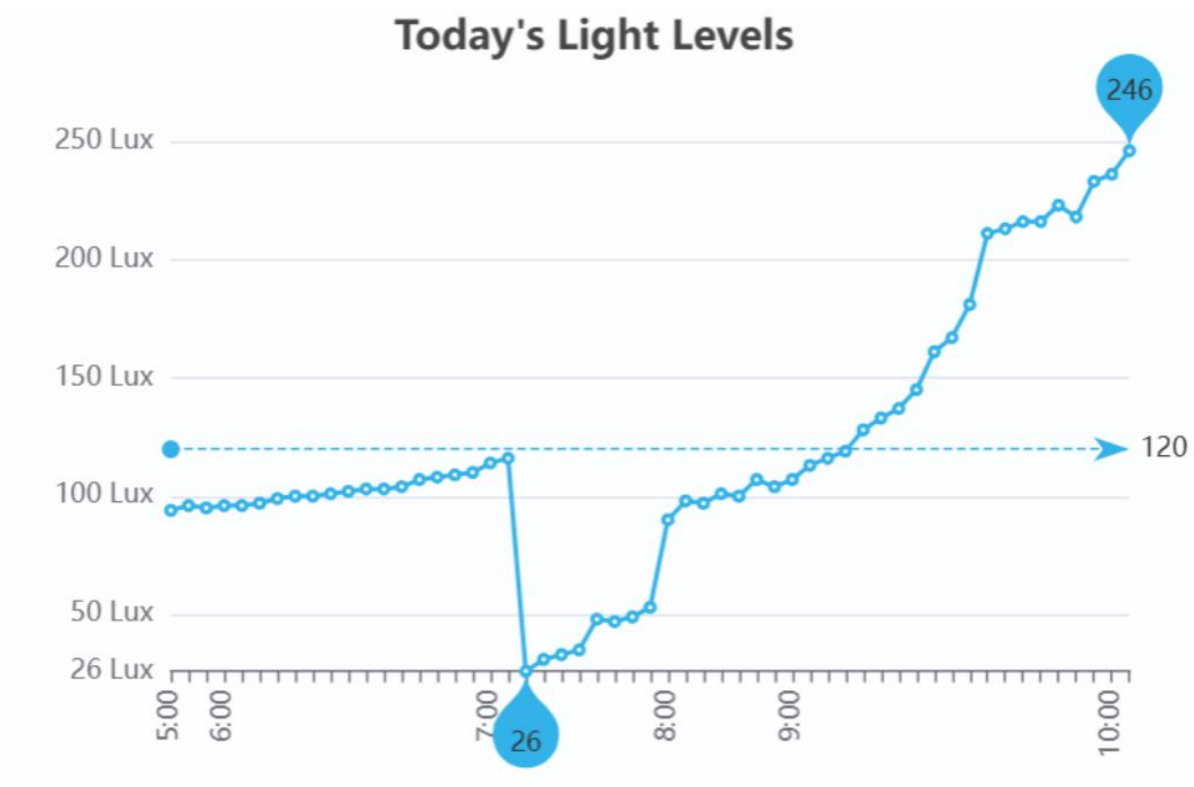


Figure 12

Historical data representation

Figure 13 shows the line graph used to show historical light data, where all the light records for the user are grouped by hour and then averaged.

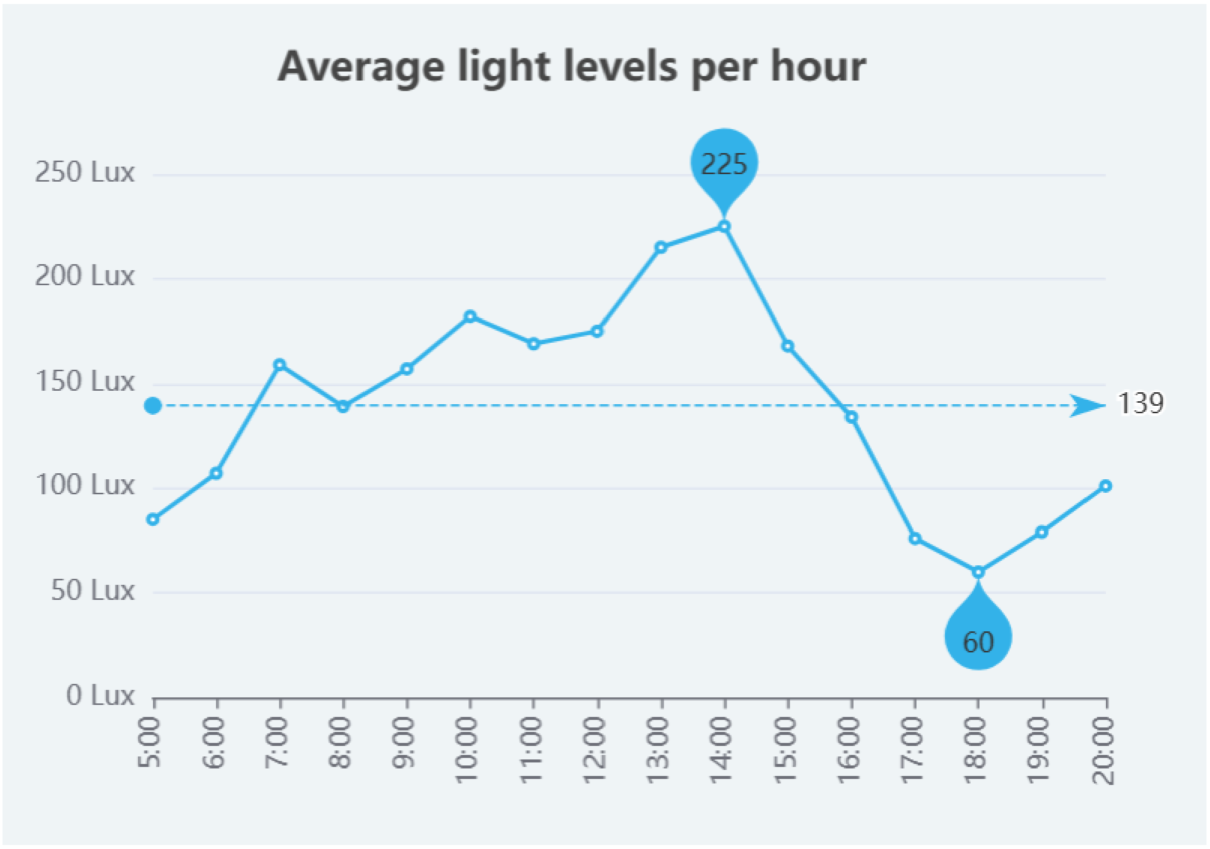


Figure 13

Lamp

Purpose

The purpose of the lamp is to introduce automation and remote control functionality. If the light sensor's automation option is enabled, then when the light levels fall below the threshold, the lamp will automatically turn on (if plugged in). Unlike all the other circuits that rely on the main power source being 5V through DC (Direct current) - which is provided from a laptop's USB port - the lamp is also plugged into the mains which provides 230V through AC (Alternating current). This means this device circuit is the most dangerous as the live wire has been handled to connect to a relay switch, and touching over 120V DC for over 3 seconds, or touching over 50V AC is considered dangerous and potentially life threatening[66].

Research options

The SRD-05VDC-SL-C mechanical relay module was picked as it has a very safe and user friendly design to handle the lamp with as it includes isolation that separates the high voltage wires from the ESP32 pins (Figure 14). A solid state relay was not chosen as that cannot do true physical disconnection which can lead to effects like light flickering, and there is more complexity to handle such as load types and heat generation[67].

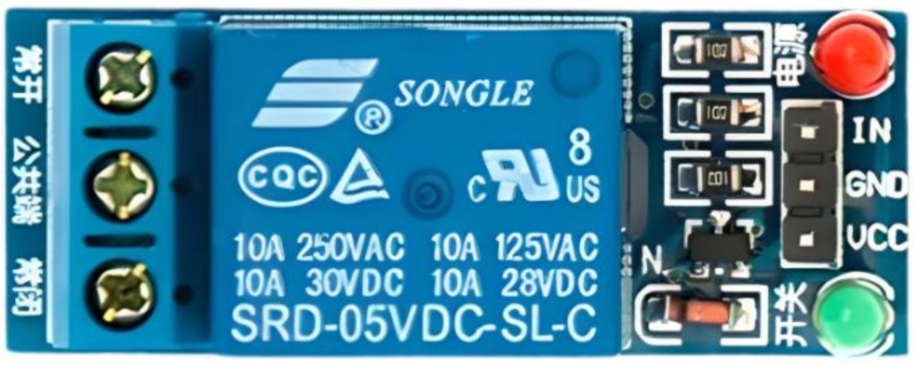


Figure 14

A lamp was chosen instead of a ceiling light due to the initial criteria of being portable and cheap. The limitation of this is that there is only a concentrated area of light on the desk rather than a natural distributed spread of light across the room which results in uneven lighting conditions.

As the guidance for light levels in an office room is 500 Lux, this requires a bulb that can generate over 500 lumens when placed a meter away. A 60W bulb was used here that stated it generates 806 lumens, which satisfies the light recommendation.

Pinout table

The circuit's pinout table is shown in Table 11.

Table 11

ESP32 Pins	Component(s) Pins
3.3V	Relay VCC LED Power
18	Relay Signal
GND	Relay GND
23	LED Red
22	LED Green
19	LED Blue

Live wire and relay handling

This section details how a retail desk lamp was modified for the project.

The outer insulation was scored with sharp blades carefully, followed by using a wire stripper to strip the insulation and expose the wires inside. The live wire was then cut and the wire stripper was used to expose the inner copper strands by a few millimetres. As the relay module uses screw terminals to securely hold the wires instead of solder, the 2 copper ends were simply inserted and screwed into their respective terminals. The input power end was inserted into the COM (Common) terminal, and the other end into the NO (Normally-Open) terminal. Finally, the relay was connected to the ESP32 which issued the on/off commands to the relay.

State handling

Figure 15 shows the dashboard display of the lamp where the user can control it remotely from.

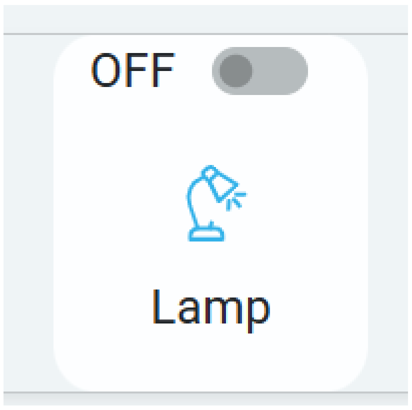


Figure 15

Temperature sensor

Purpose

Research found that the temperature for optimum work performance is around 22 degrees where beyond that, the user's performance levels decreases[68]. Temperature analytics are included in the application so the user can keep track of the current temperature and utilise the fan to ensure comfort, which improves the user's performance.

Research options

The following section explores different low cost temperature sensor modules.

DHT11

This is a digital sensor that can measure from 0 to 50 degrees, and can also measure humidity. It has a stated accuracy of +-2 degrees. DHT22 is the upgraded version and can measure from -40 to 80 degrees with +-0.5 degrees, but at a higher price.[69]

LM35

This is an analogue sensor that can measure from -55 to 150 degrees, with a stated accuracy of +-0.5 (at 25 degrees). This sensor does not comply with the project's criteria.[70]

DS18B20

This is a digital sensor that can measure from -55 to 125 degrees. It has a stated accuracy of +-0.5 from -10 to 85 degrees.[71]

Final choice

Based on the specifications, DS18B20 can be concluded to being the best pick here due to its greater accuracy and dedicated function of measuring temperature. However, this project has utilised the DHT11 instead, which is also sufficient for our use case, due to being able to easily source it during development.

Pinout table

The circuit's pinout table is shown in Table 12.

Table 12

ESP32 Pins	Component Pin(s)
3.3V	DHT11 Power LED Power
5	DHT11 Signal
GND	DHT11 GND
23	LED Red
22	LED Green
18	LED Blue

Sensor reading mechanism

The DHT11 library[72] is used to allow us to easily take a temperature reading and avoid working on low level interface configurations.

The reading flow is as follows:

1. The sensor keeps trying to take a reading until a non-erroneous value is returned (including timeout and checksum errors).
2. This is repeated 3 times and then the average value is taken.
3. The final Celsius reading is sent as command 17 and 18 to the hub.

Accuracy

The accuracy of the used sensor was tested by matching the reading taken against an infrared thermometer. The DHT11 library produces readings in integer format, so there is no decimal precision available. Testing with the thermometer provided the same value as the sensor repeatedly, so we can conclude that the sensor is accurate for our use case.

Today's data representation

Figure 16 shows the dashboard display of the current temperature reading. It displays the average of the last 3 readings received (or the general average if less than 3).

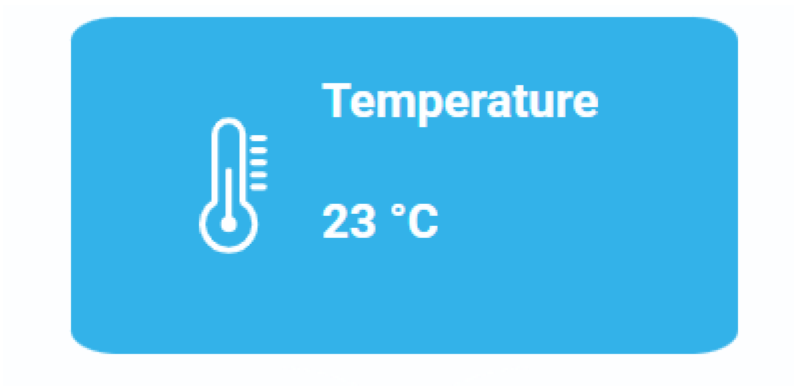


Figure 16

Historical data representation

Figure 17 shows the historical line graph used to show the average temperature reading per day.

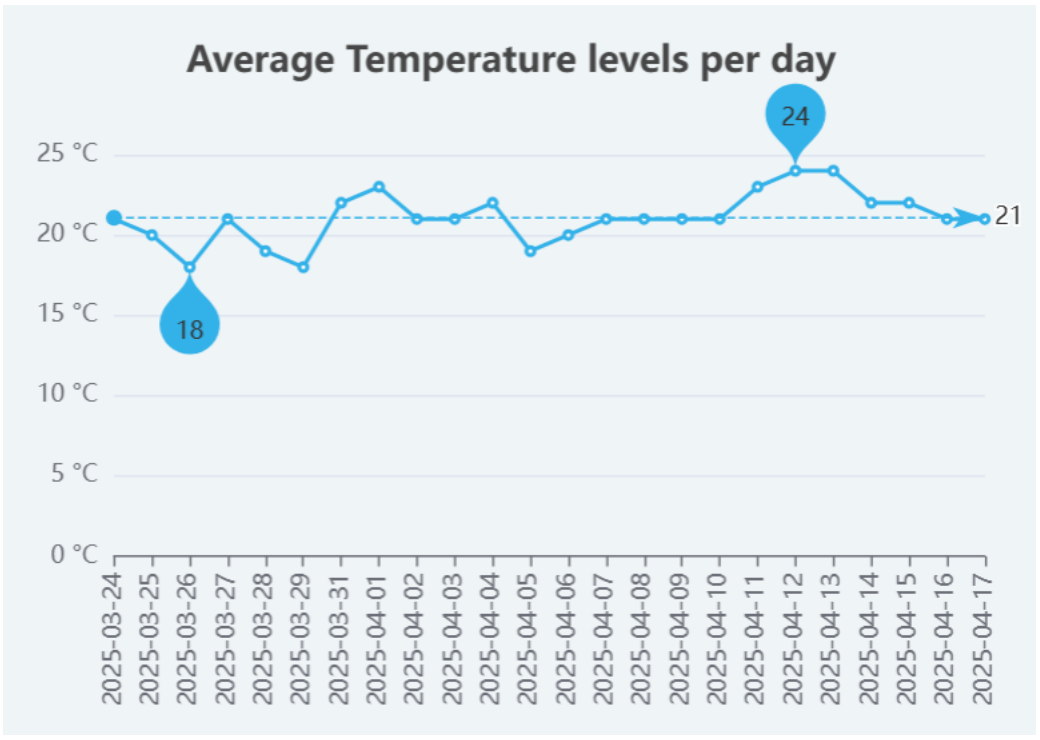


Figure 17

Fan

Purpose

This is similar to the lamp's purpose, but if the temperature levels exceed the temperature threshold, the fan will automatically turn on (if enabled).

Research options

As using a large scale fan would not meet the portable or price criteria, a DC motor with a mini fan head attached was used, but it was configured as if it was an AC device, and so was connected to a relay to control its state like the lamp.

Pinout table

The circuit's pinout table is shown in Table 13.

Table 13

ESP32 Pins	Component Pin(s)
3.3V	Relay VCC LED Power
GND	Relay GND
5	Relay Signal
23	LED Red
22	LED Green
19	LED Blue

Relay handling

The connector wire from the COM terminal is connected to 3.3V, and the wire connecting to the NO terminal is from one of the 2 terminals on the fan. The other terminal of the fan is connected to ground. So when the relay switch has been activated connecting the COM and NO wires, it completes the fan's circuit allowing it to turn on.

State handling

Figure 18 shows the dashboard display of the fan where the user can remote control it from.

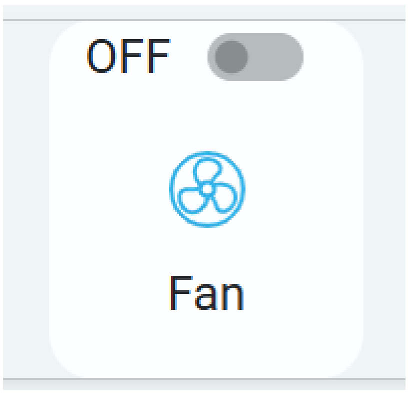


Figure 18

Heart rate sensor

Purpose

Research was found that when faced with significant mental load, the user experiences an increase in adrenaline release which triggers the "fight or flight" response where the heart rate increases significantly for a short period of time before stabilising. Frequently experiencing these events can lead to long term cardiovascular issues. [73][74]

The purpose of this sensor is to be a wearable device that can detect the user's heart rate when working. This is so the user can be more aware of their heart trends and activity to help manage any stress events more consciously, and to track their changes in trends over time. A notification is triggered to alert the user when a reading is detected above the heart rate threshold.

Research options

The following section explores different sensor implementations to consider.

HW-827 pulse sensor

This is an analogue sensor that implements PPG (the tracking of light absorption changes of the blood flow corresponding with a heartbeat) with a green LED, and has a preferable small circular shape that improves wearability.[75][76][77]

MAX30102 pulse sensor

This is a digital sensor that detects heart rate and blood oxygen with PPG. Red and infrared LEDs are included here as these are particularly needed for the blood oxygen measuring.[78][79]

ECG sensor

These measure the electrical activity of the heart using electrodes. It is an accurate method, but is not practical for this project as it is not easily removable, not user friendly, and not within budget.[80]

Final choice

The MAX30102 pulse sensor was chosen as it was the best matching the project's use case and criteria.

Pinout table

The circuit's pinout table is shown in Table 14.

Table 14

ESP32 Pin	Component(s) Pins
5V	MAX30102 VIN
21	MAX30102 SDA
22	MAX30102 SCL
GND	MAX30102 GND

There is no LED colour interface on this device only, as it would have interfered with the wearability of the device.

Sensor reading mechanism

The method used here is adapted from the example provided by Nathan Seidle at SparkFun Electronics (2/10/2016) where the heart rate is calculated by using a “Penpheral Beat Amplitude” (PBA) algorithm[81]. For every beat detected, it calculates the time difference between the current and the previous beat to estimate an instant BPM value as well as a 4 point moving average BPM value. This was adapted for this project by taking readings continuously for 30 seconds storing every BPM value, removing the outliers, and then taking a final average. The SparkFun library was used to manage the heart sensor[82]. The final reading process is as follows:

1. The sensor is initialised, and the red LED is switched on.
2. If a beat is detected where the calculated instant BPM is greater than 30 or lower than 255, and if the finger is detected (through infrared detection), then it is added to a vector of BPMs. The boundary conditions are to prevent noise data from being regarded as the user’s heartbeat.
3. After repeating step 2 for 30 seconds, the IQR method is used for removing outliers, by only keeping values that are between the 25th and 75th quartile. This produces a more representative remaining collection of values.
4. The average is calculated of the remaining values, and is regarded as the final BPM to send. If the value is lower than 45, then it is discarded, as from testing observation, this has been mostly due to when the sensor is not worn and is prone to environment noise interference which can be caused from light fluctuations, electrical noise, and vibrations. If it is valid, then the heart rate reading is sent to the hub through command 25.

Using the PBA algorithm to calculate the heart rate provides more accuracy than just counting the number of beats per second as there is an extra dimension of complexity where each beat can be used to determine an estimate of the user’s BPM due to also including time as a variable and then an overall average is taken, which provides a more representative value.

Accuracy

The accuracy of the MAX30102’s readings were tested against a finger pulse oximeter. Both were worn simultaneously on the 2 index fingers. It was found that the sensor gave near accurate results with it detecting 78 BPM whilst the oximeter returned 77 BPM. Observing the continuous instant values returned from the 2, it could be seen that the heart sensor was accurate a lot of the time, however it sometimes gave quite erroneous results (but that would most likely have been filtered out in the outliers removal).

This may be due to it not having enough consistent firm pressure applied by the worn ribbon for the sensor placement on the finger, which could have affected the time difference between detected beats, which affects the BPM. Unlike the pulse oximeter, where it clamps down on both sides of the finger for consistent placement.

Today's data representation

Figure 19 shows the dashboard display of the "Today's readings" line graph for the heart rate.

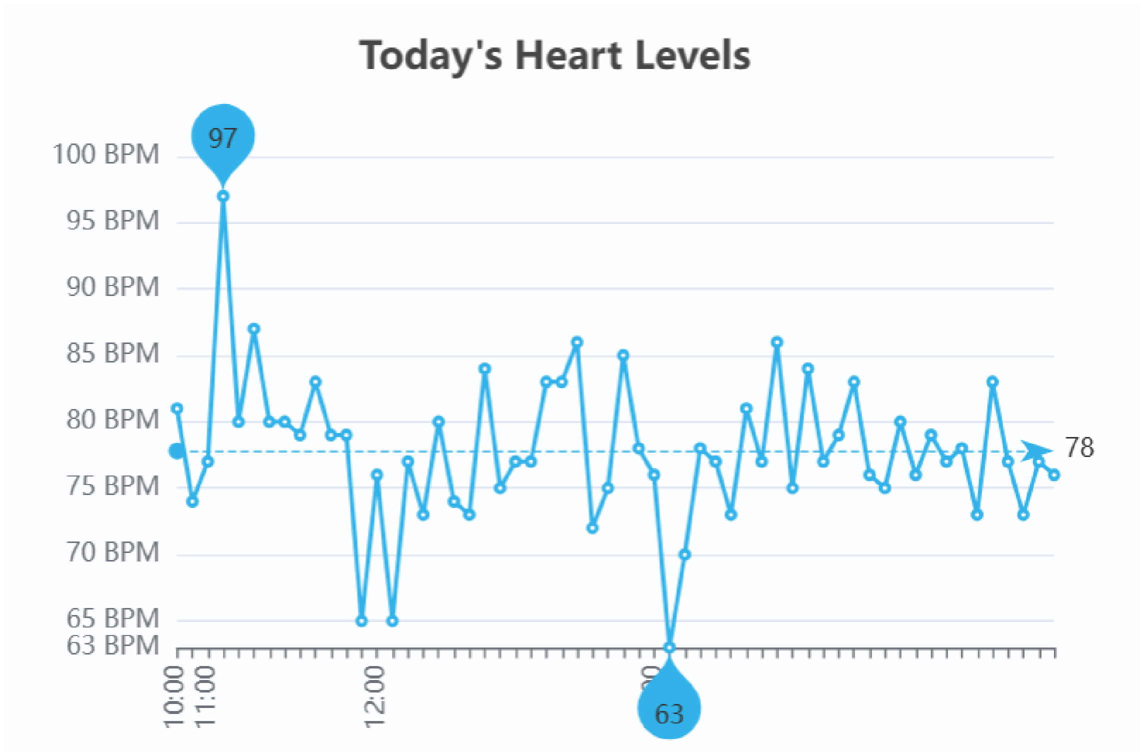


Figure 19

Historical data representation

Figure 20 shows the historical line graph used to show the average heart rate per hour.

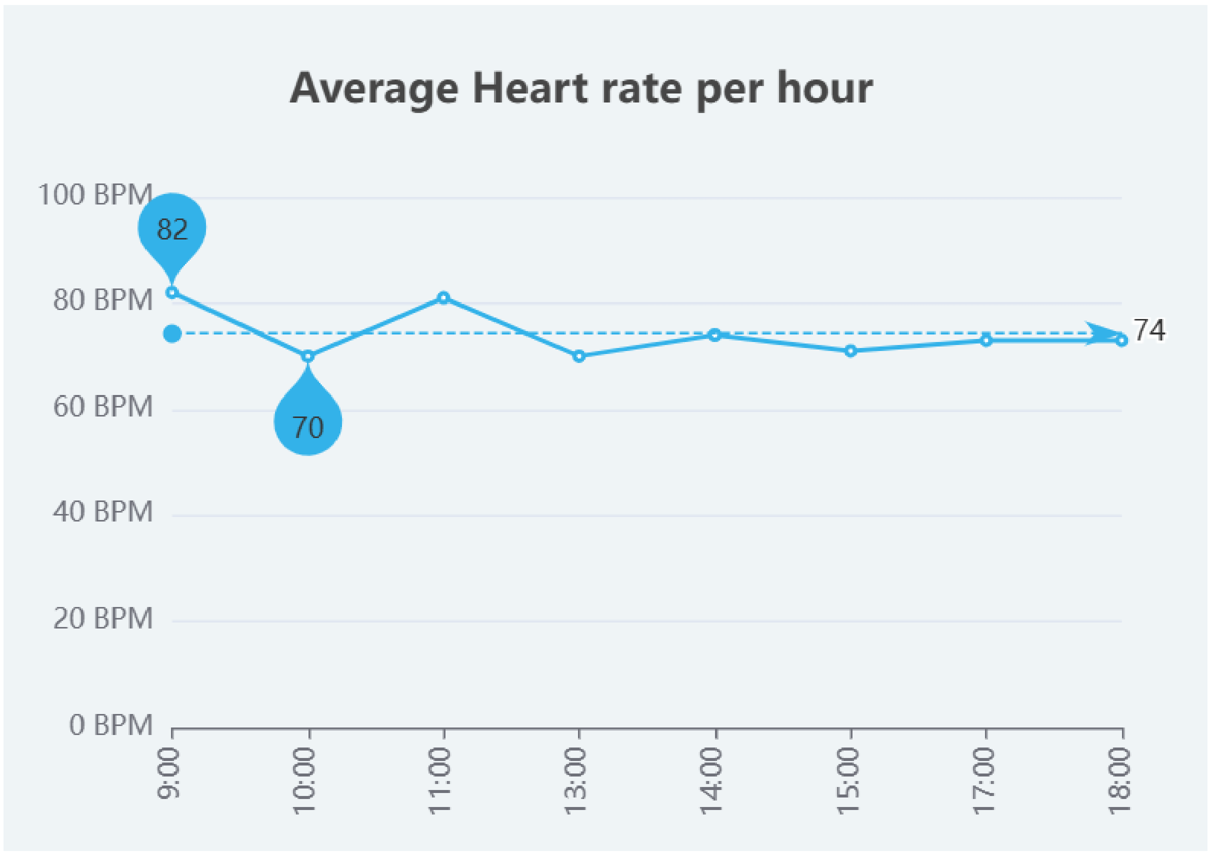


Figure 20

Web application

This section explores the design and implementation of the web application and its structural components covering the technologies used, the database schema, authentication roles, key web pages and their purposes, the notifications handling, and the results of a heuristic and usability survey.

Background research and planning

Programming languages and frameworks

The Java Spring Boot framework and MVC pattern was chosen to develop the application due to the author's familiarity and practiced experience.

A RESTful API was chosen here rather than a SOAP API due to its support of various text formats including JSON and plaintext, rather than being limited to only XML, which reduces the complexity needed and allows lightweight data transfers[83].

For the front end, the Bootstrap framework and templates[84] were used to follow the mobile-first approach in designing a responsive interface that could cater to different device sizes. HTML, CSS, and JavaScript (including jQuery and AJAX) were used for creating a dynamic user interface.

Database

A MySQL database was used due to its simplicity in setting up and its cross platform hosting ability which allows more flexibility for testers to use the database[85]. Another suitable option was MSSQL which allows Transactional-SQL which is useful for creating stored procedures and functions.

As Spring Data JPA is being used which abstracts the handling of direct SQL querying, and as MSSQL is primarily for Windows hosting, MySQL is more appropriate for the project's use case[86].

IDE

IntelliJ IDEA Ultimate was used due to familiarity, and it being an effective tool in streamlining development through its extensive range of features such as: variable auto completion, plugin marketplace, method tracking, and more.

Graphs and charts

The Apache ECharts library[87] was chosen to display the sensor readings in a meaningful and friendly format. It provides a professional feel and offers a lot of customisability.

UI and UX design

The goal of the website is to provide a user friendly interface that is clean, simple, minimalistic, professional looking, non-overwhelming, and intuitive.

The author's previous attempts at designing web interfaces have been commented for having poor aesthetics and cluttering. Due to limited web design experience, inspiration was sought for dashboard designs and overall web themes. The final website's theme and UI choices were based around Katsiatyna Lysykh's visual work (June 2022)[88] as seen in Figure 21 due to it achieving a favourable balance of desired qualities.

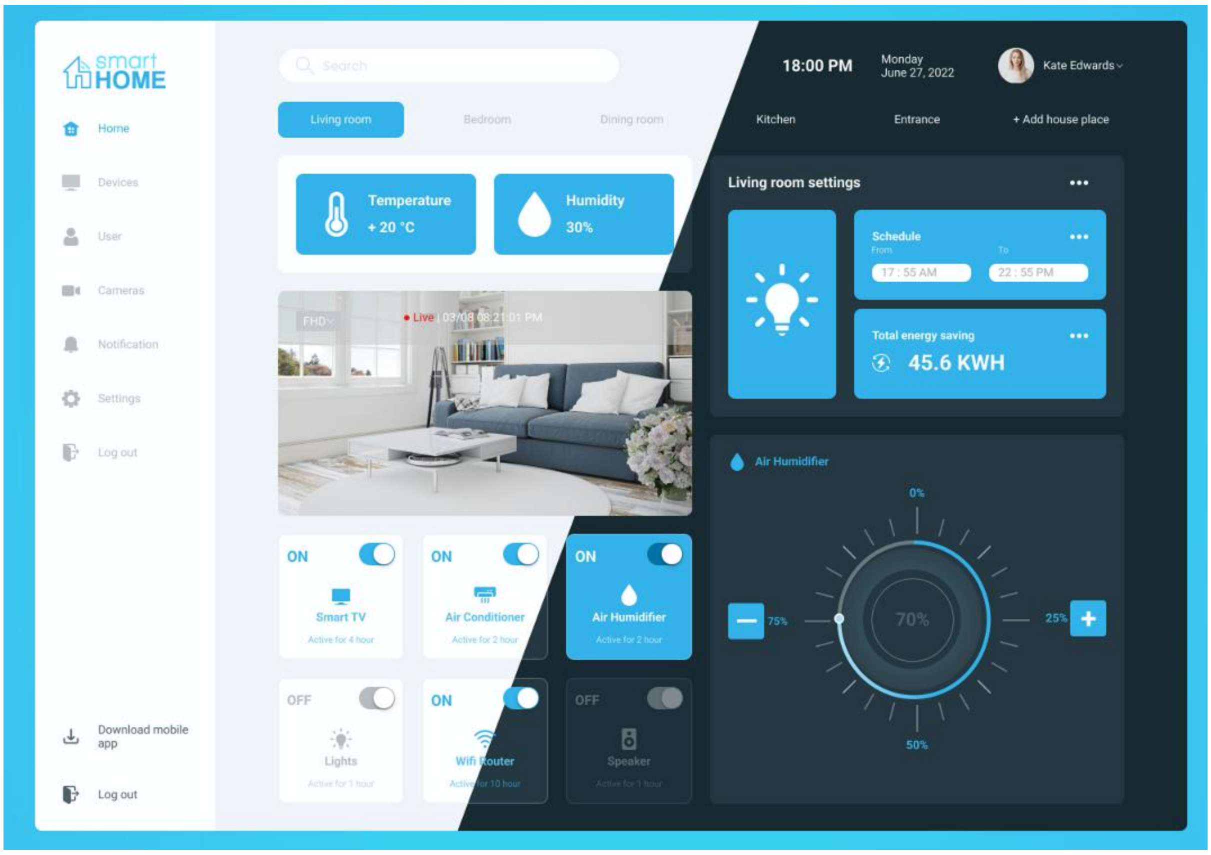


Figure 21

General design ideas were also inspired from the dashboard designs presented in Figure 22[89] and Figure 23[90].

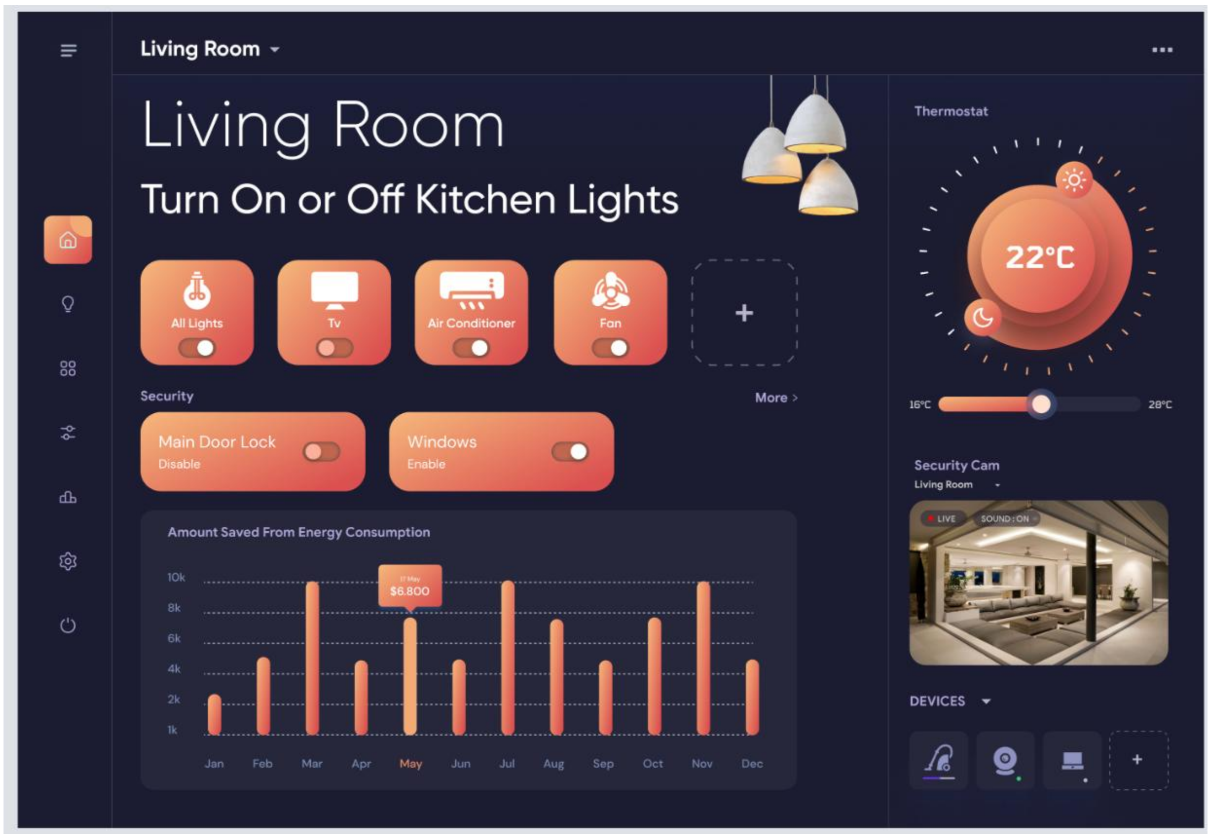


Figure 22

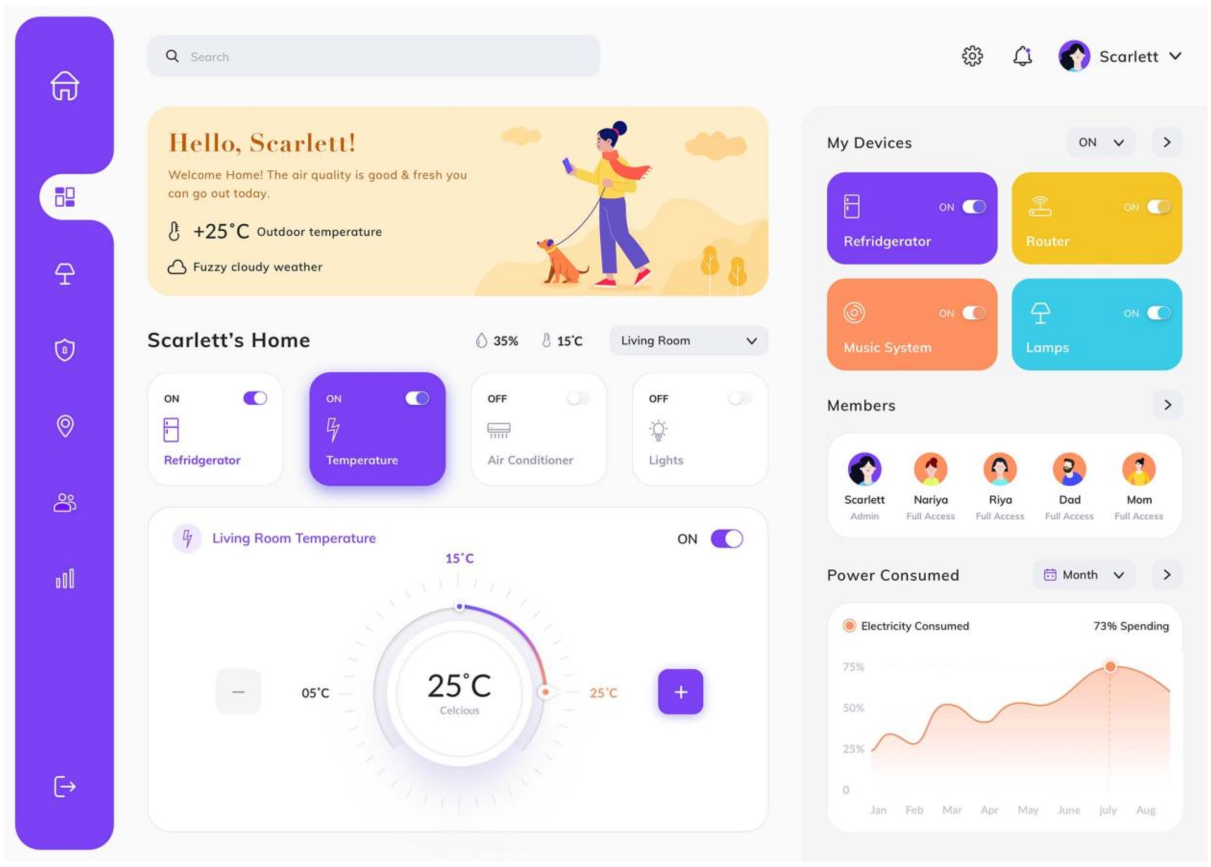


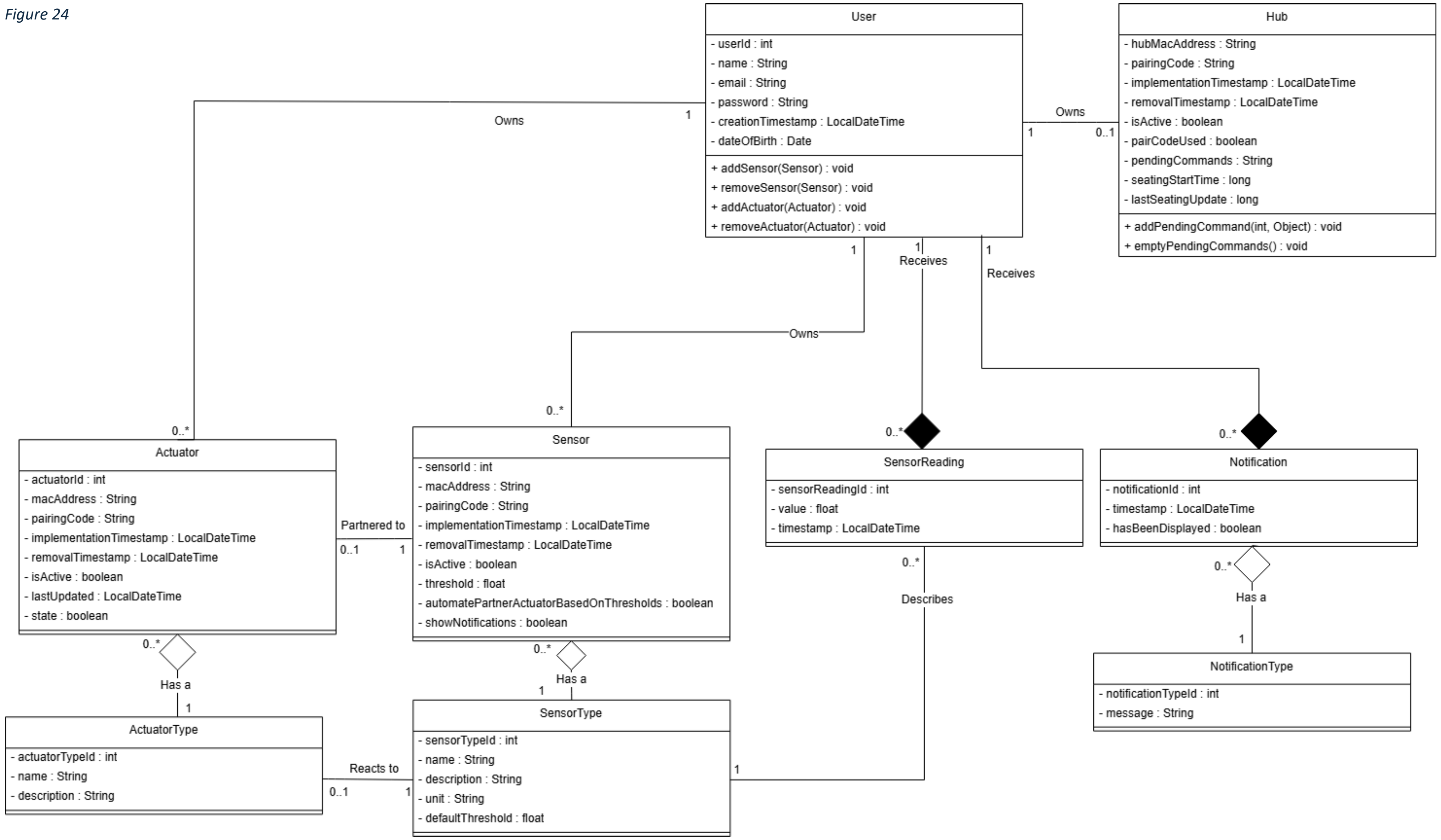
Figure 23

Implementation

Database schema

Figure 24 shows the UML class diagram of the system’s database schema which presents the entities and their relationships.

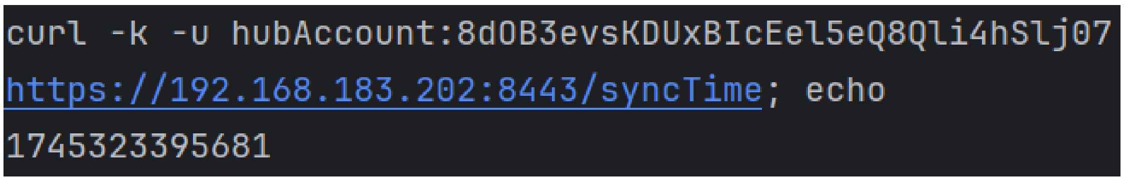
Figure 24


User
- userId : int
- name : String
- email : String
- password : String
- creationTimestamp : LocalDateTime
- dateOfBirth : Date
+ addSensor(Sensor) : void
+ removeSensor(Sensor) : void
+ addActuator(Actuator) : void
+ removeActuator(Actuator) : void
Hub
- hubMacAddress : String
- pairingCode : String
- implementationTimestamp : LocalDateTime
- removalTimestamp : LocalDateTime
- isActive : boolean
- pairCodeUsed : boolean
- pendingCommands : String
- seatingStartTime : long
- lastSeatingUpdate : long
+ addPendingCommand(int, Object) : void
+ emptyPendingCommands() : void
Owns
1
0..1
Owns
1
0..*
Owns
1
0..*
Receives
1
0..*
Receives
1
0..*
Actuator
- actuatorId : int
- macAddress : String
- pairingCode : String
- implementationTimestamp : LocalDateTime
- removalTimestamp : LocalDateTime
- isActive : boolean
- lastUpdated : LocalDateTime
- state : boolean
Partnered to
0..1
1
Sensor
- sensorId : int
- macAddress : String
- pairingCode : String
- implementationTimestamp : LocalDateTime
- removalTimestamp : LocalDateTime
- isActive : boolean
- threshold : float
- automatePartnerActuatorBasedOnThresholds : boolean
- showNotifications : boolean
SensorReading
- sensorReadingId : int
- value : float
- timestamp : LocalDateTime
Notification
- notificationId : int
- timestamp : LocalDateTime
- hasBeenDisplayed : boolean
0..*
Has a
1
0..*
Has a
1
0..*
Describes
1
0..*
Has a
1
ActuatorType
- actuatorTypeId : int
- name : String
- description : String
Reacts to
0..1
1
SensorType
- sensorTypeId : int
- name : String
- description : String
- unit : String
- defaultThreshold : float
NotificationType
- notificationTypeId : int
- message : String

User roles

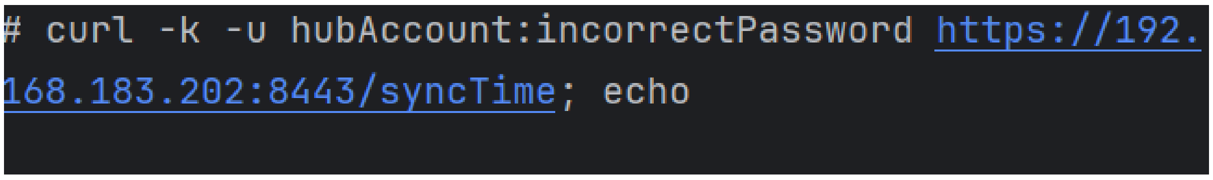
There are 3 authentication roles: “USER”,”HUB”, and “ADMIN”. The hub and admin roles are part of the static records created in the initialisation of the project, so this means only new user roles can be created at run time.

The hub can access endpoints in the HubController class. This is achieved through the use of HTTP Basic Authentication which sends the hub’s fixed credentials encoded in Base 64, over HTTPS[91]. Figure 25 shows a successful request using the correct authentication details, and Figure 26 shows the unsuccessful request (blank) using incorrect details.



```
curl -k -u hubAccount:8dOB3evsKDUxBIcEel5eQ8Qli4hSlj07
https://192.168.183.202:8443/syncTime; echo
1745323395681
```

Figure 25



```
# curl -k -u hubAccount:incorrectPassword https://192.
168.183.202:8443/syncTime; echo
```

Figure 26

The admin can access all non-hub protected endpoints, including “/swagger-ui/index.html” which is a user interface to the API. As there is not a general admin that overlooks the data of the users, the main purpose of the rights granted to this role is to be able to register new devices via a web interface through AdminController class endpoints.

Logging

The application is configured to store errors in a log file which makes it easier to debug unknown issues.

Key web pages

Each page follows a consistent format, includes relevant error handling and input sanitation, and every 30 seconds, the page checks if there are any available notifications to show and displays one if so. Animated icons are used throughout to support the interactivity, and are sourced from Lordicon’s icon library through JavaScript[92].

Dashboard

The dashboard shows graphs of today’s readings (Figure 27) and past readings (Figure 28) for each sensor. The past readings are grouped by selectable sections: “Last 7 days”, “Last 30 days”, “All time” and a “Custom range”, where they can select the start and end date durations (Figure 29).

Users can remote control the actuators from here.

Alongside the notification checking, the state of the actuators are checked every 30 seconds to dynamically update the display states.

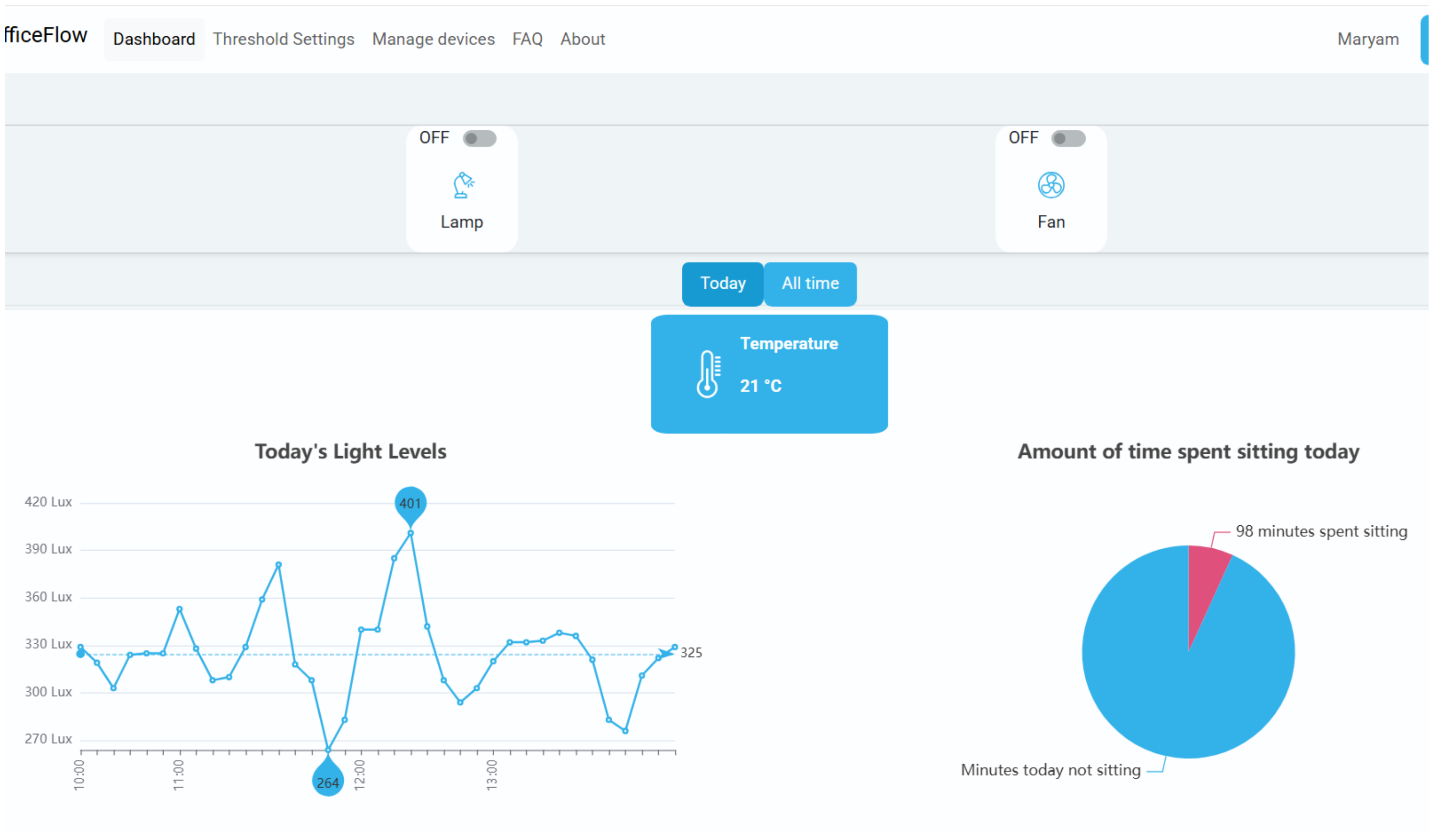


Figure 27

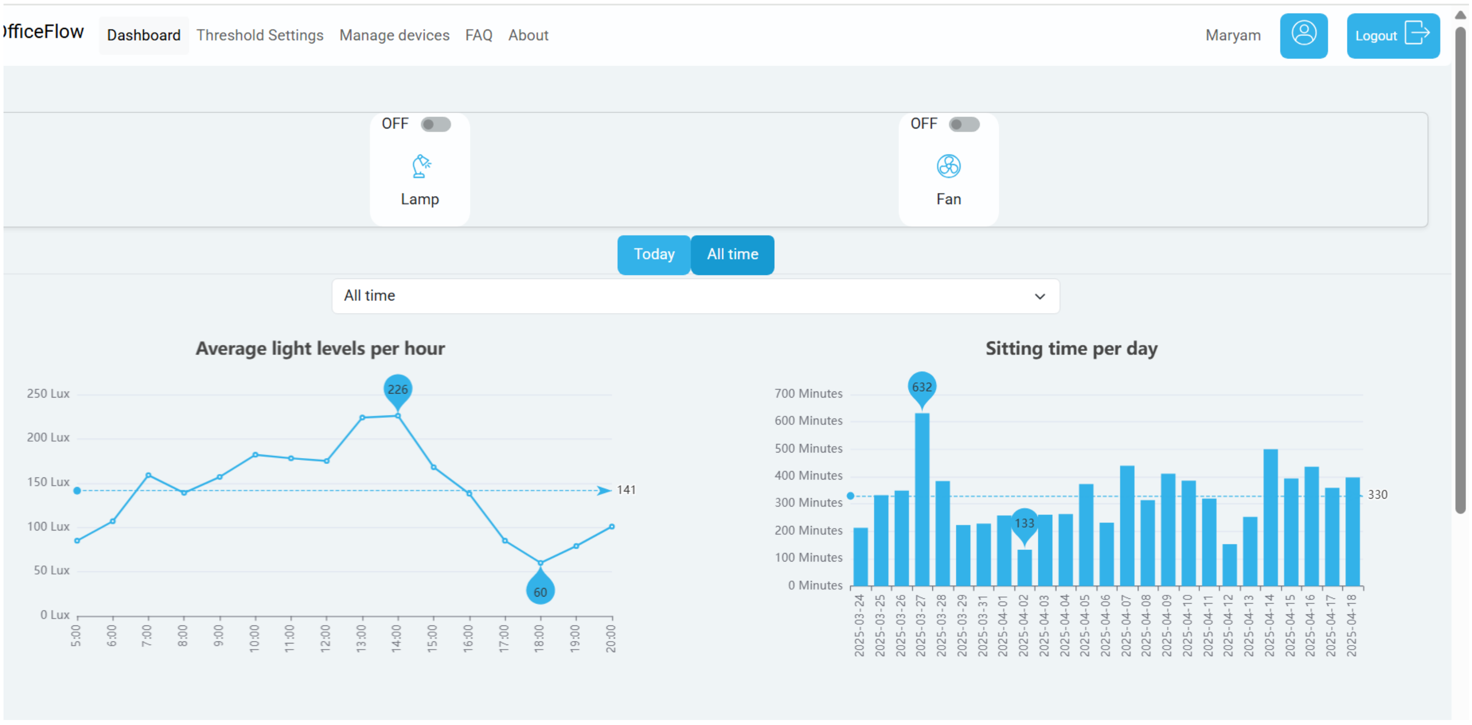


Figure 28

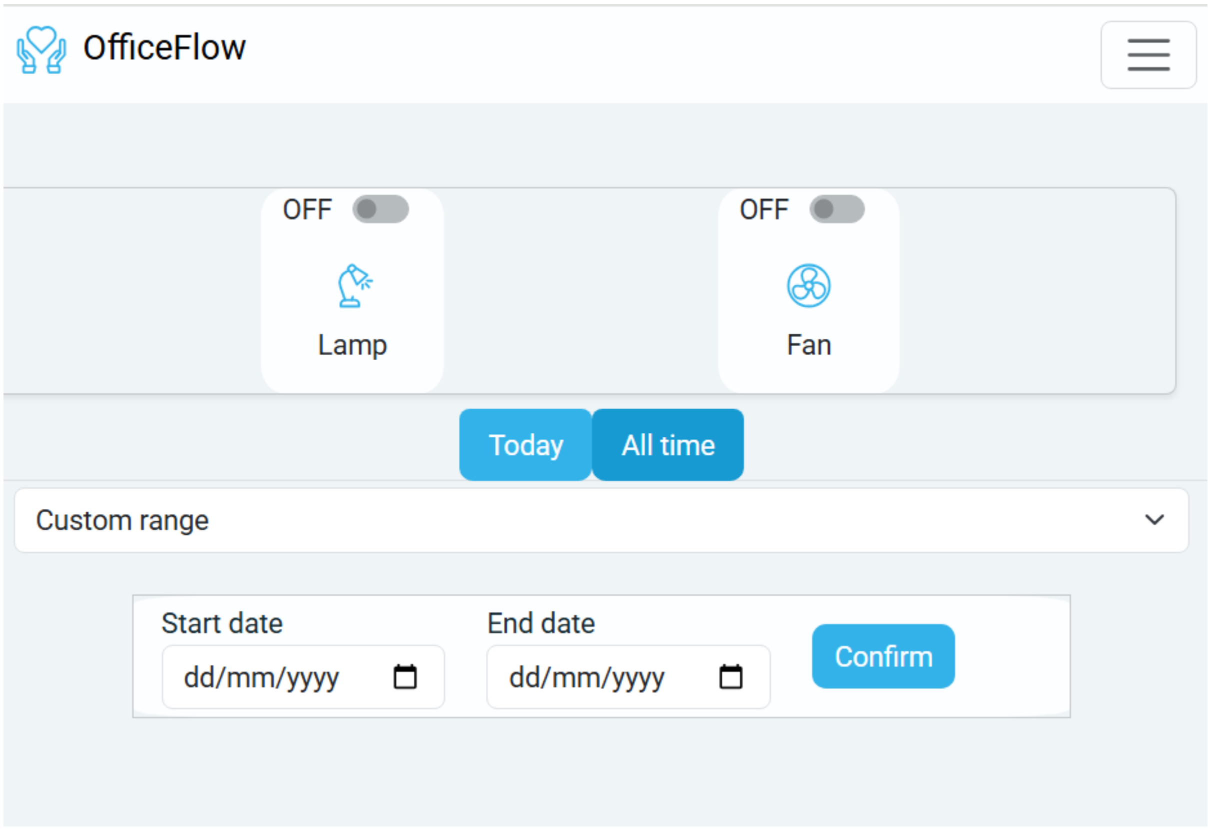
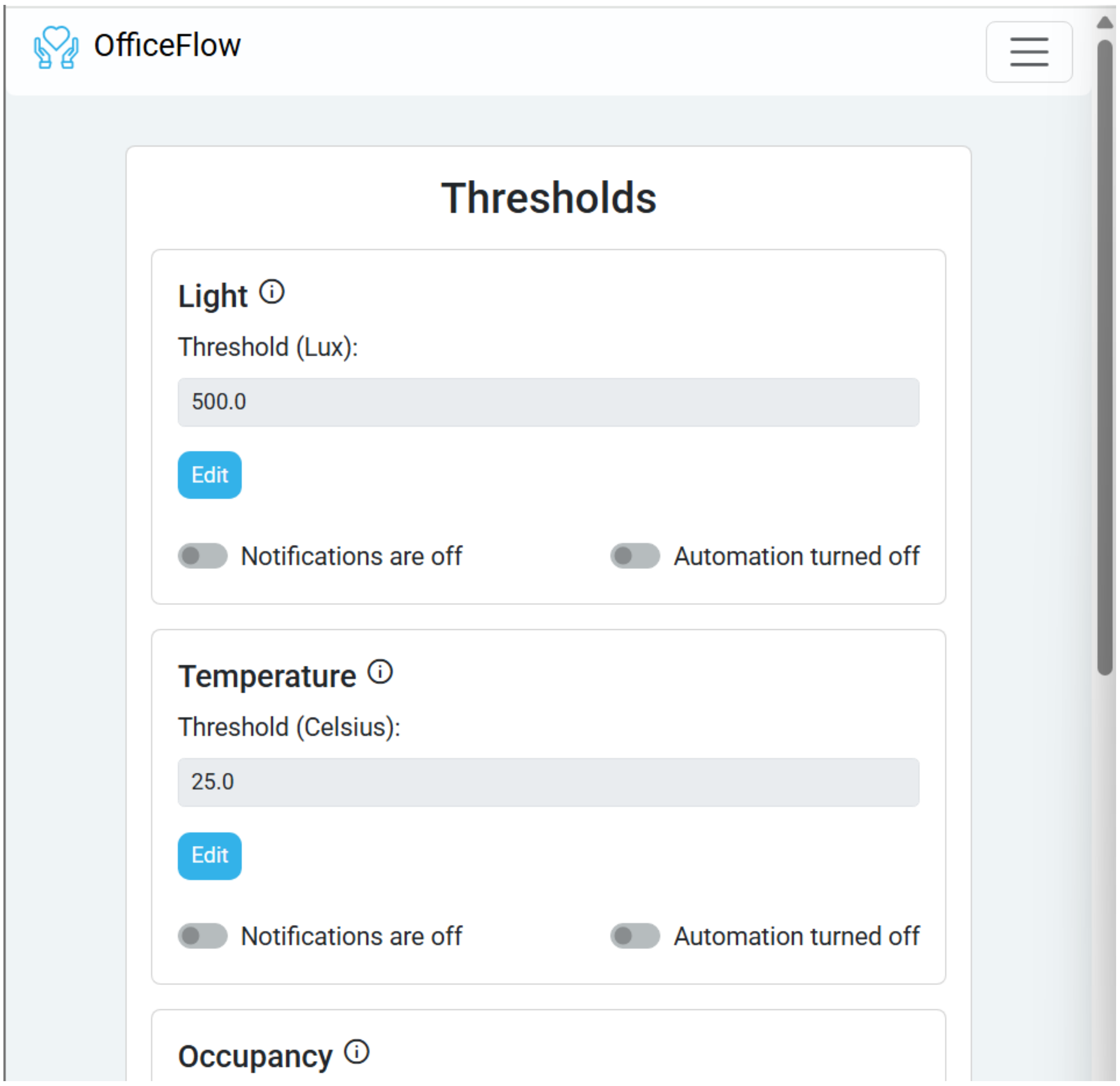


Figure 29

Threshold Settings

Figure 30 shows the threshold settings page where users can enable/disable automations and notifications, as well as adjust the threshold values. There are also info tooltips (from Bootstrap[84] and PopperJS[93]), that provides further guidance to the user, as seen in Figure 31.



OfficeFlow

Thresholds

Light ⓘ

Threshold (Lux):

500.0

Edit

Notifications are off — Automation turned off

Temperature ⓘ

Threshold (Celsius):

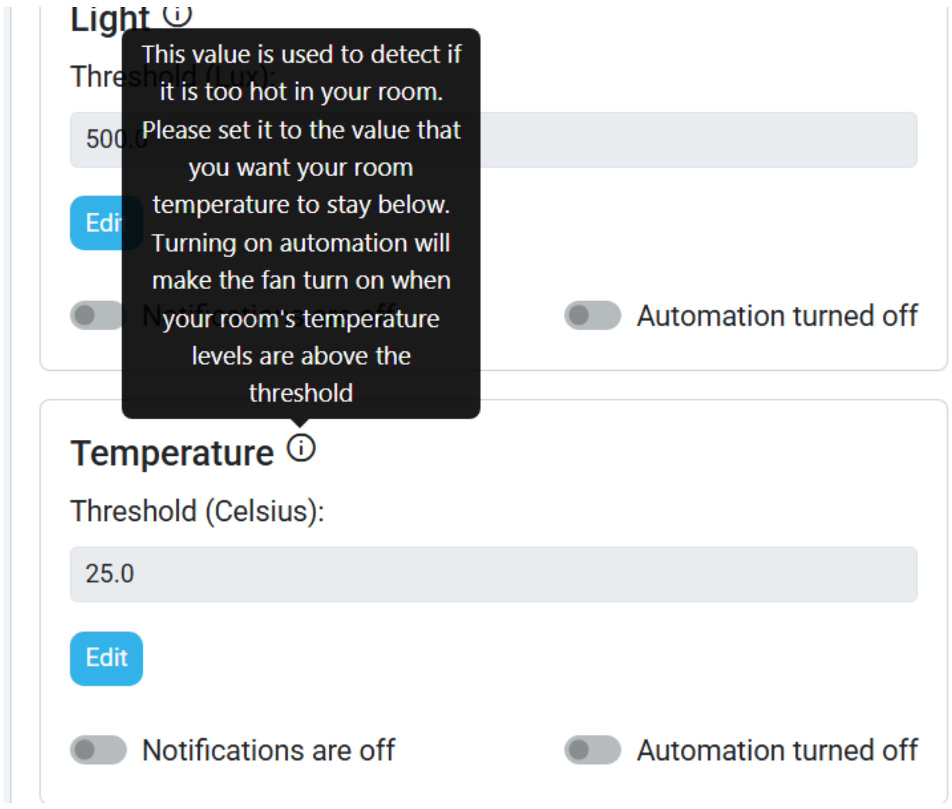
25.0

Edit

Notifications are off — Automation turned off

Occupancy ⓘ

Figure 30



Light

Threshold (Lux):

500.0

Edit

Notifications are off

Automation turned off

This value is used to detect if it is too hot in your room. Please set it to the value that you want your room temperature to stay below. Turning on automation will make the fan turn on when your room's temperature levels are above the threshold

Temperature

Threshold (Celsius):

25.0

Edit

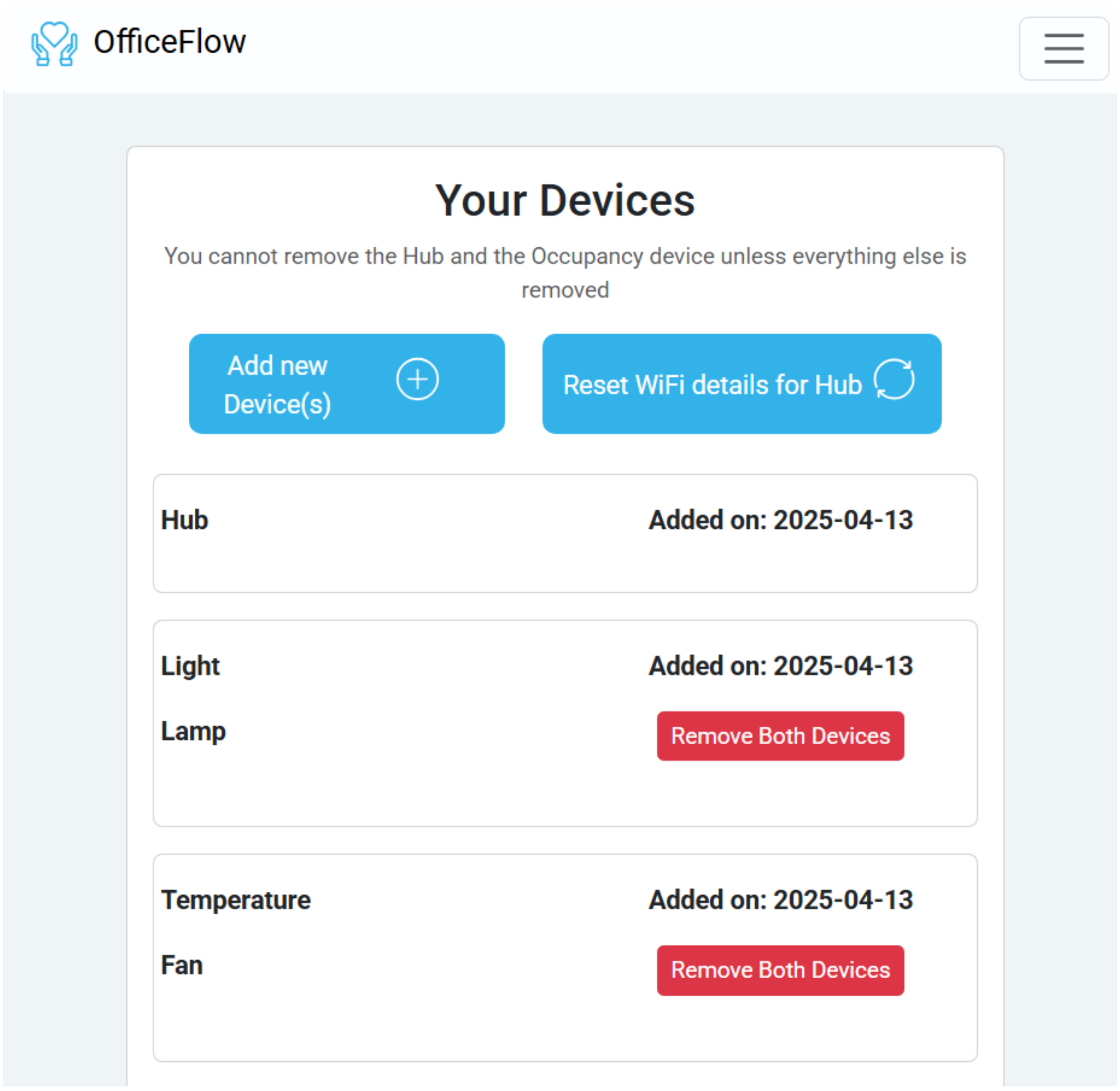
Notifications are off

Automation turned off

Figure 31

Manage Devices

Figure 32 shows the manage devices page, where users can be redirected to adding a new device, reset Wi-Fi details for the hub (by sending it command 112), and remove devices from their network (which does not delete readings). Removed devices can be paired again.


OfficeFlow

Your Devices

You cannot remove the Hub and the Occupancy device unless everything else is removed

Hub Added on: 2025-04-13

Light Added on: 2025-04-13

Lamp

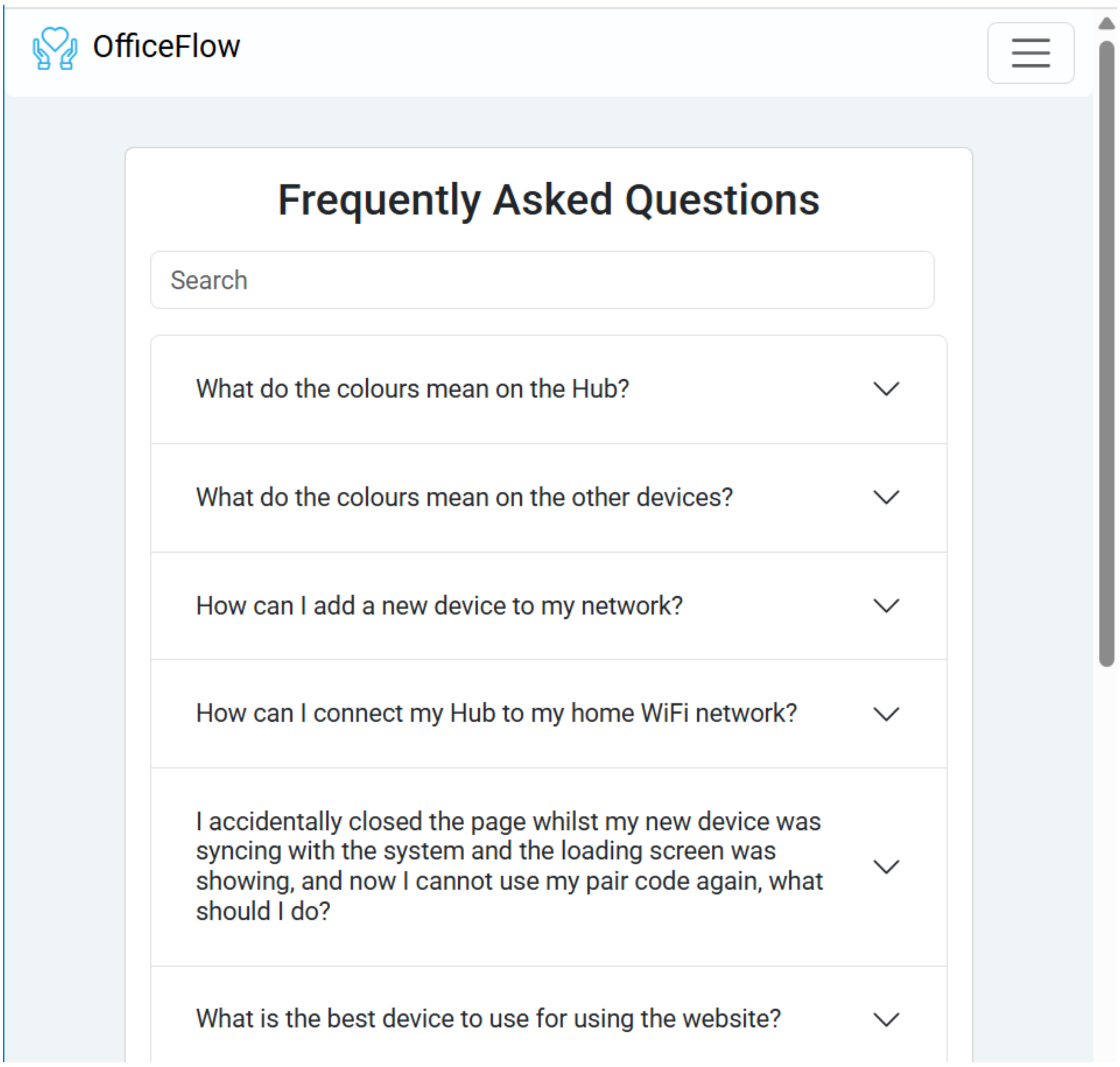
Temperature Added on: 2025-04-13

Fan

Figure 32

FAQ

Figure 33 shows the FAQ page where there is a collection of questions and answers to help guide the user. A search bar is included for easy filtering.



Frequently Asked Questions

Search

What do the colours mean on the Hub?

What do the colours mean on the other devices?

How can I add a new device to my network?

How can I connect my Hub to my home WiFi network?

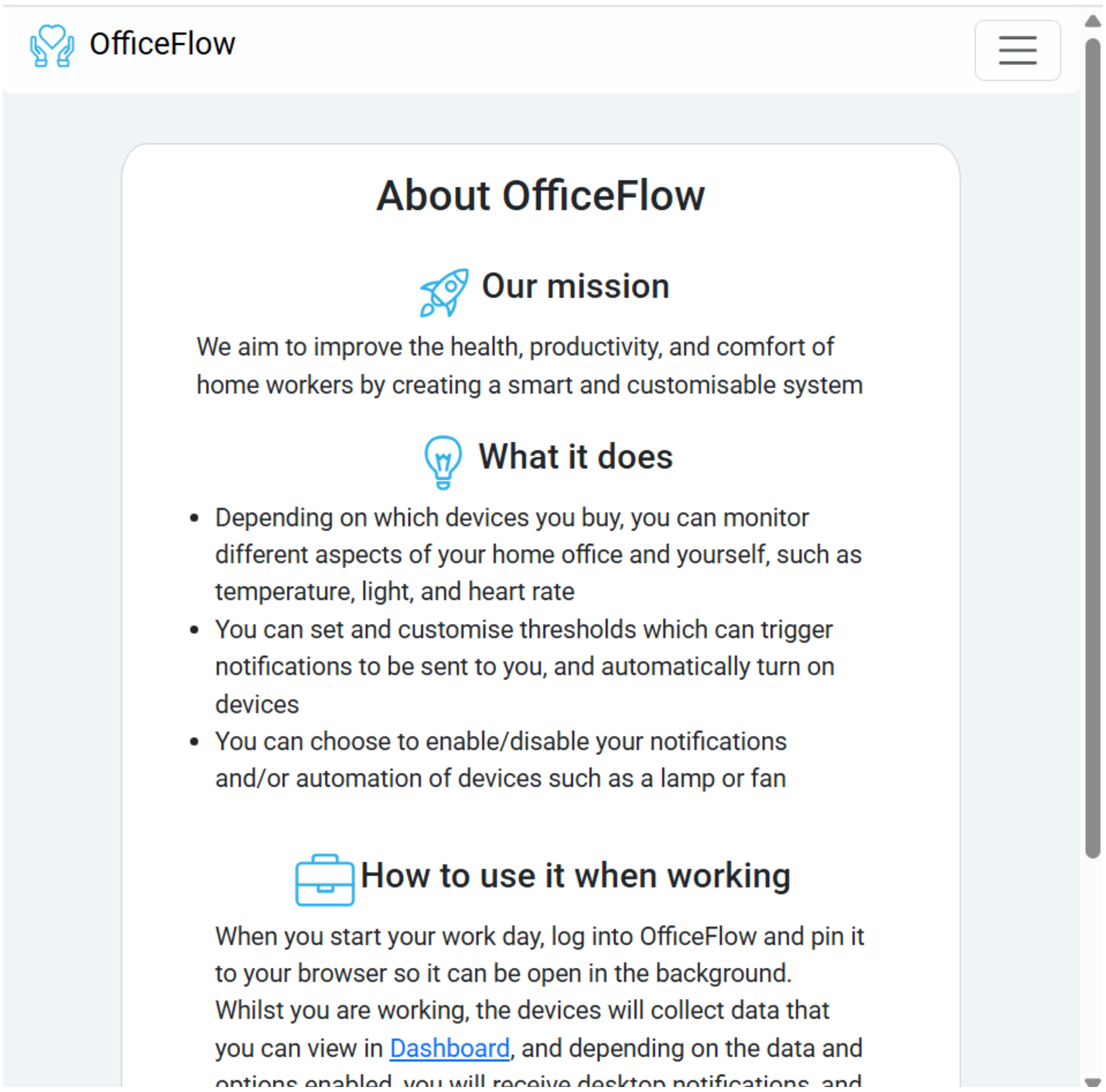
I accidentally closed the page whilst my new device was syncing with the system and the loading screen was showing, and now I cannot use my pair code again, what should I do?

What is the best device to use for using the website?

Figure 33

About

Figure 34 shows the about page which introduces the context and motivation of the system to the user, particularly as this system is seen as containing devices that can be "bought" individually or in pairs, and so needs an introduction of the selling point to the user due to the absence of promotional materials (that are out of scope for this project).


OfficeFlow

About OfficeFlow

Our mission

We aim to improve the health, productivity, and comfort of home workers by creating a smart and customisable system

What it does

- Depending on which devices you buy, you can monitor different aspects of your home office and yourself, such as temperature, light, and heart rate
- You can set and customise thresholds which can trigger notifications to be sent to you, and automatically turn on devices
- You can choose to enable/disable your notifications and/or automation of devices such as a lamp or fan

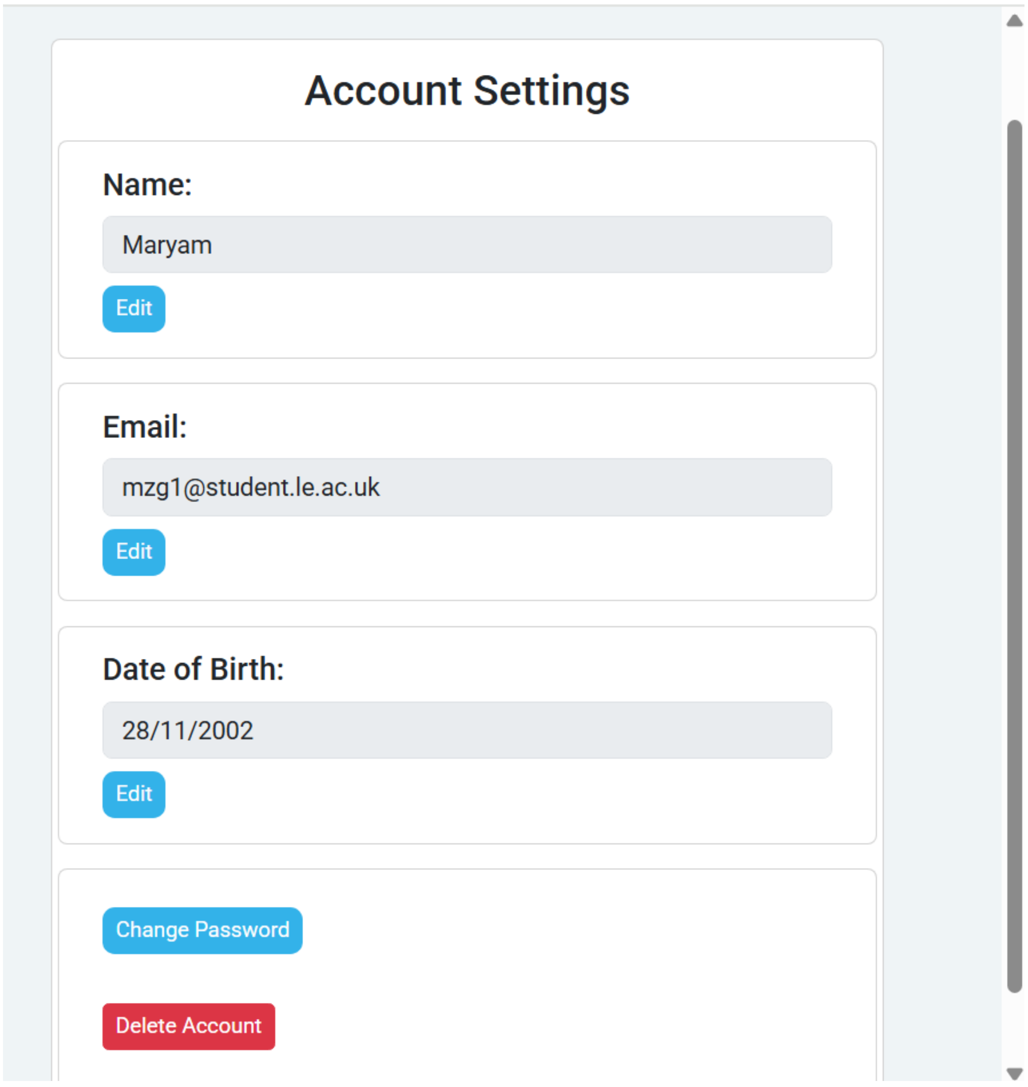
How to use it when working

When you start your work day, log into OfficeFlow and pin it to your browser so it can be open in the background. Whilst you are working, the devices will collect data that you can view in Dashboard, and depending on the data and

Figure 34

Account Settings

Figure 35 shows the account settings page which allows the user to edit their details, and delete their account which removes all their devices and deletes all their readings.


Account Settings

Name:

Maryam

Email:

mzg1@student.le.ac.uk

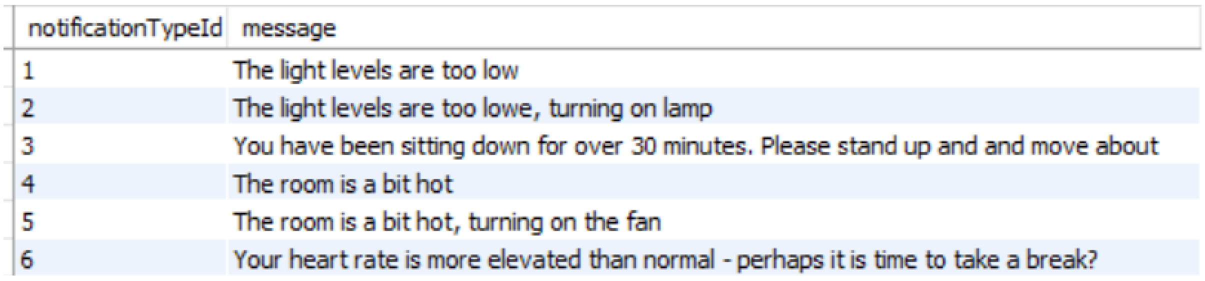
Date of Birth:

28/11/2002

Figure 35

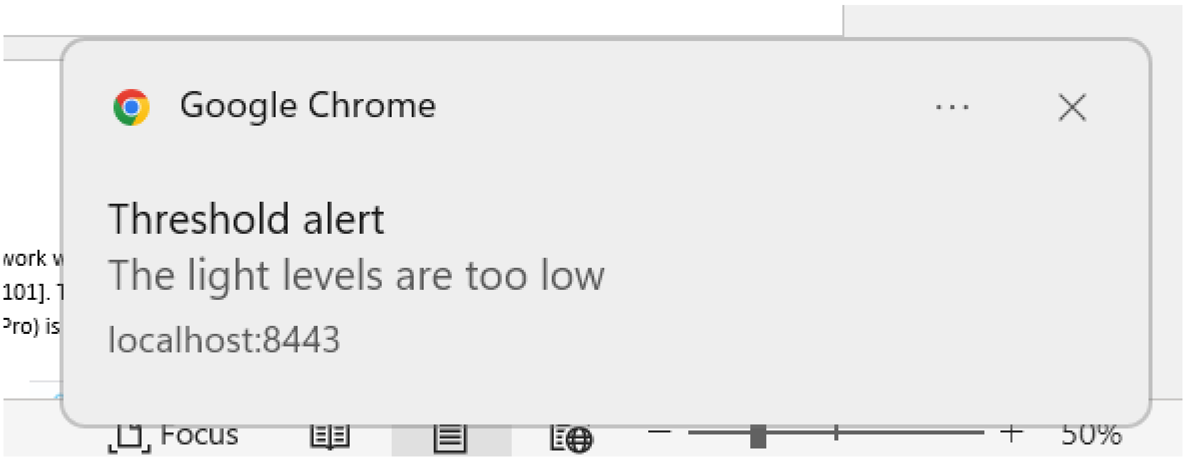
Notification alerts

Mozilla's Notification API[94] is used to implement notifications in the application. The earliest notification created in the last 5 minutes are displayed. Notifications are created when a sensor reading is uploaded. There are 6 notification types to create a notification from, and they vary depending on the sensor type and if the automation option is enabled or not, as seen in Figure 36. Figure 37 shows an example of a notification. Currently, the notification feature only works on popular browsers used on computers, such as Chrome and Firefox, but fails to show on Android device browsers such as Samsung Internet or Chrome due to compatibility issues. This is an acceptable limitation as the website is designed to be open in the background on the user's computer, whilst they are working.



notificationTypeId	message
1	The light levels are too low
2	The light levels are too lowe, turning on lamp
3	You have been sitting down for over 30 minutes. Please stand up and and move about
4	The room is a bit hot
5	The room is a bit hot, turning on the fan
6	Your heart rate is more elevated than normal - perhaps it is time to take a break?

Figure 36



Google Chrome

Threshold alert

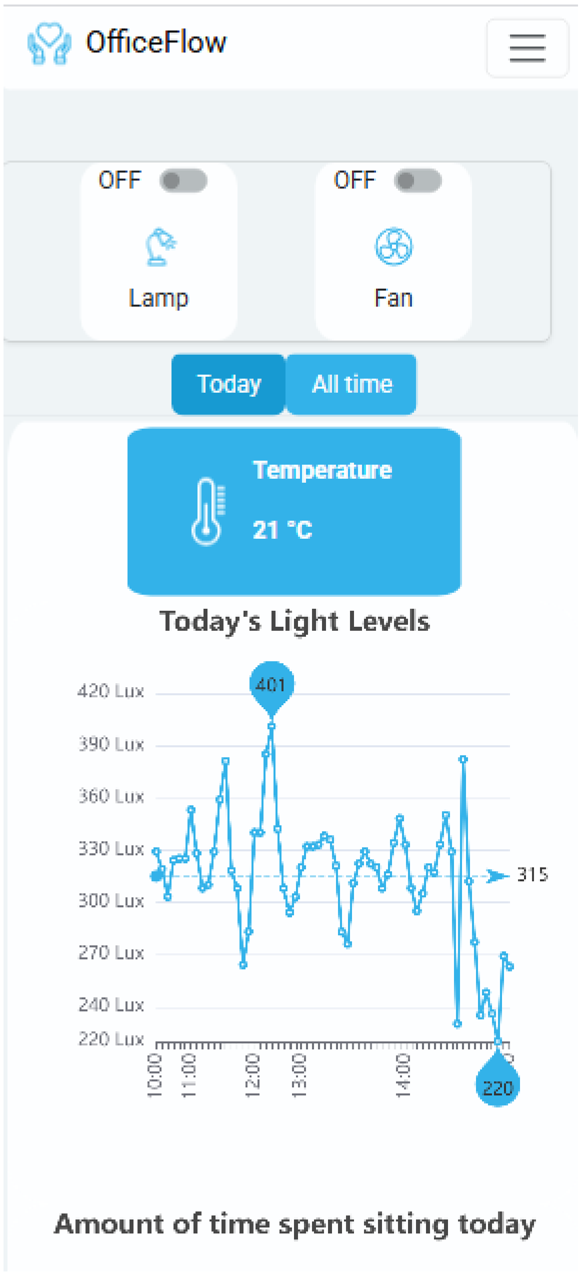
The light levels are too low

localhost:8443

Figure 37

Dynamic device sizing

The web application on a mobile’s interface (iPhone 12 Pro) is shown in Figure 38 and Figure 39.


OfficeFlow

OFF Lamp

OFF Fan

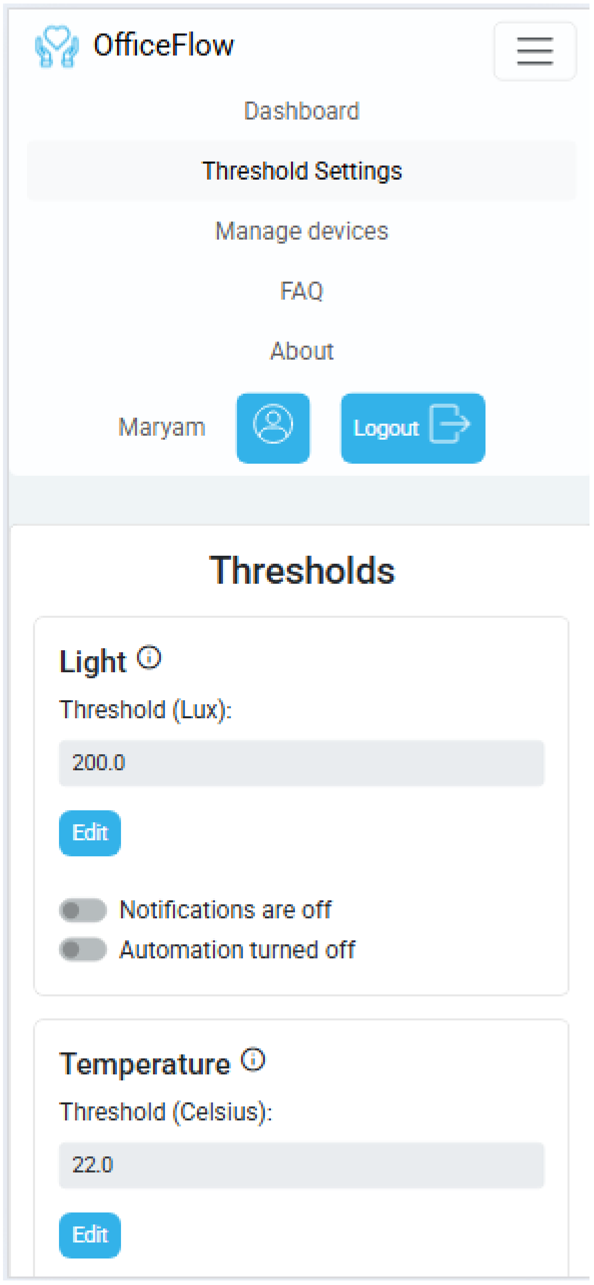
Today All time

Temperature 21 °C

Today's Light Levels

Amount of time spent sitting today

Figure 38


OfficeFlow

Thresholds

Light

Threshold (Lux): 200.0

Edit

Notifications are off

Automation turned off

Temperature

Threshold (Celsius): 22.0

Edit

Figure 39

Heuristic and usability testing

A Nielsen's heuristic and usability evaluation was performed by 3 testers on the web application that had pre-populated data (to remove the need for more hardware prototypes). They are considered technical users as they have a degree in a computing related field. The aim was to gather feedback on what end users would expect and to help cater for the oversights of the author. Figure 64 to Figure 68 in the Appendix show the distributed testing form and instructed tasks.

Insightful results were received, the following lists the key appreciated features of the application:

- The simple and easy to use UI
- The search feature in the FAQ
- The user tour (IntroJS)
- The error handling for user input fields
- The system's responsiveness of the user's actions
- The graphs' style

The following lists improvement suggestions derived from the negative feedback:

1. Desire for having the user enter the current password before changing the user's password
2. Desire for a logout confirmation button to aid user mis clicks
3. Desire for a dark mode option for accessibility
4. Desire for including an additional indicator on where to access the account settings page
5. Desire for more information about the application's context

In response to the feedback, the following changes have been made:

1. An additional field has been added to the account settings page, where the user will be unable to change their password without entering the correct current password.
2. An additional logout confirmation modal now shows before logging the user out.
3. The dark mode option was unable to be implemented due to the limited time constraint, but has been listed in the future improvements backlog.
4. A hovering tooltip has been added to the account settings navbar button for extra direction.
5. An "About" page has been implemented to provide foundational context of the project.

Setting up process flows

This following section covers the implementation flow of the system when the user sets up their hub, occupancy sensor, and a new sensor and actuator device. This will provide an in depth look at key logic and design features used in this project, as well as linking the previously mentioned components together to see them in action.

Setting up the Hub

Simplified overview

The hub has 2 set up modes, the first is when there are no connected devices stored so it starts communicating directly with the API to set up the core devices, which is the hub and the occupancy sensor. The second is the normal routine set up, where the hub has at least the occupancy sensor's details stored.

The hub and the occupancy sensor are considered the core devices due to the emphasis on the system only collecting readings whilst the user is working and present. They are 2 separate devices as they both require different placements in the room: the hub would need somewhere where there is strong Wi-Fi signal, and the sensor would need to be in close proximity to the user.

This section and the next (Setting up the Occupancy sensor) follow the hub's first set up routine. The nodes always follow the same set up routine regardless if they have been linked yet or not. If they have not previously been linked with the hub, then they are prevented from sending messages, taking readings, and doing normal operation tasks.

The following figures show the simplified overview of steps of when a user sets up their Hub in sequence.

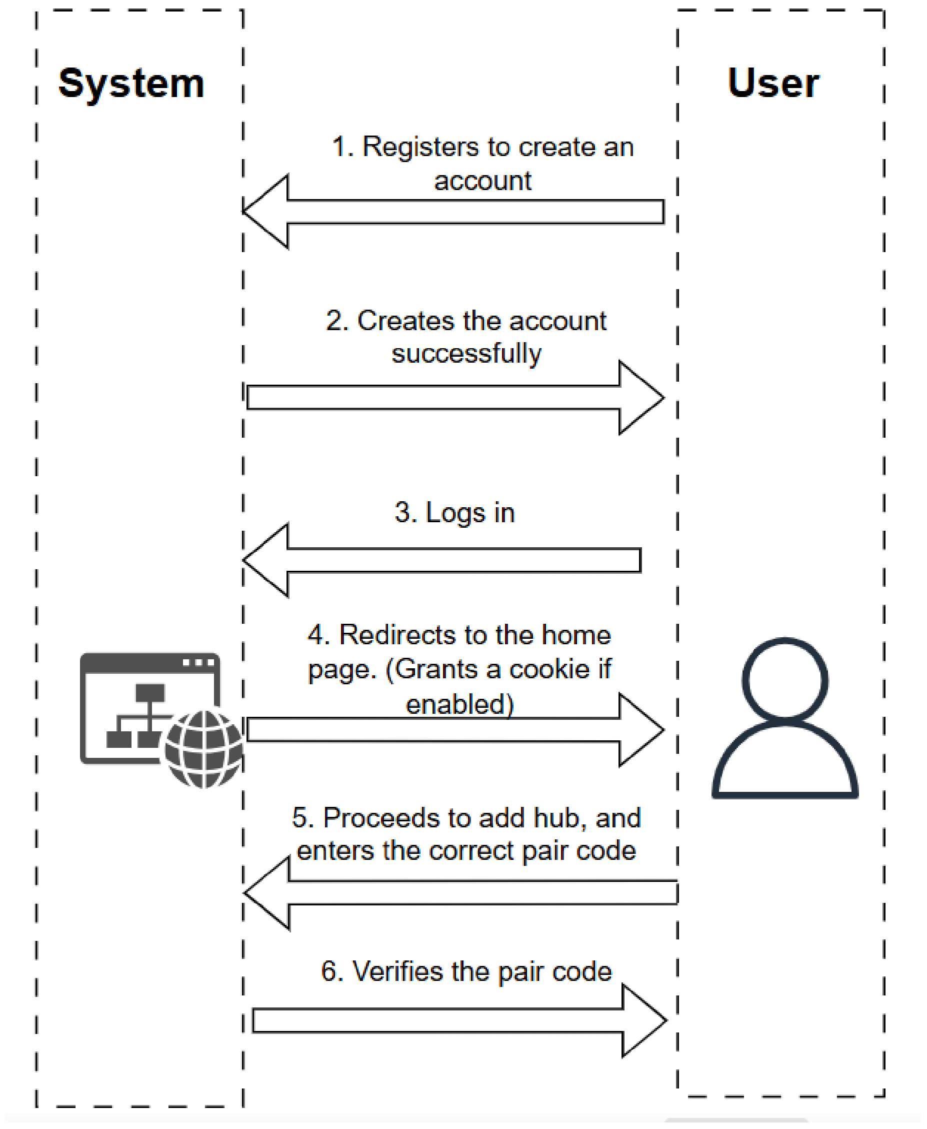

System
User
1. Registers to create an account
2. Creates the account successfully
3. Logs in
4. Redirects to the home page. (Grants a cookie if enabled)
5. Proceeds to add hub, and enters the correct pair code
6. Verifies the pair code

Figure 40

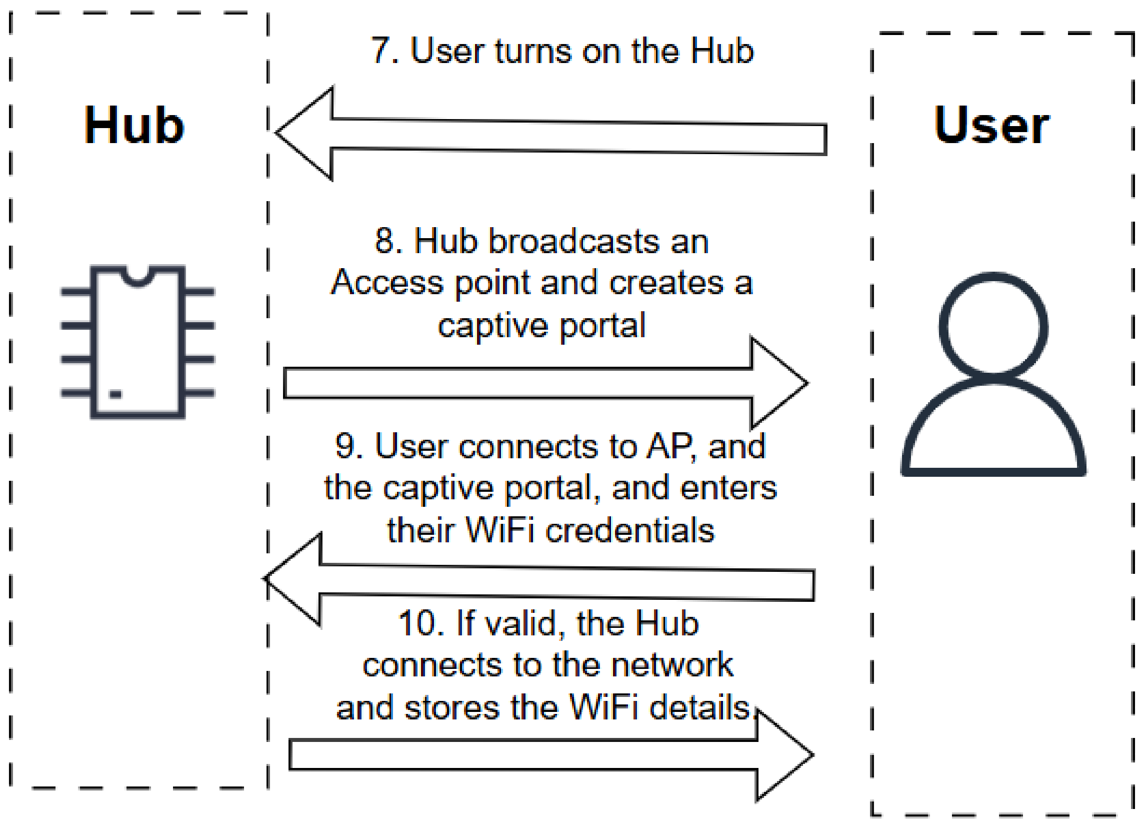

Hub
User
7. User turns on the Hub
8. Hub broadcasts an Access point and creates a captive portal
9. User connects to AP, and the captive portal, and enters their WiFi credentials
10. If valid, the Hub connects to the network and stores the WiFi details

Figure 41

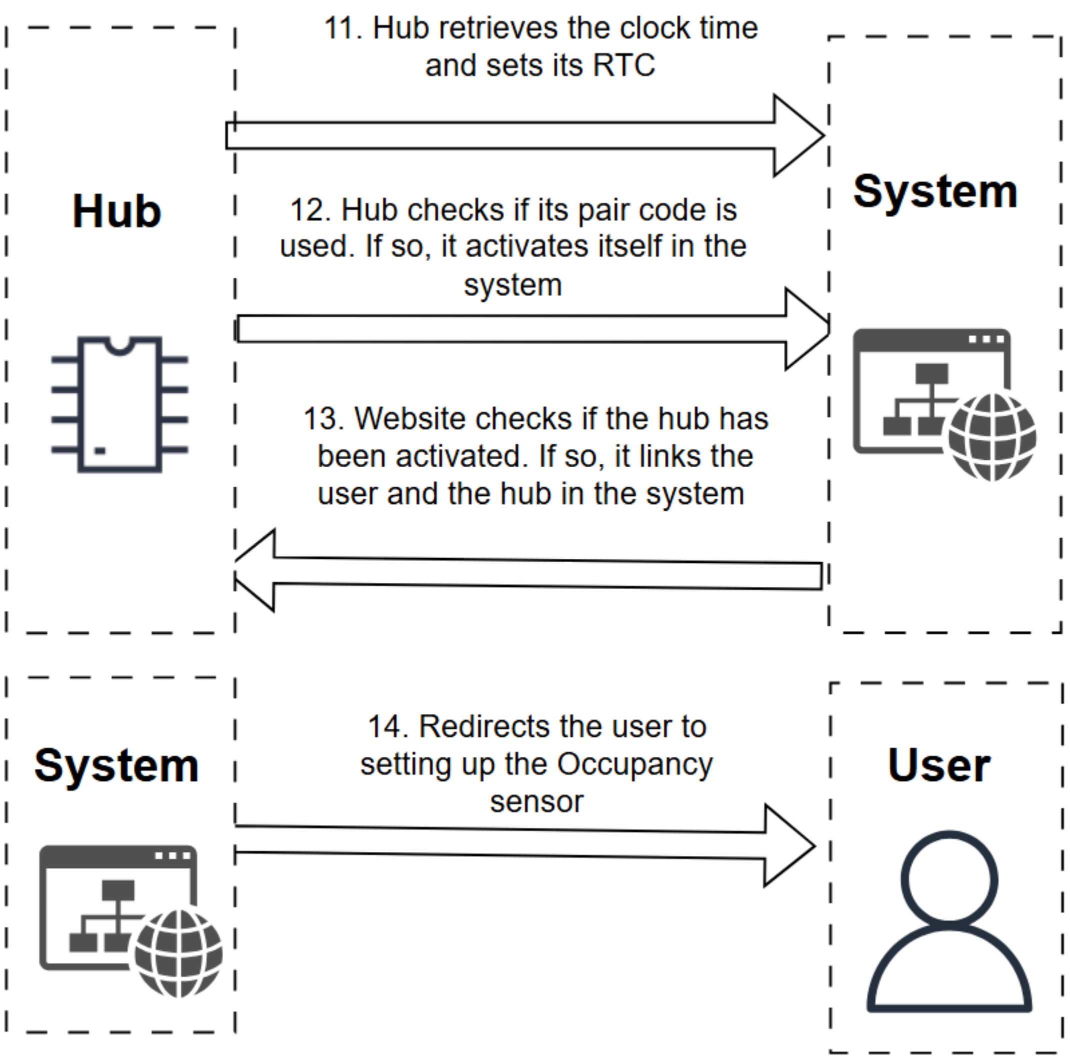
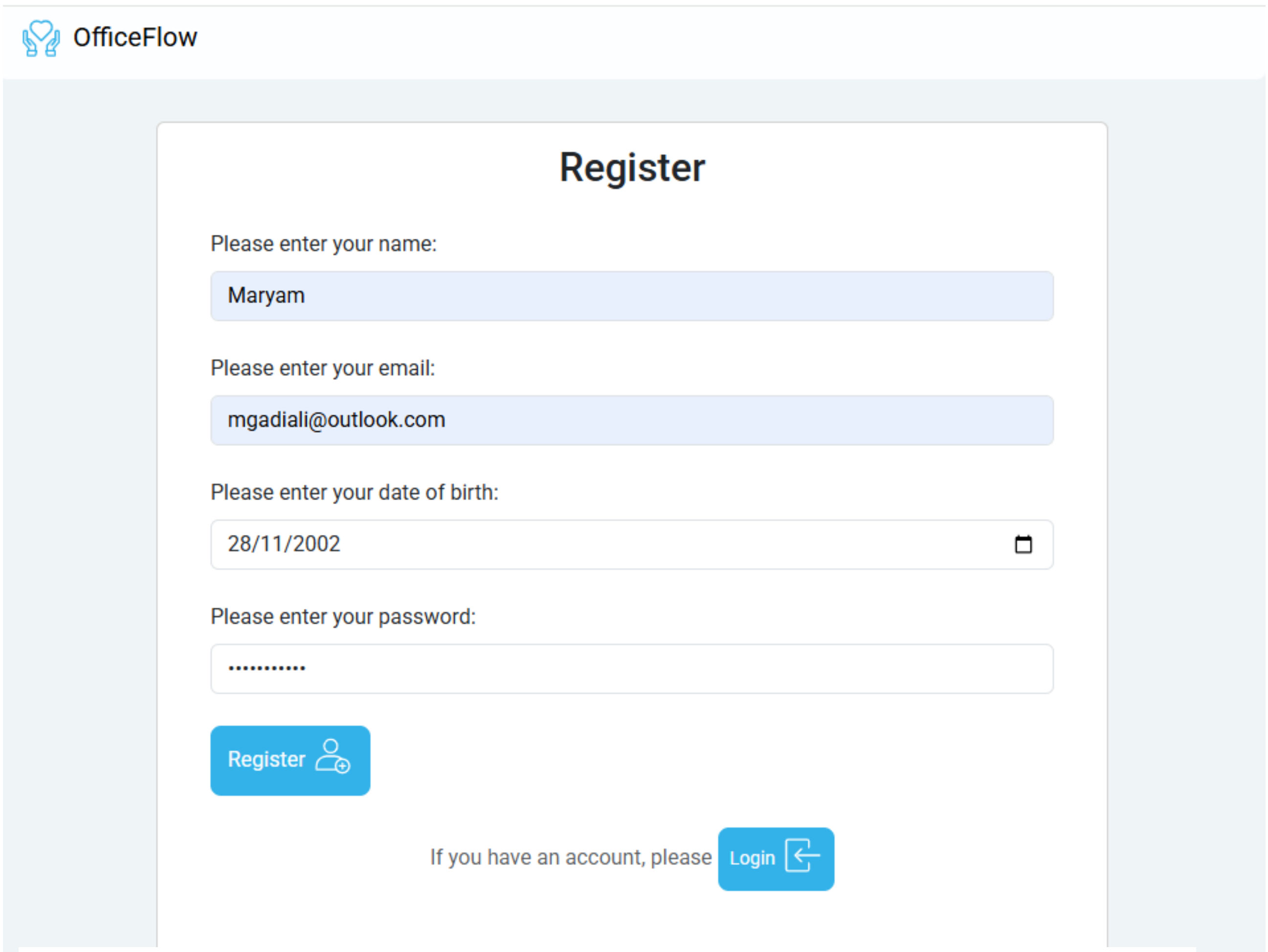

11. Hub retrieves the clock time and sets its RTC
Hub
System
12. Hub checks if its pair code is used. If so, it activates itself in the system
13. Website checks if the hub has been activated. If so, it links the user and the hub in the system
System
14. Redirects the user to setting up the Occupancy sensor
User

Figure 42

Detailed overview

This section goes into further depth of each step outlined in the above "Simplified overview" section.

1. First the user creates an account with their name, email, date of birth (DOB) and password in a register form, as seen in Figure 43. The DOB is included as a field for future potential cases where age verification may be needed in order to comply with GDPR data processing rules[95]. Another potential future use of DOB is for future identity verification in case the user forgets their credentials.



Register

Please enter your name:

Maryam

Please enter your email:

mgadiali@outlook.com

Please enter your date of birth:

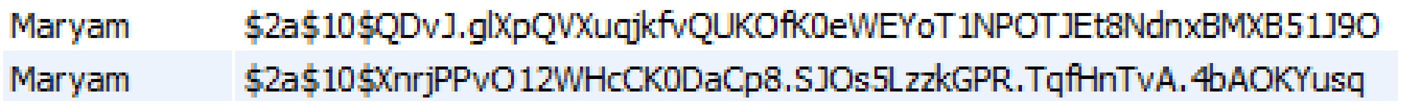
28/11/2002

Please enter your password:

•••••••••••

Figure 43

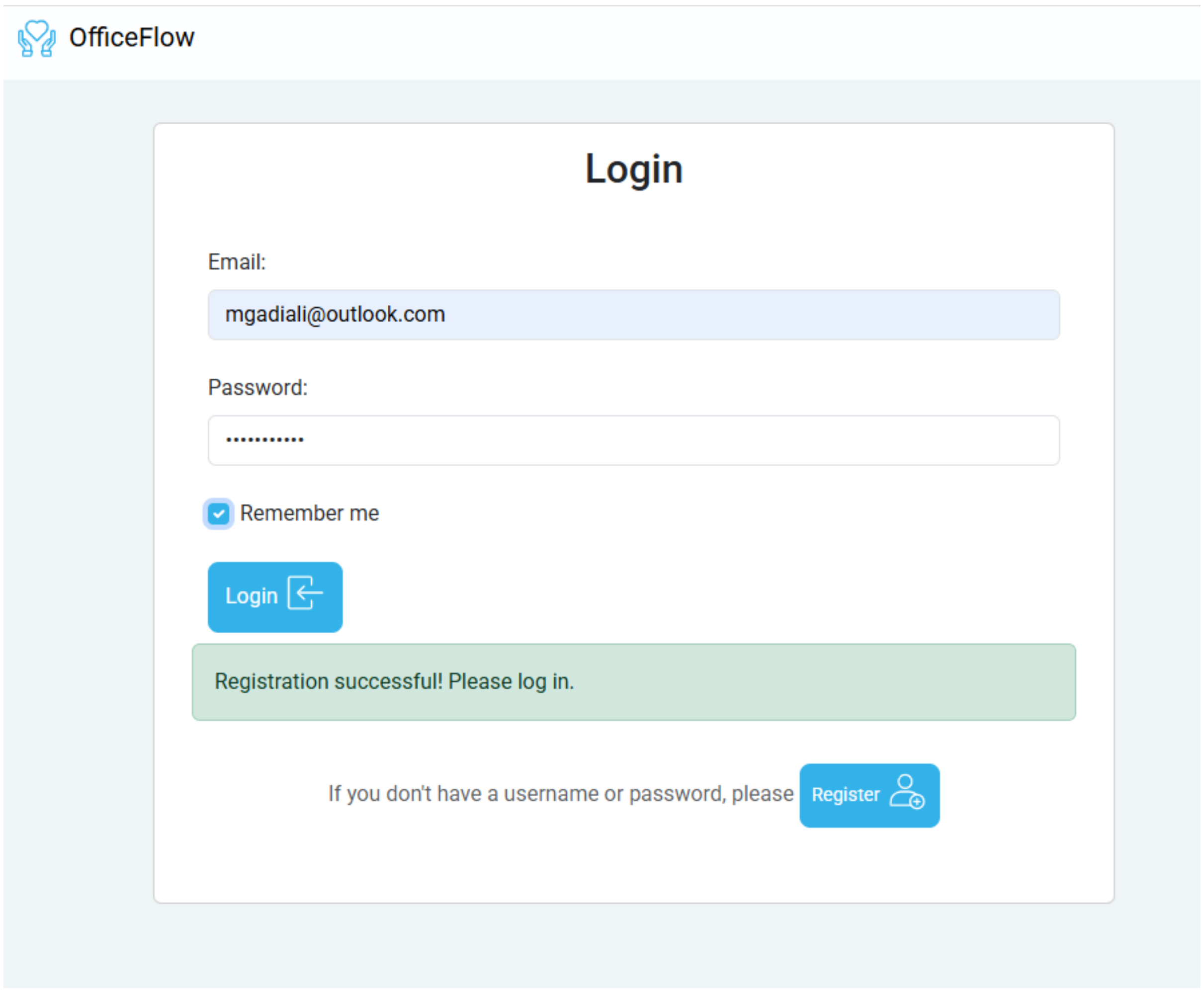
2. After the user has successfully created an account, their data is stored in the database. Their password is stored with a password encoder from Spring Security that uses the BCrypt hashing algorithm which automatically generates a randomly generated salt to produce a unique string that is stored as the password value in the database[96]. The advantage in using a salt, is that the same password input does not generate the same output values as seen in Figure 44 where the same password “bob” was used. This protects against rainbow table attacks.



Maryam	$2a$10$QDvJ.glXpQVXuqjkfvQUKOfK0eWEYoT1NPOTJEt8NdnxBMXB51J9O
Maryam	$2a$10$XnrjPPvO12WHcCK0DaCp8.SJOs5LzzkGPR.TqfHnTvA.4bAOKYusq

Figure 44

3. The user logs in and has the option to enable a “remember-me” cookie (Figure 45).


OfficeFlow

Login

Email:

mgadiali@outlook.com

Password:

•••••••••••

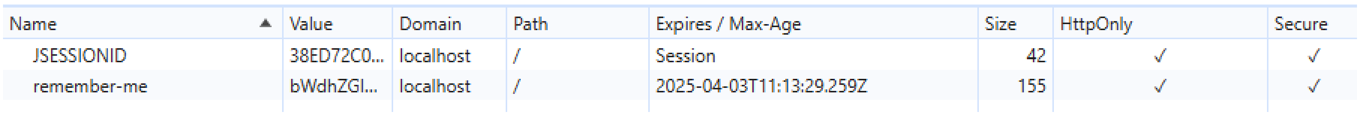
☑ Remember me

Registration successful! Please log in.

If you don't have a username or password, please

Figure 45

4. Once the user has logged in successfully, and if they enable the cookie option, they will receive a Spring Security remember-me cookie that expires after 24 hours as seen in Figure 46. This value was chosen due to the context in which a user would activate the devices at the start and for the duration of their work day, so if the user logged in on a public or shared computer, it would prevent other people from being able to access their details. Spring Security automatically creates a session and grants a matching session cookie that is used to help establish the "SecurityContext" where this is used throughout to identify the user who is interacting with the server and do subsequent user handling operations[97].
   The user is then redirected to the dashboard page.



Name	Value	Domain	Path	Expires / Max-Age	Size	HttpOnly	Secure
JSESSIONID	38ED72C0...	localhost	/	Session	42	✓	✓
remember-me	bWdhZGl...	localhost	/	2025-04-03T11:13:29.259Z	155	✓	✓

Figure 46

5. The application detects that they are a new user, and so an introduction tour is provided through the IntroJS library[98] to guide their setting up process as seen in Figure 47. This library was chosen as it makes the user follow instructions step by step and highlights HTML elements to focus on.
   The user then proceeds to the hub set up page, and enters the correct pair code (Figure 48). Pair codes are unique human-friendly 6 character strings that maps to a unique MAC address in the database. It identifies the type of device (sensor, actuator, hub), and their specific function (light, temperature etc.). Determining the correct MAC address is necessary to establish ESP-NOW communication amongst the IoT devices.

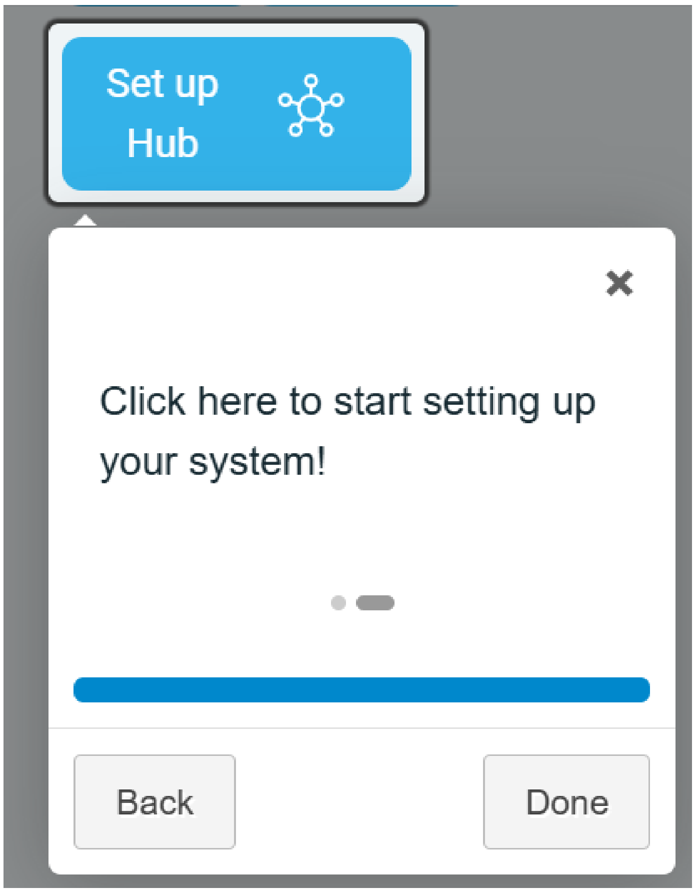


Figure 47

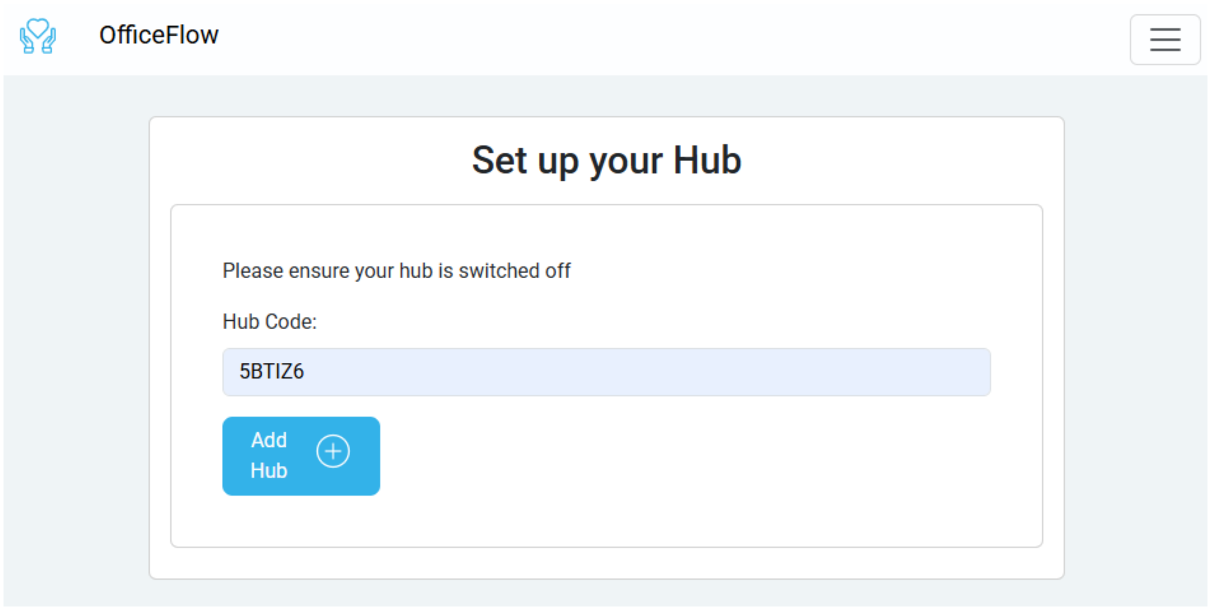


Figure 48

6. The application verifies that it is the correct pair code, then guides the user on the next steps throughout, which is switching on the hub and setting up the Wi-Fi details for the hub to connect to and store for future use.
7. The user switches on the hub.
8. The WiFiManager library[99] is used to configure a user friendly way of setting up the hub's Wi-Fi connection. It causes the hub to broadcast an AP for the user to connect to which is named "HubSetup" (Figure 49) .

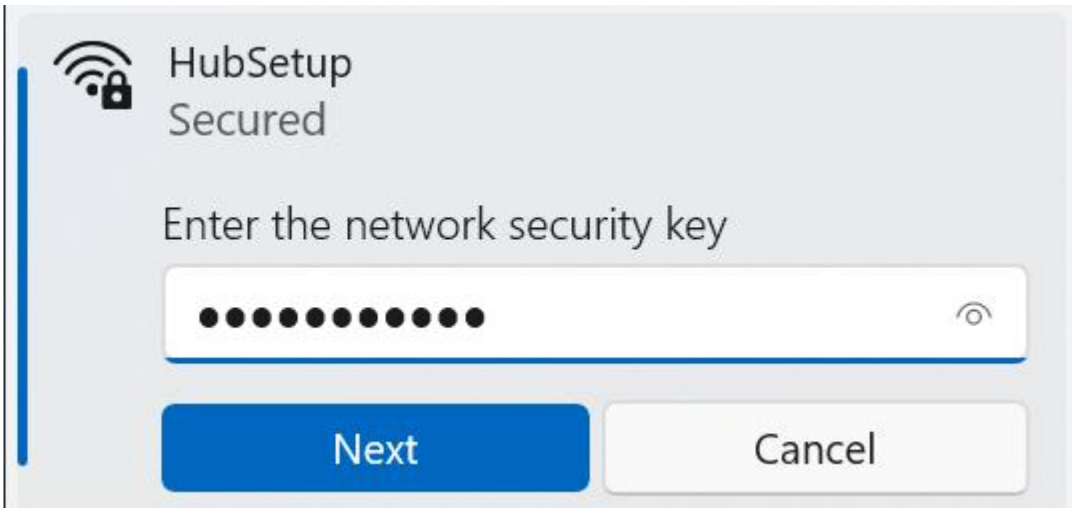


Figure 49

9. Once the user has connected, they are told to navigate to “192.168.4.1/wifi” which leads to a captive portal handled by the library, where the user can select their home Wi-Fi from the list of available networks, enter the relevant details and then save it (Figure 50).

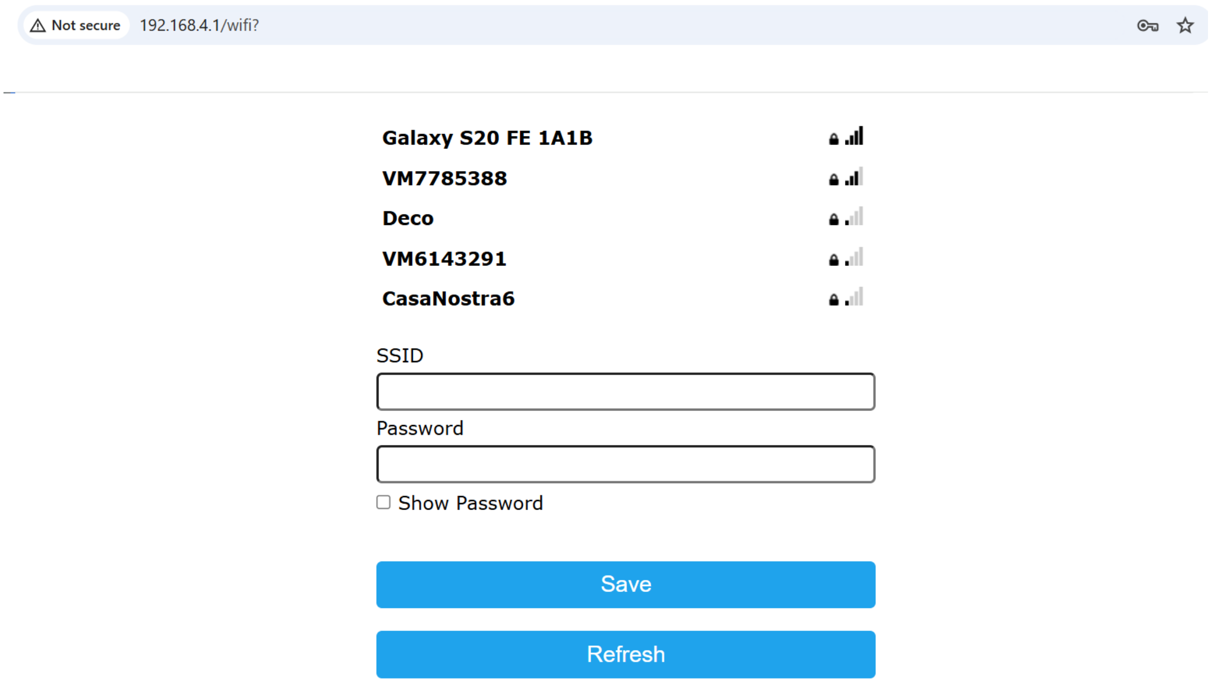
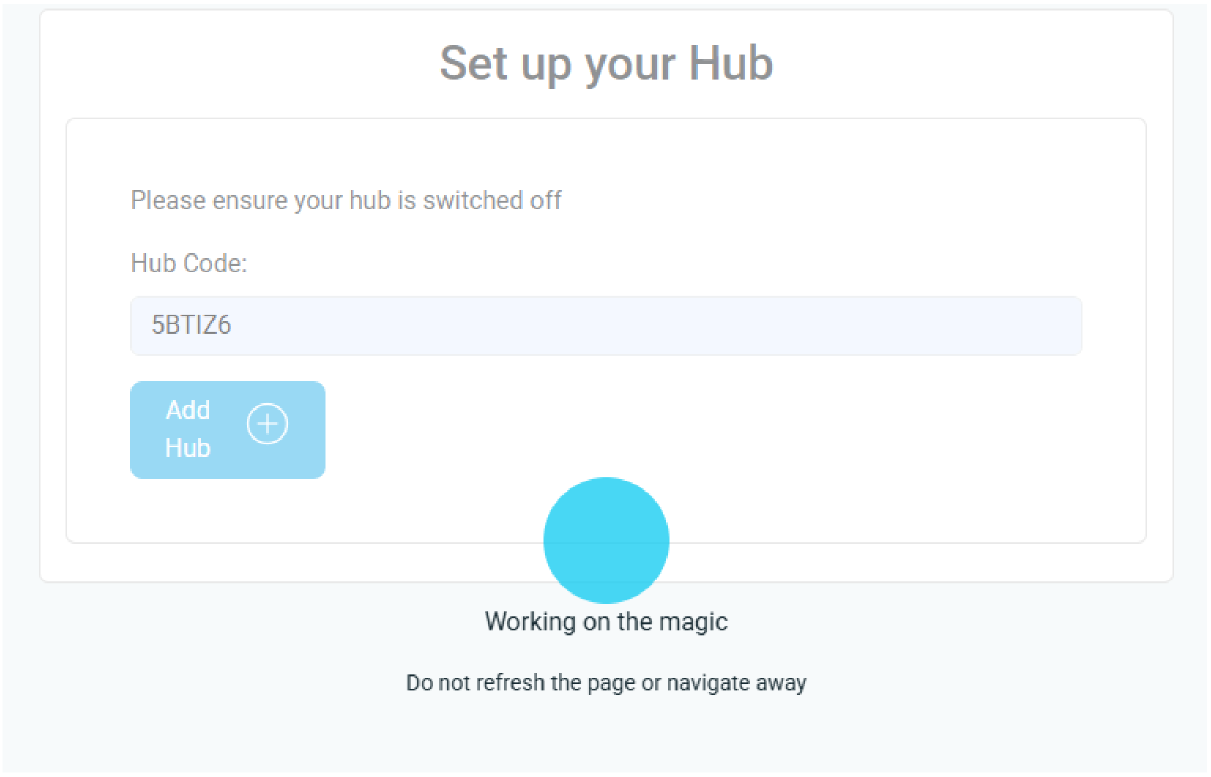


Figure 50

10. This Wi-Fi set up only needs to be done once, as the library handles the storing of the Wi-Fi details for future start-ups.
11. The hub retrieves the current clock time from the API, to set its RTC. This is because ESP32 RTCs do not keep track of live time and so needs an external reference point or a specific component to set or persist the time[100].
12. It checks if its pair code has been used yet by a user, and if so, it activates itself in the system database.
13. During this time, the user’s page on the client side polls and checks the hub’s activeness every 5 seconds whilst the hub is being set up by using JavaScript’s “setInterval” function to poll an API endpoint using AJAX. To the user, this is being displayed as the loading screen (Figure 51). If it receives the response that the hub has been activated, it proceeds to link the user and hub together in the database.



Set up your Hub

Please ensure your hub is switched off

Hub Code:

5BTIZ6

Working on the magic

Do not refresh the page or navigate away

Figure 51

14. Now the hub setting up is completed, and the user is redirected to setting up the occupancy sensor.

Setting up the Occupancy sensor

Simplified overview

The following figures show the simplified overview of steps of when a user sets up their Occupancy sensor in sequence.

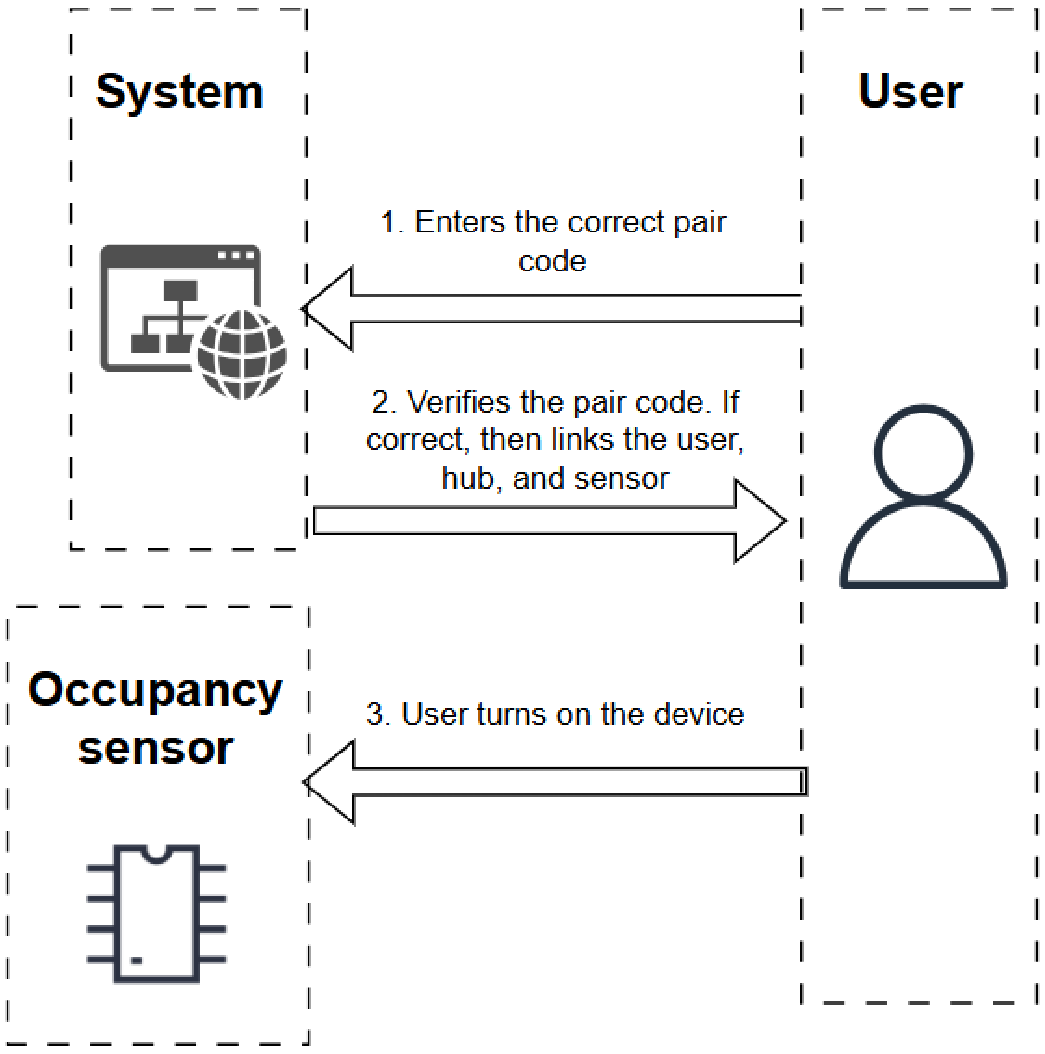

System
User
1. Enters the correct pair code
2. Verifies the pair code. If correct, then links the user, hub, and sensor
Occupancy sensor
3. User turns on the device

Figure 52

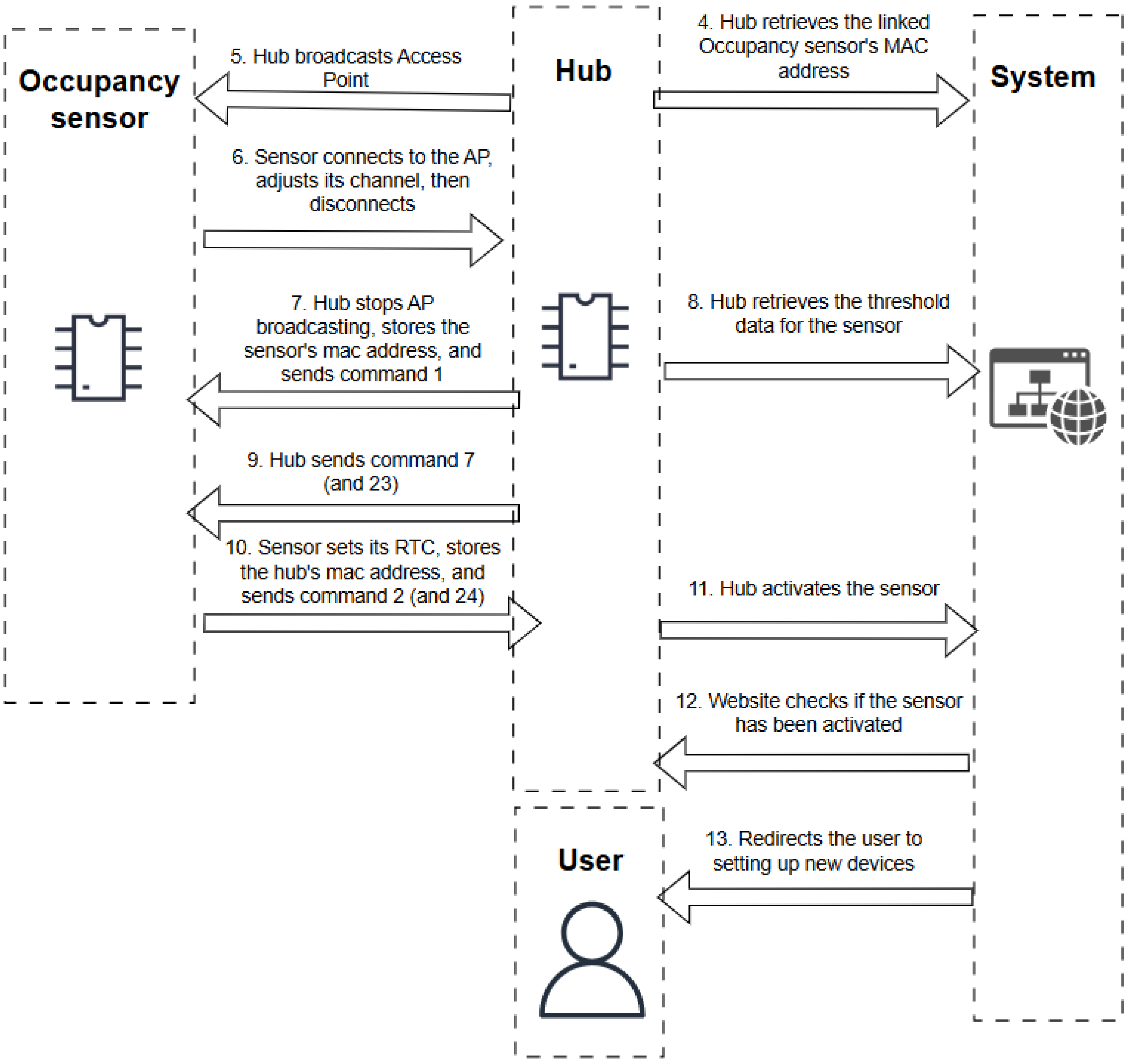
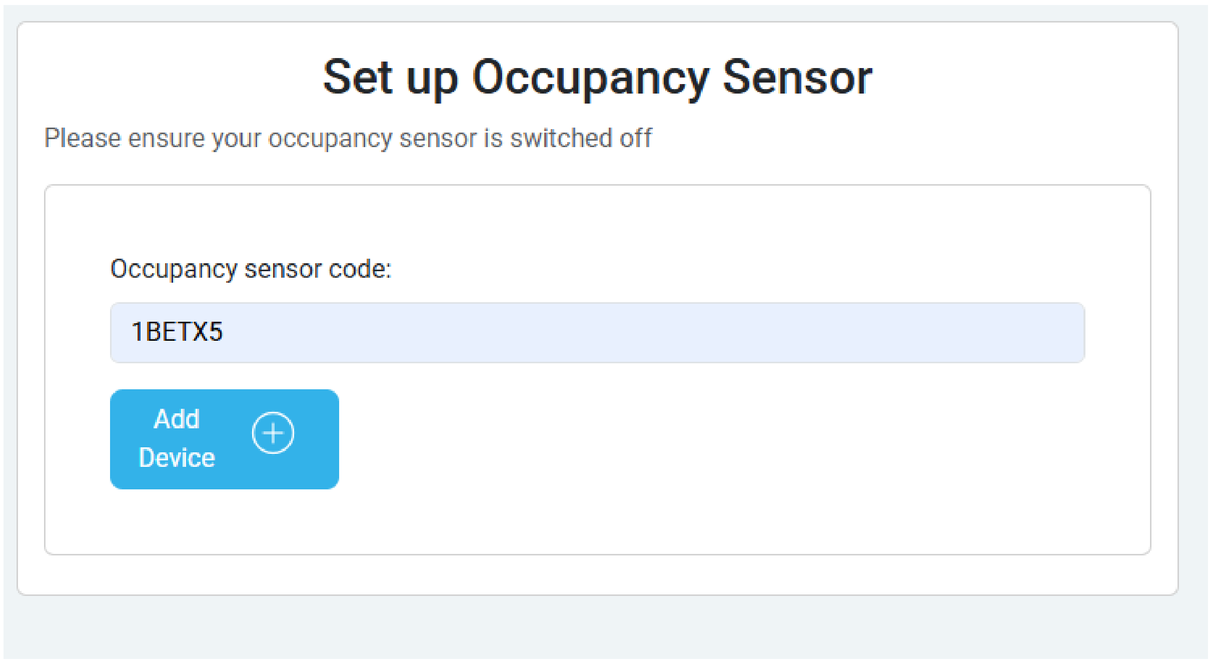

Occupancy sensor
Hub
System
User
4. Hub retrieves the linked Occupancy sensor's MAC address
5. Hub broadcasts Access Point
6. Sensor connects to the AP, adjusts its channel, then disconnects
7. Hub stops AP broadcasting, stores the sensor's mac address, and sends command 1
8. Hub retrieves the threshold data for the sensor
9. Hub sends command 7 (and 23)
10. Sensor sets its RTC, stores the hub's mac address, and sends command 2 (and 24)
11. Hub activates the sensor
12. Website checks if the sensor has been activated
13. Redirects the user to setting up new devices

Figure 53

Detailed overview

This section goes into further depth of each step outlined in the above "Simplified overview" section.

1. User enters the Occupancy sensor's pair code (Figure 54).


Set up Occupancy Sensor

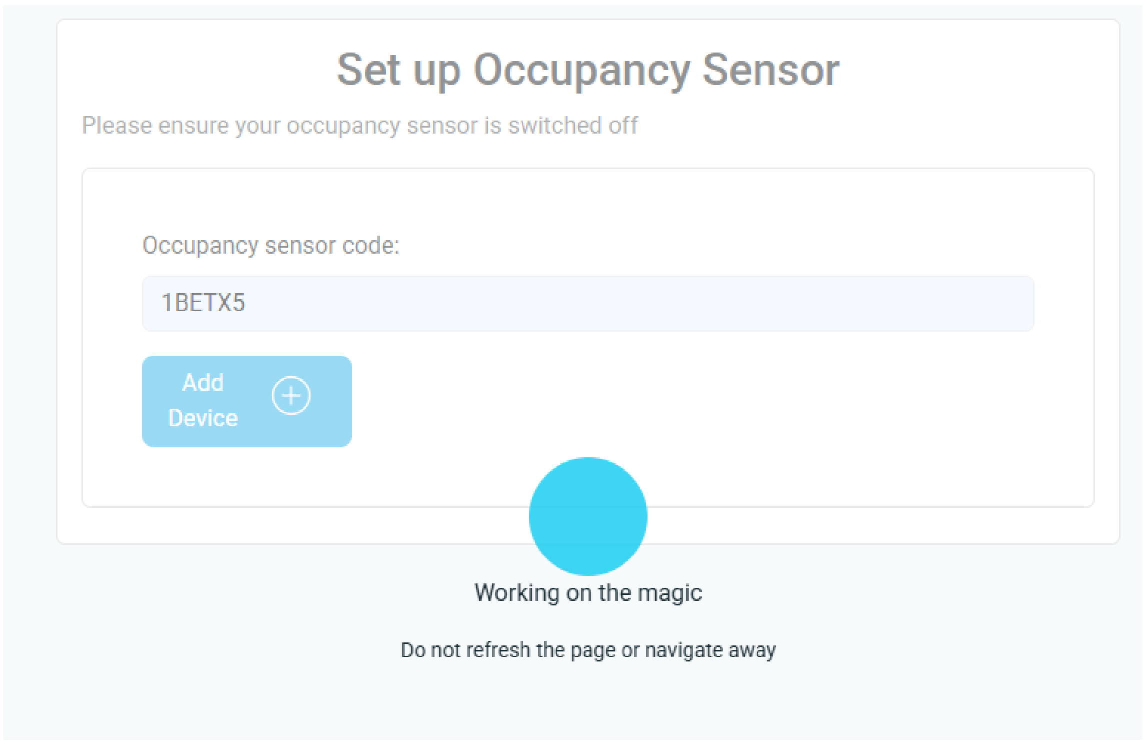
Please ensure your occupancy sensor is switched off

Occupancy sensor code:

1BETX5

Figure 54

2. The application verifies the pair code, if it is correct, then the hub, user and sensor is linked in the database. The user then receives an instruction to turn on the device. At this point, the application shows a loading display until the matching sensor is activated (Figure 55).


Set up Occupancy Sensor

Please ensure your occupancy sensor is switched off

Occupancy sensor code:

1BETX5

Working on the magic

Do not refresh the page or navigate away

Figure 55

3. The user turns on the Occupancy sensor, and it initiates in Wi-Fi station mode where it is trying to locate the AP called “ESP32HUB”.
4. At this point, the hub would have been polling the API until it received a response for a valid MAC address for the Occupancy sensor.
5. Once the hub receives it, it proceeds to broadcast the AP point called “ESP32HUB”. Only the ESP32 nodes can connect to this AP, not users, as the connection credentials are hardcoded in. The reason for having the nodes connect to the hub’s AP point is due to ESP-NOW.
   In order to use ESP-NOW successfully, there are 3 main components needed:
   1. ESP-NOW has to be initiated
   2. The devices have to be on the correct channel
   3. The MAC address of the target device has to be known by the sender

   In order to satisfy criteria 2, we first have to understand that the ESP32 supports WIFI connections on the 2.4GHz band, meaning it can support 13 channels[30][101]. If any WIFI station device connects to an AP, the station device is set to the channel that the AP is set at. From observation, channels 1, 6 and 11 are most commonly used. This means that as the hub is the only device connected to a network AP (the home Wi-Fi), it cannot manually set its own fixed preset channel number as that is decided by the network AP, so the hub’s channel cannot be known beforehand.

   The resolution for this is first having the hub connect to the home Wi-Fi, which automatically sets the hub’s channel, then having the hub host its own AP so the nodes can temporarily connect to it to automatically set their channels in order to satisfy criteria 2. This is more efficient and faster than the method of having all nodes loop through each channel to send a message and waiting a certain time period to see if there is a response back before moving onto the next channel.
6. The sensor connects to the hub’s AP for a second to adjust its channel, then initiates ESP-NOW. It is prevented from proceeding until it receives command 7 for setting its RTC.
7. The hub stops broadcasting when it detects that the sensor has connected. Using the MAC address retrieved from step 4, it stores the sensor’s MAC address in the ESP32’s flash memory using the LittleFS library[102]. LittleFS is a lightweight filesystem that creates and uses a dedicated partition on the ESP32 to enable standard file handling operations[103]. In this file, the hub will store all the connected nodes’ type names (as the key) and their MAC addresses, in direct byte format.
   The hub then sends an ESP-NOW message to this address with command 1 which will be added to the Occupancy sensor’s message queue.
8. The hub then retrieves the thresholds from the API. The thresholds include details only for all the enabled automated devices (none will appear yet), and the

Occupancy sensor. For the purpose of the latter, this is so the hub can detect if the user is absent or not, so it can determine if a pause command should be issued to all the other devices.

9. The hub sends command 7 which allows the sensor to set its own RTC for the purpose of later creating readings for the API which contains the timestamp. Now the sensor has finished the setup section and can move onto the loop for handling queued commands. The hub also sends command 23 which initiates the heartbeat protocol check.
10. The sensor sets its RTC from command 7, and handles command 1 which stores the hub's MAC address using LittleFS - this is the only detail the nodes ever store. The sensor then sends command 2, which confirms that the sensor has completed set up. It also sends command 24 as the heartbeat response.
11. The hub handles command 2 by activating the sensor in the system
12. The website is continuously checking if the sensor has been activated yet
13. If so, then the user is directed to the setting up new devices page

Setting up new non-core devices

Simplified overview

The following figures show the simplified overview of steps of when a user sets up a new sensor and matching actuator, in sequence.

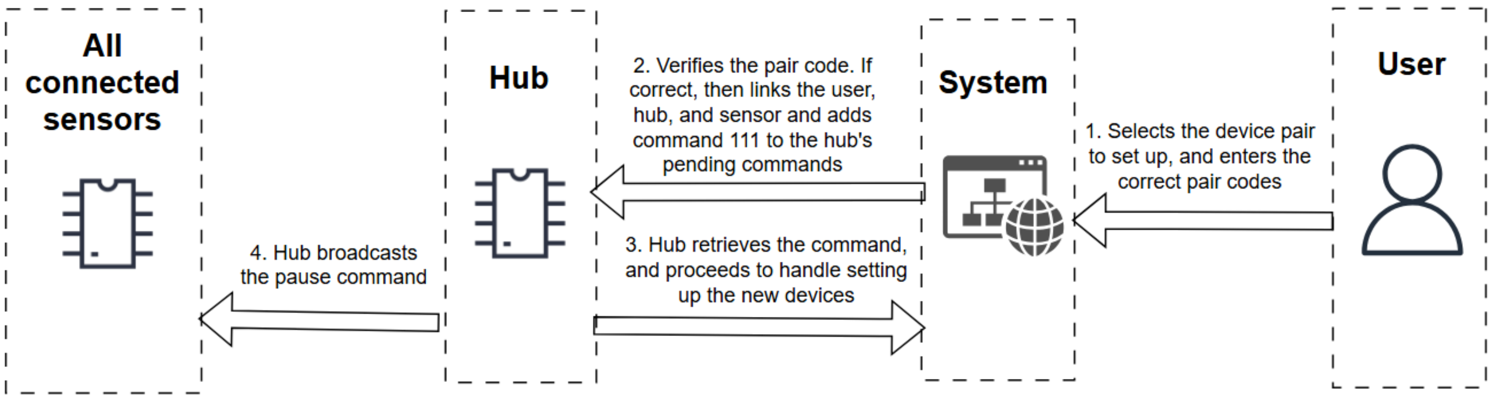

All connected sensors
Hub
System
User
1. Selects the device pair to set up, and enters the correct pair codes
2. Verifies the pair code. If correct, then links the user, hub, and sensor and adds command 111 to the hub's pending commands
3. Hub retrieves the command, and proceeds to handle setting up the new devices
4. Hub broadcasts the pause command

Figure 56

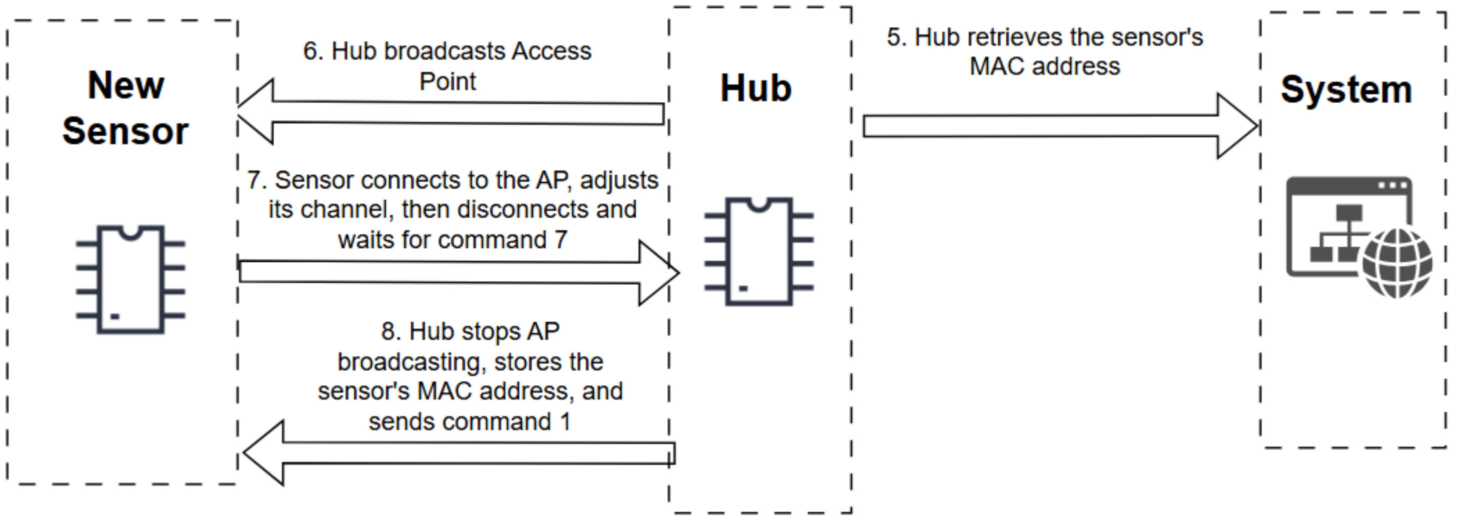

New Sensor
6. Hub broadcasts Access Point
Hub
5. Hub retrieves the sensor's MAC address
System
7. Sensor connects to the AP, adjusts its channel, then disconnects and waits for command 7
8. Hub stops AP broadcasting, stores the sensor's MAC address, and sends command 1

Figure 57

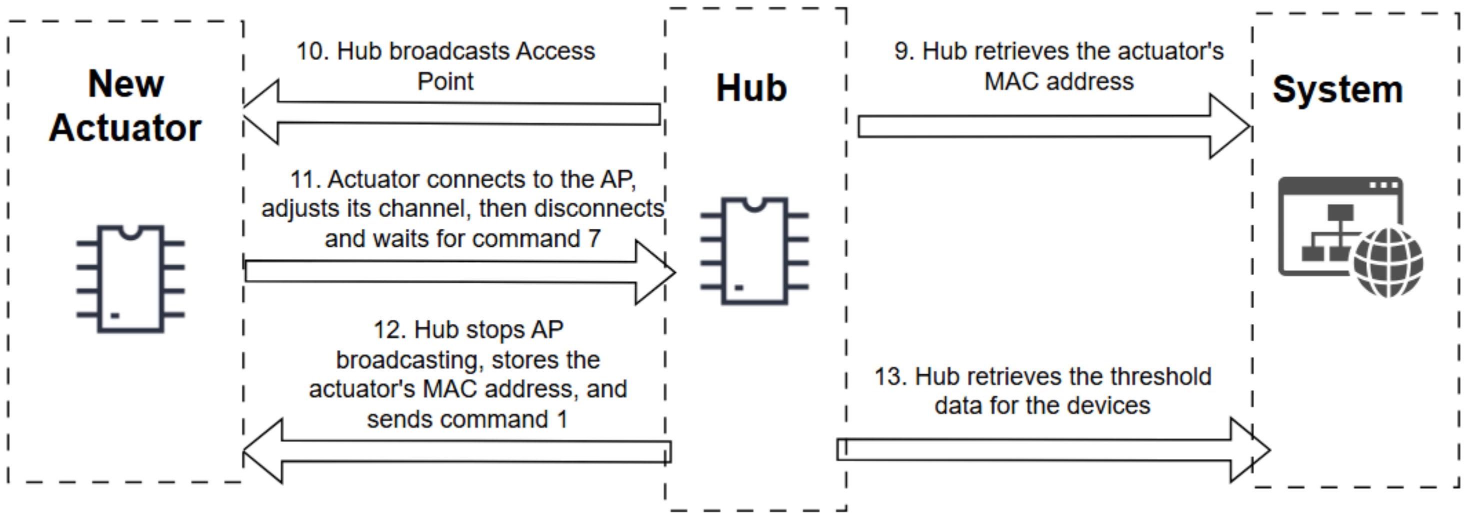

New Actuator
10. Hub broadcasts Access Point
Hub
9. Hub retrieves the actuator's MAC address
System
11. Actuator connects to the AP, adjusts its channel, then disconnects and waits for command 7
12. Hub stops AP broadcasting, stores the actuator's MAC address, and sends command 1
13. Hub retrieves the threshold data for the devices

Figure 58

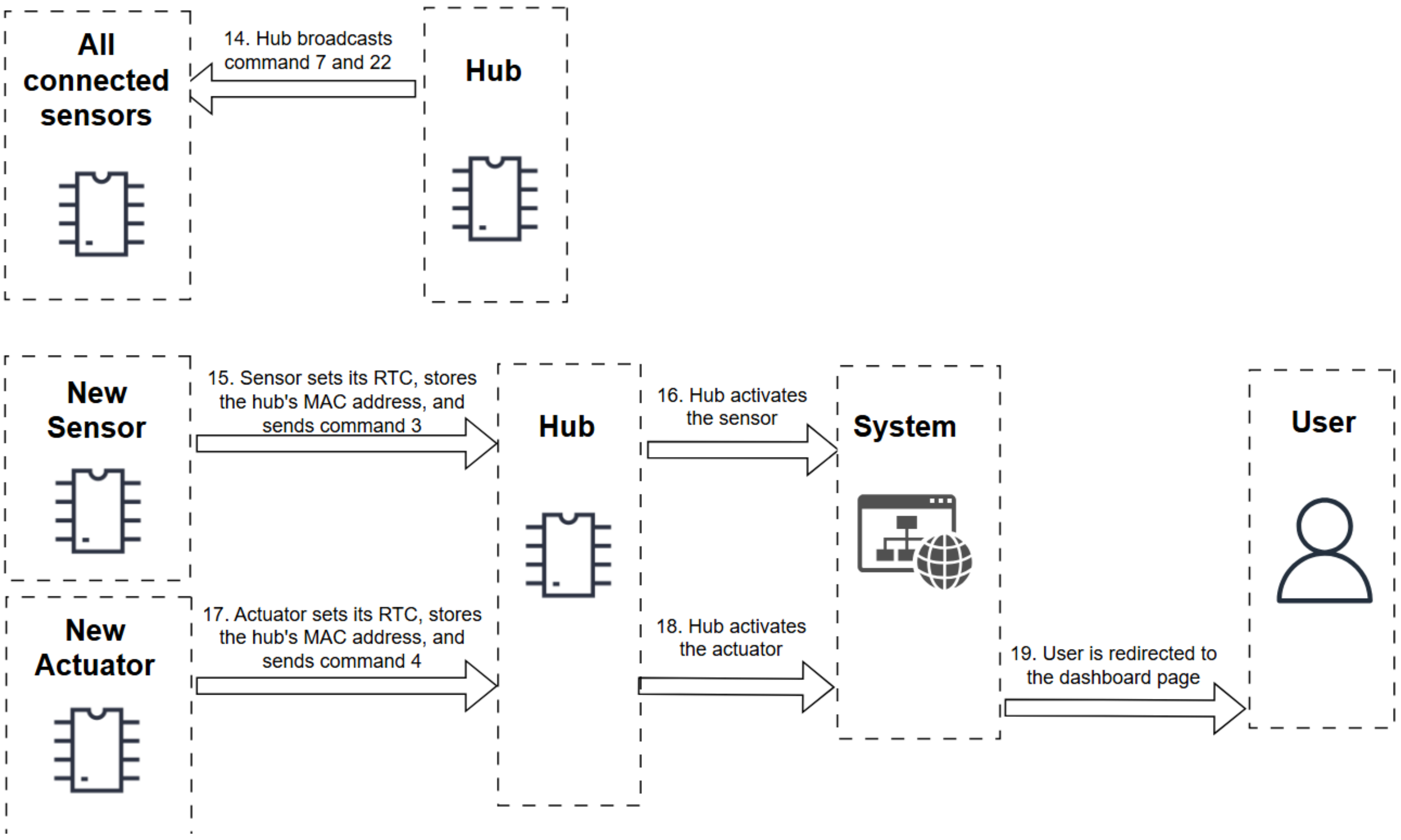

All connected sensors
14. Hub broadcasts command 7 and 22
Hub
New Sensor
15. Sensor sets its RTC, stores the hub's MAC address, and sends command 3
Hub
16. Hub activates the sensor
System
User
New Actuator
17. Actuator sets its RTC, stores the hub's MAC address, and sends command 4
18. Hub activates the actuator
19. User is redirected to the dashboard page

Figure 59

Detailed overview

This set up process is very similar to the steps covered for the Occupancy sensor's set up, except that 2 devices can be set up at once if the sensor has a matching actuator type, such as light and lamp, and temperature and fan. The user can choose which devices to set up from a drop down list (Figure 60). After the setup is complete, then the user is redirected to the dashboard page. If only one sensor is being set up (when there is no matching actuator type – such as the heart rate sensor), then steps 9,10,11,12,17 and 18 are omitted.

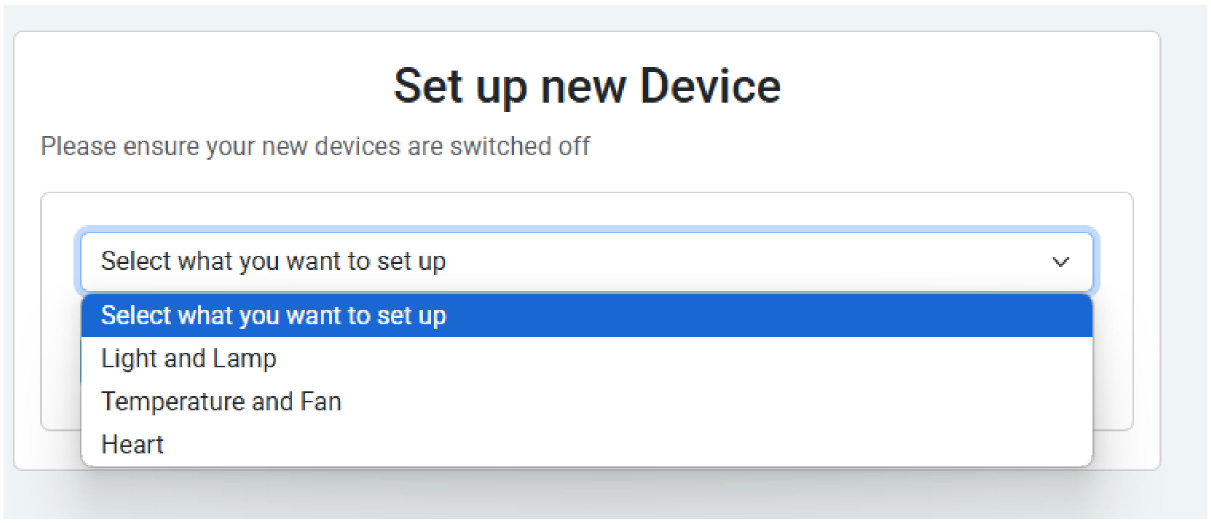

Set up new Device
Please ensure your new devices are switched off
Select what you want to set up
Select what you want to set up
Light and Lamp
Temperature and Fan
Heart

Figure 60

Evaluating times

The author recorded the time taken for them to set up 4 devices: the Hub, Occupancy sensor, Light sensor and Lamp. With their knowledge of shortcuts and experience, they were able to complete this process in under 4 minutes.

The same task was given to a 22 year old student who is a casual technology user, and they were able to complete the task in under 8.5 minutes. This result is satisfactory as the author's original target was to have the user complete it in under 10 minutes, and is a good indicator that the setting up process is easy to understand for the general user.

Whilst observing this user go through the process, there was mild frustration seen regarding the time duration of the task (when they were faced with the loading screens). To resolve this issue, more feedback should be added in the future, such as progress indicators when a certain task is complete, or providing a time expectation value when the loading screen appears. More participant testing would be beneficial to find what the average time taken is for non-technical users – unfortunately this could not take place in the current project timeline as only one version of the hardware system has been developed due to resources available.

Normal routines

This section provides an overview of the start-up flow and normal operation flow for when all the devices have already been activated in the system. For the hub, this follows the second set up process – for normal routine.

Start up

Figure 61 shows the start-up process of the IoT network.

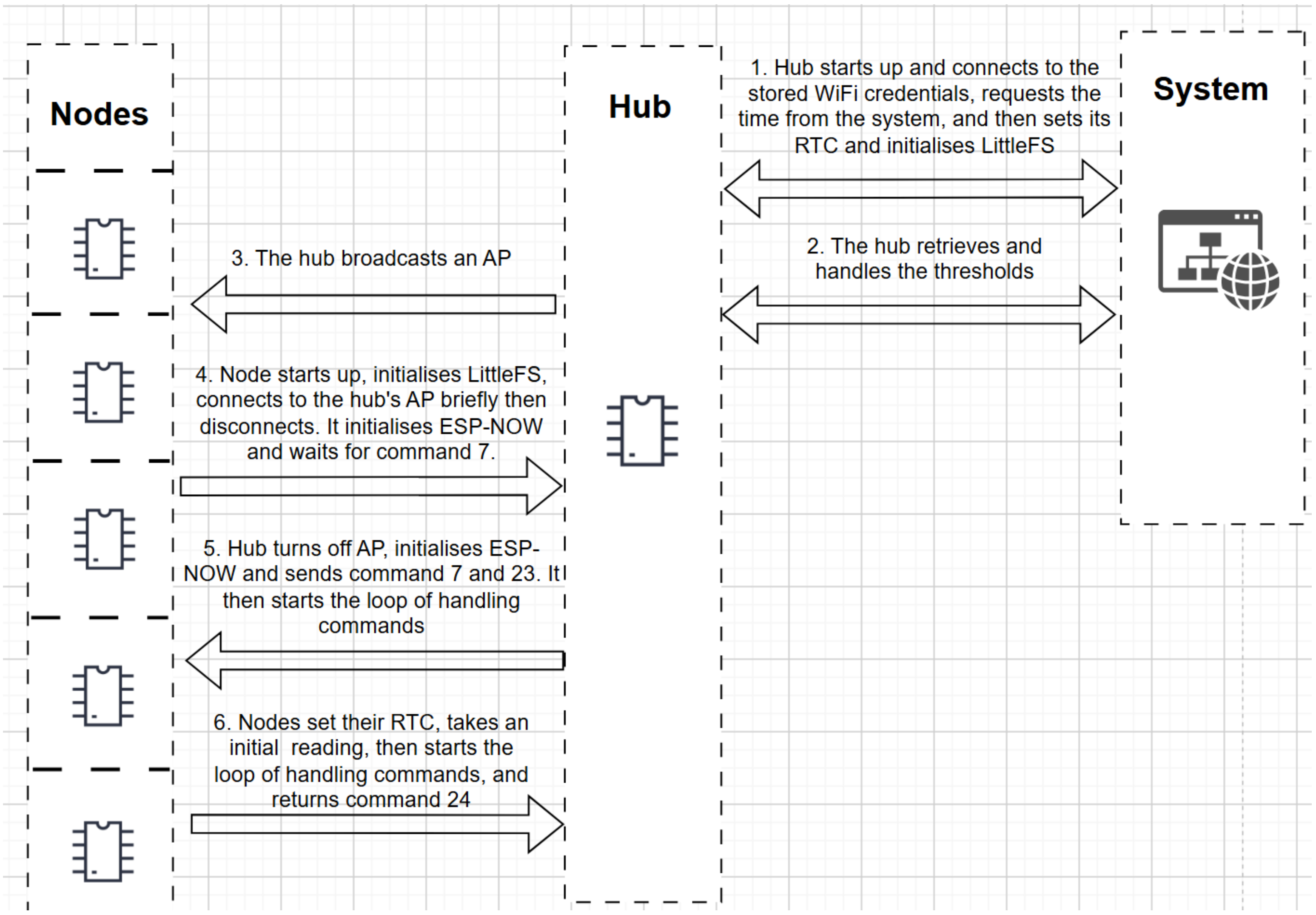

Nodes
Hub
System
1. Hub starts up and connects to the stored WiFi credentials, requests the time from the system, and then sets its RTC and initialises LittleFS
2. The hub retrieves and handles the thresholds
3. The hub broadcasts an AP
4. Node starts up, initialises LittleFS, connects to the hub's AP briefly then disconnects. It initialises ESP-NOW and waits for command 7.
5. Hub turns off AP, initialises ESP-NOW and sends command 7 and 23. It then starts the loop of handling commands
6. Nodes set their RTC, takes an initial reading, then starts the loop of handling commands, and returns command 24

Figure 61

Normal operation loop

Once the devices have finished initialising, they go into a loop where they handle commands on the message queue. The following lists regular time triggered actions during normal operation:

- Hub checks pending commands up to every 15 seconds
- Hub sends a clock reset and heartbeat command every 3 minutes if all the linked devices are active, if not, then it broadcasts the local AP every 2 minutes for 15 seconds and then sends out the heartbeat and clock reset (after retrieving the time from the API). The RTC reset is needed to help prevent clock drift which would affect

timestamps recorded for readings. The hub determines how many devices are active by recording how many command 24s are received after. The broadcast is needed to connect to any devices that are switched on after the Hub's start up process.

- The Occupancy sensor takes a reading every 60 seconds (1 minute)
- The Light sensor takes a reading every 180 seconds (3 minutes)
- The actuator devices send their power state every 60 seconds (1 minute)
- The Temperature sensor sends a reading every 300 seconds (5 minutes)
- The Heart rate sensor sends a reading every 300 seconds (5 minutes)
- If the node has not received a message from the hub in 6 minutes, it restarts itself. This is to resolve the scenario where the hub has reconnected to a Wi-Fi network or to a new one at run time, which may have changed its channel number, meaning the nodes channel would need to be reset in order to reestablish communication.

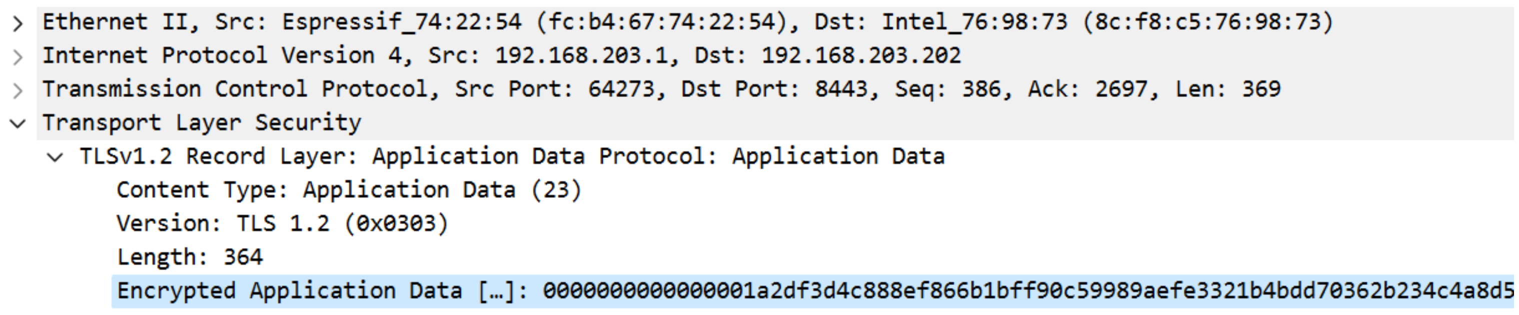
Security overview

The emphasis on including security aspects in this system is due to the lack of security in modern day IoT devices. The biggest critical vulnerability, as highlighted by several studies, was users using default or weak credentials[104].

All of the electronic IoT devices in this system do not have any user interface where the user can login and 'assign' themselves to the device hardware. The devices operate blindly of the user, only sending the sensor data collected to the API with just the hub as the identifier. The hub is the only device that can receive commands, and only from the specified server. Overall, this means that the IoT network is protected from being able to communicate with third party connections, particularly as the hub is the one that always initialises the connection and actively sends and receives data from the API, the application does not actively communicate with the hub.

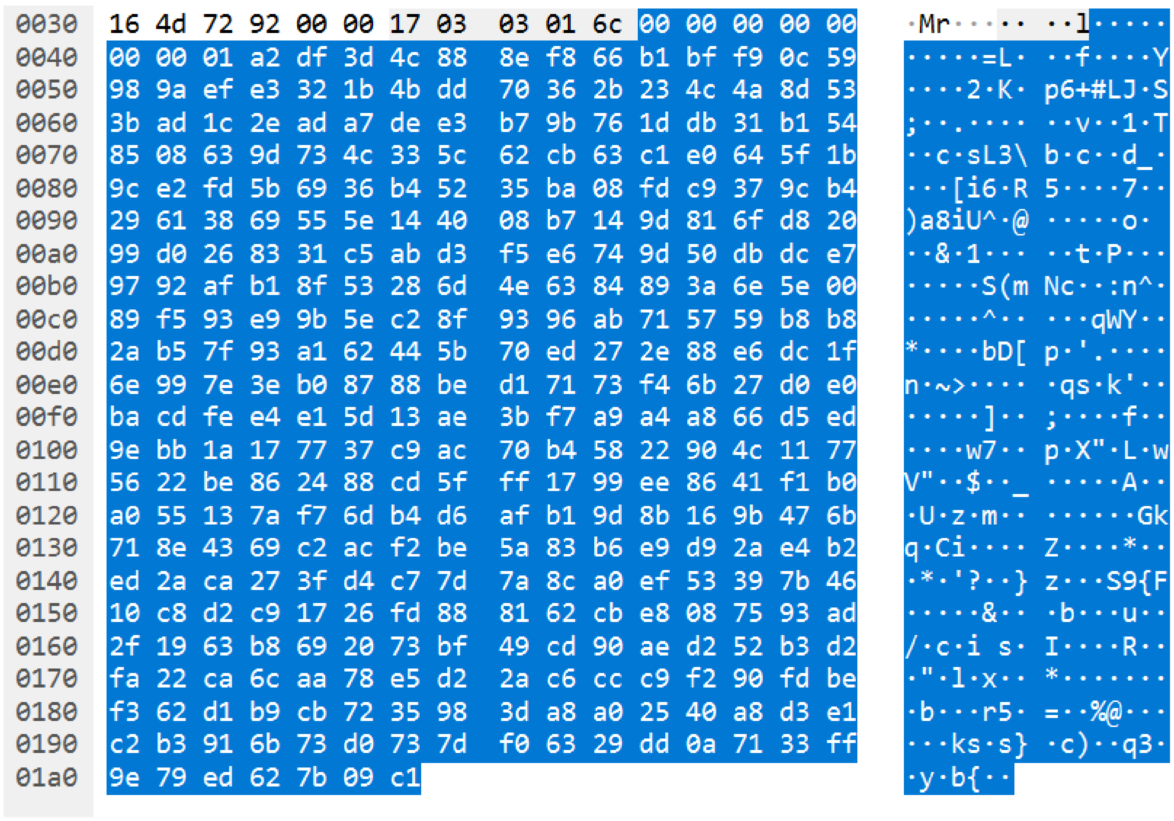
Another common security risk is the use of plaintext in network communications due to the use of HTTP rather than HTTPS[104]. This compromises the confidentiality and integrity of the messages. Research was conducted into the vulnerabilities of IoT cameras, and found that over 1 million surveillance cameras and over 125,000 surveillance servers exposed to the internet had 90% of them using HTTP[105]. A critical risk here is the sending of credentials in plaintext where it is easy to sniff and identify the packets through tools like Wireshark.

This project's application only runs over HTTPS, allowing the hub to communicate securely with it. The hub is the only exposed device to the internet, which limits any potential compromise on other nodes. Figure 62 and Figure 63 shows a capture of a Wireshark packet and its corresponding packet bytes of the hub sending a message to the API, which we can see is encrypted in transit.



```
> Ethernet II, Src: Espressif_74:22:54 (fc:b4:67:74:22:54), Dst: Intel_76:98:73 (8c:f8:c5:76:98:73)
> Internet Protocol Version 4, Src: 192.168.203.1, Dst: 192.168.203.202
> Transmission Control Protocol, Src Port: 64273, Dst Port: 8443, Seq: 386, Ack: 2697, Len: 369
v Transport Layer Security
  v TLSv1.2 Record Layer: Application Data Protocol: Application Data
      Content Type: Application Data (23)
      Version: TLS 1.2 (0x0303)
      Length: 364
      Encrypted Application Data […]: 0000000000000001a2df3d4c888ef866b1bff90c59989aefe3321b4bdd70362b234c4a8d5
```

Figure 62



```
0030  16 4d 72 92 00 00 17 03  03 01 6c 00 00 00 00 00   ·Mr····· ··l·····
0040  00 00 01 a2 df 3d 4c 88  8e f8 66 b1 bf f9 0c 59   ·····=L· ··f····Y
0050  98 9a ef e3 32 1b 4b dd  70 36 2b 23 4c 4a 8d 53   ····2·K· p6+#LJ·S
0060  3b ad 1c 2e ad a7 de e3  b7 9b 76 1d db 31 b1 54   ;··.···· ··v··1·T
0070  85 08 63 9d 73 4c 33 5c  62 cb 63 c1 e0 64 5f 1b   ··c·sL3\ b·c··d_·
0080  9c e2 fd 5b 69 36 b4 52  35 ba 08 fd c9 37 9c b4   ···[i6·R 5····7··
0090  29 61 38 69 55 5e 14 40  08 b7 14 9d 81 6f d8 20   )a8iU^·@ ·····o· 
00a0  99 d0 26 83 31 c5 ab d3  f5 e6 74 9d 50 db dc e7   ··&·1··· ··t·P···
00b0  97 92 af b1 8f 53 28 6d  4e 63 84 89 3a 6e 5e 00   ·····S(m Nc··:n^·
00c0  89 f5 93 e9 9b 5e c2 8f  93 96 ab 71 57 59 b8 b8   ·····^·· ···qWY··
00d0  2a b5 7f 93 a1 62 44 5b  70 ed 27 2e 88 e6 dc 1f   *····bD[ p·'.····
00e0  6e 99 7e 3e b0 87 88 be  d1 71 73 f4 6b 27 d0 e0   n·~>···· ·qs·k'··
00f0  ba cd fe e4 e1 5d 13 ae  3b f7 a9 a4 a8 66 d5 ed   ·····]·· ;····f··
0100  9e bb 1a 17 77 37 c9 ac  70 b4 58 22 90 4c 11 77   ····w7·· p·X"·L·w
0110  56 22 be 86 24 88 cd 5f  ff 17 99 ee 86 41 f1 b0   V"··$··_ ·····A··
0120  a0 55 13 7a f7 6d b4 d6  af b1 9d 8b 16 9b 47 6b   ·U·z·m·· ······Gk
0130  71 8e 43 69 c2 ac f2 be  5a 83 b6 e9 d9 2a e4 b2   q·Ci···· Z····*··
0140  ed 2a ca 27 3f d4 c7 7d  7a 8c a0 ef 53 39 7b 46   ·*·'?··} z···S9{F
0150  10 c8 d2 c9 17 26 fd 88  81 62 cb e8 08 75 93 ad   ·····&·· ·b···u··
0160  2f 19 63 b8 69 20 73 bf  49 cd 90 ae d2 52 b3 d2   /·c·i s· I····R··
0170  fa 22 ca 6c aa 78 e5 d2  2a c6 cc c9 f2 90 fd be   ·"·l·x·· *·······
0180  f3 62 d1 b9 cb 72 35 98  3d a8 a0 25 40 a8 d3 e1   ·b···r5· =··%@···
0190  c2 b3 91 6b 73 d0 73 7d  f0 63 29 dd 0a 71 33 ff   ···ks·s} ·c)··q3·
01a0  9e 79 ed 62 7b 09 c1                               ·y·b{··
```

Figure 63

Spring Security was used to secure the application, the implementation of which has been covered (authorisation, authentication, HTTPS, password encoding).

Testing

Manual testing was performed over a month by the author using the system daily when working in their home space, and then fixing according to observed bugs, subtle inconsistencies, and errors. This covers all the physical hardware and software behaviours unlike automated testing which cannot account for physical issues.

As an end user, the author gained useful insights from the dashboard data in regards to their room's light levels throughout the day and the effect on it from the windows, blinds and the light bulb. At the time of writing, the season is slowly approaching the Summer months, and this general trend increase in sunshine received can be tracked by comparing historical graphs with more recent ones. Similarly, the general increase in temperature trends are also observed and provides useful insight to the author on which degrees they start feeling uncomfortable to work in and to wear lighter clothing or use a fan in response.

However, the author has struggled to gain useful insights yet from the heart rate graphs. This is mostly in part due to the sensor's placement being designed for the tip of the finger which restrains the author from typing efficiently for their daily work, and so it was challenging to gather a wide range of records for this metric. Future improvements should explore a wrist wearable device and relevant modules that can effectively measure the wrist BPM. Challenges include keeping the module in place at a certain location on the wrist, and applying a consistent firm pressure.

Critical Appraisal

What went well

The aims, objective, and requirements were achieved to create a reliably working smart home office health system, with extra features included that allowed the user more control over their system such as options to enable/disable automation and notifications, and edit thresholds.

The author's skills and capability were sufficient enough to deliver the project against the challenging time requirement.

The author is satisfied with the result and quality of this project.

Personal development

The author has learnt a lot of new skills and enhanced their analytical and problem solving abilities. This project, being their first non-guided work handling both hardware and software, opens up a whole new subject field that is now accessible to them and enables them to pursue further inventor projects, using this one as a template and guide.

The author has learnt new DIY skills, such as how to solder, use a multi-meter, handle AC wires, and debug hardware problems.

The author has learnt how to prioritise time efficiently by learning what their limits and skilled areas are, and to practice patience when progress was slow, particularly as this is their first C++ real world application project.

The author has also learnt valuable lessons for the future through continuous reflection, such as the risks of being overambitious, the importance of having a work-life balance, setting realistic goals, and getting an informed second opinion from an experienced person before committing to a large task relative to their abilities and capacity.

Challenges

This section covers key challenges faced in the project.

Power distribution issues

As 7 devices were created and were intended to form a network where they run simultaneously, a 7-port extender was used to overcome the limitation of the development laptop having only 2 ports. The laptop's USB ports provide 5V. When using sensor modules designed for Arduino programming, most of the time, they either need 3.3V or 5V as a power source.

Using the port extender invited additional unwelcome resistance that affected the amount of power reaching the circuits. A multi-meter was used to detect the voltage received in different scenarios for the power pins on the light sensor device. Table 15 shows the results.

Table 15

	Connected directly to laptop (V)	Connected to port extender and being the only device switched on (V)	Connected to port extender with all 7 devices switched on (V)
5V power pin	4.8	4.7	4.35
3.3V power pin	3.3	3.3	3.3

The results show that modules using 3.3V would face no issues as there is sufficient amount delivered. However, for modules requiring 5V, there is a significant decrease in voltage received which affected the device circuits, causing issues such as power crashes and resets, light flickering, and failed readings. This also caused the circuits to be more affected by movement of the USB power cable, and the length of connector wires used, as it was observed that centimetre differences in length would cause issues.

Incorrect details and hardware whack-a-mole

Alongside the power distribution issues, there were also issues relating to the sensor modules. This included receiving incorrect details by the seller regarding important specifications such as the power source voltage needed, and also facing hidden mechanical faults that caused failures.

When it was hard to make out pin labels, research was used to find diagrams for the component, which turned out to be incorrect and with consequence (Figure 69 in Appendix).

Unexplainable laptop crashes and restarts when using the system caused the theory to appear that there was a short circuit. Testing the circuits returned no faults, and the cuplrit was found to be a regularly used USB wire. As implied with the above, the challenge was determining if my work or the purchased tools were causing problems.

IDE issues

The Arduino IDE was used to compile the code to the hardware, and the speed of the compile time was dependent on the laptop's resources available. Generally, compilation took a few minutes which added up and impacted development speed particularly when waiting for simple minimal code changes such as a debug line entry, or finding out after the compilation attempt that there was a semicolon missing which required a re-compile.

Large codebase

After having established the templated and reusable code across the 7 devices, changing one aspect meant manually changing it 7 more times and recompiling the code, which took up a lot of time.

Learning curve and overambition

This project was proposed underestimating the complexity, scale, time, and learning curve required against the author's skill level. Maximising time efficiency from early stages became key to delivering a completed project whilst also balancing performance in other university modules.

There was a lack of guidance available online of similar non-hobbyist programming of microcontrollers and so there was a lot of initial trial and error of code and ideas.

Health and safety

Due to the author's lack of formal education in hardware, and lack of professional equipment and workstation, health and safety practices had to be learnt from online sources and performed in a DIY fashion. A lot of time was spent learning and gaining confidence before executing related tasks, such as wiring the AC lamp, and soldering.

Future improvements

The following section describes future improvement suggestions of the project which were not able to be implemented due to time constraints or lack of resources:

- Automating the templated code so that changes to one will automatically refactor other files thus saving time.
- Adding ESP-NOW encryption. This was implemented for a while, however it was found that the results were not always deterministic and further research was needed to implement this, so the current final system has unencrypted ESP-NOW communication.
- Upgrading the presentation of the hardware devices with 3D printing. The author does not have access to this, and so cardboard cut covers were used to abstract the circuit details from the user.
- Finalising the circuits on PCBs (Printed Circuit Boards) instead of breadboards to move on from prototyping to a production releasable version.
- Exploring wrist wearable solutions for the heart rate sensor.
- Implementing batteries to the devices for portability and avoiding power distribution issues. Batteries were avoided due to its need of replacing when empty of charge, and risk of causing the device to fail in the final demonstration of the system.
- Hosting the application on a cloud server to provide a fixed domain name. Currently the hub is hardcoded with the private IP address of the development laptop on the same network. Providing a fixed domain name would remove the issue of the IP address's dynamic nature and allow the hub flexibility to connect to any network.
- Updating the database schema relationship to handle when one sensor can take more than one type of reading. For example, the DHT11 can provide both temperature and humidity readings, and the MAX30102 can take both heart rate and blood oxygen levels.
- Updating the Wi-Fi manager's captive portal display or creating a custom captive portal to make it more user-friendly. This requires learning of how to host and configure an API server on the ESP32.
- Providing offline capability of the system. Currently the whole system is dependent on connection to Wi-Fi. Options should be explored on how to provide offline functionality, which can include the development of a mobile app.
- Adding more complexity into the analysis of heart rate readings or exploring other heart rate sensors. Unlike light and temperature where patterns and useful insights can easily be derived from the produced graphs, the heart rate graphs don't provide this well as seen in Figure 19, which may be to do with the hardware or the algorithm used. More solutions for this should be explored, perhaps emphasising more importance on the user's heart rate variability from the PBA algorithm used.

- Implementing an over-the-air update system where the devices' code can be updated at run time.
- Providing a user-friendly option of allowing users to delete specific sensor readings.
- Implementing more advanced data analytics utilising machine learning, and including notification analytics which can help track user improvement in their office conditions.
- Spending time on finding an IDE that offers useful tools to streamline hardware development.
- Refactoring the codebase for efficiency. This could include merging the occupancy sensor code with the other reusable sensor code, instead of repeating similar steps. However, this did provide great benefit in debugging issues.
- Setting up an identity verification process, perhaps including an email server, for when the user forgets their password.
- Providing more visual progress feedback on the loading screen for when new devices are being set up.
- Exploring and implementing automated testing for a more complete and faster testing solution.
- Exploring more notification solutions to overcome the compatibility issues.
- Implementing a dark mode colour scheme.

Conclusion

This project provides an undergraduate's take on the foundational base for a low cost smart home office health system with areas for further improvement. It is not a complete solution and can be expanded to include new sensors and actuators to take advantage of its scalable design.

References

[1] G. Ráthonyi *et al.*, ‘Changes in Workers’ Physical Activity and Sedentary Behavior during the COVID-19 Pandemic’, *Sustainability*, vol. 13, no. 17, Art. no. 17, Jan. 2021, doi: 10.3390/su13179524.

[2] S. W. P. and J. J. Redmond, ‘The rise in remote work since the pandemic and its impact on productivity’, Bureau of Labor Statistics. Accessed: Apr. 12, 2025. [Online]. Available: https://www.bls.gov/opub/btn/volume-13/remote-work-productivity.htm

[3] E. Participation, ‘The Workplace (Health, Safety and Welfare) Regulations 1992’. Accessed: Apr. 12, 2025. [Online]. Available: https://www.legislation.gov.uk/uksi/1992/3004/contents

[4] E. Participation, ‘Health and Safety at Work etc. Act 1974’. Accessed: Apr. 12, 2025. [Online]. Available: https://www.legislation.gov.uk/ukpga/1974/37/contents

[5] ‘The Health and Safety at Work Act Explained’, British Safety Council. Accessed: Apr. 12, 2025. [Online]. Available: https://www.britsafe.org/training-and-learning/informational-resources/the-health-and-safety-at-work-act-explained

[6] E. Participation, ‘The Health and Safety (Display Screen Equipment) Regulations 1992’. Accessed: Apr. 12, 2025. [Online]. Available: https://www.legislation.gov.uk/uksi/1992/2792/contents

[7] S. Morrison, ‘Workplace Health Safety and Welfare Regulations 1992 – A Complete Guide’, Human Focus. Accessed: Apr. 12, 2025. [Online]. Available: https://humanfocus.co.uk/blog/workplace-health-safety-and-welfare-regulations-1992/

[8] M. S. Wütschert, D. Romano-Pereira, L. Suter, H. Schulze, and A. Elfering, ‘A systematic review of working conditions and occupational health in home office’, *WORK*, vol. 72, no. 3, pp. 839–852, Jul. 2022, doi: 10.3233/WOR-205239.

[9] M. J. Espinosa-Gavira, A. Agüera-Pérez, J. C. Palomares-Salas, J. M. Sierra-Fernandez, P. Remigio-Carmona, and J. J. González de-la-Rosa, ‘Characterization and Performance Evaluation of ESP32 for Real-time Synchronized Sensor Networks’, *Procedia Comput. Sci.*, vol. 237, pp. 261–268, Jan. 2024, doi: 10.1016/j.procs.2024.05.104.

[10] C. G. C. Carducci, A. Monti, M. H. Schraven, M. Schumacher, and D. Mueller, ‘Enabling ESP32-based IoT Applications in Building Automation Systems’, in *2019 II Workshop on Metrology for Industry 4.0 and IoT (MetroInd4.0&IoT)*, Jun. 2019, pp. 306–311. doi: 10.1109/METROI4.2019.8792852.

[11] ‘Alexa-compatible smart home devices 2017-2020’, Statista. Accessed: Apr. 14, 2025. [Online]. Available: https://www.statista.com/statistics/912893/amazon-alexa-smart-home-compatible/

[12] ‘Control smart home devices added to the Google Home app - Google Nest Help’. Accessed: Apr. 14, 2025. [Online]. Available: https://support.google.com/googlenest/answer/7073578?hl=en

[13] ‘Meet the Google Home app - Android - Streaming Help’. Accessed: Apr. 14, 2025. [Online]. Available: https://support.google.com/chromecast/answer/7071794?hl=en&co=GENIE.Platform%3DAndroid

[14] ‘Amazon Alexa App @ Amazon.com’. Accessed: Apr. 14, 2025. [Online]. Available: https://www.amazon.com/Alexa-App/b?ie=UTF8&node=18354642011

[15] ‘Cloud-to-cloud’, Google Home Developers. Accessed: Apr. 14, 2025. [Online]. Available: https://developers.home.google.com/cloud-to-cloud

[16] ‘Understand Smart Home Skills | Alexa Skills Kit’, Amazon Alexa. Accessed: Apr. 14, 2025. [Online]. Available: https://developer.amazon.com/en-US/docs/alexa/smarthome/understand-the-smart-home-skill-api.html

[17] ‘Nest Hub (2nd Gen)’, Google Store. Accessed: Apr. 14, 2025. [Online]. Available: https://store.google.com/gb/product/nest_hub_2nd_gen?hl=en-GB

[18] ‘Echo Show 5 (Newest gen) I Smart display and alarm clock with clearer sound I Cloud blue : Amazon.co.uk’. Accessed: Apr. 14, 2025. [Online]. Available: https://www.amazon.co.uk/echo-show-5-3rd-gen/dp/B09B2RV31Z?th=1

[19] ‘Nest Mini’, Google Store. Accessed: Apr. 14, 2025. [Online]. Available: https://store.google.com/gb/product/google_nest_mini?hl=en-GB

[20] ‘Learn about the LED lights on your speaker - Google Nest Help’. Accessed: Apr. 14, 2025. [Online]. Available: https://support.google.com/googlenest/answer/7073219?hl=en-GB#mini&zippy=%2Cgoogle-home-max

[21] ‘What Do the Lights on Your Echo Device Mean? - Amazon Customer Service’. Accessed: Apr. 14, 2025. [Online]. Available: https://www.amazon.co.uk/gp/help/customer/display.html?nodeId=GKLDRFT7FP4FZE56

[22] ‘Echo Dot (Newest gen) | Big vibrant sound Wi-Fi and Bluetooth smart speaker with Alexa | Charcoal : Amazon.co.uk: Amazon Devices & Accessories’. Accessed: Apr. 14, 2025. [Online]. Available: https://www.amazon.co.uk/echo-dot-2022/dp/B09B96TG33?th=1

[23] ‘Hive Hub | Hive Home’. Accessed: Apr. 14, 2025. [Online]. Available: https://www.hivehome.com/shop/smart-home/hive-hub

[24] ‘Hive Hub | Hive Home’. Accessed: Apr. 14, 2025. [Online]. Available: https://www.hivehome.com/shop/smart-home/hive-hub

[25] ‘The Hive App | Hive Home’. Accessed: Apr. 14, 2025. [Online]. Available: https://www.hivehome.com/hive-app

[26] ‘Will my Hive products continue to work if my broadband is turned off or stops working? | FAQs | Hive Home IE’. Accessed: Apr. 14, 2025. [Online]. Available: https://www.hivehome.com/ie/support/Our_Hive_Products/OHP_General/Will-my-Hive-products-continue-to-work-if-my-broadband-is-turned-off-or-stops-working

[27] ‘What to do if your Hive Hub is offline | FAQs | Hive Home IE’. Accessed: Apr. 14, 2025. [Online]. Available: https://www.hivehome.com/ie/support/Help_Using_Hive/Hive_Hub/What-to-do-if-your-Hive-Hub-is-offline

[28] H. Assistant, ‘Home Assistant’, Home Assistant. Accessed: Apr. 14, 2025. [Online]. Available: https://www.home-assistant.io/

[29] I. Plauska, A. Liutkevičius, and A. Janavičiūtė, ‘Performance Evaluation of C/C++, MicroPython, Rust and TinyGo Programming Languages on ESP32 Microcontroller’, *Electronics*, vol. 12, no. 1, Art. no. 1, Jan. 2023, doi: 10.3390/electronics12010143.

[30] ‘esp32-wroom-32_datasheet_en.pdf’. Accessed: Apr. 15, 2025. [Online]. Available: https://www.espressif.com/sites/default/files/documentation/esp32-wroom-32_datasheet_en.pdf

[31] C. Gomez, J. Oller, and J. Paradells, ‘Overview and Evaluation of Bluetooth Low Energy: An Emerging Low-Power Wireless Technology’, *Sensors*, vol. 12, no. 9, Art. no. 9, Sep. 2012, doi: 10.3390/s120911734.

[32] D. Eridani, A. F. Rochim, and F. N. Cesara, 'Comparative Performance Study of ESP-NOW, Wi-Fi, Bluetooth Protocols based on Range, Transmission Speed, Latency, Energy Usage and Barrier Resistance', in *2021 International Seminar on Application for Technology of Information and Communication (iSemantic)*, Sep. 2021, pp. 322–328. doi: 10.1109/iSemantic52711.2021.9573246.
[33] 'ESP-NOW Wireless Communication Protocol | Espressif Systems'. Accessed: Apr. 15, 2025. [Online]. Available: https://www.espressif.com/en/solutions/low-power-solutions/esp-now
[34] 'Device to Device Communication with ESP-NOW | Arduino Documentation'. Accessed: Apr. 15, 2025. [Online]. Available: https://docs.arduino.cc/tutorials/nano-esp32/esp-now/
[35] 'MQTT - AWS IoT Core'. Accessed: Apr. 15, 2025. [Online]. Available: https://docs.aws.amazon.com/iot/latest/developerguide/mqtt.html
[36] 'MQTT - The Standard for IoT Messaging'. Accessed: Apr. 15, 2025. [Online]. Available: https://mqtt.org/
[37] B. Blanchon, *bblanchon/ArduinoJson*. (Apr. 20, 2025). C++. Accessed: Apr. 21, 2025. [Online]. Available: https://github.com/bblanchon/ArduinoJson
[38] 'FreeRTOS', GitHub. Accessed: Apr. 21, 2025. [Online]. Available: https://github.com/FreeRTOS
[39] 'ESP-NOW - ESP32 - — ESP-IDF Programming Guide v5.4.1 documentation'. Accessed: Apr. 03, 2025. [Online]. Available: https://docs.espressif.com/projects/esp-idf/en/v5.4.1/esp32/api-reference/network/esp_now.html#receiving-esp-now-data
[40] M. A. Huysmans, H. P. van der Ploeg, K. I. Proper, E. M. Speklé, and A. J. van der Beek, 'Is Sitting Too Much Bad for Your Health?', *Ergon. Des.*, vol. 23, no. 3, pp. 4–8, Jul. 2015, doi: 10.1177/1064804615585410.
[41] D. W. Dunstan, B. Howard, G. N. Healy, and N. Owen, 'Too much sitting – A health hazard', *Diabetes Res. Clin. Pract.*, vol. 97, no. 3, pp. 368–376, Sep. 2012, doi: 10.1016/j.diabres.2012.05.020.
[42] T. Yates *et al.*, 'Metabolic effects of breaking prolonged sitting with standing or light walking in older South Asians and White Europeans: a randomized acute study', *J. Gerontol. A. Biol. Sci. Med. Sci.*, vol. 75, no. 1, Art. no. 1, Jan. 2020, doi: 10.1093/gerona/gly252.
[43] S. Komarizadehasl, B. Mobaraki, H. Ma, J.-A. Lozano-Galant, and J. Turmo, 'Low-Cost Sensors Accuracy Study and Enhancement Strategy', *Appl. Sci.*, vol. 12, no. 6, Art. no. 6, Jan. 2022, doi: 10.3390/app12063186.
[44] 'Gravity: I2C Non-contact IR Temperature Sensor For Arduino (MLX90614-DCC)', The Pi Hut. Accessed: Apr. 17, 2025. [Online]. Available: https://thepihut.com/products/gravity-i2c-non-contact-ir-temperature-sensor-for-arduino-mlx90614-dcc
[45] 'Contactless measurement with OMRON D6T Thermal Sensors'. Omron. Accessed: Apr. 18, 2025. [Online]. Available: https://omronfs.omron.com/en_US/ecb/products/pdf/en_D6T_catalog.pdf
[46] '16977-TFMini-S_-_Micro_LiDAR_Module-Product_Manual.pdf'. Accessed: Apr. 17, 2025. [Online]. Available: https://cdn.sparkfun.com/assets/8/a/f/a/c/16977-TFMini-S_-_Micro_LiDAR_Module-Product_Manual.pdf
[47] P. Denysyuk, V. Teslyuk, and I. Chorna, 'Development of mobile robot using LIDAR technology based on Arduino controller', in *2018 XIV-th International Conference on*

Perspective Technologies and Methods in MEMS Design (MEMSTECH), Apr. 2018, pp. 240–244. doi: 10.1109/MEMSTECH.2018.8365742.

[48] 'HC-SR04-Ultrasonic.pdf'. Accessed: Apr. 17, 2025. [Online]. Available: https://www.handsontec.com/dataspecs/HC-SR04-Ultrasonic.pdf

[49] 'ELEGOO Ultrasonic Sensor, HC-SR04 Ultrasonic Distance Sensor Kits for Arduino UNO MEGA R3 Raspberry Pi,Datasheet Available to Download : Amazon.co.uk: Business, Industry & Science'. Accessed: Apr. 07, 2025. [Online]. Available: https://www.amazon.co.uk/ELEGOO-Ultrasonic-Raspberry-Datasheet-Available/dp/B01M0QL1F1

[50] 'CIBSE Recommended Lighting Levels - Mount Lighting'. Accessed: Apr. 07, 2025. [Online]. Available: https://mountlighting.co.uk/technical/cibse-recommended-lighting-levels/

[51] 'CIBSE Recommended Lux Levels'. Accessed: Apr. 07, 2025. [Online]. Available: https://www.kellwoodlighting.co.uk/led-lighting-help/cibse-recommended-lux-levels

[52] 'SLL Lighting Publications'. Accessed: Apr. 07, 2025. [Online]. Available: https://www.cibse.org/knowledge-research/knowledge-resources/engineering-guidance/sll-lighting-publications/

[53] F. Roberts, M. White, S. Memon, B.-J. He, and S. Yang, 'The Application of Human-Centric Lighting in Response to Working from Home Post-COVID-19', *Buildings*, vol. 13, no. 10, Art. no. 10, Oct. 2023, doi: 10.3390/buildings13102532.

[54] A. N. Coogan, A. L. Baird, A. Popa-Wagner, and J. Thome, 'Circadian rhythms and attention deficit hyperactivity disorder: The what, the when and the why', *Prog. Neuropsychopharmacol. Biol. Psychiatry*, vol. 67, pp. 74–81, Jun. 2016, doi: 10.1016/j.pnpbp.2016.01.006.

[55] W. J. M. van Bommel, 'Non-visual biological effect of lighting and the practical meaning for lighting for work', *Appl. Ergon.*, vol. 37, no. 4, pp. 461–466, Jul. 2006, doi: 10.1016/j.apergo.2006.04.009.

[56] 'TEMT6000 Ambient Light Sensor Hookup Guide - SparkFun Learn'. Accessed: Apr. 08, 2025. [Online]. Available: https://learn.sparkfun.com/tutorials/temt6000-ambient-light-sensor-hookup-guide/all

[57] 'bh1750fvi-e-186247.pdf'. Accessed: Apr. 08, 2025. [Online]. Available: https://www.mouser.com/datasheet/2/348/bh1750fvi-e-186247.pdf?srsltid=AfmBOopmiNpuOJNl6PbFnyuC7jsfCRJDO-Sy-mjywbUfswY07p-oWVPy

[58] 'BH1750 Light Sensor.pdf'. Accessed: Apr. 08, 2025. [Online]. Available: https://www.handsontec.com/dataspecs/sensor/BH1750%20Light%20Sensor.pdf

[59] 'TSL2561.pdf'. Accessed: Apr. 08, 2025. [Online]. Available: https://cdn-shop.adafruit.com/datasheets/TSL2561.pdf

[60] '(PDF) Determination of Low-Cost Arduino Based Light Intensity Sensors Effectiveness for Agricultural Applications', *ResearchGate*, Dec. 2024, doi: 10.1590/1678-4324-2022220172.

[61] 'Infrared Waves - NASA Science'. Accessed: Apr. 08, 2025. [Online]. Available: https://science.nasa.gov/ems/07_infraredwaves/

[62] S. Santos, 'Arduino with BH1750 Ambient Light Sensor | Random Nerd Tutorials'. Accessed: Apr. 08, 2025. [Online]. Available: https://randomnerdtutorials.com/arduino-bh1750-ambient-light-sensor/

[63] claws, *claws/BH1750*. (Jan. 28, 2025). C++. Accessed: Apr. 08, 2025. [Online]. Available: https://github.com/claws/BH1750

[64] ekinorak, ‘Flashlight on the Samsung s8/s8+’, r/flashlight. Accessed: Apr. 08, 2025. [Online]. Available: https://www.reddit.com/r/flashlight/comments/82k3zb/flashlight_on_the_samsung_s8s8/

[65] ‘lux’, Metric System. Accessed: Apr. 22, 2025. [Online]. Available: https://metricsystem.net/derived-units/special-names/lux/

[66] ‘EMF-Portal | Limit values’. Accessed: Apr. 18, 2025. [Online]. Available: https://www.emf-portal.org/en/cms/page/home/more/electrical-injuries/limit-values

[67] C. Green, ‘Electromechanical Relays Versus Solid-State: Each Has Its Place’, Library.Automationdirect.com. Accessed: Apr. 18, 2025. [Online]. Available: https://library.automationdirect.com/electromechanical-relays-versus-solid-state-each-has-its-place-issue-9-2007/

[68] O. Seppanen, W. J. Fisk, and Q. H. Lei, ‘Effect of temperature on task performance in office environment’, Jul. 2006, Accessed: Apr. 25, 2025. [Online]. Available: https://eta-publications.lbl.gov/sites/default/files/lbnl-60946.pdf

[69] ‘dht.pdf’. Accessed: Apr. 18, 2025. [Online]. Available: https://cdn-learn.adafruit.com/downloads/pdf/dht.pdf

[70] ‘LM35-DATASHEET.pdf’. Accessed: Apr. 18, 2025. [Online]. Available: https://edn.com/electroschematics/wp-content/uploads/2010/02/LM35-DATASHEET.pdf

[71] D. S. Corp, ‘Programmable Resolution 1-Wire Digital Thermometer’. [Online]. Available: https://cdn.sparkfun.com/datasheets/Sensors/Temp/DS18B20.pdf

[72] D. Saha, *dhrubasaha08/DHT11*. (Apr. 07, 2025). C++. Accessed: Apr. 08, 2025. [Online]. Available: https://github.com/dhrubasaha08/DHT11

[73] ‘Stress’, British Heart Foundation. Accessed: Apr. 18, 2025. [Online]. Available: https://www.bhf.org.uk/informationsupport/risk-factors/stress

[74] J. Kato, A. Kojima, and Y. Niiyama, ‘CARDIAC RHYTHM AND ADRENALINE EXCRETION AS PHYSIOLOGICAL PARAMETERS FOR MENTAL STRESS FOR THE STUDY OF WORK PHYSIOLOGY’, *Ind. Health*, vol. 3, no. 1–2, pp. 1–8, 1965, doi: 10.2486/indhealth.3.1.

[75] M. A. Almarshad, M. S. Islam, S. Al-Ahmadi, and A. S. BaHammam, ‘Diagnostic Features and Potential Applications of PPG Signal in Healthcare: A Systematic Review’, *Healthcare*, vol. 10, no. 3, p. 547, Mar. 2022, doi: 10.3390/healthcare10030547.

[76] ‘PulseSensorAmpedGettingStartedGuide.pdf’. Accessed: Apr. 09, 2025. [Online]. Available: https://cdn-shop.adafruit.com/product-files/1093/PulseSensorAmpedGettingStartedGuide.pdf

[77] ‘PulseSensor Prep’, World Famous Electronics llc. Accessed: Apr. 18, 2025. [Online]. Available: https://pulsesensor.com/pages/pulsesensor-manual

[78] S. K. Longmore, G. Y. Lui, G. Naik, P. P. Breen, B. Jalaludin, and G. D. Gargiulo, ‘A Comparison of Reflective Photoplethysmography for Detection of Heart Rate, Blood Oxygen Saturation, and Respiration Rate at Various Anatomical Locations’, *Sensors*, vol. 19, no. 8, Art. no. 8, Jan. 2019, doi: 10.3390/s19081874.

[79] ‘MAX30102--High-Sensitivity Pulse Oximeter and Heart-Rate Sensor for Wearable Health’. Maxim Integrated Products, Inc., 2018. [Online]. Available: https://www.analog.com/media/en/technical-documentation/data-sheets/max30102.pdf

[80] 'Gravity: Analog Heart Rate Monitor Sensor (ECG) For Arduino', The Pi Hut. Accessed: Apr. 18, 2025. [Online]. Available: https://thepihut.com/products/gravity-analog-heart-rate-monitor-sensor-ecg-for-arduino

[81] *'SparkFun_MAX3010x_Sensor_Library/examples/Example5_HeartRate/Example5_HeartRate.ino at master · sparkfun/SparkFun_MAX3010x_Sensor_Library'. Accessed: Apr. 26, 2025. [Online]. Available: https://github.com/sparkfun/SparkFun_MAX3010x_Sensor_Library/blob/master/examples/Example5_HeartRate/Example5_HeartRate.ino*

[82] *sparkfun/SparkFun_MAX3010x_Sensor_Library*. (Apr. 13, 2025). C++. SparkFun Electronics. Accessed: Apr. 18, 2025. [Online]. Available: https://github.com/sparkfun/SparkFun_MAX3010x_Sensor_Library

[83] '(PDF) Web Services: A Comparison of Soap and Rest Services', *ResearchGate*, Oct. 2024, doi: 10.5539/mas.v12n3p175.

[84] M. O. contributors Jacob Thornton, and Bootstrap, 'Bootstrap'. Accessed: Apr. 19, 2025. [Online]. Available: https://getbootstrap.com/

[85] 'MySQL :: MySQL Workbench'. Accessed: Apr. 14, 2025. [Online]. Available: https://www.mysql.com/products/workbench/

[86] erinstellato-ms, 'Download SQL Server Management Studio (SSMS)'. Accessed: Apr. 14, 2025. [Online]. Available: https://learn.microsoft.com/en-us/ssms/download-sql-server-management-studio-ssms

[87] 'Apache ECharts'. Accessed: Apr. 22, 2025. [Online]. Available: https://echarts.apache.org/en/index.html

[88] Behance, 'Smart Home Dashboard - Katsiatyna Lysykh', Behance. Accessed: Apr. 02, 2025. [Online]. Available: https://www.behance.net/gallery/149895495/Smart-Home-Dashboard

[89] Behance, 'Smart Home Dashboard - AR Shakir', Behance. Accessed: Apr. 02, 2025. [Online]. Available: https://www.behance.net/gallery/125301817/Smart-Home-Dashboard

[90] Unblast, 'Free Smart Home Dashboard Ui Template (Sketch)', Unblast. Accessed: Apr. 02, 2025. [Online]. Available: https://unblast.com/free-smart-home-dashboard-ui-template-sketch/

[91] 'Authorization - HTTP | MDN'. Accessed: Apr. 02, 2025. [Online]. Available: https://developer.mozilla.org/en-US/docs/Web/HTTP/Reference/Headers/Authorization

[92] '32,600+ animated icons - Lordicon'. Accessed: Apr. 19, 2025. [Online]. Available: https://lordicon.com/

[93] 'Popper (v2.×)'. Accessed: Apr. 22, 2025. [Online]. Available: https://popper.js.org/docs/v2/

[94] 'Notification - Web APIs | MDN'. Accessed: Apr. 19, 2025. [Online]. Available: https://developer.mozilla.org/en-US/docs/Web/API/Notification

[95] 'Art. 8 GDPR – Conditions applicable to child's consent in relation to information society services', General Data Protection Regulation (GDPR). Accessed: Apr. 16, 2025. [Online]. Available: https://gdpr-info.eu/art-8-gdpr/

[96] 'BCryptPasswordEncoder (spring-security-docs 6.4.4 API)'. Accessed: Apr. 04, 2025. [Online]. Available: https://docs.spring.io/spring-security/site/docs/current/api/org/springframework/security/crypto/bcrypt/BCryptPasswordEncoder.html

[97] ‘Who Generates the JSESSIONID Cookie in Spring Security? - CodingTechRoom’. Accessed: Apr. 18, 2025. [Online]. Available: https://codingtechroom.com/question/who-generates-jsessionid-cookie-in-spring-security

[98] *usablica/intro.js*. (Apr. 04, 2025). TypeScript. usablica. Accessed: Apr. 04, 2025. [Online]. Available: https://github.com/usablica/intro.js

[99] tzapu, *tzapu/WiFiManager*. (Apr. 03, 2025). C++. Accessed: Apr. 04, 2025. [Online]. Available: https://github.com/tzapu/WiFiManager

[100] ‘System Time - ESP32 - — ESP-IDF Programming Guide v5.4.1 documentation’. Accessed: Apr. 16, 2025. [Online]. Available: https://docs.espressif.com/projects/esp-idf/en/stable/esp32/api-reference/system/system_time.html

[101] ‘Wi-Fi - - — ESP-FAQ latest documentation’. Accessed: Apr. 16, 2025. [Online]. Available: https://docs.espressif.com/projects/esp-faq/en/latest/software-framework/wifi.html#what-is-the-definition-for-wi-fi-channel-can-i-select-any-channel-of-my-choice

[102] ‘lorol/arduino-esp32littlefs-plugin: A LittleFS wrapper for Arduino ESP32 of Mbed LittleFS Library’. Accessed: Apr. 05, 2025. [Online]. Available: https://github.com/lorol/arduino-esp32littlefs-plugin?tab=readme-ov-file

[103] S. Santos, ‘ESP32: Write Data to a File (LittleFS) - Arduino | Random Nerd Tutorials’. Accessed: Apr. 05, 2025. [Online]. Available: https://randomnerdtutorials.com/esp32-write-data-littlefs-arduino/

[104] ‘(PDF) Smart home, security concerns of IoT’, ResearchGate. Accessed: Apr. 15, 2025. [Online]. Available: https://www.researchgate.net/publication/342783170_Smart_home_security_concerns_of_IoT

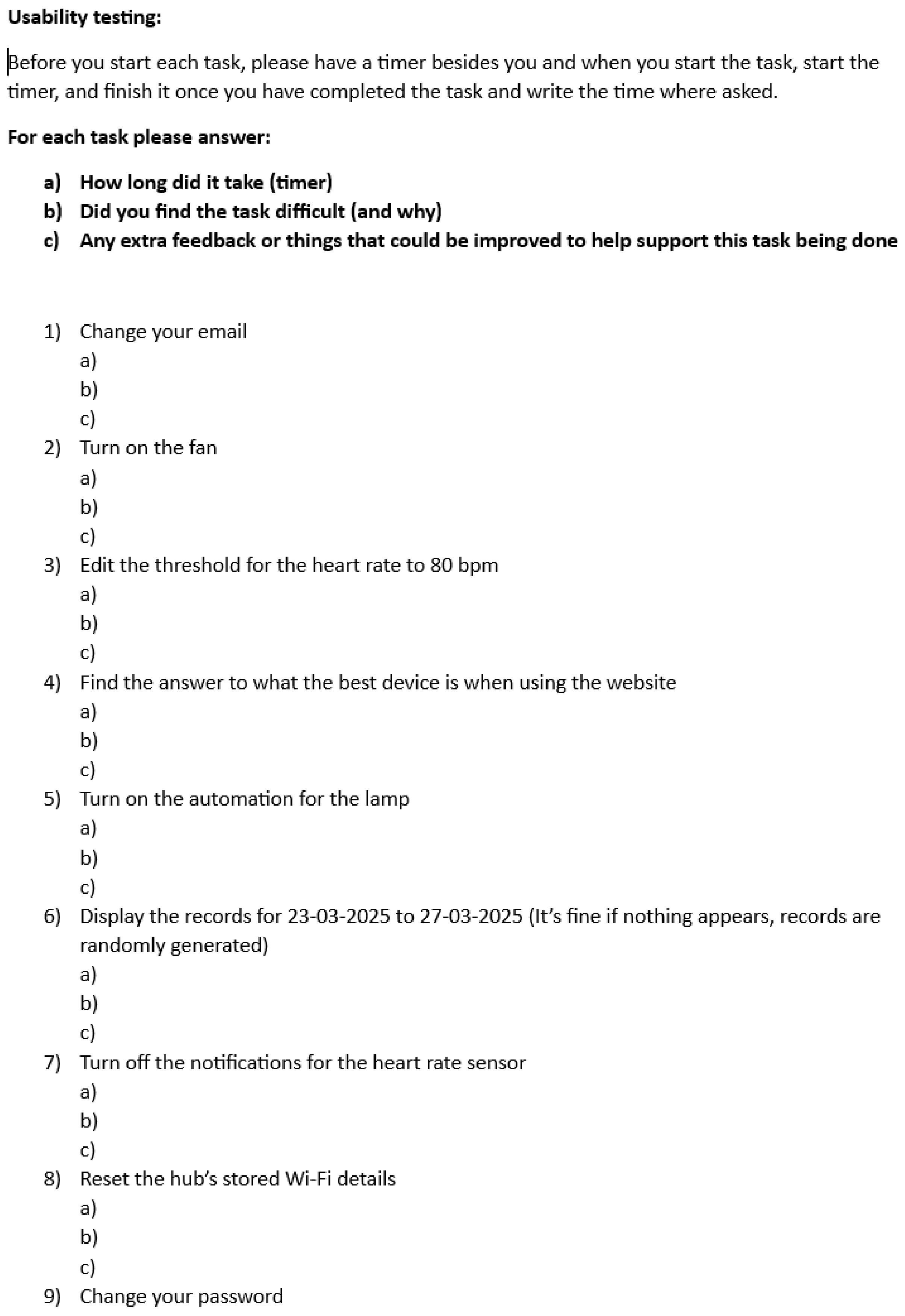
[105] N. Kalbo, Y. Mirsky, A. Shabtai, and Y. Elovici, ‘The Security of IP-Based Video Surveillance Systems’, *Sensors*, vol. 20, no. 17, Art. no. 17, Jan. 2020, doi: 10.3390/s20174806.

Appendix

Table 16 – Key Arduino libraries

Library name	Source URL locations
WiFiClientSecure	https://github.com/espressif/arduino-esp32/tree/master/libraries (Implicitly included in ide/board set up)
HTTPClient	https://github.com/espressif/arduino-esp32/tree/master/libraries (Implicitly included in ide/ board set up)
WiFiManager	https://github.com/tzapu/WiFiManager
esp_now	https://github.com/espressif/arduino-esp32/tree/master/libraries (Implicitly included in board set up)
ESP32Time	https://github.com/fbiego/ESP32Time
ArduinoJson	https://github.com/bblanchon/ArduinoJson
WiFi	https://github.com/espressif/arduino-esp32/tree/master/libraries (Implicitly included in board set up)
LittleFS	https://github.com/lorol/arduino-esp32littlefs-plugin https://github.com/espressif/arduino-esp32/tree/master/libraries (Implicitly included in board set up)
FreeRTOS	https://github.com/FreeRTOS
DHT11	https://github.com/dhrubasaha08/DHT11
BH1750	https://github.com/claws/BH1750
MAX30105	https://github.com/sparkfun/SparkFun_MAX3010x_Sensor_Library

Figure 64 – Usability testing - 1



Usability testing:

Before you start each task, please have a timer besides you and when you start the task, start the timer, and finish it once you have completed the task and write the time where asked.

For each task please answer:

a) How long did it take (timer)
b) Did you find the task difficult (and why)
c) Any extra feedback or things that could be improved to help support this task being done

1) Change your email
   a)
   b)
   c)
2) Turn on the fan
   a)
   b)
   c)
3) Edit the threshold for the heart rate to 80 bpm
   a)
   b)
   c)
4) Find the answer to what the best device is when using the website
   a)
   b)
   c)
5) Turn on the automation for the lamp
   a)
   b)
   c)
6) Display the records for 23-03-2025 to 27-03-2025 (It's fine if nothing appears, records are randomly generated)
   a)
   b)
   c)
7) Turn off the notifications for the heart rate sensor
   a)
   b)
   c)
8) Reset the hub's stored Wi-Fi details
   a)
   b)
   c)
9) Change your password

Figure 65 – Usability testing - 2

a)
b)
c)

10) Remove all the devices from your network
    a)
    b)
    c)

Preference questions:

Please answer the following questions;
1) Do you like the website design? Any positives/negatives/improvements?

2) Did you find the website easy to use? Were there some parts that were more difficult than others?

3) Are there any things you liked in particular?

4) Are there any things you think could be improved overall?

Figure 66 – Usability testing - 3

Heuristic questions:

There are 10 Nielsen heuristics. Please give a severity ranking score for each heuristic from this severity ranking table:

Severity Rating	Description
0	I don't agree that this is a usability problem at all.
1	Cosmetic problem only. Need not be fixed unless extra time is available on project.
2	Minor usability problem. Fixing this should be given low priority.
3	Major usability problem. Important to fix, so should be given high priority.
4	Usability catastrophes. Imperative to fix this before product can be released.

Please give a severity ranking for each heuristic. For each point, I have given the name of the heuristic, followed by the definition in brackets (from slides), followed by a second bracket with questions to help support you and non-technical people come up with an appropriate ranking. If you can, please give a reason why you gave that ranking. You do not have to answer each questions in the second bracket, they're just there to help you understand the definition of the heuristic.

1) **Visibility of System status** (Need to let the user know what is going on at a given time – e.g. not letting the user wonder whether the system is working)
(Is it clear to users what is happening at all times while they interact with the system? Do they understand when their actions are being processed?)

   Severity ranking:
   Reason for severity ranking:
   Any extra feedback:

2) **Match between the system and the real world** (Need to support intuitive and familiar-to-the-user UI – e.g. not letting the user wondering what features, labels, concepts mean)
(Does the language and terminology used in the system align with what users expect or understand? Do the actions or options available in the system resemble real-world objects or concepts that users are familiar with?)

   Severity ranking:
   Reason for severity ranking:
   Any extra feedback:

Figure 67 - Usability testing - 4

3) **User control & freedom** (Need to let users 'control' their interactions with your UI – e.g. allowing users to 'undo', 'redo' and 'exit' actions)
(Can users easily navigate through the system and go back or undo actions if they make a mistake?, Are there safeguards in place to prevent accidental or irreversible actions?)

Severity ranking:
Reason for severity ranking:
Any extra feedback:

4) **Consistency and standards** (Need to meet users' expectations through a consistent design that follows international standards)
(Does the system follow consistent patterns, such as using common icons or layout elements? Does it adhere to established design and interaction standards that users are accustomed to?)

Severity ranking:
Reason for severity ranking:
Any extra feedback:

5) **Error prevention** (Need to provide warnings to users to prevent them from making errors – support error messages)
(Are there clear instructions or cues provided to guide users and prevent them from making errors? Are there constraints in place to prevent users from entering invalid or incorrect data?)

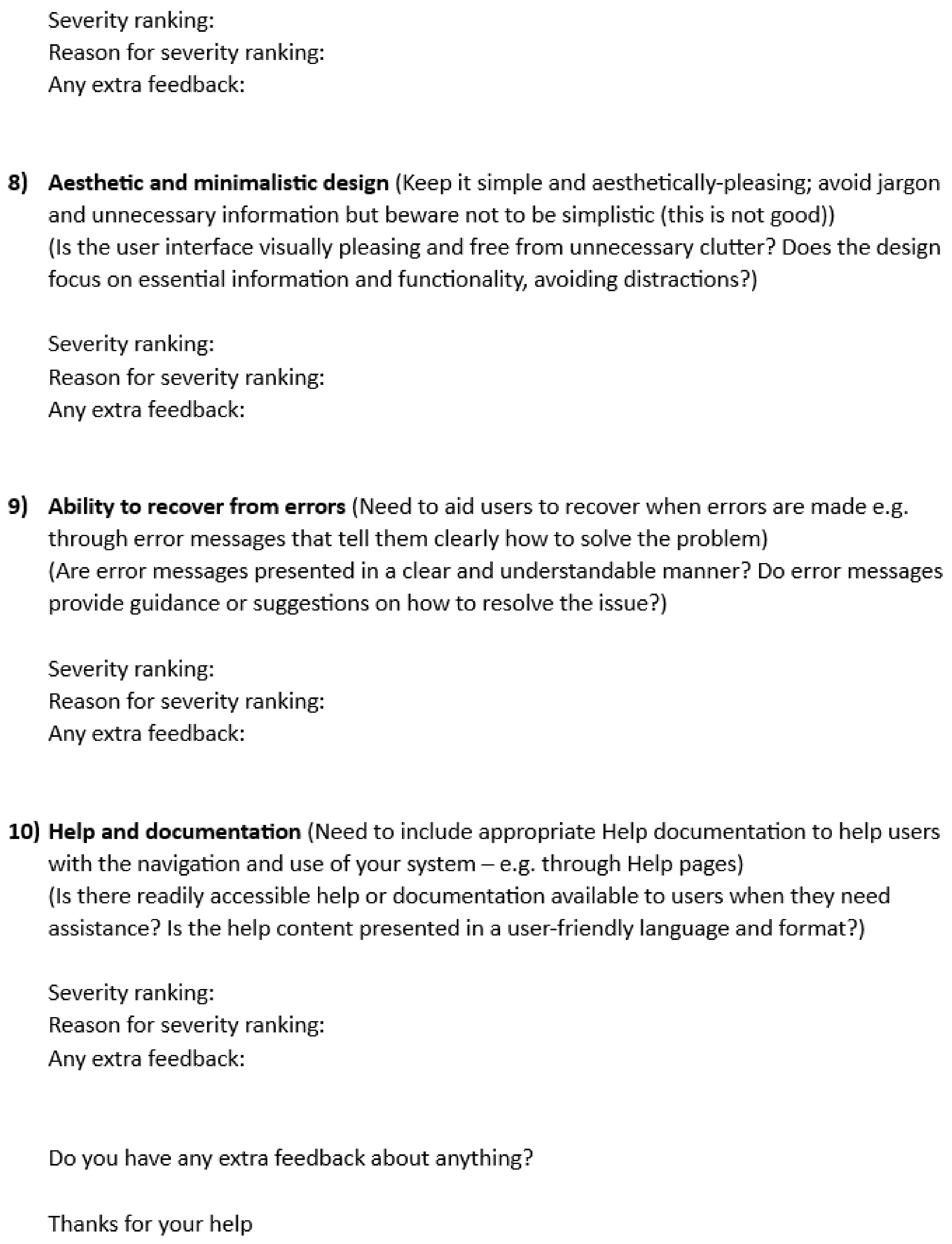
Severity ranking:
Reason for severity ranking:
Any extra feedback:

6) **Recognition rather than recall** (Need to aid users with their memory constraints – e.g. not letting the user recall information on your UI but instead recognise it – e.g. when navigating across pages; careful use of menus)
(Are the labels, instructions, or options presented in a way that users can easily recognize and understand? Are users required to remember information from one screen or step to another, or is it readily available?)

Severity ranking:
Reason for severity ranking:
Any extra feedback:

7) **Flexibility and efficiency of use** (Need to aid users to optimise their interactions – e.g. through customisation options, accelerators and personalisation)
(Are there shortcuts or alternative methods available for performing tasks to cater to different user levels? Can experienced users navigate the system more quickly or efficiently than beginners?)

Figure 68 – Usability testing - 5



Severity ranking:
Reason for severity ranking:
Any extra feedback:

8) **Aesthetic and minimalistic design** (Keep it simple and aesthetically-pleasing; avoid jargon and unnecessary information but beware not to be simplistic (this is not good))
(Is the user interface visually pleasing and free from unnecessary clutter? Does the design focus on essential information and functionality, avoiding distractions?)

Severity ranking:
Reason for severity ranking:
Any extra feedback:

9) **Ability to recover from errors** (Need to aid users to recover when errors are made e.g. through error messages that tell them clearly how to solve the problem)
(Are error messages presented in a clear and understandable manner? Do error messages provide guidance or suggestions on how to resolve the issue?)

Severity ranking:
Reason for severity ranking:
Any extra feedback:

10) **Help and documentation** (Need to include appropriate Help documentation to help users with the navigation and use of your system – e.g. through Help pages)
(Is there readily accessible help or documentation available to users when they need assistance? Is the help content presented in a user-friendly language and format?)

Severity ranking:
Reason for severity ranking:
Any extra feedback:

Do you have any extra feedback about anything?

Thanks for your help

Figure 69 – A fried DHT11 as a result of incorrect online pin diagram