Cardiff School of Computer Science & Informatics

IoT Device for Home Monitoring

Author:
Georgi Pramatarov

Supervised by:
Dr Philipp Reinecke

Moderator:
Dr Jing Wu

May 11, 2018

Acknowledgement

I would like to thank my supervisor Dr Philipp Reinecke for his the incredible advice and guidance during this project. And also another thank you to Laurence Semmens, who helped my with his impeccable soldering skills

Contents

List of Figures

List of Tables

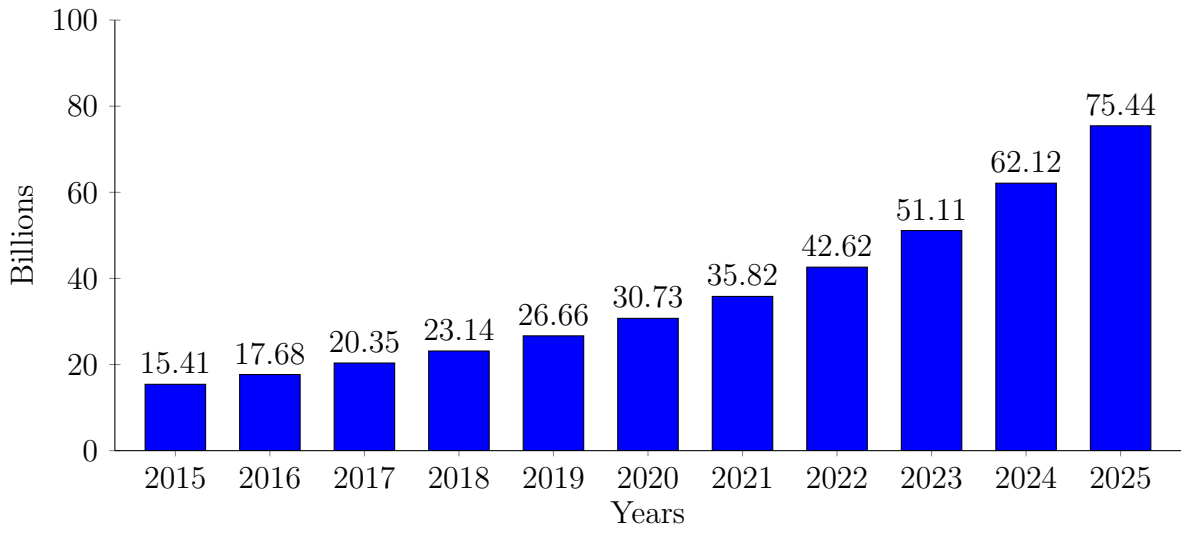
1. Introduction

The aim of this project is to explore the possibilities of developing a fully modular IoT device with a smaller price tag. The focus of this project will primarily gravitate around achieving the modular design, which will provide more flexibility for its users. This project will also aim to follow the currently ongoing standards applied by companies involved it the development of similar devices. In addition, a supporting application will be developed, to provide the user with a full control over the device. The secondary focus will be achieving a smaller price tag, without compromising functionality. In most cases functionality is sacrificed in the name of a smaller price tag. This project will try to alleviate that by providing the user with the freedom of choosing the components that they want, rather than being forced to buy pre-build kits. Focus will also be placed on the dependability and security of the system. The project will also provide a view of the current state of the IoT market and analysis of competing devices.

The structure of the report follows the project development in a chronological order and each section acts as a stepping stone for the next one. Starting with the Background chapter, which includes more information about the motivation behind this project as well as a marketing research on similar devices that compete directly with this one. Also, included in this section is a quick analysis of the current market state of smart home monitoring devices. The System Design chapter acknowledges some of the problems that this project needs to solve, as well as the steps taken to solve them. It also shows the justifications behind each decision and what other alternatives were considered, but disbanded from the final design. The System Design chapter is structured from the highest levels of abstraction moving to the lowest. The focus of the System Design is to describe the design decisions that were made without going into too much technical details. Located directly after the System Design is the Implementation chapter, which depicts the technical decisions that were taken in order to comply with the framework introduced in the system design. The Implementation chapter only focuses on the most important aspects of the system and thus does not include all of the code that will be provided for this project. The rest of the code is provided in a separate zip file. After the Implementation section is the Product Evaluation chapter, which will focus on testing the different segments implemented in the system. The evaluation will be recorded in the form of test cases, which will deliver an insight on the success of the project. The Conclusion and Future Work chapter will provide a brief analysis of the finished project, focusing on the distinct qualities of the system. In addition, the conclusion and future work section will include a discussion on the possible future development as well as the features which were only partially finished or completely left out, due to the time constraints. The Self-Reflection chapter provides an overview of the knowledge that was gained from this project and a reflection on the decisions that ware taken.

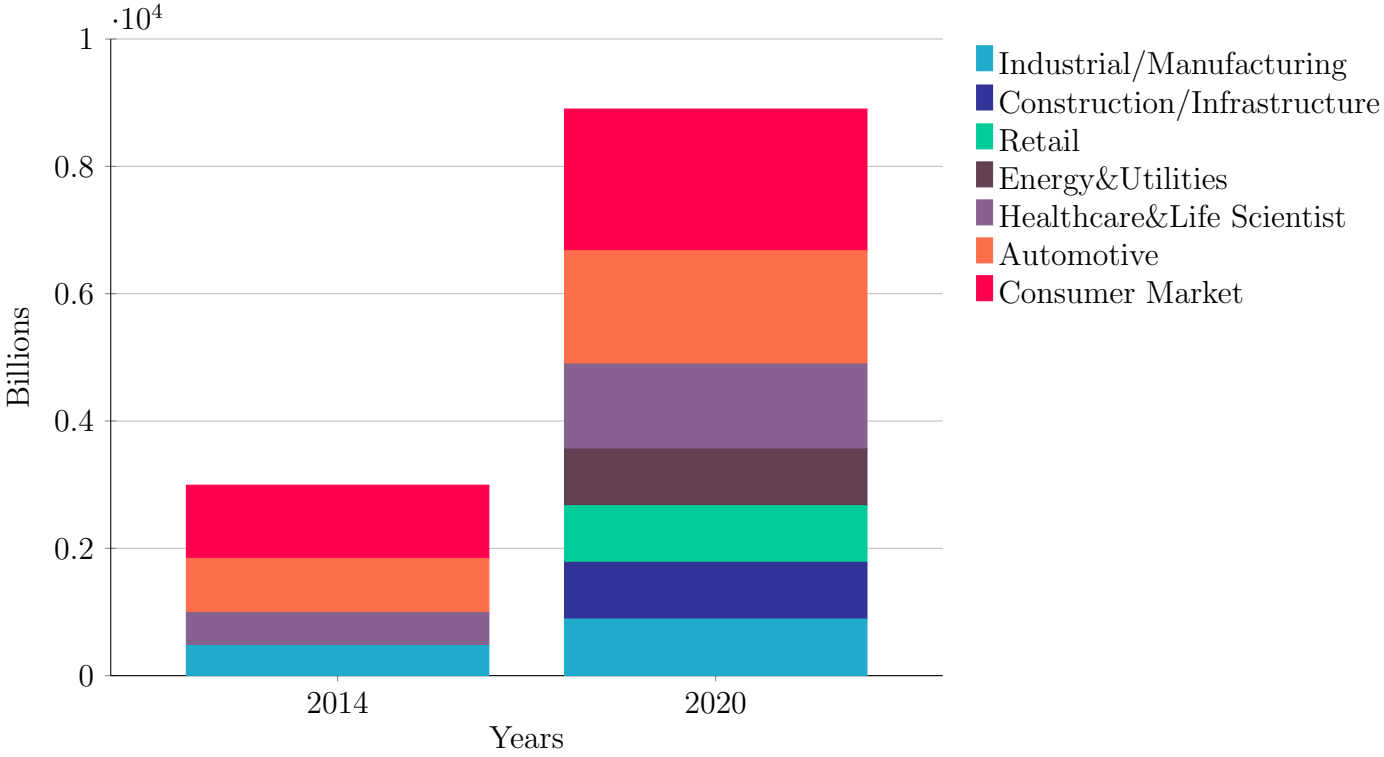
2. Background

Undoubtedly during the past few years one certain technology has established it's importance and that is the Internet of Things. The idea behind The Internet of Things is to connect all smart sensors to the Internet. This technology has already proved its importance in the industrial and agricultural sector and now it's settling its way into people's homes. There is no denying that the Internet of Things has had a positive impact on the way in which organisations operate and manage their business. This is proven by the amount of resources that companies pour in development of this vastly expanding sector. In addition, the IoT sector also serves as the birthplace for a growing empire of new companies who devote themselves to building the new IoT infrastructure. This makes the amount of new IoT devices rapidly climbing, which according to Statista [17] will surpass 75 billion IoT devices by the year 2025. Although these numbers are just predictions and the actual number can vary in both directions, the popularity of IoT devices continues to climb and the historical data confirms that. Between 2015 and 2017 nearly 5 billion more IoT devices were connected to the Internet. In 2018 nearly 3 billion more IoT devices are expected to join the IoT world.



Investment in this sector is also predicted to rise from $2.99 trillion in 2014 to the whopping $8.9 trillion in 2020, which results in an compound annual growth rate of 19.92% [8]. So far these analysis regarded the IoT global market as a whole, however this sector is split into a few categories. The category that this project falls under is the consumer market, which is just a part of the global market pie. Delving deeper revealed that the share taken by the consumer market is the biggest, which in 2014 occupied 38% of the whole IoT market. The predictions for the expansion of the consumer share are also in its favour, showing an incredible increase. This increase according to Statista is going to reach 93% by 2020 and take 25% of the global

IoT market. In addition, the consumer share is predicted to hold its position as the biggest slice of the global IoT pie all the way to 2020.



Despite being just predictions, these analysis paint a really promising picture for the future of the IoT sector. This also shows a huge promise for newer companies, who want to enter the consumer IoT market. However, this increasing growth also posses a lot of challenges for newer companies that want to take part in this highly competitive field. For these companies it is really important to research the current trends and determine what are the users expecting in an IoT device. That is what this segment of the project aims to accomplish. As this project is going to focus on developing a system that can monitor the users home, its important to know whether the consumer will be interested in such a device. For the research of the users current trends the use of the Google Trends tool was employed. Google Trends is a tool that shows what are people searching and most interested in. And according to google trends [19] in the past 12 months peoples interest in the smart home security devices has peaked. The number one most searched type of smart home security device is the IP Camera. In that sense the Internet of Things has the power to revolutionise the way in which people monitor their homes. It can provide users with the ability to remotely monitor their homes for any abnormal events. These events can include a wide variety of different scenarios and being able to detect them really gives the users of this technology a piece of mind. This huge interest in the IoT sector, has disrupted the market of smart sensors in a very positive way. Now not only are sensors getting smarter and smaller in size, but they are also getting cheaper, which means that vendors can develop cheaper devices for the consumer market. However that is not always the case, as some vendors provide their devices as a part of a contract, which can tie the user for upwards of 2 years. Here it should be mentioned that those types of devices will usually offer more security and support, however at a higher price.

Another part of this project was to venture in this highly competitive market and explore the competitors offerings. It is pointless to develop a device that does not offer anything special when compared to its competitors in the segment. Analysis of the segment revealed that ABUS

[1], YALE [20] and ERA [11] have devices that shared some resemblance between each other and this project. All of them have the ability to add extra modules in order to expand the user's network, allowed the user to add alert triggers as well as provide the user with a smartphone application for easy control of the smart network. ERA even went one step further and provided their users with the capability of pairing Lighthouse smart products to their HomeGuard Smart Alarm device. ERA also has a RFID reader for authenticating the user. Disregarding these extra features that ERA provides to their users, all of the fore-mentioned devices are very similar. The main feature that this project is interested in is the modular design of these devices. And ERA scores top marks in this section, allowing the users to connect up to 110 components to their modular device. ABUS Smart Vest ranks second with support for up to 32 components and YALE allows connecting up to 20 components to their Smart Home device. Ideally the device developed in this project would either match or surpass the amount of connected components that the competitors rate their devices for. In addition the second point of this project is to develop a similar device at a lower price.

	Abus SmartVest	ERA HomeGuard	YALE Smart Home	GuerillaGuard Home
Smart Hub	✓	✓	✓	✓
Easy Device Setup	✓	✓	✓	✓
Alarm Scheduling	✓	✓	✓	✓
Supporting Application	Android and IOS	Android and IOS	Android and IOS	Android
Extendability	Support for up to 32 components	Support for up to 50 sensors, 50 RFID tags and 10 remotes	Support for up to 20 components	Support for up to 30 components
Novelty Features		Compatible with Lightwave smart products		Biometric authentication

So far this chapter provided a brief overview of the importance and growth of IoT devices. It also included the main competition in the consumer market as well as a direct comparison against the devices produced by competitors. The next part of this chapter is devoted to the motivation behind the development of this project. As mentioned previously a lot of smart home monitoring devices on the market, are still very expensive and most of them are sold as

pre-build kits. This locks the user into potentially spending more money for components that he might not need. In addition, not every device sold on the market allows the user to expand and potentially upgrade it, without having to buy a whole new kit and scrap the old one. This is the motivation behind the modular design, which is the main focus of this project. Technology is aways going to pave the way ahead and technology developed today will be obsolete tomorrow as it makes way for newer and more advanced one. A great example is the smartphone market, not a long time ago, the typical user would replace their device every 2 years. However, now it seems that users tend to switch their devices every year or in some cases 6 months. The reason for this trend is clear, the market is overflowing with newer devices, and vendors are in a constant state of war with each other, each trying to release newer and better products as fast as they can and in an ever increasing rate. The same could be said about the IoT sector, which although fairly new to the general consumer is rising in popularity. For that reason vendors are constantly pushing new and more advanced products to the market and if the user wants that new fancy device, they have to scrap their old system, because of incompatibility issues. That is why this projects sets out to develop a system in which the various elements can be replaced with newer ones and thus give the user the flexibility to personalise. Developing a modular system, will allow the user to expand it whenever they decide, rather than being forced to buy pre-build kits, this will also allow the user to choose the components they most need. A modular system will also give the user future upgradability, if a certain module gets older or it does not satisfy the user any more, they can simply swap it for a newer version. This type of design will also lower the price of the device, because it allows freedom when choosing between the different components. However in order to be able to achieve all that, there must be some framework that all devices adhere to. In order to be able to promise future upgradability to the user, the system must be designed in a way that is future proof. This is the main constraint of this approach, trying to design a system in a way such that it is compatible with module devices that are not developed yet. In order to combat that, the software that drives the system also has to be designed with modularity taken into consideration. The idea is that an older block of the system can be replaced with a newer one. This however does not solve the whole problem, because the newer software block should be able to work with older module devices. However, breaking the system into modules and software blocks, will allow the user to personalise it however they want. It also means that the system can be expanded in the future and extra functionality can be added later on. This extra functionality can be replacing older algorithms with newer ones, adding support for specialised components and etc. This approach introduces a lot of issues and providing a solution to them will be the focus of the next chapter.

3. Requirements

The point of this project is to not only create a competing device for the IoT market but also to develop a framework. This framework would make the development of new devices much easier. Thats why the point of this chapter is twofold, firstly to establish the requirements that must be satisfied by such a framework and secondly focus on the requirements that need to be satisfied by the IoT device developed in this project.

3.1 Non-Functional

System Structure

- The system must be devided into separate hardware and software modules
- The system must provide a well defined structured framework, which will allow others to develop IoT devices
- The framework must define a clear data and control flow
- The framework must establish precedence and relationships between the different components
-

Ease of Use

- The framework must be intuitive and easy to use
- The supporting smartphone application must provide ease of use
- The supporting smartphone application must be organised
- The supporting smartphone application should display help messages to notify the user of the current state

Interactive

- The supporting smartphone application could provide statistical analysis
- The supporting smartphone application could provide support for multiple languages

Dependability

- The system should be able to recover from connection drops
- The system must be able to work comfortably within a slower network
- The system must be able to handle a large amount of data throughput

Security

- The system must support current security standards
- The system must provide access only to the authenticated users
- The system must provide secure communication channels between the Central Hub and the supporting smartphone application

3.2 Functional

Requirement:

The smartphone application must allow the user to add new edge devices.

Acceptance Criteria:

- The smartphone application opens a dialog in which the user can choose the name of the new edge device and add it to the network.

Requirement:

The smartphone application must allow the user to adjust the Alert trigger.

Acceptance Criteria:

- The smartphone application opens a dialog in which the user can adjust the timeframe for the alert trigger.

Requirement:

The smartphone application must allow the user to authenticate with the biometric scanners.
Acceptance Criteria:

- The smartphone application must allow the user to scan their fingerprint and authenticate.

Requirement:

The smartphone application must allow the user to control the camera stream.

Acceptance Criteria:

- The smartphone application must allow the user to start the camera stream.
- The smartphone application must allow the user to stop the stream.

Requirement:

The smartphone application could show the live data flow from the sensors.

Acceptance Criteria:

- The smartphone application could display cards with information from the sensors.

Requirement:

The smartphone application could allow the user to change the camera streaming quality.

Acceptance Criteria:

- The smartphone application could display a dialog for choosing the stream quality

---

Requirement:

The smartphone application could display the currently connected edge devices.

Acceptance Criteria:

- The smartphone application displays a field with all of the currently active devices

---

Requirement:

The Door Lock must allow the user to reset it whenever they wish to.

Acceptance Criteria:

- The Door Lock provides a hardware button for manually reseting it

---

Requirement:

The Door Lock must provide an alternative way of unlocking the door.

Acceptance Criteria:

- The Door Lock must provide the user with the ability to unlock the door with a key if the sensor fails.

---

4. System Design

Before starting with the system design I had to step back and look at the big picture and establish exactly what requirements must be satisfied. As mentioned in the Background chapter focus should be placed on the modularity of the system. The benefit of developing a modular system is the ability to interchange the different blocks of that system, be that software or hardware. The goal, then is to develop a modular system that can accommodate a big sensor network and is also capable of future expansion. My approach was to begin with understanding the main purpose of IoT devices. In general IoT devices are used for collecting data from the surrounding environment, deriving information from that data and presenting it to the user. Another really important point here is the seamless work of the device, IoT devices consist of multiple components that need to work in unison to provide the user with an enjoyable experience. This is particularly important in the case of this project, because of its modular nature. There needs to be an established framework that will ensure the seamless communication between the different software and hardware components of the system. Therefore this chapter will focus on establishing such a framework. This framework will include identifying the separate stages of the data lifecycle and defining what each of those stages will be responsible for. The framework will also define a mechanism for communication as well as the structure of the data exchanged between the different stages of the data lifecycle. In addition to the abstract model design of the system, this chapter will also focus on the design of the separate components. Emphasis will be placed on their importance and uses within the framework.

4.1 Data and Control Flow

Starting with the abstract model design of the device, which is comprised of 3 stages (Data Collection, Data Processing and Data Presentation), starting from low to high level of integration. These stages can also be interpreted as a stack, the low level contains the raw and unprocessed data, while the higher level presents the already processed data into a more user friendly format. Every stage in the data stack has its own responsibility and each stage requires different hardware and software to achieve that. This stack effectively represents the data flow starting from the first stage(Data Collection) of this system, which is responsible for extracting machine readable data from the surrounding environment and passing it to the next stage of the process stack. This is achieved with the use of sensors/actuators and micro-controllers, which can be summarised as Edge Devices. The job of the sensors are to collect readings from the environment and turn them into machine readable data. Actuators on the other hand are used to change physical conditions based on the generated data. In this case micro-controllers supply the sensors and actuators with a wireless connection, which provides a rather convenient way of communication. The second stage(Data Processing) is the brain of the stack, taking the raw data collected from the first stage, extracting additional information from it and sending it to the next stage. This is achieved by using a central hub, which keeps track of all the sensors in the network, records their data and performs machine

learning algorithms on that data. The third stage(Data Presentation) is where the data is presented in a more user friendly format, such as graphs and tables. This is achieved by using a smartphone application, that takes the processed data from the second stage and displays it in a nicer fashion. With the data flow established the system needs to have some mechanism for the highest level to interact with the lowest. Since the user's only interaction will be with the Data Presentation stage, there needs to be a well established control flow, allowing the user to easily control the lowest levels of the stack. Compared to the data flow, the control flow takes the opposite direction and it moves from the highest to the lowest levels of the data stack. Thus allowing the user to interact with the edge devices, such as smart based locks, thermostats, light switches and etc. The data and control flow complete the data lifecycle.

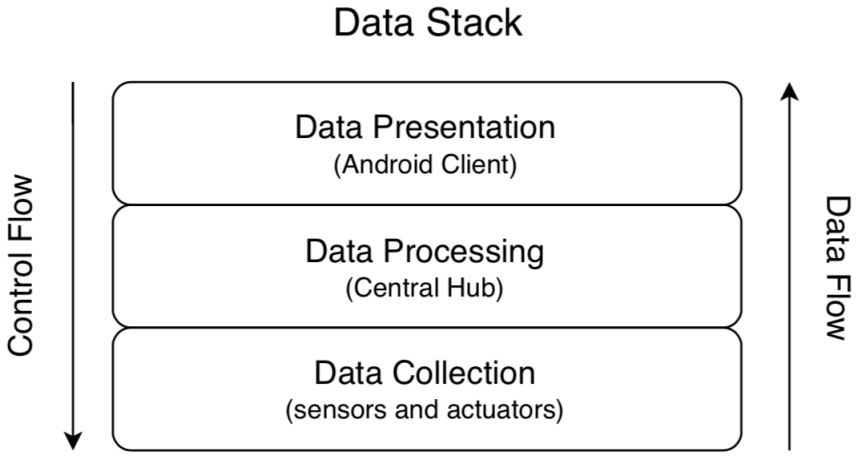


Figure 4.1: DataStack

4.2 System Structure

After the data and control flow were established, the next step was to choose a mechanism for communication between the different levels of the data stack. Since the components of IoT devices are usually not that powerful and they always try to be as efficient as possible and are usually deployed in places where the internet connection is not that fast. Which meant that the communication protocol had to be lightweight so it can run on the chosen hardware and also have a small network footprint. As it turns out there is a plethora of internet communication protocols that fitted this description, however the one that caught my attention was the Message Queuing Telemetry Transport (MQTT) protocol [15]. MQTT was specifically designed to be light-weight publish/subscribe messaging transport protocol . It is very useful for communication over networks where network bandwidth is a premium as well as for communication where a small code footprint is required and it is well supported by the components chosen for this project. In addition, MQTT is well documented and even though it was developed in the 1990s, it is very popular amongst IoT devices and it is still widely supported. After choosing the mechanism for communication between the different levels of the data stack, the next step was to determine the channels for communication. In order to create a more efficient design, the different levels of the data stack should be treated differently. The initial plan for communication between the stack levels was to have a separate

communication channel for each device that connects to the Hub. However that proved to be inefficient when adding large number of devices, each of which had to reserve a separate channel which would strain the system. Rather than creating a separate communication channel for each device, this framework defines just six communication channels for the entire system. These communication channels are split between the different levels of the data stack. Focusing on the communication between the lower two stages of the data stack, the framework defines three communication channels. These are the data, control and configuration channels, each playing a different role. Starting with the data channel, which is responsible for carrying data from the edge devices to the Hub. The control channel operates in the opposite direction to the data channel and carries control commands from the Hub to the edge devices. The configuration channel is used for transmission of configuration details when a new device connects to the Hub. In addition, the configuration channel is bidirectional and establishes a two way communication between the edge device and the Hub. Where as the data and control channels are omnidirectional, like a one way street, they can send data in one direction only. Communication channels between the highest levels of the data stack use a similar methodology. Again the android data and control channels are the main communication channels, with the difference of the naming scheme. This naming scheme was introduced to remove the confusion between the naming of the channels. In addition to that there is the media channel, which is responsible for carrying the camera streaming data from the Hub to the Android Client. More detailed information about the camera functionality will be included later on. In addition, because this system is modular, extra communication channels can be added in future development if needed. My motivation behind designing the communication channels in such a way was to make the system more simplistic and efficient. This way the system is more organised and can be easily confined within a framework that is intuitive and easy to understand. Applying these changes to the data stack, so now it will reflect the communication channels, resulted in the following schematic.

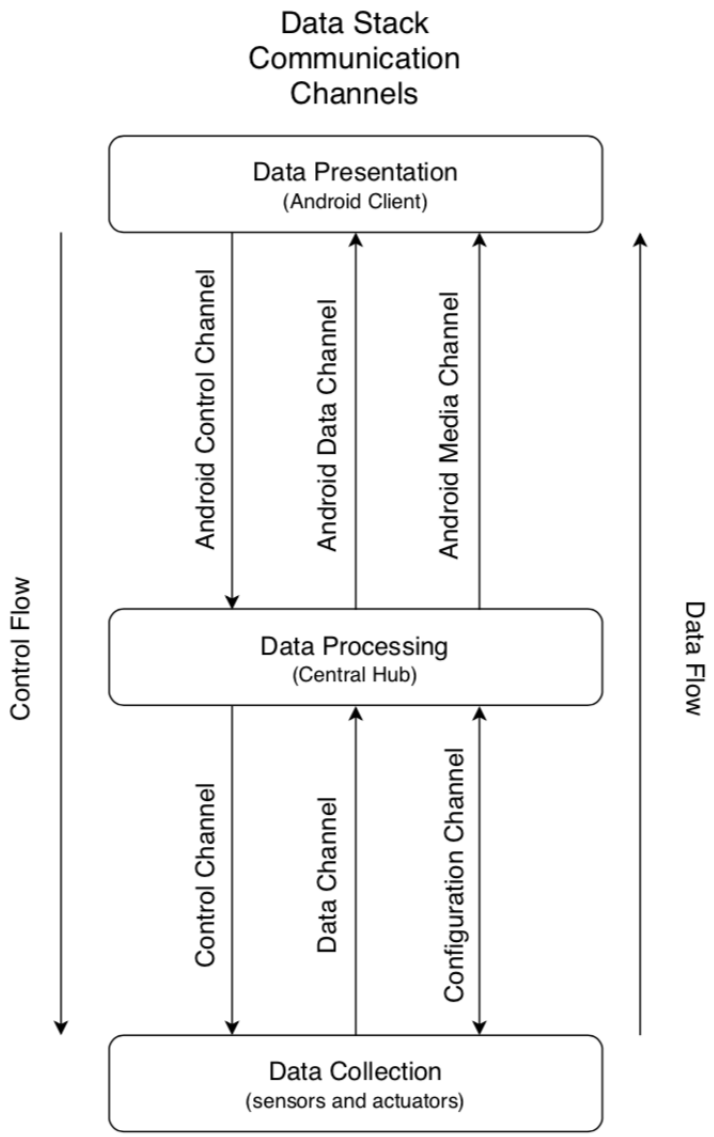


Figure 4.2: DataStackCom

The data stack graph represents the abstract model of the system. It describes its main blocks, while the system design graph below shows the real world application of the abstract design. Taking into account all of the design choices taken so far produces the following schematic. The Data collection is represented by the edge devices located on the left in the graph, the Data Processing block is represented by the Hub and finally the final block the Data Representation is represented by the smartphone. Both of the graphs represent the same system, however the extended data graph shows the abstract representation of the system. While the second graph shows the real scenario for this system.

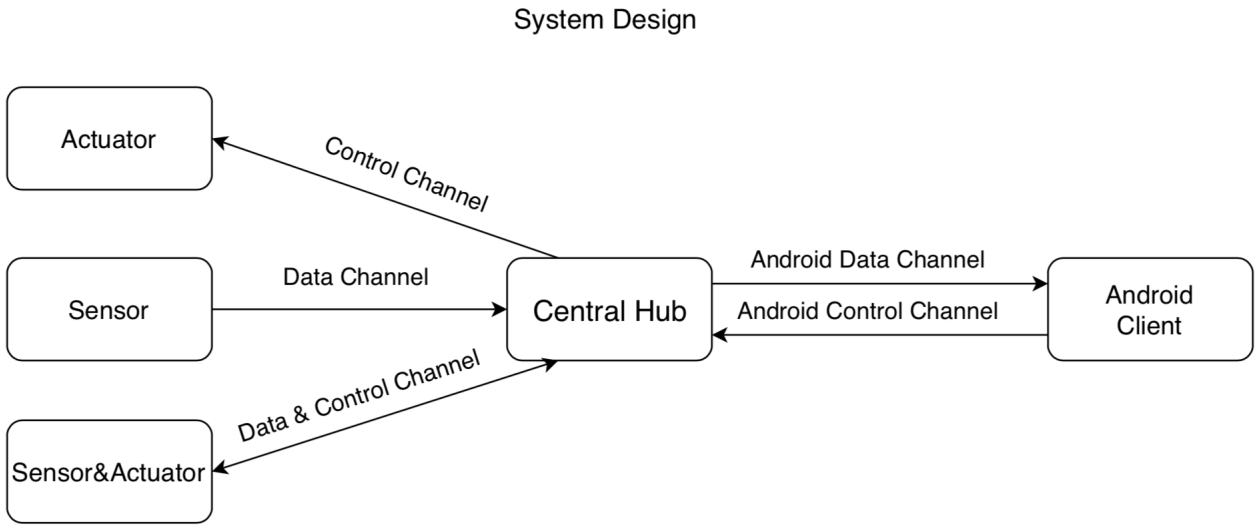


Figure 4.3: SystemDesign

In addition to the communication channels, my design also categorises all of the edge devices into three types. The three edge device types are as follows: sensors, actuators and sensor&actuators. In IoT devices such as this one sensors and actuators are a given, however sometimes a combination of both can also be used in some instances, such as the smart door lock. Depending on the type of edge device a different communication channel is used, either the data channel or the control channel. The sensors make use of the data channel while the actuators use the control channel. In the case of the sensor&actuator both the data and control channels can be utilised to achieve the required bidirectional communication with the Hub. This distinction between the edge devices made the system even more organised and made it much easier to develop new edge devices, once this framework is adhered to. Developing a new edge device now comes to choosing what type of device the developer is building and that would immediately suggest the communication channel they need to use. The Android Client can also be considered as a type of an edge of device, since it is located at the edge of the network. However, the Android Client makes use of different communicational channels and it should not be confused with the edge devices that belong to the Data Collection stage. To disperse any confusion and bring more clarity, the table below shows what channels each device connected to the Hub should support. Again as previously mentioned, when developing a new device, picking the type of device would immediately suggest the use of the correct communication channels.

	Sensor	Actuator	Sensor&Actuator	Android Client
Data Channel	✓	-	✓	-
Controll Channel	-	✓	✓	-
Configuration Channel	✓	✓	✓	-
Android Data Channel	-	-	-	✓
Android Control Channel	-	-	-	✓
Media Streaming Channel	-	-	-	✓

This narrowing of the channels introduced earlier, creates another problem and that is the distinction between the different edge devices of the same type. Edge devices already have three categories that they can fall under, however devices from the same type will be able to share the communication channels. This can create confusion between the devices. To alleviate this problem there needs to be an established identification between the edge devices. This can be solved by introducing a unique identifier that uses information about the edge device and converts that into an Identification number. The mechanism behind the generation of the unique identifiers will be explored further in the Implementation section. For now lets assume that each edge device has a unique identifier or ID number. The framework so far has defined the data and control flow as well as the communication channels between the data stack levels. However, communication can not be established yet due to the absence of a clear messaging structure. This messaging structure will ensure that no conflict is raised between the edge devices. It will also preserve the consistency and integrity of the transmitted data. However, because the structure of the system design consist of more than one communication channel and each of them performs a different task. Therefore each communication channel will make use of a slightly different messaging structure, which is shown in the table below.

	Message Structure	Data Type
Data Channel	[SensorID/Payload] or [SensorID/Payload/UserID]	Encoded Byte Stream
Control Channel	[SensorID/Command]	Encoded Byte Stream
Configuration Channel	[SensorName/SensorType] and [SensorID]	Encoded Byte Stream
Android Data Channel	[SensorID/Payload]	Encoded Byte Stream
Android Control Channel	[ClientID?=Feature?=Command]	Encoded Byte Stream

The first three entries in the table show the message structure of the communication channels between the first and second levels of the data stack. Starting with the Data Channel, which can have two different message structures, depending on the type of edge device. The first message structure ([SensorID/Payload]) should be implemented and used by the sensors. However, as pointed out earlier there could be a scenario where a combination of sensor and actuator can be used. In this case the messaging structure can be different. For example the DoorLock which is a Sensor&Actuator type of edge device, where knowing the user who accessed the door is important. That is the purpose of the second messaging structure, it allows the system to monitor who triggered the Sensor&Actuator device. The Control channel's messaging structure uses two parameters namely the sensor identification number and the command. The identification number of the sensor is used to preserve the data integrity, because as mentioned previously devices of the same type can access the same communication channels. The Configuration channel is bidirectional and requires a different approach to the messaging structure. The structure from the edge device to the Hub requires two parameters and they are the name of the sensor and its type. While the structure in the other direction, from the Hub to the edge device requires only one parameter which is the newly generated Identification number.

As mentioned previously there is a third communication channel between the second and third stage of the data stack. This channel is specifically reserved for the camera streaming functionality. The design of the camera streaming functionality is simple. The camera module is directly connected to the Hub and communicates with the Hub via a ribbon cable. As defined by the raspberry pi foundation, there are two main data buses for communication between the camera module and the Hub. These are the I2C bus which is a relatively low bandwidth and it is used to carry commands and configuration information from the Hub to the camera module. The second connection is the CSI bus which operates at a much higher bandwidth and it carries pixel data from the camera module to the Hub. In this iteration of the project the camera is directly connected to the Hub, which can be improved in future generations. This introduces the final change to the System Design. The complete system now looks like.

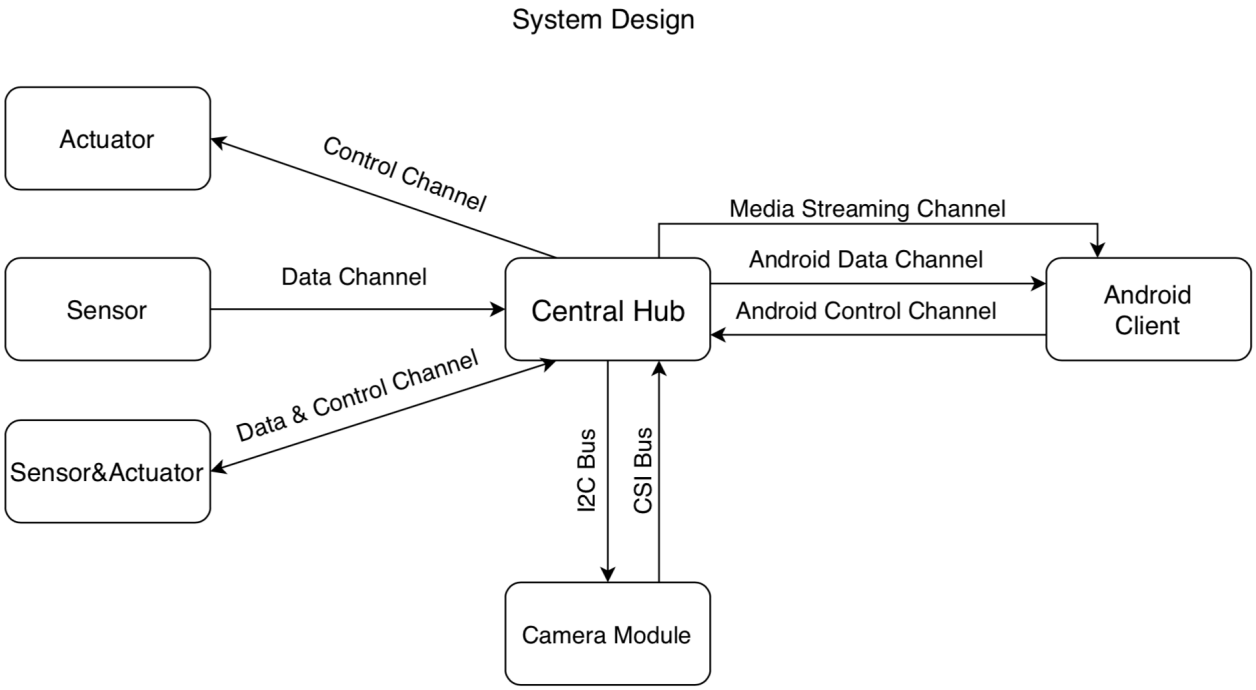


Figure 4.4: SystemDesign

This concludes the system design and structure. The main points here are the data stack, which defines the data lifecycle as well as establishing the data and control flow. In addition to the data stack, establishing the communication channels, provided the vital connections between the stages of the data stack. Furthermore, introducing a predefined message structure made communication trough the channels less ambiguous. Categorisation of the edge devices also delivered a much needed structure to the edge device network. It also added some composure in the development of new edge devices. The next sections will layout the design of the software that is housed in the main hardware components included in the project.

4.3 Edge Devices

Starting with the first level of the data stack, which is occupied by the Edge Devices. They act as the eyes and ears of the whole system, constantly sensing and registering changes in the surrounding environment as well as providing a translation between the virtual and physical environment. These changes then need to be transmitted to the next data stack level. It is vital for these devices to be accurate and dependable, since all of the data that accumulates within the system is generated by them. Incorporating these devices into the previously defined system design, does not require much work from the developer. The fore-mentioned framework makes the development of new edge devices straightforward and easy. This system is modular so there are no limits, when it comes to developing new devices. However, this modular design also creates another problem and that is the connection of new edge devices to the existing network. The focus of the next section will be the design of the mechanism for adding new edge devices to the network.

4.3.1 Addition of new Edge Device

Ideally the procedure for adding new edge devices should be easy, intuitive and would not require the user to remember tedious details like usernames and passwords. Ideally the user would just press one button and that would conclude the addition of new devices. However, providing some personalisation should also be taken into consideration. For example allowing the user to chose the name of the device they wish to add is a nice touch. The final design combined both simplicity and personalisation by allowing the user to chose a name for the new device and to add it to the network with the press of one button. The name for the sensor is a required field and it is used for the ID generation process. After naming the new device the Android Client will send the required configuration information to the Edge Device. The transmission of the configuration data between the Android Client and the Edge Device should not be confused with the configuration channel defined previously. This communication channel between the Android Client and the new Edge Device is different from the one defined earlier. Only when the Edge device receives the required information can the second stage of the configuration process begin. The second stage of this process is to connect the Edge Device to the Central Hub.

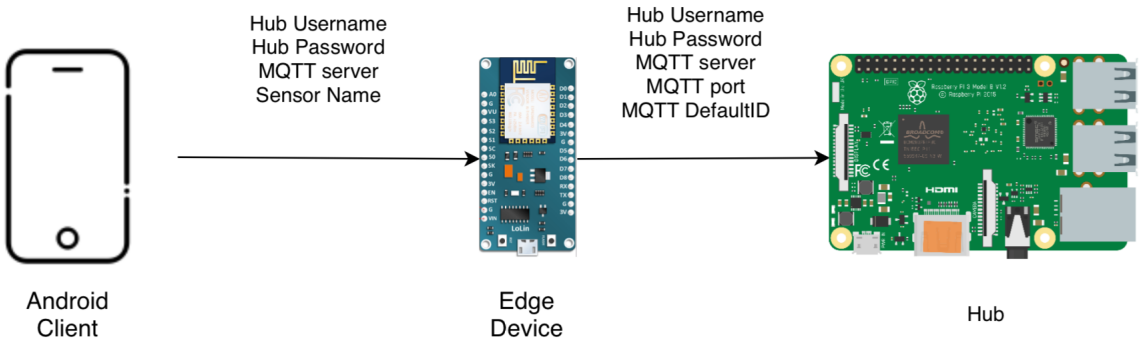


Figure 4.5: New Edge Device Setup

On its first connection to the Hub, the edge device will connect using a default ID number and will send its name and type trough the configuration channel that was defined earlier. This step is really important and each new Edge Device needs to use the configuration channel on its first setup. The messaging structure is also of great importance, because the Hub will expect the device to send its name and type in this exact order. Once the connection is established and the configuration details received by the Hub, can the next stage begin. This stage is responsible for generating the unique ID number and sending it back to the edge device. Then the Edge device has to reconnect back to the Hub with its new Identification number.

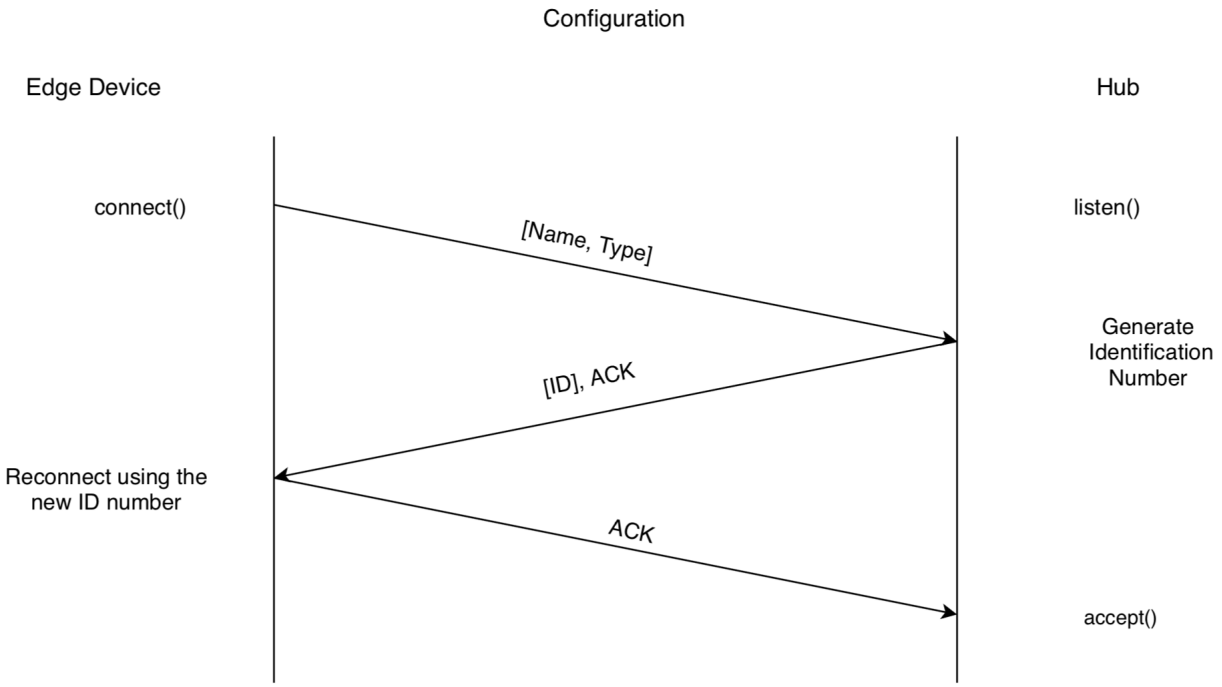


Figure 4.6: ID Configuration

This concludes the setup process for the new edge device and the device can now enter its normal state. If any issues occur during the setup of the device the standard procedure is to restart the device via the RST button and start the setup process again. It should be noted that the distance between the edge device and the Hub will affect the time it takes for the edge device to connect. In addition, depending on that distance the normal operating state of the edge device might also be affected.

4.4 Central Hub

As mentioned before the Central Hub is the brain of this IoT device and its main task is to process the raw data from the edge devices. In addition to data processing, the Hub also provides a connection between the lowest and highest data stack levels. Since the system is modular the Central Hub is also split into separate block, this will allow the addition or removal of sections if that is needed in future updates. In its current state the Central Hub consists of three main block. These are the MQTT Broker, Database and Media server block. The aim of this section is to explain the design of each of those blocks as well as establishing a relationship between them.

The first block is the MQTT Broker, which is split into three segments. First of which is the data receiver segment, which is responsible for handling messages sent on the communication channels.Once the messages are received they are handed to the next segment of the system. This position is fulfilled by the data processing segment. This segment is responsible for extracting additional information from the raw data captured by the edge devices. After processing the data, the next step is to send the extracted information to the smartphone client, which is fulfilled by the departure segment. The combination of these three parts completes the communication between the data stack stages. These three blocks of the system are essential, because they are the glue that holds the system together. Without them

the edge devices wont be able to communicate with the smartphone client. The graph below shows the design of the Hub, with the incorporation of these blocks. The combination of these three blocks creates the MQTT Broker.

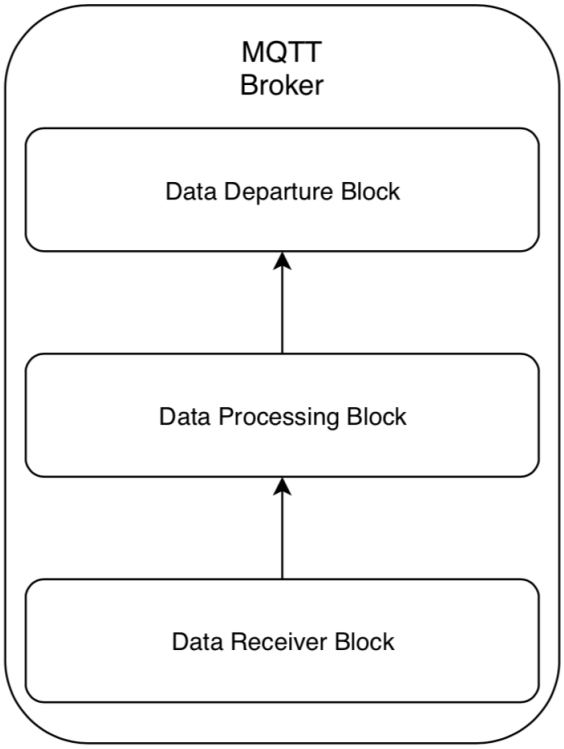


Figure 4.7: MQTT Broker Design

The Hub also needs a mechanism to store information that passes trough it. This information can be accessed later on by other blocks of the system. That can be achieved with the use of database. Choosing a database comes down to whether a structured approach is needed, which can be satisfied by an SQL Database. The other path is choosing a NoSQL approach, which in this case was unnecessary, as the information is not going to be stored across multiple devices. The nature of this project, if extended, can make use of a NoSQL approach, however for the time being a more structured approach was needed. The best solution in this case was to use a MySQL database, which would provide a structure for the stored data. The database is accessed every time information passes trough the MQTT Broker's receiver segment. The job of the database is not only for data collection, but also storing the users authentication information. In its final design the database includes three tables, the SensorMap, the DataCollection and the Users table. Starting with the SensorMap, which is used to store the mapping between the edge device IDs, names and types. The DataCollection table is used for collecting any information sent from the edge devices as well as the smartphone client. Finally the Users table holds all of the current system users that are authenticated to use the device.

The Hub also houses the media server, which is responsible for handling the streaming functionalities of the system. Complying with the modular design of the system, the streaming server should also be treated as a separate block, much like the Database and

MQTT Broker. The media server plays the role of reading and encoding the pixel data sent from the camera module. Once the data is encoded the media server has to transmit that data to the smartphone client. This is the final block of the Central Hub's design. The final design of the Central Hub is represented by the graph shown below.

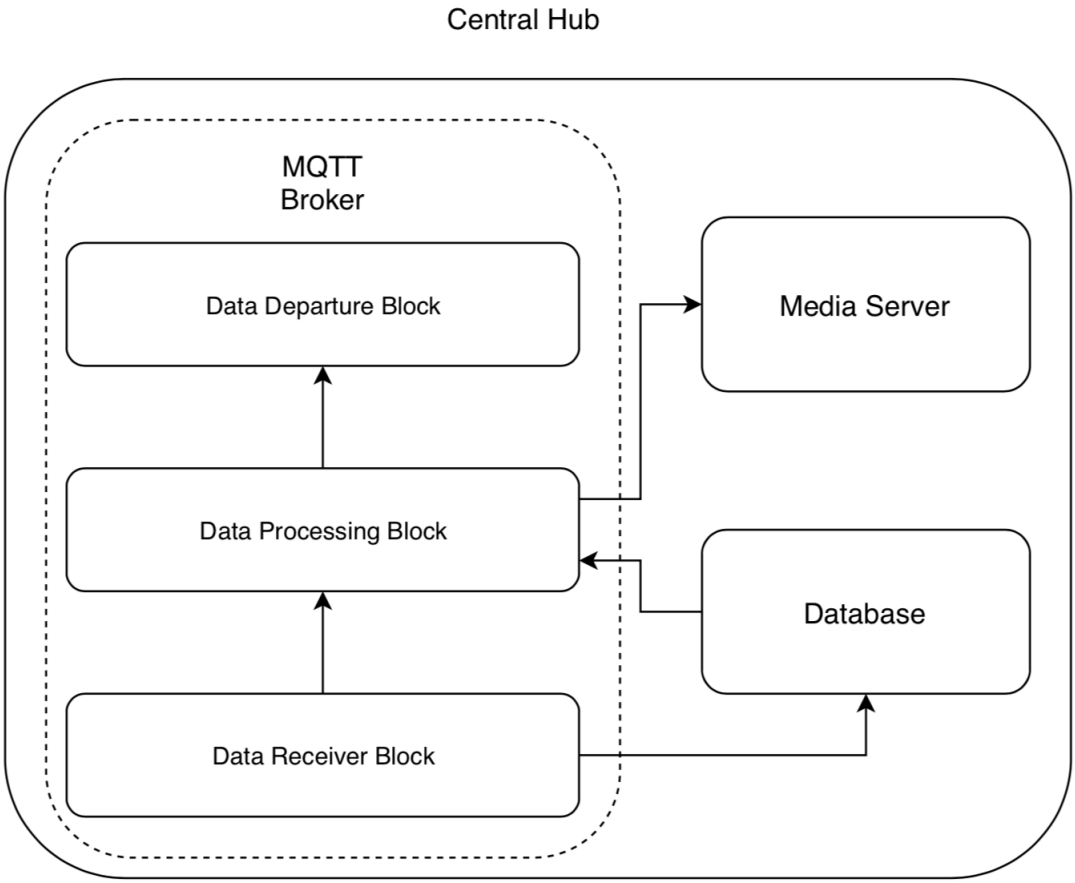


Figure 4.8: Central Hub Design

The design consists of three main blocks as mentioned previously. The sections above focus on the individual design of the different blocks that build up the Central Hub. This section will focus on the relationship between these blocks. The first relationship is between the receiver segment and the Database block, this relationship is vital and provides the receiver segment with the capability to save every piece of data that passed trough it. The second really important relationship is between the data processing segment and the Media stream server, the idea here is to allow the data processing block to control the media server. Whenever the user starts the stream the data block check if the user is authenticated to do so and it that is the case, then the media server can be started. That is why the other vital relationship is between the Database and data processing block. This connection allows the data processing block to access the database when it needs to. This action can be performed to authenticate a user or get access to historical data that the user might request.

4.5 Smartphone Client

Nowadays everyone has a smartphone in their pocket and this allows us to connect to the rest of the world. A big part of the Home monitoring device is the ability to be controlled remotely when ever the user wishes to. Here the smartphone comes into its own, providing that flexibility to the user. The development of a supporting application for controlling this IoT device is crucial. The design of this application in this project is a proof of concept. The smartphone client can be created in any way as long as it abides to the fore-mentioned framework. The full design and implementation of the supporting smartphone application will be further discussed in the Implementation chapter.

To recap, this chapter defines the data stack, which lays the foundations of the abstract design of the system. The data and control flow show how the system should behave and the precedence of each stage in the data lifecycle. In addition, this framework also establishes communication channels between the different levels of the data stack and a mechanism for using these communication channels. One more important thing here is the structure of the messages that are transmitted over the communication channels. This structure should be followed, otherwise the Hub wont be able to save and process the data. Also discussed in this section is the design of the edge devices, the Hub and the Android Client. Starting with the Edge devices, my main goal was to make it easy for the developers, and so with this design the developers have a lot of freedom, the only constraint that they need to abide by is the framework that defines the communication channels and the messaging structure. The design of the Hub was focus on modularity as well, so that the software can be broken into separate blocks and can be upgraded in the future. These upgrades can either add extra functionality or they can replace older blocks with newer, more advanced ones. The next section will focus on Implementing the system design in a real world scenario.

5. Implementation

This chapter will focus on the techniques used to implement the system design. The chapter is broken into five sections, starting with a brief overview of the hardware components used for the implementation. Followed by the Software tools section, which primarily provides justification for the tools that were used to help with the implementation process. The following three chapter describe the steps taken to develop each component of the system. Starting with the Door Lock implementation and moving towards the Central Hub implementation and the final section is the implementation of the smartphone application. This chapter will depict only the main functionalities of the system, therefore most of the code that was used will not be present. The full code that was implemented will be provided in a separate zip file.

5.1 Hardware components

Since the project is based around the idea of cheap hardware, the parts list provided below shows the devices used in the development of this project. Just as a disclaimer, all of the following parts were purchased from amazon.co.uk at retail prices. Some of the parts can be found for less on eBay or directly from the manufacturer's websites. However, since almost all sensor manufacturers are based in China, it takes longer for the parts to be delivered(approximately one month, according to estimates provided by the couriers). Also some manufacturers do not always agree on sending just one sensor. Considering the time frame in which this project had to completed, I have opted to purchase the parts listed below from amazon, so that I can take advantage of the one day prime delivery. Furthermore if this project is upscaled, the parts can be ordered directly from the manufacturer in bigger quantities and that can reduce the price per sensor. In addition, some of the parts used in this project can be substituted with other alternatives. Below I have listed the parts that I have used to develop the project and my dependability evaluation is based on these parts.

Raspberry Pi 3 [4]
The Raspberry pi is a great addition to this project, thanks to its smaller price tag and incredible versatility. The 3'rd edition of the Raspberry pi also adds WiFi and Bluetooth, which makes it ideal for this project. This part can be substituted with any linux device that has a WiFi anthena and similar specification to the Raspberry Pi 3.

MakerHawk NodeMCU ESP32 [2]
The ESP32 is a low-cost WiFi chip with a full TCP/IP stack and can provide WiFi to any micro controller. That combined with the NodeMCU architecture delivers a very powerful micro-controller in a small form factor. In addition, the ESP32 micro-controller adds a dual-core to the NodeMCU architecture, which provides much more versatility. This part can be substituted with the NodeMCU ESP8266 board.

MakerHawk v1.3 Raspberry Pi 3 Camera Module [3]

The SmartHawk camera module was chosen for this project, because it was cheaper than the official camera module, produced by the raspberry foundation. In addition, the camera module has the capability to capture video at 1080p, despite its smaller resolution of 5mp. It was also build to be compatible with the Raspberry Pi, thus it is well supported and documented.

XCSOURCE PN532 NFC RFID reader/writer [5]
This sensor was chosen, because of its flexibility when it comes to I/O communication, it provides SPI/I2C/HSU connections. Furthermore it was well documented, cheap and vastly supported amongst the community. Since most libraries for it are community based.

Smartphone with a fingerprint scanner running Android OS
The decision behind this choice were the very expensive fingerprint scanners and the limited libraries for implementing them with the nodeMCU ESP32 micro-controller. The most cheap alternative was to use the fingerprint of a smartphone and most phones today have fingerprint scanners. In addition, Android provides libraries for implementing this functionality.

5.2 Software tools

This project included the use of multiple software tools for achieving the development of the final product. Starting with the micro-controller programming which was done trough the Arduino IDE [6] [**androiddoc**]. This particular IDE provided the most support for the micro-controllers that were chosen for this project. In addition, the Arduino IDE provided a wide range of libraries, which made the whole development process much easier. For the programming of the Hub the main software tool that was used was the Atom IDE [7]. It is similar to Sublime Text and provides an overall nicer development environment. The rest of the Hub implementation was done trough the terminal with the help of the nano text editor. The Android application was developed in Android Studio [12], which is the official software tool for Android application development provided by Google. Although other environments such as Eclipse are available, the Android Studio provides the best support for Android application development in multiple versions of the Android operating system. In addition, the provided gradle build tools made it easy to add any external libraries, which made development easy and efficient.

5.3 DoorLock

The smart door lock is not a new idea and there are quite a few smart door locks out there, the difference is that this design uses the biometric scanner of the user's smartphone. This approach was inspired from the smartphone contactless payment, which takes advantage of the user's fingerprint scanner and NFC anthena to authenticate the transaction. The design of the DoorLock follows this exact same principle. Simply put, the DoorLock is a micro-controller connected to a NFC sensor and a door solenoid actuator. In this project the DoorLock can be regarded as a proof of concept, since the door solenoid is not present. To mitigate the absence of the actuator, an LED light will be used instead. This light will simulate the door solenoid open and close states. Communication between the two components is achieved trough a SPI interface. Other interfaces like I2C were also considered, however in the end the SPI interface was preferred. The full duplex of the SPI interface, the ability to transfer more data at once and the synchronous mode of operation were the perfect fit for this project. The SPI interface uses a master-slave design principle. The master device has full control over the slave device over a four wire serial bus. This four wire serial bus consists of the SCK: Serial Clock, MOSI: Master Output Slave Input, MISO: Master Input Slave Output, SS: Slave Select. The SCK,

MOSI and SS wires are the output from the master to the slave, while the MISO wire outputs from the slave to the master. In addition to these four serial bus wires, there are two more connections going from the micro-controller to the NFC sensor and they are the Power and Ground wires. As mentioned earlier the inclusion of a LED will simulate the absence of an actuator. This LED is directly soldered to the micro-controller, which provides the power and control. The scheme below shows the design of the DoorLock.

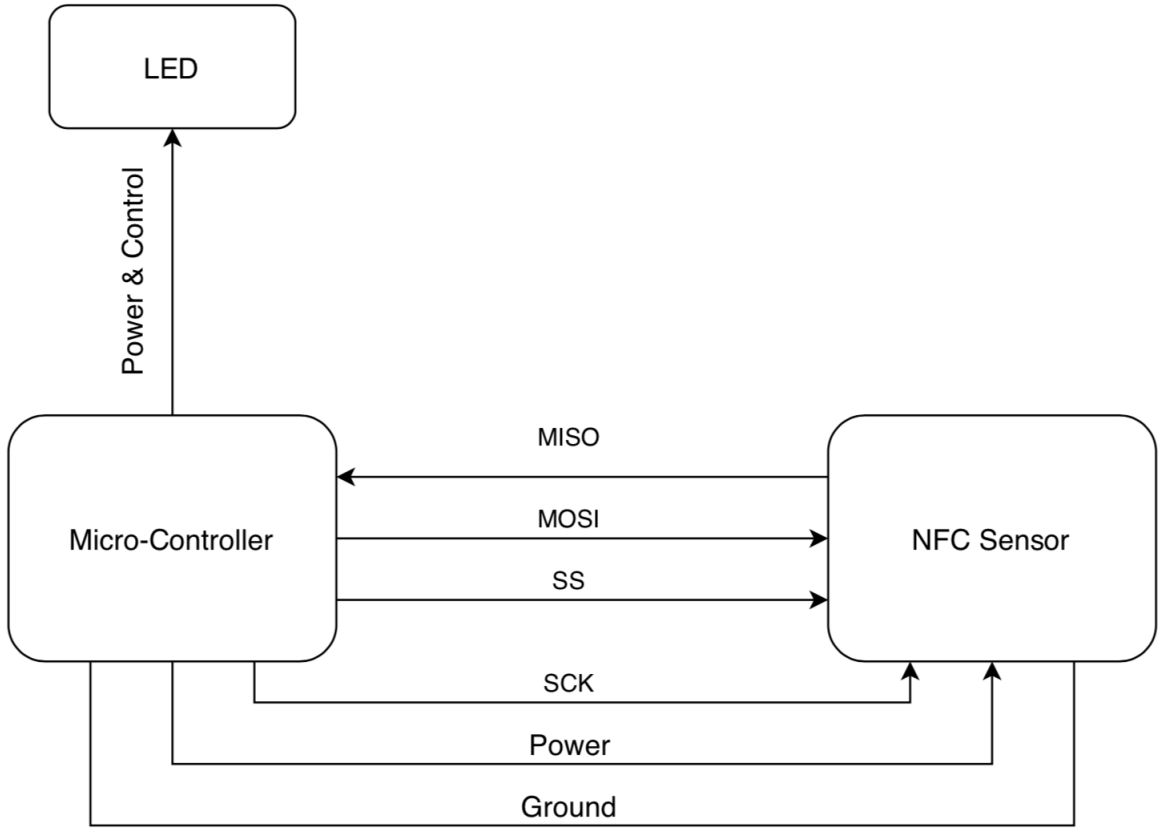


Figure 5.1: Door Lock Design

With the design of the DoorLock taken care of, the first step was to implement the connection between the NFC sensor and the micro-controller. This was achieved with the help of a few community based libraries, mainly the PN532, PN532 SPI and PN532 Interface [10]. To program the micro-controller the Arduino IDE defines two main functions, the first one is the setup and the second is the loop. All of the initialisation code goes into the setup function. This includes all of the constructer methods from the PN532 libraries, the initialisation of the server and access point, which are going to be discussed later on. In addition, the setup function also binds the two micro-controller cores to their respective tasks, which are going to be executed later on in the loop functions. Because this is a dual core board, there are two loop functions and each of them is bound to a separate core. One of those cores is used for the PN532 listener function and it is executed every few seconds. When the NFC sensor is triggered the application protocol data unit(APDU) [16] is checked. The APDU defines a communication structure between a smart card reader and a smart card in this case the smartphone.

However to be able to completely authenticate the user, the Door Lock needs to be connected to the Hub. So far the PN532 libraries have only acted as a bridge between the smartphone and the NFC scanner, this is only part of the authentication process. The next step is to crosscheck the information received by the NFC reader with the one that is stored on the Hub. This was introduced to prevent valuable information from being stored on the Door Lock itself. However in order to accomplish that we have to implement the mechanism for pairing Edge Devices to the Hub. But before that can be achieved, there should be a defined way of transmitting configurational data from the smartphone to the Door Lock. This was achieved by implementing a server and switching the ESP32 onboard WiFi card into access point mode. The idea behind that is to connect the smartphone to the Door Lock via WiFi and transmit the necessary configuration data with a HTTP Post request.

```
WiFi.softAP(ssidOfSensor, passOfSensor); //can remove pass argument to set up an
    open access point
  //gets the IP adress of the access point
  IPAddress myIp = WiFi.softAPIP();
  //prints the IP to the serial monitor
  Serial.println(myIp);
  //starts the server
  server.on("/", handleRoot);
  server.begin();
```

This process can also be implemented by substituting the WiFi connectivity with Bluetooth which is up to the person developing the edge device. The WiFi was chosen, because it provides backward compatibility with older micro-controllers. The transmitted data will be encoded in a JSON format. Then it is just a matter of parsing the JSON object and extracting the necessary setup information.

```
// Check if the HTTP header has present arguments
  if (server.args() > 0) {
    // unpacs the server arguments
    for ( uint8_t i = 0; i < server.args(); i++ ) {
      if (server.argName(i) == "PostData") {
        Serial.println(" Input received was: ");
        Serial.println(server.arg(i));
        StaticJsonBuffer<300> JSONBuffer;
        JsonObject& parsed = JSONBuffer.parseObject(server.arg(i));
        NetworkSSID = parsed["NetworkSSID"];
        NetworkPassword = parsed["NetworkPassword"];
        mqttServer = (const char *)parsed["mqttServer"];
        server.stop();
        WiFi.softAPdisconnect();
      }
    }
  }
```

This setup information is then used to connect to the Hub. Connection is established in the following fashion, first with the Hub and then with the MQTT Broker. It is vital that the setup is followed in this fashion, or the device would not be able to finish the setup process.

```
 WiFi.mode(WIFI_STA);
  WiFi.begin(NetworkSSID, NetworkPassword);

  while (WiFi.status() != WL_CONNECTED) {
    delay(500);
    Serial.print ("Connecting... to ");
    Serial.println(NetworkSSID);
  }

  Serial.println("Connected to ");
```

```
11  Serial.println(NetworkSSID);
12
13  client.setServer(mqttServer.c_str(), mqttPort);
14  client.setCallback(callback);
15
16  while (!client.connected()) {
17    Serial.println("Connecting to MQTT...");
18
19    if (client.connect("ESP32Client", mqttUser.c_str(), mqttPass.c_str())) {
20      Serial.println("connected");
21    } else {
22
23      Serial.print("failed");
24      Serial.print(client.state());
25      delay(2000);
26    }
27  }
28
```

Before connecting to the MQTT Broker, the Door Lock needs to implement the communicational channels introduced by the software design framework. Following the system design makes this process easy and straight forward. The first step is to determine what type of device the Door Lock is. Analysing the design of the Door Lock, we have a sensor which is represented by the NFC anthena. There is also an actuator which is simulated by a LED. Taking these into consideration defines the Door Lock as a sensor&actuator. Going back to the system design, each sensor&actuator must implement three communication channels, which are the data, control and configuration channels. After choosing the channels we also need to select the message structure for the communication. The system design also provides this by showing the messaging structure that each channel must use. With the communication channels taken care of, the next step is to implement the mechanism for pairing the Door Lock with the MQTT Broker. Initially the device will connect to the Broker with a default Identification number, which will be changed later on. After the initial connection to the MQTT Broker the device will send its configuration details on to the configuration channel and will wait for the Broker to assign it with a new Identification number. After the new Identification number is assigned to the edge device, it will disconnect from the Broker, thus destroying the default Identification number that was provided by the initial setup process. If the disconnection is successful the edge device will now attempt to reconnect with the new Identification number that was generated by the MQTT Broker. If a problem occurs during the reconnection process the user will be notified and the recommended procedure in this case would be to restart the edge device and restart the setup process. This can be done by pressing the reset button on the device, which will destroy all previous data and start a fresh setup process. Upon a successful setup process the device will start its normal operation and the user will be notified of that action. In addition, the device will disable the AP mode and will enter into its STA (station) mode. The initial implementation featured a single core micro-controller which was causing issues, regarding the task scheduling. Even though concurrency can still be accomplished with software interrupts and task scheduling, the decision was to move to a micro-controller with two cores. This made the DoorLock more robust and responsive and completely eliminated the need for software scheduling.

5.4 Hub

The Hub that is used in this project is a Raspberry Pi 3, but any device running Linux will be sufficient as long as that device has a similar or better hardware. As mentioned in the system

design the Hub is responsible for keeping track of all edge devices connected to it, in addition it must be capable of handling the data throughput. The system design also split the Central Hub into segments and the implementation will follow this. Starting with the implementation of the MQTT Broker which is the main segment that is responsible for controlling the other segments. The implementation of the MQTT Broker is developed using the Python 3.6 environment with the help of the paho.mqtt library [14] and a mosquito server [9]. The paho library defines two functions which are the on_connect and the on_message. On_connect function is the callback when the server receives a CONNACK response. The CONNACK response marks a successful connection to the Hub. The On_message callback function is triggered when a PUBLISH message is received from the client. In addition, the Hub is also responsible for storing all of the collected data from the edge devices as well as the user's authentication information. That is why the database logic was merged with the MQTT Broker, and every time the Broker receives information the database controller will record it. The Hub is also responsible for the generation of unique identification number for each newly connected device. This was achieved by taking the name of the sensor and appending a number at the end of it. This number is created by taking the count of all the edge devices that are currently connected and adding one to it. For example if there are 2 edge devices currently active the generation of the Identification number for the new device will be the "device name" + 2(number of the currently connected devices) + 1. The listing below shows how this was achieved.

```
def generate_ID(sensorName, numberOfSensors):

    if(not isinstance(sensorName, str)):
        raise TypeError("sensorName must be of type str")
    elif(not isinstance(numberOfSensors, int)):
        raise TypeError("numberOfSensors must be of type int")

    senosorID = sensorName + str(numberOfSensors + 1)

    return senosorID
```

Another distinct feature is the Alert Trigger functionality, which allows the user to set a time during which they wont be home. This is set trough the smartphone application and sent to the Hub via the communication channels defined by the framework. More on the smartphone implementation will be discussed in the following section. The only thing that we need to know right now is that the smartphone application will use the control channel to send the set Alert Trigger timeframe. Thus the first step of the implementation was to save the timeframe to a configuration file. The configuration file houses two entries, one entry is used just for the time in hours and minutes while the other entry is used for saving the weekdays. Only the days in which the trigger will be active are saved.

```
def trigger_processing():
    #pull data from DB
    time_trigger = datetime.strptime(datetime.now().strftime('%Y-%m-%d %H:%M:%S'
    ),'%Y-%m-%d %H:%M:%S')
    dt1 = datetime.strptime((config.get('TimeTrigger', 'timeStart') + ":00"), '%
    H:%M:%S').time()
    dt2 = datetime.strptime((config.get('TimeTrigger', 'timeStop') + ":00"), '%H
    :%M:%S').time()

    today = datetime.today().weekday()
    timeToday = datetime.strptime(datetime.now().time().strftime("%H:%M:%S"), '%
    H:%M:%S').time()

    if str(today) in config.get('weekdayTrigger', 'weekdayList'):
        if (dt1 < timeToday and timeToday < dt2):
            return True
```

```
        else:
            return False
```

Once the trigger information is saved the next step was to provide a function for checking if the user authenticates in the correct time or not.The precedence in which the authentication is carried out is the following. Firstly check if the user is authenticated if they are then check whether they are authenticated in or outside the set hours. Depending on that the system will respond differently and will send a different response to the smartphone application. The rest is left to the smartphone application to notify the user. Another function implemented is the camera control, which is straight forward. The user is checked for access rights to use the camera stream, if access to the camera stream is granted the system checks whether the smartphone application sent a message with flag "1" or "0". If it is "1" the system starts the camera if it is "0" the camera is stopped.

```
cursor.execute(check_user_access)
            for i,j in cursor:
                if(userID == i and message == j):
                    cursor.execute(insert_data_data_collection, (sensorID,
    userID, message, datetime.now().strftime('%Y-%m-%d %H:%M:%S')))
                    if(trigger_processing()):
                        client.publish(androidDataChannel, sensorID + ":anomaly"
    )
                    else:
                        client.publish(androidDataChannel, sensorID + ":normal")
                    print("granted")
                    client.publish(controlChannel, "1")
                else:
                    #unothorised activity, send notification
                    client.publish(androidDataChannel, sensorID + ":unauthorised
    ")
                    client.publish(controlChannel, "0")
                    print("denied")
            cnx.commit()
```

The next important functionality is the control of the media server, which is a matter of starting the uv4l server with the help of the python subprocess library. This library allows the python server to execute terminal commands. Stopping the server is handled the same was as starting it.

```
if(temp[0] == checkUserID and temp[1] == "cameraControl" and temp[2] == "1"):
            fr.media_Stream("uv4l --auto-video_nr --driver raspicam --encoding
    mjpeg", True)
            print("Stream started")
        elif(temp[0] == checkUserID and temp[1] == "cameraControl" and temp[2]
    == "0"):
            fr.media_Stream("pkill uv4l", False)
            print("Stream stopped")
```

These three functionalities are the main features if the Hub design. There other functions are defined by the paho library and are used to handle the connection with the edge devices as well as receiving messages from them. To initialise the MQTT Broker, paho defines the following functions:

This concludes the implementation of the Hub, the next section will focus on the design and development of the smartphone application. As mentioned previously this application was developed in the Android system environment.

5.5 Android Client

The system design explored the benefits of a smartphone application and how it can improve the IoT device by providing the ability to access it remotely. This section is devoted to the development of a smartphone application, namely in the android environment. The android environment was preferred as it is a free platform and was the only one available for this project. The development of the application will be mainly focused on the design and user interface the code will be provided in the appendix and in the whole implementation will be included in a separate zip file.

5.5.1 Design and User Interface

The main goal with the design of the user interface was to make it as simple as possible and to constantly keep the user aware of the state in which the application is, by displaying useful help messages. The design of the application was developed following the android design guidelines for simplicity and optimal functionality. Launching the application, the user is greeted by the application's home screen, which is encapsulated between the navigation bar at the bottom of the application screen and the action bar, located at the top. Following android's design guidelines, the title of the current state in which the user is located, is displayed on the left side of the action bar. To the right side of the action bar is the icon for the preferences settings fragment, which can be accessed from any of the three main states of the application. Switching between the three main states of the application can be done by taping on the icon of the state to which the user wants to switch. The icons that represent this functionality are located in the navigation bar located at the bottom of the application screen. The default screen in which the application greats the user is the home screen.

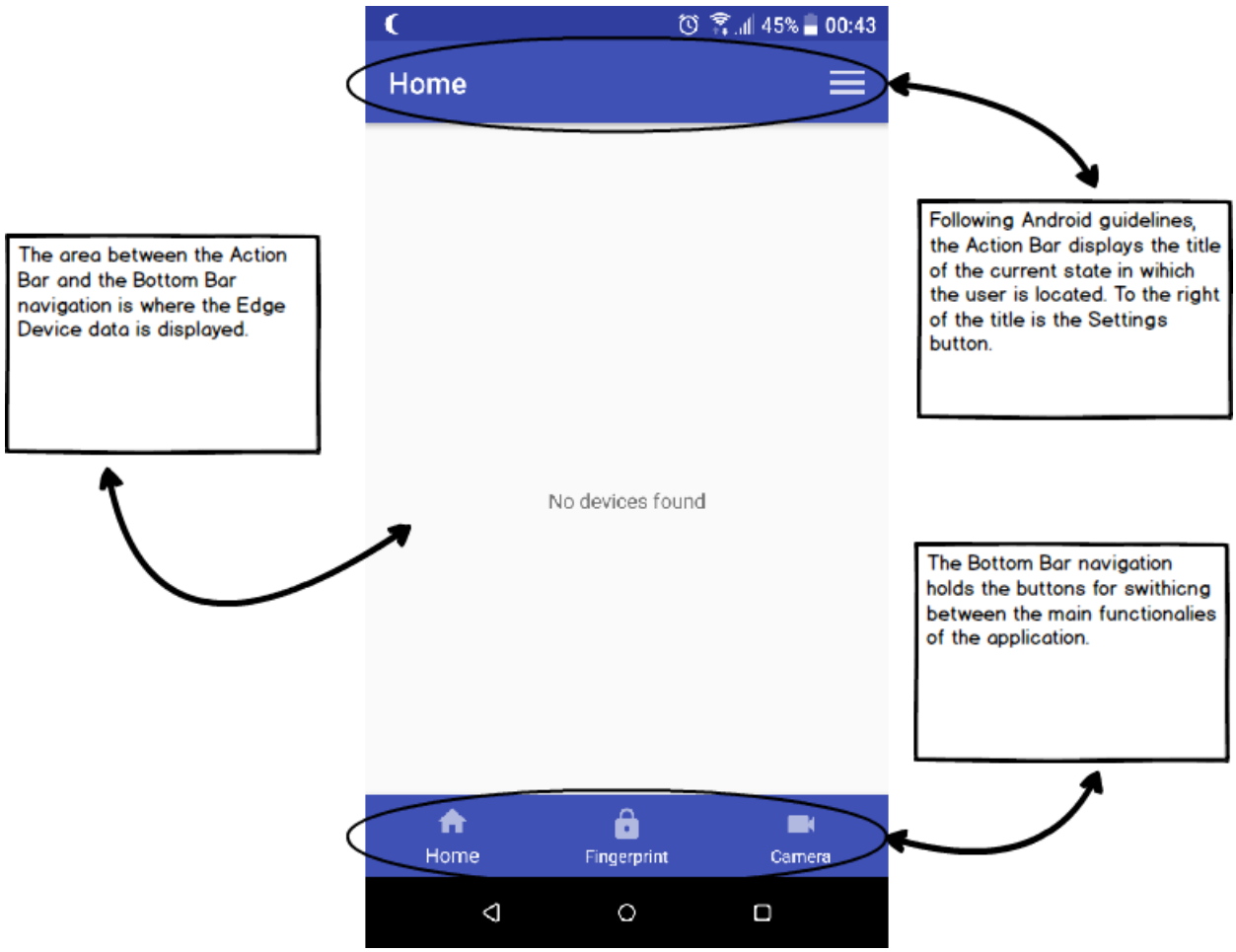


Figure 5.2: Home Screen

Located in the middle of the bottom bar navigation is the button for the fingerprint reader [18] [13]screen. Here the system also displays help messages, when the user misuses the fingerprint scanner. In addition it provides guidance for the door unlocking process. The messages are displayed on the bottom of the Fingerprint screen, which complies with Android guidelines for usability.

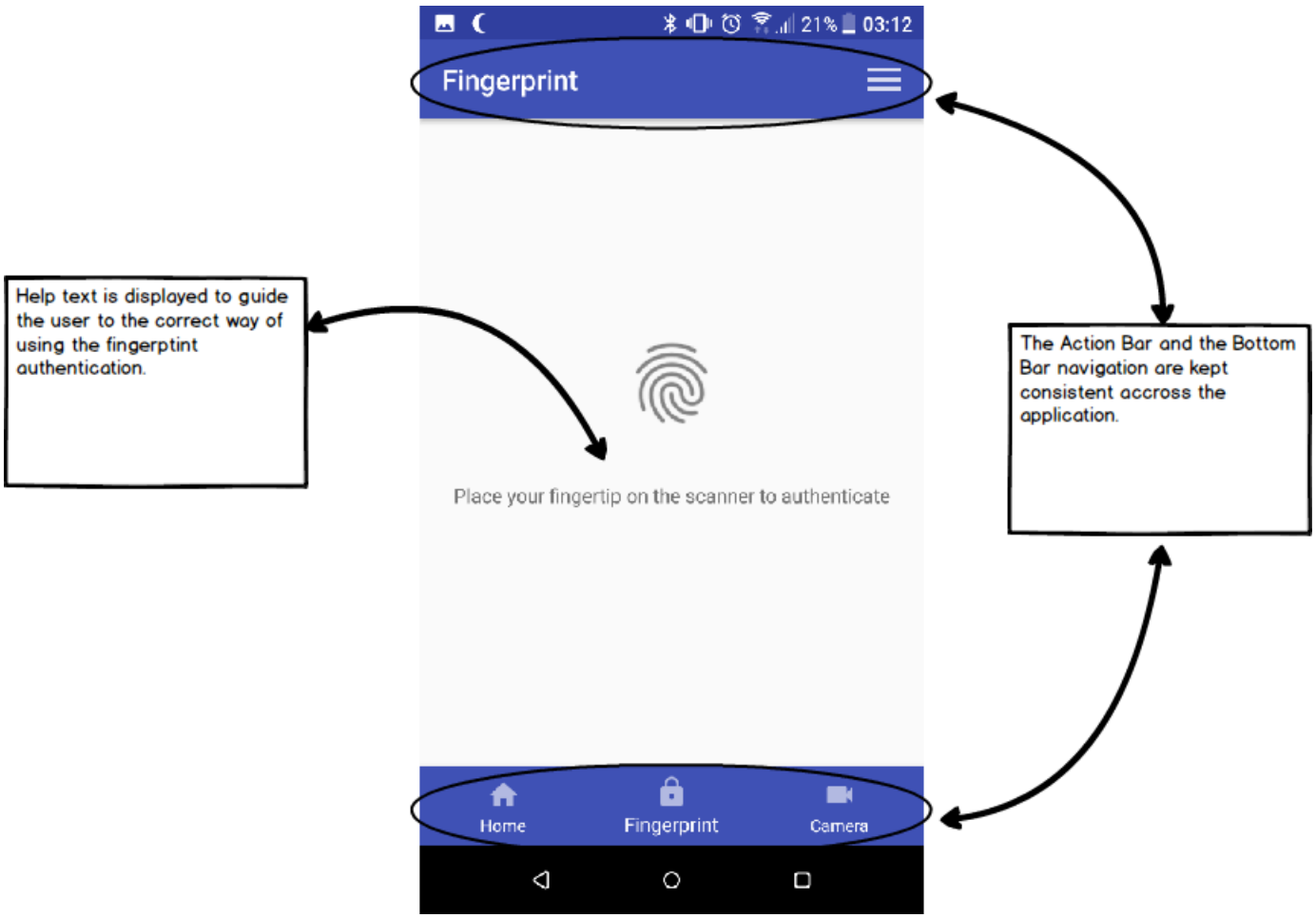


Figure 5.3: Fingerprint Screen

Located in the middle of the bottom bar navigation is the button for the fingerprint reader screen. Here the system also displays help messages, when the user misuses the fingerprint scanner. In addition it provides guidance for the door unlocking process. The messages are displayed on the bottom of the Fingerprint screen, which complies with Android guidelines for usability.

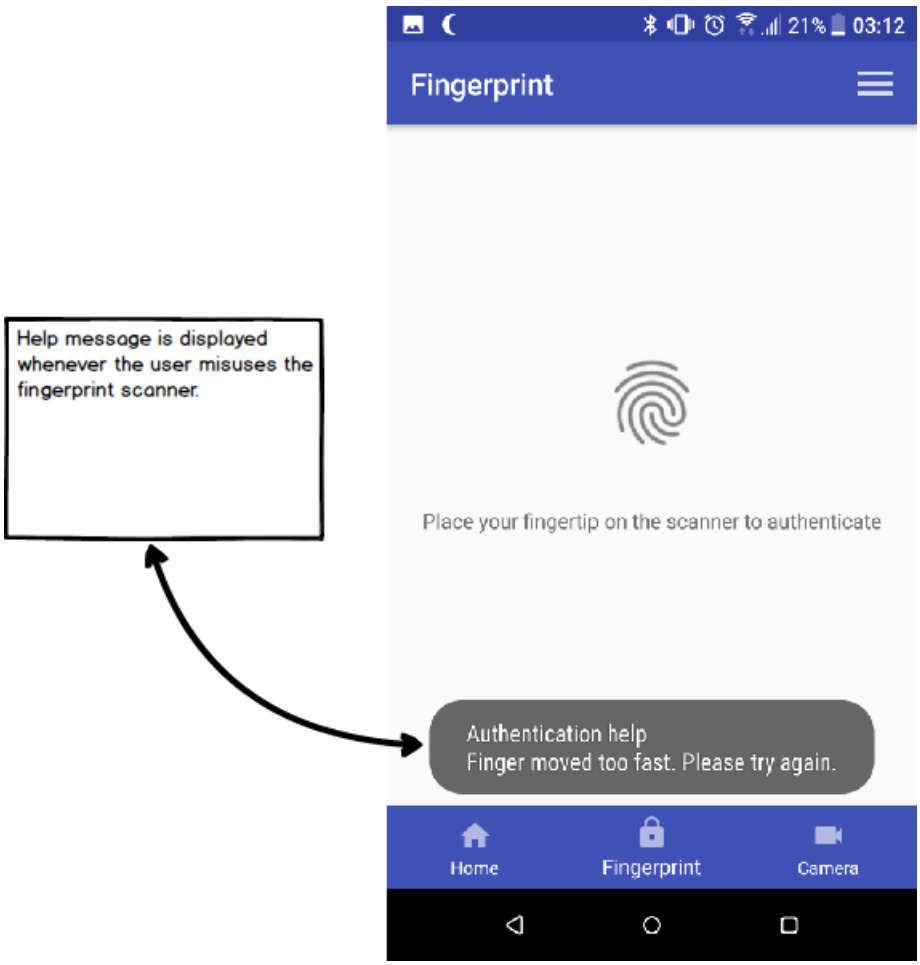


Figure 5.4: Fingerprint Screen Help Message

This message is displayed whenever the user does not completely cover the fingerprint scanner or when they removed their fingers from the scanner during the authentication process.

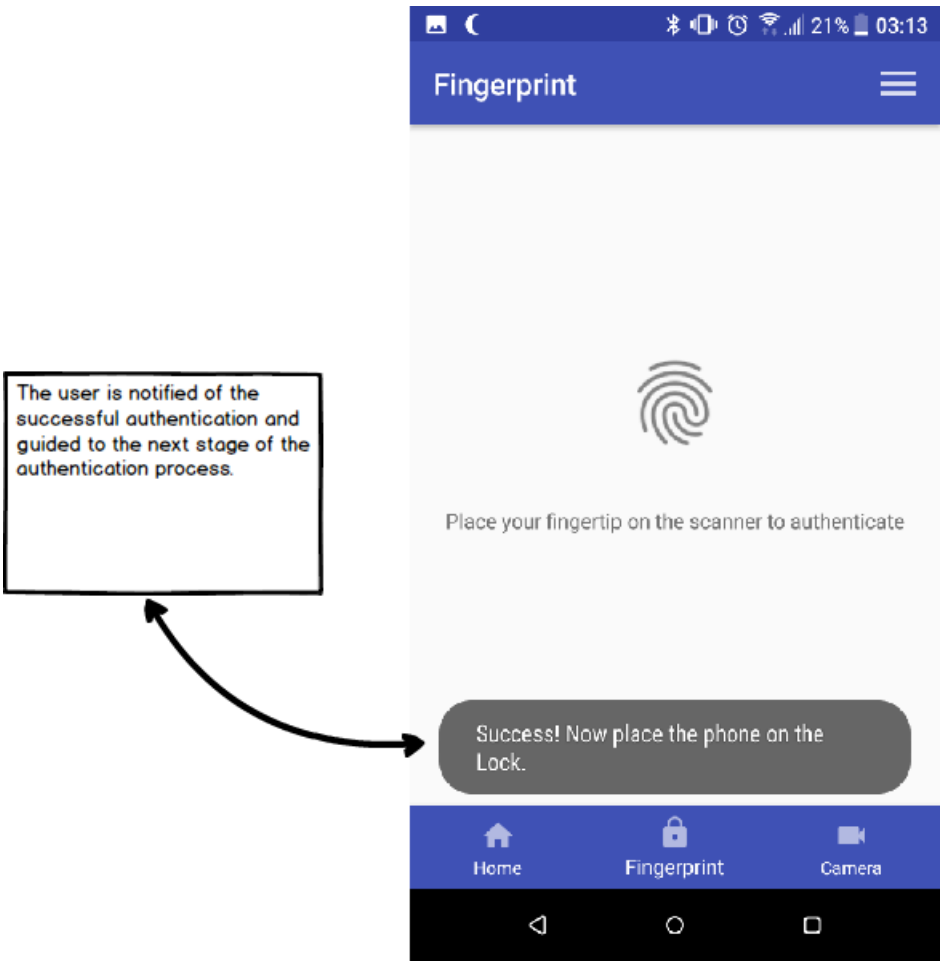


Figure 5.5: Fingerprint Screen Successful Authentication

Next to the fingerprint screen is the camera stream viewer. The control buttons for the camera stream are located just above the bottom bar navigation. Pressing the Play button will start the camera stream, to stop the camera stream the Stop button must be pressed.

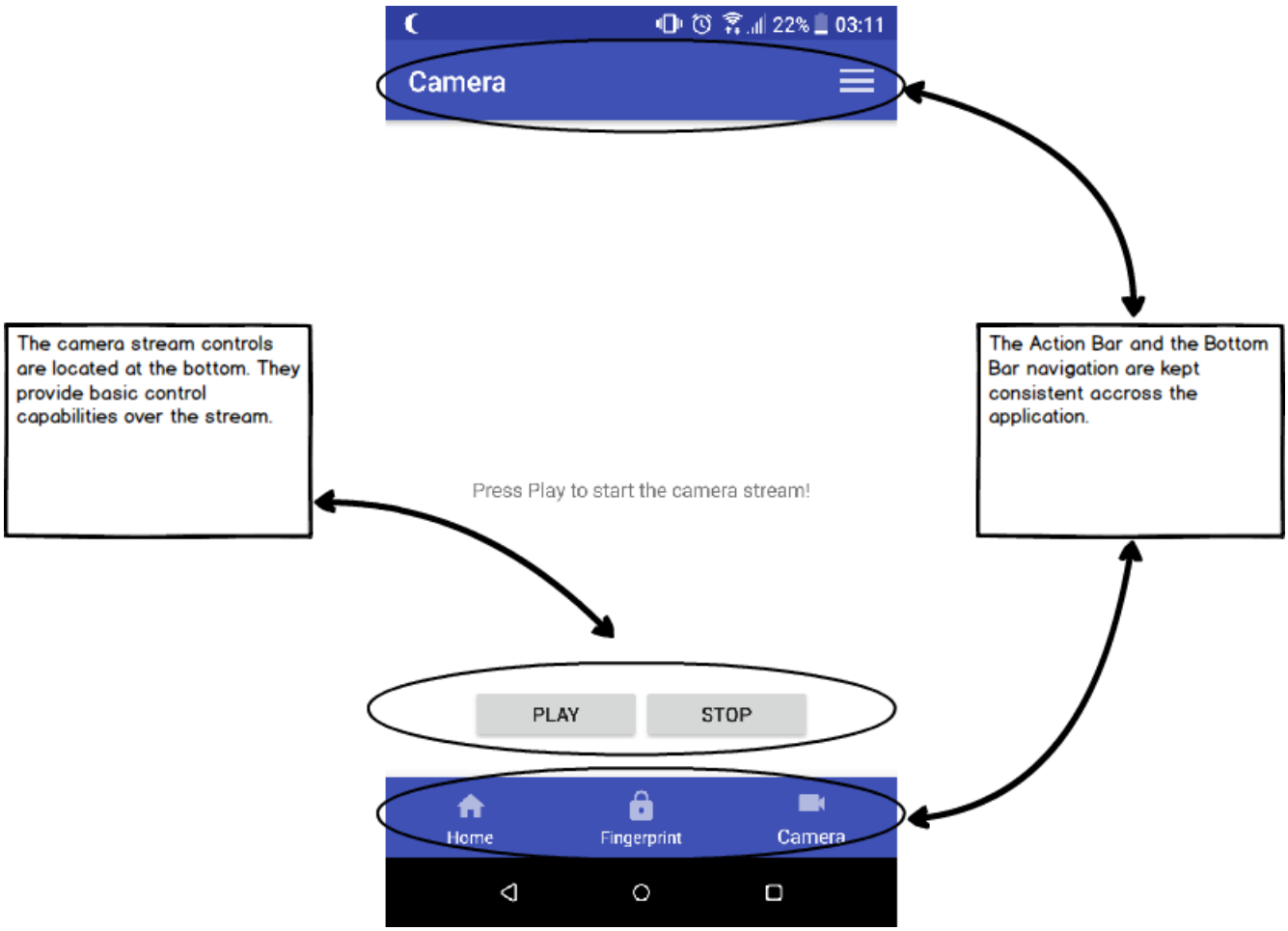


Figure 5.6: Camera Viwer Screen

Moving the user's attention to the right side of the action bar is the Preference button, which leads the user to the Preference window. The Preference window allows the user to change the settings of the system. The structure that I decided to use for the preference tab is to group all of the functionalities in a separate sections. The sections are arranged from most frequently used by the user to the least. The categories are as follows Alert Trigger and Edge Device Control.

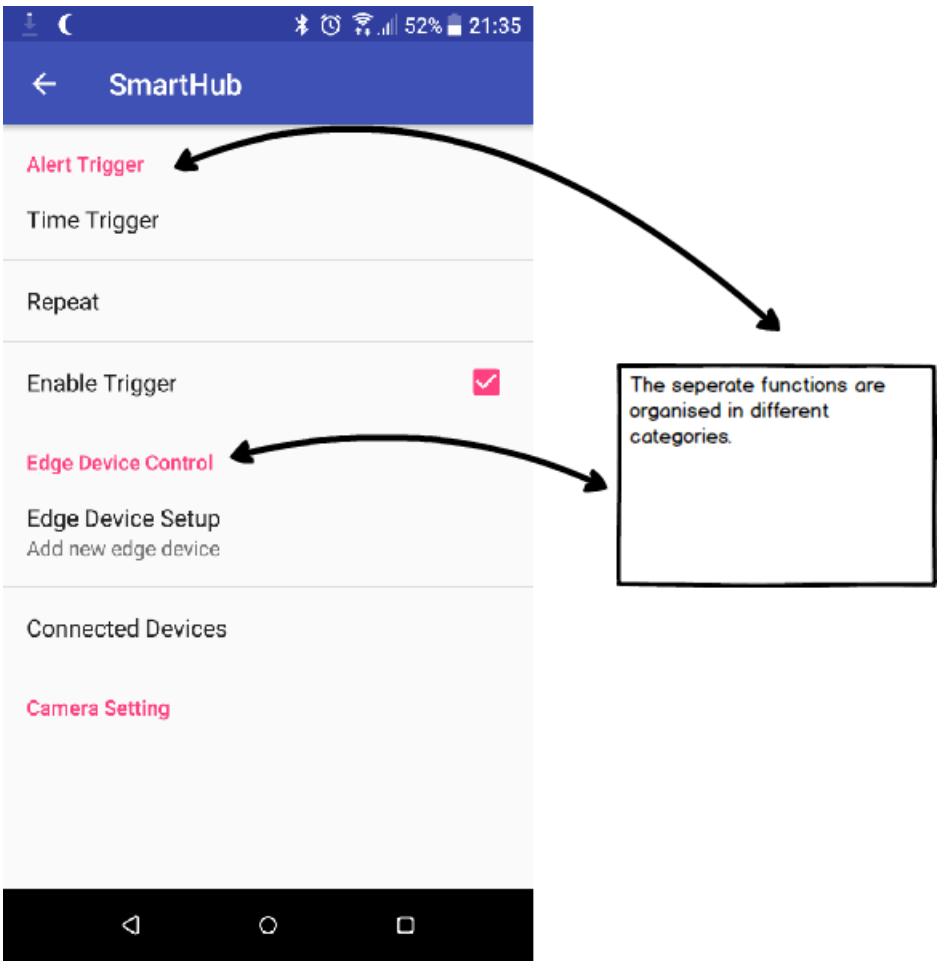


Figure 5.7: Preference Menu

Focusing on the Alert Trigger, which allows the user to set a preferred time when they are not going to be home. Time selection was implemented using the time picker clock, which gives the user a separate dialog for choosing the time. In addition, this implementation separated the time picker in two different sections. The first one is the time picker dial and the second one is the weekday checklist, this separation gives the system more functionality.

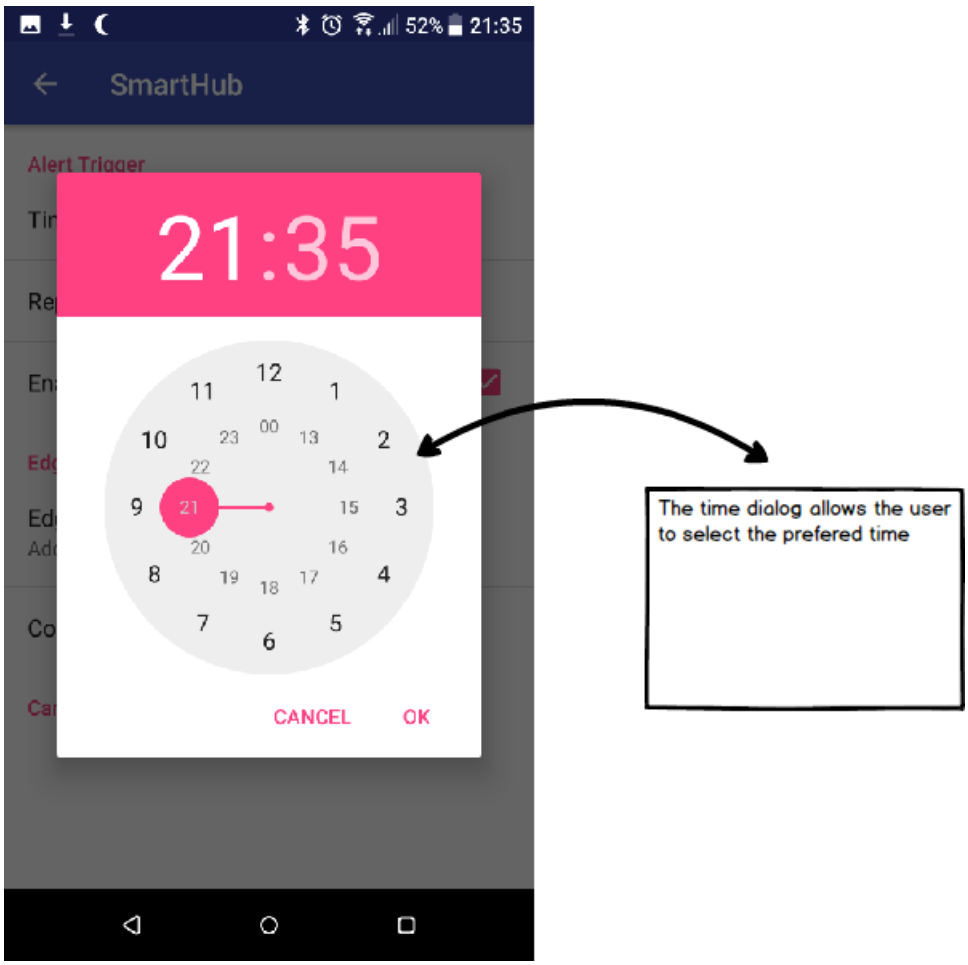


Figure 5.8: Time Picker Dialog

From this menu the user can chose in which weekdays they want the alert trigger to be active. The Implementation uses a multi-selection list.

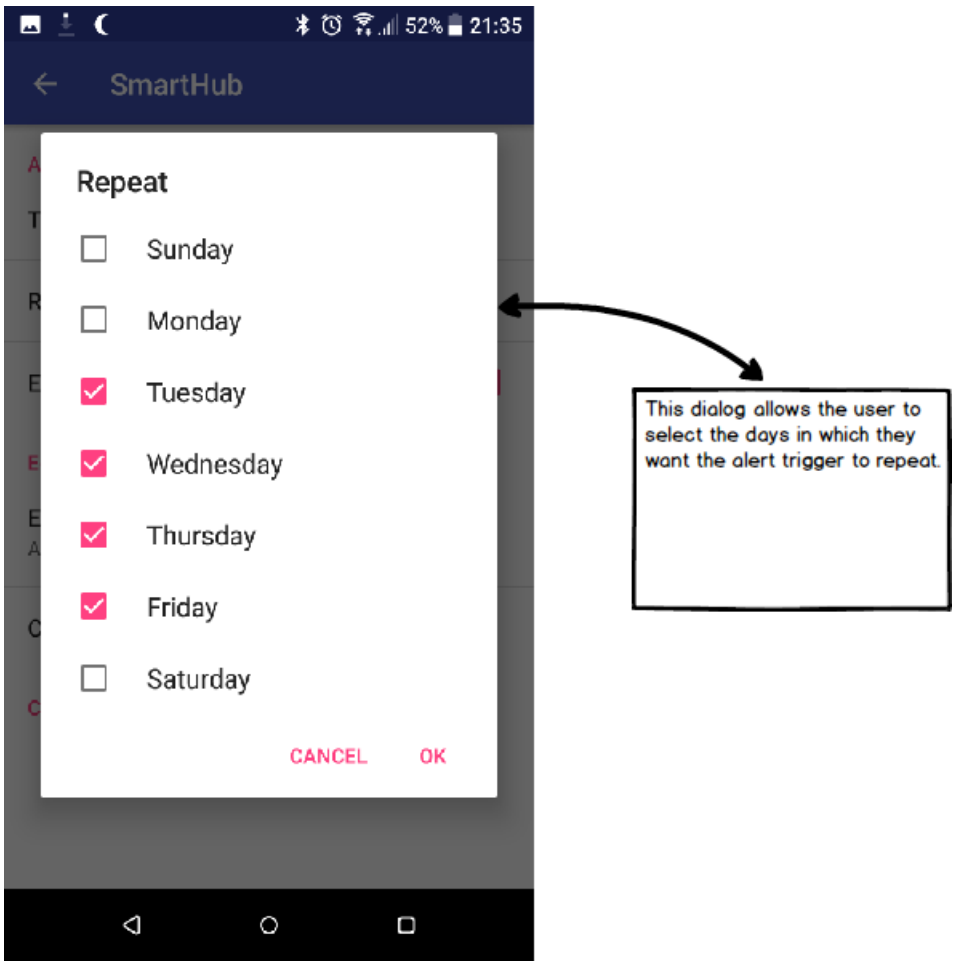


Figure 5.9: Weekday Picker Dialog

The second important functionality of the Preference Menu is the add new edge device field, which when selected opens a new dialog window in which the user can enter the name for the new device. When the user has entered the name hitting OK will send all of the configuration data to the Edge Device. However before that the user must be connected to the Edge Device WiFi server.

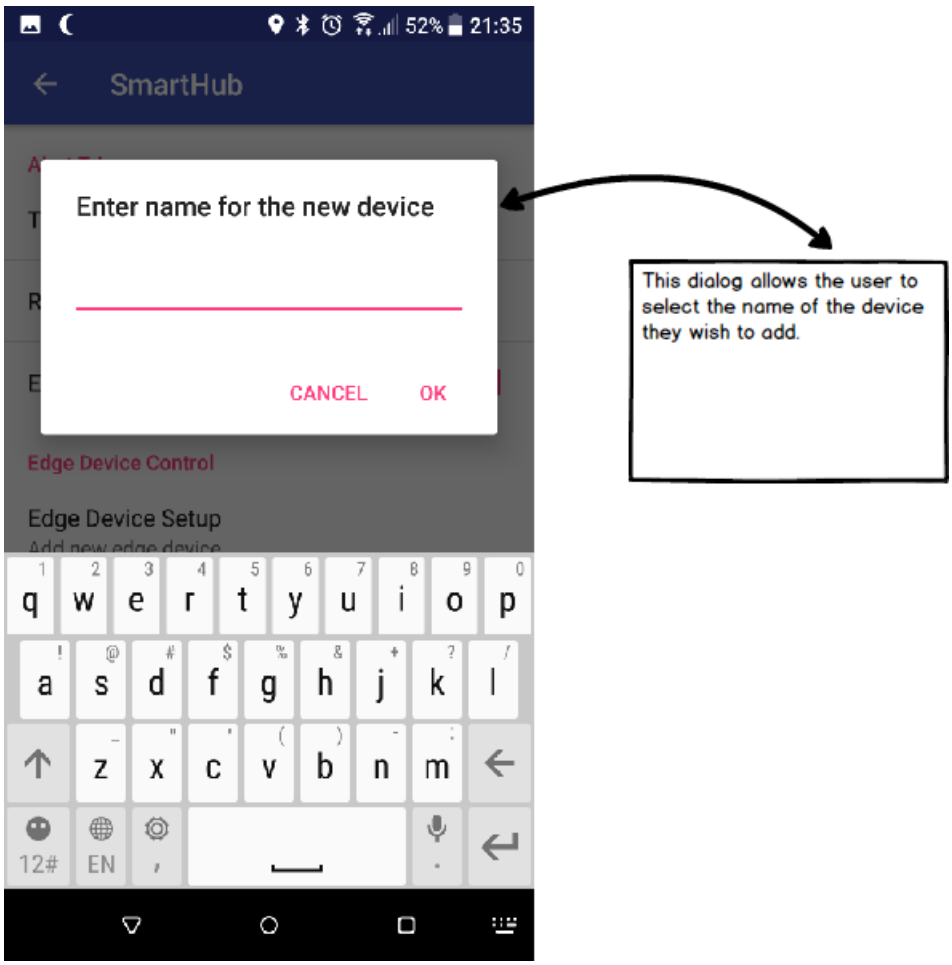


Figure 5.10: Add New Device Dialog

This section explained the Implementation decisions taken to comply with the system design. The next section will focus on the evaluation of the product that was developed in this project.

6. Product Evaluation

An important part of the development of any new device is the evaluation of the finished project. Evaluation of the functionalities of the any product can show how many of the requirements are satisfied. It can also show potential bugs and issues that can be fixed either before launch or afterwards with an update. This section focuses on the evaluation of the implementation of the software that was developed in this project. The evaluation was conducted and recorder in the form of test cases. The test cases are separated in three categories, each of which will focus on a different section of the product. The focus of this chapter is to establish the dependability of each component and how well they cooperate. The test cases will include the name of the module that is being tested as well as the priority of this module. In addition, a brief description of the task that is executed by that module will be included. The test cases will also be accompanied by a brief explanation, which will outline the reason for conducting the particular test.

6.1 DoorLock Evaluation

Starting with the evaluation of the DoorLock implementation. Analysed more abstractly the DoorLock should react when the user activates the sensor. However there are some scenarios where the unauthenticated personal may try to bypass the DoorLock, in which cases the system should respond by alerting the user. In addition, the DoorLoc should respond differently according to the alert trigger times set by the user. Thus, there are a few use cases that need to be considered and the DoorLock should respond accordingly. The proper behaviour of the DoorLock should provide the user with the appropriate alert notification, so that the user can react in time. The cases that this evaluation takes into consideration are the following. Starting with the user authentication, which can result in a few different system behaviours. These are the authentication with normal behaviour, authentication with abnormal behaviour and unauthorised access. The system authenticates the user with a normal behaviour when the user accesses the door outside the set Alert Trigger times. Alert notification in this case is still send to the user notifying that the door has been opened. The other case is when the system authenticates the user with abnormal behaviour. Here the correct user is still authenticated, however the system will notify them that the door has been accessed during the set Alert Trigger time. The last scenario is when an unauthorised personnel tries to open the door without having the access rights to do so. In this case the system responds by sending an alert notification to the user, letting them know that someone has accessed the DoorLock. The table below shows the location of all the test cases that were created for this section. The test cases are located in the appendix and they provide more detailed and structured information about the evaluation that was carried out.

	Test Case Name	Test Case ID	Test Case Page Referance
Test Case 1	Door Lock authenticate with normal behaviour	DL-001	[49]
Test Case 2	Door Lock authenticate with abnormal behaviour	DL-002	[50]
Test Case 3	Door Lock unauthorised access	DL-003	[51]

6.2 Android Client Evaluation

This section includes the evaluation of the complete set of functionalities that were implemented in the Android Client. The Android Client consists of a few functionalities and to make the evaluation more accurate they are split into segments. These include the Fingerprint Authentication, Camera Stream Viewer and the Preference tab. Starting the evaluation with the Fingerprint authentication, where the user can be authenticated either successfully or unsuccessfully. The successful authentication is pretty straight forward, the user places their finger tip on the fingerprint scanner and if the fingerprint matches, the authentication is successful. However there are a few different states with which the system reacts when it comes to the unsuccessful authentication. An unsuccessful attempt would be if the user uses a fingerprint that is not recognised by the system. Another possibility here is if the user does not cover scanner completely as well as leave enough time for the scanner to read their fingerprint. Every fingerprint scanner is different and each requires a different time to read the fingerprint correctly and the system takes that into consideration. If the wrong fingerprint is used the system will display a failed message. The next functionality is the Camera Stream Viewer, here two evaluation tests were conducted and they were the start and stop stream cases. To start the stream the user needs to press the Play button, the system has to react by starting the stream. When the user presses the stop button the system has to react by stopping the stream. Next part of the system is the Preference tab, which has more moving parts than the other states of the system. As mentioned in the implementation section the Preference tab is segregated into three different segments, the alert trigger, add new device and the stream quality. Starting with the alert trigger, the user has to press the Time Trigger field and the system has to respond by displaying the time preference wheel. After choosing the time the user can adjust the Repeat period. The system has to react to the user pressing the Repeat field, by displaying a multi select list dialog. When the user is finished choosing the repeat period, they need to press the ok button which sets the repeat periods. Pressing the cancel button will disband any previous changes made. This section also provides more technical information in a the form of test cases, which are going to be placed in the appendix of the report. The table below provides the name of the test cases and a link to the place of the actual test case.

	Test Case Name	Test Case ID	Test Case Page Referance
Test Case 1	Set Alert Trigger time	AC-001	[52]
Test Case 2	Addition of new edge device	AC-002	[53]
Test Case 3	Fingerprint Authentication with success	AC-003	[54]
Test Case 4	Fingerprint Authentication with failure	AC-004	[55]
Test Case 5	Play the camera stream	AC-005	[56]
Test Case 6	Verify if the camera is not available	AC-006	[57]
Test Case 7	Stop the Camera Steam	AC-007	[58]

This concludes the product evaluation section, which focused on evaluating the different sections of the implementation. The section was delivered in the the form of test cases and explanation was provided for each of the possible scenarios with which the system reacts in the different situations. The next chapter will focus on the outcome of this project as well as the future improvements that could be added to the system.

7. Conclusion and Future Work

This project set out to explore the possibilities of developing a fully modular device at a smaller price. The journey this project took was to analyse the current market and establish the popularity of these devices amongst users. The report also ventured into analysing the current competition and their solutions to the problems introduced by this project. In addition, the project developed and encapsulated the modular design principles into a framework that can be used and extended by others interested in this concept. The main part of this project is the software development chapter which introduced a way of making IoT devices modular. The Implementation chapter provided the proving grounds for this framework. Wile the software design chapter introduced the abstract model of the data stack with its data and control flow, the Implementation section provided a translation between this model and the real world. After the Implementation of the software design the next focus of the report was the product evaluation. While the Implementation section served as a proof of how easy to use the framework is, the evaluation was the proof of the success of this design model. However, there are still some improvements to be made to the design and implementation. This project laid the foundation of the modular design principles and if extended can become really useful not only in the consumer market. For the whole of this project the main target for this device was the consumer market, however further into the development process this project showed a hidden potential. Given the right hardware and proper Implementation this design model can be extended and upscaled to accommodate bigger networks. Regarding the Implementation of the software design that this project introduced there is still some room for improvements.Throughout this project the main spotlight was taken by the development and implementation of the modular system design. This left little time for pampering the android application experience, which should be further improved to bring more functionality and ease of use. A big hole was also left by the camera streaming functionality, which had more be desired from, regarding the quality and the decoding standards. The future improvement of the camera streaming will bring support for the h.264 standard, which is widely adopted by most other devices from the same caliber. Right now the system is limited to handling just mjpeg streams which provides worse quality and less frames per second than the h.264 standard. Another future improvements will be the introduction of machine learning algorithms to help with the automation of certain areas of the system. One area that will benefit from this is the alert trigger, which in the current state of the system depends on user input. However, with the introduction of machine learning this can be greatly improved and would automate the process and require less input from the user. Security improvements will also need to be carried out to keep the system up to date with the security standards and provide more resistance to attacks.

8. Self-Reflection

Coming to the end of this project I spent some time reflecting on the skills that I have acquired from this project. During the development of this project I have experimented with the development of low level programming fo the development of the micro-controllers. In addition, I have also had the opportunity to develop an application for the Android OS platform, which proved to be more challenging. Even though I spent 3 months working with those environments I still fell I need more time to completely understand them. This project also created the opportunity to experience working on my own big project and this allowed me to immerse myself into the Software Development process. This process taught me a lot of important skills, which I am going to use and improve in the future.

During the development of this project there were a lot of ups and downs and that sometimes made progress really slow and tedious. However, a big part of the learning process is to identify you weaknesses by learning from your failures. Every time I started to develop a new part of the system I would get so concentrated on it, that I would forged about the rest. This costed me some precious time that could be used in the development of other more important blocks of the system. And most of the time the part that I would get so focused on, was not even as important. Even though I was warned by my supervisor about this pitfall I still did not notice falling into it. Most of the time when that happened and the achieved end result did not satisfy my and I had to drop it and start from scratch. But the one thing that threw me off-guard was when I decided to take a completely different approach for the development of the system design. This required my to start completely from scratch and change a lot of the other blocks that I had already finished. I realise now that I started this project from the wrong angle, by developing the edge devices and the smartphone app first, instead of focusing my attention on the system design.

Furthermore the wrong approach to the development of this project, tilted the balance of my work plan. This setback meant that I had lost nearly two months trying to develop something that in the end was completely useless. To my relief I managed to create and document the new system design in a few days and I still had some time left for the Implementation, however little it was. Despite that I still feel that I could not achieve the final version of the device, that I wanted to. However, this experience taught me a lot of lessons and it also gave me the opportunity to acquire new skills. For the future to come I will try to improve these skills and will continue to develop and expand my knowledge in relation to Software Design principles. In general I had fun working on this project and I am happy I had the opportunity to discover my strengths and weaknesses.

9. Appendix

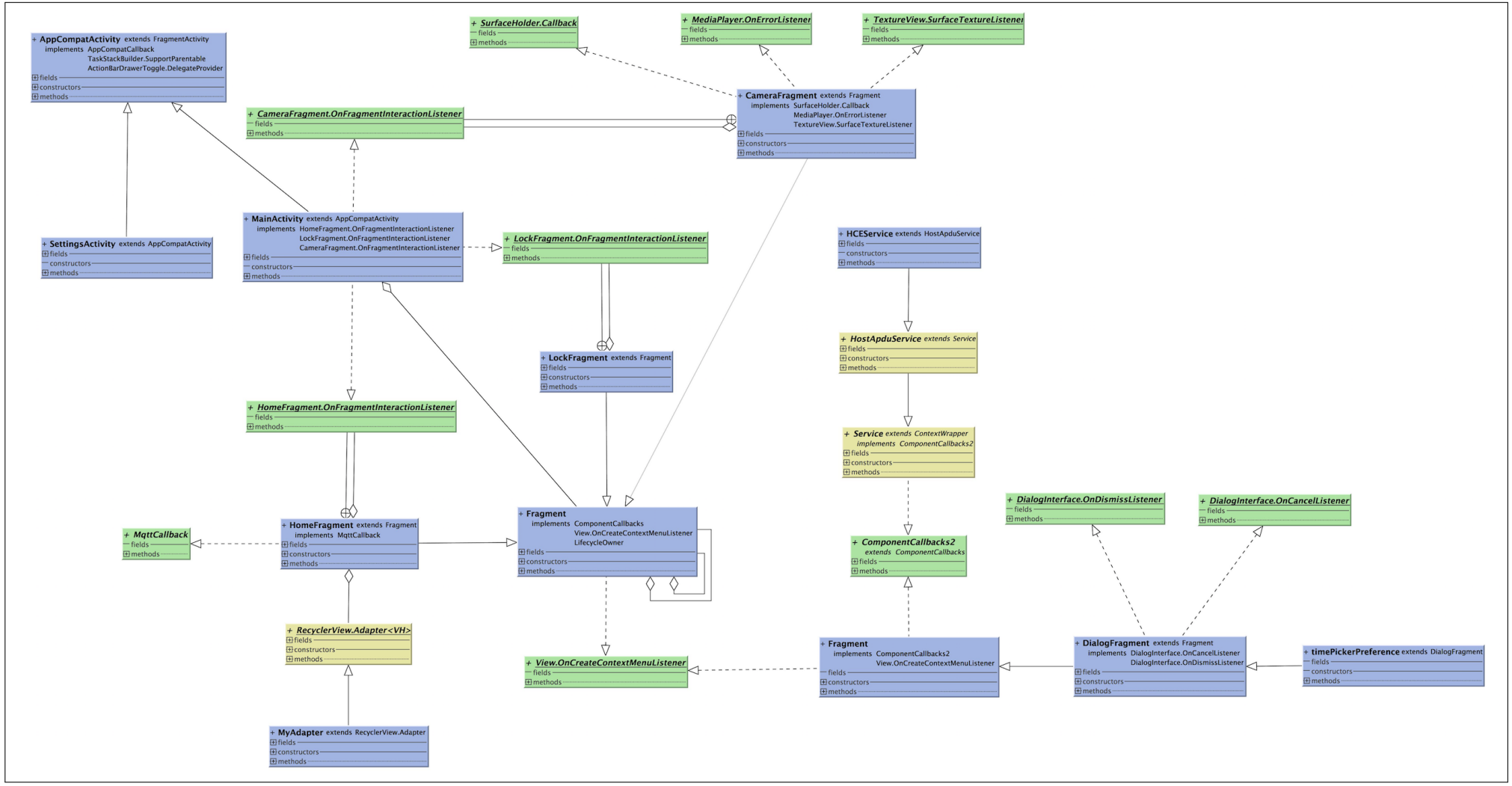


Figure 9.1: Android Client Design

Table 9.1: Test Case 1

Project Name: IoT Device for Home Monitoring	
Test Case	
Test Case ID: DL-001	**Test Designed by:** Georgi Pramatarov
Test Priority (Low/Medium/High): High	**Test Designed date:**
Module Name: DoorLock authentication	**Test Executed by:** Georgi Pramatarov
Test Title: User authenticated with normal behaviour	**Test Execution date:**
Description: Verify if the user is authenticated outside the designated hours	
Pre-conditions: User is authenticated in the Android Client application	
Test Data	Username:User1
	User Authentication code:48656c6c6f9000

Step	**Step Details**	**Expected Results**	**Actual Results**	**Status (Pass/Fail)**	**Notes**
1	User places the smartphone on the DoorLock NFC Sensor	User is authenticated	As expected	Pass	
2	System alerts the users that the door was unlocked	System sends a normal behaviour alert	as expected	Pass	
Post-conditions: User is validated with the database and the door is unlocked.					

Table 9.2: Test Case 2

Project Name: IoT Device for Home Monitoring	
Test Case	
Test Case ID: DL-002	**Test Designed by:** Georgi Pramatarov
Test Priority (Low/Medium/High): High	**Test Designed date:**
Module Name: DoorLock authentication	**Test Executed by:** Georgi Pramatarov
Test Title: User authenticated with abnormal behaviour	**Test Execution date:**
Description: Verify if the user is authenticated inside the designated hours	
Pre-conditions: User is authenticated in the Android Client application	
Test Data	Username:User1
	User Authentication code:48656c6c6f9000

Step	**Step Details**	**Expected Results**	**Actual Results**	**Status (Pass/Fail)**	**Notes**
1	User places the smartphone on the DoorLock NFC Sensor	User is authenticated	As expected	Pass	
2	System alerts the users that the door was unlocked	System sends an abnormal behaviour alert	as expected	Pass	

Post-conditions: User is validated with the database, the door is unlocked and the system send an abnormal behaviour alert.

Table 9.3: Test Case 3

Project Name: IoT Device for Home Monitoring	
Test Case	
Test Case ID: DL-003	**Test Designed by:** Georgi Pramatarov
Test Priority (Low/Medium/High): High	**Test Designed date:**
Module Name: DoorLock authentication	**Test Executed by:** Georgi Pramatarov
Test Title: Unauthenticated user	**Test Execution date:**
Description: Verify if the user is an authenticated, regardless the designated hours	
Pre-conditions: User does not have access rights for the Door Lock	
Test Data	Username:User1
	User Authentication code:48656c6c6f9000

Step	**Step Details**	**Expected Results**	**Actual Results**	**Status (Pass/Fail)**	**Notes**
1	User places the smartphone on the DoorLock NFC Sensor	User is not authenticated	As expected	Pass	
2	System alerts the authenticated users that the door was unlocked	System sends an unauthorised access alert	as expected	Pass	

Post-conditions: The user is not authenticated and the system sends an unauthorised access alert

Table 9.4: Test Case 4

Project Name: IoT Device for Home Monitoring					
Test Case					
Test Case ID: AC-001			**Test Designed by:** Georgi Pramatarov		
Test Priority (Low/Medium/High): High			**Test Designed date:**		
Module Name: Android Client			**Test Executed by:** Georgi Pramatarov		
Test Title: User adjusts time scope			**Test Execution date:**		
Description: Verify if the user has adjusted the time scope					
Pre-conditions: User is located in the settings preference screen					
Step	**Step Details**	**Expected Results**	**Actual Results**	**Status (Pass/Fail)**	**Notes**
1	User presses the time scope field	System opens the time picker dialog	As expected	Pass	
2	User picks the beginning of the time scope	System opens the second time picker dialog	As expected	Pass	
3	User picks the end of the time scope	System closes the dialog	As expected	Pass	
Post-conditions: The system updates the summary of the time scope field to reflect the newly chosen values.					

Table 9.5: Test Case 5

Project Name: IoT Device for Home Monitoring	
Test Case	
Test Case ID: AC-002	**Test Designed by:** Georgi Pramatarov
Test Priority (Low/Medium/High): High	**Test Designed date:**
Module Name: Android Client	**Test Executed by:** Georgi Pramatarov
Test Title: User adds a new edge device	**Test Execution date:**
Description: Verify if the user has added a new device	
Pre-conditions: User is located in the System Preferences screen	

Step	**Step Details**	**Expected Results**	**Actual Results**	**Status (Pass/Fail)**	**Notes**
1	User pressed the Edge Device Setup field	System opens the add new device dialog	As expected	Pass	
2	User pressed the name input field	The system displays the keyboard	As expected	Pass	
3	User enters the name of the device and presses OK	The system closes the dialog	As expected	Pass	

Post-conditions: User is notified for the successful addition of the new edge device

Table 9.6: Test Case 6

Project Name: IoT Device for Home Monitoring					
Test Case					
Test Case ID: AC-003			**Test Designed by:** Georgi Pramatarov		
Test Priority (Low/Medium/High): High			**Test Designed date:**		
Module Name: Android Client			**Test Executed by:** Georgi Pramatarov		
Test Title: Fingerprint authentication			**Test Execution date:**		
Description: Verify if the user is successfully authenticated					
Pre-conditions: User is located in the FingerUnlock screen					
Step	**Step Details**	**Expected Results**	**Actual Results**	**Status (Pass/Fail)**	**Notes**
1	User places their finger on the fingerprint scanner	System displays a message to inform the user of the successful authentication	As expected	Pass	
Post-conditions: User is successfully authenticated and a guidance message is displayed					

Table 9.7: Test Case 7

Project Name: IoT Device for Home Monitoring					
Test Case					
Test Case ID: AC-004			**Test Designed by:** Georgi Pramatarov		
Test Priority (Low/Medium/High): High			**Test Designed date:**		
Module Name: Android Client			**Test Executed by:** Georgi Pramatarov		
Test Title: Fingerprint authentication			**Test Execution date:**		
Description: Verify if the user is not authenticated					
Pre-conditions: User is located in the FingerUnlock screen					
Step	**Step Details**	**Expected Results**	**Actual Results**	**Status (Pass/Fail)**	**Notes**
1	User places their finger on the fingerprint scanner	System displays a message to inform the user of the unsuccessful authentication	As expected	Pass	
Post-conditions: User is not authenticated					

Table 9.8: Test Case 8

Project Name: IoT Device for Home Monitoring					
Test Case					
Test Case ID: AC-005			**Test Designed by:** Georgi Pramatarov		
Test Priority (Low/Medium/High): High			**Test Designed date:**		
Module Name: Android Client			**Test Executed by:** Georgi Pramatarov		
Test Title: Camera control			**Test Execution date:**		
Description: Verify if the camera is available					
Pre-conditions: User is located in the Camera Viewer screen					
Step	**Step Details**	**Expected Results**	**Actual Results**	**Status (Pass/Fail)**	**Notes**
1	User presses the play button	System displays the camera output in the viewer	As expected	Pass	
Post-conditions: User is able to view the output camera stream.					

Table 9.9: Test Case 9

Project Name: IoT Device for Home Monitoring					
Test Case					
Test Case ID: AC-006			**Test Designed by:** Georgi Pramatarov		
Test Priority (Low/Medium/High): High			**Test Designed date:**		
Module Name: Android Client			**Test Executed by:** Georgi Pramatarov		
Test Title: Camera control			**Test Execution date:**		
Description: Verify if the camera is not available					
Pre-conditions: User is located in the Camera Viewer screen					
Step	**Step Details**	**Expected Results**	**Actual Results**	**Status (Pass/Fail)**	**Notes**
1	User presses the play button	System displays an error dialog	As expected	Pass	
Post-conditions: User is notified that the camera is currently unavailable					

Table 9.10: Test Case 10

Project Name: IoT Device for Home Monitoring					
Test Case					
Test Case ID: AC-007			**Test Designed by:** Georgi Pramatarov		
Test Priority (Low/Medium/High): High			**Test Designed date:**		
Module Name: Android Client			**Test Executed by:** Georgi Pramatarov		
Test Title: Camera control			**Test Execution date:**		
Description: Stop the camera stream					
Pre-conditions: User is located in the Camera Viewer screen					
Step	**Step Details**	**Expected Results**	**Actual Results**	**Status (Pass/Fail)**	**Notes**
1	User presses the stop button	System stops the camera stream and closes the viewer	As expected	Pass	
Post-conditions: User is notified that the camera stream is stopped					

Bibliography

[1] *ABUS Smartvest - Wireless SmartPhone Alarm.* URL: https://www.safe.co.uk/products/abus-smartvest-wireless-smartphone-alarm.html.

[2] Amazon. *MakerHawk ESP32 Development Board WiFi Bluetooth Dual Cores Ultra Low Power Consumption ESP-32 Board.* URL: https://www.amazon.co.uk/MakerHawk-Development-Bluetooth-Consumption-ESP-32/dp/B072JVQVND/ref=sr_1_2?ie=UTF8&qid=1525989728&sr=8-2&keywords=esp32+makerhawk.

[3] Amazon. *MakerHawk Raspberry Pi Camera for Raspberry Pi Model A/B/B+, Pi 2 and Raspberry Pi 3 and Adapter Cable for Zero/Zero W Camera and Camera Stand.* URL: https://www.amazon.co.uk/MakerHawk-Raspberry-Camera-Model-Adapter/dp/B07516S8PG/ref=sr_1_25?s=computers&ie=UTF8&qid=1525989817&sr=1-25&keywords=raspberry+pi+camera.

[4] Amazon. *Raspberry Pi 3 Model B Quad Core CPU 1.2 GHz 1 GB RAM Motherboard.* URL: https://www.amazon.co.uk/Raspberry-Pi-Model-Quad-Motherboard/dp/B01CD5VC92/ref=sr_1_4?ie=UTF8&qid=1525989658&sr=8-4&keywords=raspberry+pi+3.

[5] Amazon. *XCSOURCE® NXP PN532 NFC RFID Module V3 Kits Reader Writer For Arduino Android Phone TE314.* URL: https://www.amazon.co.uk/XCSOURCE%EF%BF%BD%EF%BF%BD-Module-Arduino-Android-TE314/dp/B014QZRF2U/ref=sr_1_fkmr0_1?s=computers&ie=UTF8&qid=1525989899&sr=1-1-fkmr0&keywords=XCSOURCE+PN532+NFC.

[6] *Arduino IDE Mac OS.* Feb. 2018. URL: https://www.arduino.cc/en/Main/Software.

[7] Atom. *A hackable text editor for the 21st Century.* URL: https://atom.io.

[8] Louis Columbus. *2017 Roundup Of Internet Of Things Forecasts.* URL: https://www.forbes.com/sites/louiscolumbus/2017/12/10/2017-roundup-of-internet-of-things-forecasts/#5ff8be7d1480.

[9] Eclipse. *Eclipse Mosquitto™ An open source MQTT broker.* URL: https://mosquitto.org.

[10] Elechouse. *NFC library for Arduino using PN532.* URL: https://github.com/elechouse/PN532.

[11] *ERA HomeGuard Smart Alarm Pro - Starter Kit.* URL: https://www.safe.co.uk/products/era-homeguard-smart-alarm-pro-starter-kit.html.

[12] *Everything you need to build on Android.* Feb. 2018. URL: https://developer.android.com/studio/features.html.

[13] Sharif Khaleel. *Android How to Add Fingerprint Authentication.* URL: https://www.androidhive.info/2016/11/android-add-fingerprint-authentication/.

[14] Roger Light. *paho-mqtt 1.3.1.* URL: https://pypi.org/project/paho-mqtt/.

[15] *MQ Telemetry Transport(MQTT)*. Feb. 2018. URL: http://mqtt.org/faq.

[16] C. Enrique Ortiz. *An Introduction to Java Card Technology*. URL: http://www.oracle.com/technetwork/java/javacard/javacard1-139251.html.

[17] Statista. *Internet of Things (IoT) connected devices installed base worldwide from 2015 to 2025 (in billions)*. URL: https://www.statista.com/statistics/471264/iot-number-of-connected-devices-worldwide/.

[18] Jessica Thornsby. *android authority fingerprint*. URL: https://www.androidauthority.com/how-to-add-fingerprint-authentication-to-your-android-app-747304/.

[19] Google Trends. *Smart Home Security*. URL: https://trends.google.co.uk/trends/explore?date=2016-01-01%202018-02-19&q=smart%20home%20security.

[20] *Yale SR-320 Smart Home Alarm Kit*. URL: https://www.safe.co.uk/products/yale-smart-home-alarm-kit-SR-320.html.