A Home Energy Management System with Focus on Energy Optimization

Master Thesis

Hans Hansen
Mikkel Skovsmose Tewes
Nicolai Foldager

Aalborg University
Department of Electronic Systems
Fredrik Bajers Vej 7B
DK-9220 Aalborg

Department of Electronic Systems
Fredrik Bajers Vej 7
DK-9220 Aalborg Ø
http://es.aau.dk

AALBORG UNIVERSITY
STUDENT REPORT

Title:
A Home Energy Management System with Focus on Energy Optimization

Theme:
Master Thesis

Project Period:
Spring 2018

Project Group:
ICTE4SER 4.1

Participants:
Hans Hansen
Mikkel Skovsmose Tewes
Nicolai Foldager

Supervisor:
Per Lynggaard
Niels Koefoed

Copies: 1

Page Numbers: 164

Date of Completion:
June 6, 2018

Abstract:

The purpose of this project was to develop a home energy management system that utilized machine learning in order to reduce energy. A proposed system was developed that made use of smart plugs, smart lights, and a smart thermostat. An Android application was developed that allowed the user to control and monitor the smart devices. A web server was used as a communication medium between the hub and the application. Azure Machine Learning was used as the cloud environment for performing machine learning operations.
Data such as power consumption and setpoint temperature was gathered from the smart devices, and the user's presence in the home was detected through the Wi-Fi on the smartphone application. Two machine learning models were trained in order to automatically turn smart devices on or off depending on whether the user was at home, away, or asleep.
The results showed that there was a potential for energy reduction, however, more data would be needed in order to run the system in real test, where the system automatically controls the devices during the test.

Contents

1 Introduction **1**

1.1 Project Motivation . . . 2

1.2 Problem Statement . . . 2

1.3 Delimitations . . . 3

2 Methodology **4**

2.1 Development Strategies & Practices . . . 5

3 State of the Art **6**

3.1 Electricity Usage in Households . . . 6

3.2 Heating Usage in Households . . . 8

3.3 Energy Management in households . . . 9

3.3.1 Danfoss Link CC . . . 9

3.3.2 Netatmo Smart Thermostat . . . 10

3.3.3 Tado Smart Thermostat . . . 10

3.3.4 Smart Lighting . . . 11

3.3.5 Smart Plugs . . . 11

3.3.6 SmartThings . . . 12

3.4 Machine Learning in Home Energy Management Systems . . . 13

3.5 Presence Detection . . . 15

3.5.1 Geofencing . . . 16

3.5.2 Beacons . . . 16

3.5.3 Wi-Fi Tracking . . . 17

3.5.4 Passive Infrared Sensor . . . 18

3.6 Machine Learning Processing Architecture . . . 18

3.6.1 Cloud . . . 19

3.6.2 Local . . . 19

3.6.3 Hybrid . . . 20

4 Analysis **21**

4.1 Scenarios . . . 22

4.1.1 Scenario 1 - Existing Solutions . . . 22

4.1.2 Scenario 2 - Basic System Usage and Data Gathering . . . 24

4.1.3 Scenario 3 - Applied Machine Learning . . . 26

4.2 Use Cases . . . 27

4.2.1 Hub . . . 28

4.2.2 Application . . . 30

4.2.3 Server . . . 32

4.3 Services . . . 35

4.3.1 Service 1 - Control & Monitor Smart Devices . . . 35

4.3.2 Service 2 - Adaptive Heating . . . 36

4.3.3 Service 3 - Adaptive Electricity . . . 38

4.3.4 Service 4 - Energy Visualization . . . 39

4.3.5 Service 5 - Price notifier . . . 39

4.4 Key Element 1 - Smart Electricity & Heating Devices . . . 40

4.5 Key Element 2 - User Home/Away Status . . . 40

4.6 Key Element 3 - Processing Architecture . . . 43

4.6.1 Computational Power & Storage . . . 43

4.6.2 Data Ownership, Privacy & Trust . . . 44

4.6.3 Development & Dependencies . . . 44

4.6.4 Selected Architecture . . . 45

4.7 Key Element 4 - Machine Learning . . . 45

4.7.1 Supervised learning . . . 45

4.7.2 Algorithms, Relevant Data & Features . . . 46

4.7.3 Prediction Goals . . . 47

4.8 Requirement Specifications . . . 49

4.8.1 Hub . . . 50

4.8.2 Android Application . . . 53

4.8.3 Server . . . 56

4.8.4 Cloud-based Environment . . . 57

5 Design **59**
5.1 Hub & Application . . . 60
5.1.1 Hub & Application Setup . . . 60
5.1.2 Communication Between Hub & Smart Devices . . . 62
5.1.3 Price Notifier . . . 63
5.1.4 Data Visualization . . . 64
5.2 Server . . . 65
5.2.1 Registration of Hub & Application . . . 65
5.2.2 Electricity Prices Fetching . . . 65
5.3 Data Storage . . . 66
5.4 Key Element 1 - Smart Electricity & Heating Devices . . . 67
5.5 Key Element 2 - User Home/Away Status . . . 68
5.6 Key Element 3 - Processing Architecture . . . 69
5.7 Key Element 4 - Machine Learning . . . 71
5.7.1 Data Preprocessing For The Real-time Model . . . 73
5.7.2 Classification Algorithms To Evaluate . . . 74
5.7.3 Data Preprocessing For The Forecasting Model . . . 74
5.7.4 Regression Algorithms To Evaluate . . . 76
5.8 Application User Interface . . . 77

6 Implementation **80**
6.1 Hub & Application Setup . . . 80
6.1.1 Send Hub IP To Server . . . 80
6.1.2 Register Application To Server . . . 82
6.1.3 Application Request Server IP . . . 84
6.2 Wi-Fi State Update . . . 87
6.3 Data Gathering For Machine Learning Features . . . 91
6.4 Price Notifier . . . 96
6.5 Data Visualization . . . 98
6.6 Machine Learning . . . 100
6.6.1 Model For Forecasting User Status . . . 101
6.6.2 Model For Real-Time Classification . . . 105
6.7 Requirements Fulfillment . . . 107

7 Test & Evaluation **108**
7.1 Test Environment and Setup . . . 108
7.2 Data Collected . . . 109
7.2.1 Data Presentation . . . 110
7.3 Forecasting Model Evaluation . . . 117
7.4 Real-time Model Evaluation . . . 118
7.5 Energy Consumption Reduction . . . 119
7.6 Requirements Testing . . . 124

8 Discussion **125**
8.1 Test Environment . . . 125
8.1.1 Electricity & Heating Data Analysis . . . 126
8.2 User Location . . . 127
8.3 Machine Learning . . . 127
8.3.1 Sleep Status . . . 128
8.3.2 Energy Reduction . . . 128
8.4 Prototype Architecture . . . 131

9 Conclusion **132**

10 Future Works **134**
10.1 Multiple Users . . . 134
10.2 Calendar Integration . . . 134
10.3 Reinforced Learning . . . 135
10.4 Different Profiles for Different Seasons . . . 135

Bibliography **136**

Appendix **148**

Abbreviations & Meanings

AI	Artificial Intelligence
API	Application Programmable Interface
AWS	Amazon Web Services
App	Application
BDTR	Boosted Decision Tree Regression
BEC	Building Energy Codes
BLE	Bluetooth Low Energy
BLR	Bayesian Linear Regression
C	Celsius
CPU	Central Processing Unit
CSV	Comma-seperated Values
DFR	Decision Forest Regression
FR	Functional Requirement
GDPR	General Data Protection Regulation
GPS	Global Positioning System
HEMS	Home Energy Management System
IDE	Integrated Development Environment
ISP	Internet Service Provider
IaaS	Infrastructure as a Service
MAC	Media Access Control
MDF	Multiclass Decision Forest
MDJ	Multiclass Decision Jungle
ML	Machine Learning
MLR	Multiclass Logistic Regression
MNN	Multiclass Neural Network
NFR	Non-functional Requirement
NPE	Neural Processing Engine
NPM	Node Package Manager
NPU	Neural Processing Unit
PIR	Passive Infrared Sensor
R^2	Coefficient of Determination
RPi	Raspberry Pi
SSID	Service Set Identifier
STD	Standard Deviation
SaaS	Software as a Service
UI	User Interface
VCS	Version Control System
W	Watt

1 Introduction

From 1990 to 2014 the total energy consumed by the world increased by 151% [1]. In the same period, the amount of the world population who have access to electricity grew from 73.45% to 85.32% [2]. As the majority of this energy generation comes from fossil fuels that release greenhouse gasses when combusted, the nature has been negatively affected by the rising CO_2 levels in the atmosphere. As the world has slowly become aware of the negative effects, emphasize has been put on utilizing resources better for an eco-friendly way of living. This is especially present if one looks at the many different solutions that already has been developed to reduce waste and limit the $_2$ emission. More and more companies develops solutions to reduce the energy consumption. The Danish company Danfoss has developed a smart radiator thermostat that automatically turns off the heating based on a schedule entered by the user, while companies such as Tesla Motors have revolutionized electric cars by making them able to drive longer with smaller batteries. These public and well-known products have helped put eco-friendly and energy saving on the global agenda.

Similarly to growth that eco-friendliness has experience, artificial intelligence (AI) has made huge advancements. In 1952 IBM's Arthur Lee Samuel wrote a computer program that not only knew how to play checkers, but also got better at it the more it played [3]. Since then researchers and developers have worked on improving cognitive computing by going from theory to reality. Recent major advancements were seen in 2011 when IBM Watson beat two champions in Jeopardy! [4] and in 2016 when Google's AlphaGo was the first AI to win against a professional Go player [5].

The advancement in these research topics have found them to be combined. As households increase their energy consumption, both for heating and for electricity, it has been more and more prevalent to control this consumption. Home energy management systems (HEMS) allow the residents to schedule e.g. heating for when they are home. However, many of these are reliant of a static schedule defined by the resident. Combining a HEMS and AI may allow the system to dynamically reduce the energy consumption based on the user's presence in the house.

1.1 Project Motivation

The premise of this project is based on the idea that machine learning can help reduce the energy consumption of households. A study done by Attari et al. [6] found that the majority of their test subjects' energy saving strategies were to turn off light, drive less, turn down the heat before leaving the apartment, etc. However, all of these are directly conscious decisions from the users, none of the strategies are automated. The most efficient way of reducing the energy consumption is simply to turn off all electrical products and turn off the radiators, however, this would essentially make the house uninhabitable. Due to this the comfort of the user and their routine are important aspects to consider. Gram-Hanssen [7] found that there are four aspects which affects the energy consumption of a household; knowledge, technologies, habits, and the importance of residential heat comfort. All of which are essential to consider when automating parts of a user's life. The patterns of a resident (*their habits*) can be used to train a machine learning model (*the technology*) to reduce the energy consumption of the household while keeping the home pleasant to stay in (*heating comfort*) and informing the user of possible energy saving actions they can take (*knowledge*). As such, the goal of this project is to train a machine learning model to autonomously reduce the energy consumption of the house based on the user's habits, for example how they consume energy and their presence in the home.

1.2 Problem Statement

The following problem statement was defined based on the idea that machine learning can help households to reduce their energy consumption. In addition to the problem statement, three sub-questions were defined in order to support the problem statement.

How can a smart energy management system be developed in order to reduce the amount of energy spent on heating and electricity in a household?

The sub-questions can be seen below. These will help ensure that the problem statement can be answered fully in the conclusion of the report. Furthermore, they help narrow the scope of the report to focus on the important aspects of the research areas.

- How can the architecture be designed?
- How can electricity and heating be controlled in an intelligent way to reduce the energy consumption?
- How can the location of the user be used?

1.3 Delimitations

It was decided to define delimitations of the project in order to narrow down the scope of the project and increase focus on specific research areas instead of glancing them over at a high level. For this project four delimitations have been chosen.

1. **Focus only on district heating in Denmark:** The project research and test will be based on Danish standards. As district heating using radiators is the most popular heating method in Denmark [8], it was decided to focus only on this.
2. **Focus on software prototyping:** For the development of the prototype there will be no consideration on what kind of hardware that is used. The focus will be on building the software platform. The impact on the test results between using a Raspberry Pi and an Intel Nuc is minimal, while the impact of two different machine learning algorithms may be greater.
3. **1-person household:** The prototype will only support households with a single resident in it to focus on answering the problem statement rather than the accessibility of the system.
4. **No focus on usability:** There will be no focus on user experience or usability of the user interface due to the same reason as the previous delimitation.

2 Methodology

As this project covers the development of a prototype and delivering a report about the researched area, which seeks to answer the problem statement defined in section 1.2, it was vital for the group to manage their time in order to utilize the specified time given. As such, different software components were used in order to make the process as smooth as possible. The tools and their purpose can be seen in table 2.1.

Table 2.1: Primary software used through this project.

Name	**Purpose**
Google Drive	Notes and resource sharing
Trello	Timeline and project management
ShareLatex	Text and report editor
Android Studio	IDE for Android application development
IntelliJ IDEA & Atom	IDEs for Node.js development
Git	Used for version control system

As seen in table 2.1, we used a project collaboration and management tool called Trello. The group has used Trello before, which made it easy to setup the Trello boards that can be seen in Appendix A. The group had two boards, one for the report, and one for the prototype. It was primarily used to create backlogs to get an overview of tasks to do during the project.

Google Drive was used to take notes during the supervisor meetings and for sharing resources, such as picture files, used in the report. Furthermore, a Gantt-Chart was created in order to give an overview of the deadlines set by the group. The Gantt-Chart can be seen in Appendix B. The report is written in LaTeX, which is a document preparation system. It allows the users to freely define the style of the report, labels, and more. Furthermore, the service ShareLatex, allows multiple users to edit the document simultaneously.

The group had meetings three times a week in the start of the project, but this increased to 5-6 times a week as the required work load increased. Every group meeting started with defining an agenda on a whiteboard, designating tasks that should be completed either that day or the day after. The project was heavily influenced by desktop research, which includes using and reading relevant articles, books, white papers, and technical documentation.

2.1 Development Strategies & Practices

As all of the group members participated in developing the system, a version control system (VCS) was used. Git is a reliable VCS which helps the developers see saved changes, while allowing them to rollback to a previous version if needed. Git also allows the developers to each have their own branch, which allows them to work on the same project simultaneously, but on different features in the system. When a developer has finished working on a branch they can commit and perform a merge request to the master branch, this contains the final version for this iteration.

Pair programming was used throughout the development of the system as it helped the developers better understand the codebase, while also making sure that the developers had same definition of the task at hand. Furthermore, it was used to ensure a high code quality and to avoid bugs. In addition, as the group members had different competences it was used to share knowledge.

Throughout the duration of the project, the group had meetings with the project supervisors once every week. The two supervisors provided feedback through thorough discussions. Furthermore, the two supervisors helped define scenarios which later were used to define the project's requirement specification. The meetings helped the group to continuously move forward, as an updated version of the report had to be delivered once a week.

As the project contains two aspects, the report and the prototype, the time was split into two segments as both were required to be delivered simultaneously. Approximately 80% of the time was used writing the report and the remaining 20% was used developing the prototype. In order to speed up the process of developing the prototype, only the most vital functionality and core components were developed. This is also reflected in the requirement specification, as there are requirements which were deemed unnecessary for the first iteration. The unnecessary requirements are the ones that are not prioritized with "must".

While the report is written in the style of a waterfall model, the project overall did not follow that model. The requirements defined in chapter 4 were revisited several times, in order to increase the flexibility of the project, avoiding the scenario where too many requirements were defined and could not be met. For further development of the prototype, the next step would be to go through the current requirements and define new ones for the next iteration.

A framework named FURPS was used as a guideline when defining the requirements for the prototype. FURPS divides the requirements into two categories, the functional ones which describe the use cases of the system and the non-functional ones such as usability, reliability, performance and supportability [9].

3 State of the Art

In this chapter different areas related to the research question and sub-questions will be presented. First, electricity and heating usage in households is researched in order to gain a better understanding of what has been done in the area of optimizing and reducing energy consumption. Next, current solutions that are on the market are presented. These include smart thermostats, lighting control, smart plugs, and smart hubs. Research in machine learning for home energy management systems will be presented, together with different technologies to track the user. Different processing architectures for machine learning are researched to help define if the system should be local, in the cloud, or a hybrid.

3.1 Electricity Usage in Households

In Europe the majority of the households allows for a high standard of living which is also reflected in the energy consumption, where almost a third of the consumed electricity is from the residential sector [10]. Even though, as presented in section 1.1, people are willing to perform energy conserving actions, however, there are some barriers which hinders this.

Nachreiner and Matthies [10] finds that one of these barriers is the very general energy saving strategy, where companies and municipalities provide examples of where it is possible to save energy, as these are not successful and require improvements. Instead they suggests that a personalized information strategy should be used, where the strategy is about providing information and enlightening the users about how to conserve energy. Furthermore, they suggest that an algorithm to group the residents of the household based on their current context and their appliances could be used. This would give more suitable, precise and helpful information about energy conserving actions which are applicable for the residents in their current context.

Steg [11] builds on top of the aforementioned, providing two additional strategies that can be utilized in order to conserve energy. These two being psychological strategies and structural strategies. The psychological strategies are about affecting the persons perception, knowledge, and motivation related to energy conservation [11]. The structural strategies are where the context of decision-making are affected, and examples of this can be seen as pricing policies, and new and better products or services [11]. Furthermore, several different aspects should be taken into consideration when trying to conserve energy, whereas one of these are

the contextual and individual factors, as there might be aspects, which one cannot have an impact on. Some of these aspects could be a badly insulated household, old windows, poorly placed radiators, and stock appliances with bad energy markings or labels.

As non of the strategies towards conserving energy seem to be efficiently on its own, they can be combined into a single strategy which then can be used to inform, affect, and help the user to better grasp and understand how energy can be conserved. As presented in section 1.3, this projects will be designed and implemented for the population of Denmark. Therefore, the general energy conserving methods that will be provided may only be applicable for Danish households. Several of the large danish energy groups (Ørsted, SEAS-NVE, Energi Fyn, Energi Nord) provide data for different agencies that uses it to establish guidelines for the Danish population on how to save and conserve energy and electricity. Tænk, Elforbundet, Bolius, Energihjem, Ewii, and SparEnergi [12, 13, 14, 15, 16, 17] presents the Danes' electricity consumption based on different categories, and provide several actions that the population can take, in order to lower and conserve the amount of electricity that is used during a year. In figure 3.1 a presentation of the average danish family's electricity usage is shown. As it can be seen, the majority of the electricity usage are used on entertainment, namely 39%, then comes Washing 20%, lighting 12%, cooling and freezing appliances 12%, are those areas where the largest amount of energy dispersion is. Hence reducing the amount of energy used in these areas, would be beneficial.

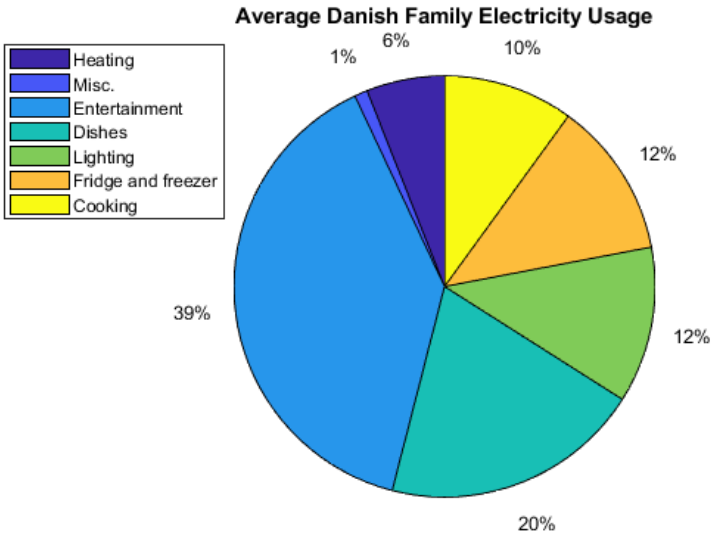


Figure 3.1: Electricity usage for an average danish family. Data from [14].

Several groups, herein Tænk, Elforbundet, Bolius, Energihjem, Ewii, and SparEnergi [12, 13, 14, 15, 16, 17], provides a lot of examples, on what can be done in order to reduce the amount of electricity used. In the kitchen area you should make sure to completely fill the dishwasher before starting it, and instead of using a program which washes the dishes at 65° Celsius, use a program which does it at 50-55° Celsius, as this uses 10-20% less electricity. Furthermore, make sure that the refrigerator is at 5° Celsius and the freezer is not lower than -18° Celsius, and that it is defrosted. Each declining degree increases the energy consumption by 5% and 2-3%, respectively. In the bathroom area, the guidelines states that when using the washing machine, you should consider washing your clothes at 20° Celsius instead of using a program which uses 30-40° Celsius, as this reduce the amount of electricity consumed with 55% for that specific task. Instead of using the dryer, it is much better to hang the clothes to dry as this does not use any electricity, and with a dryer in energy-class A, a cycle costs around 6 DKK. A survey also found that in general, the Danes only wash about 3.2 kilogram of clothes in each wash, whereas the majority of the washing machines has the capacity to wash 5-8 kilogram of cloths for each wash [12]. General overall things where energy conservation can be applied to is lights and stand-by statuses on appliances. Remember to turn off the lights if you are not in a room, furthermore, if any of the appliances has a standby function, it is more electricity conserving to completely turn it off.

3.2 Heating Usage in Households

In order to lower the amount of energy waste 118 countries have since 2010 implemented different energy policies [18]. One of these policies regards the design of constructing new buildings. The Building Energy Codes (BEC) sets a minimum energy performance requirement for the building [18]. Other policies aims towards enlightening, educating, and informing the users of the benefits of buying energy efficient appliances [18]. Another policy emphasizes that newly developed appliances is required to uphold a certain threshold of energy efficiency [18]. With all of the aforementioned policies, there is a common goal to lower the energy waste, by reducing the amount of energy used by everyday appliances. Furthermore, the implementation and adoption of smart meters adds another incentive to decrease energy consumption by the changing prices of energy based on the peak load of the energy grid. The time between 17:00 and 20:00 will be the most expensive, as this is where the energy production peaks during the day [19].

Several studies [6, 20, 7] show how house- and apartment owners are interested in consuming less energy, and hence saving money. As this project is done in Denmark, good heating practices created by Danish district heating companies are presented [21, 22, 23, 24]. 1) If

there are more than one heating unit located in a room, make sure to use both of them. It is more economically beneficial to split the work load between the heating units, instead of placing all the work load on a single unit. 2) Check the return water temperature from the radiators. This should be as cold as it can get, otherwise there is the possibility that you will receive a fine, if the return water is higher than a certain threshold. 3) Keep the indoor temperature around 20-21° Celsius, as every increased degree above 20° Celcius will increase the heating bill by approximately 5%. 4) Keep the indoor temperature at around 15° Celsius in rooms which are not used as often, as this will help keep away moist being created. Furthermore, if the room contains a door, it should be kept shut, as it hinders the room to consume the heating produced in the used rooms. 5) Ventilate the household two to three times a day for 5 minutes each time, as this provides a good indoor climate, hindering the occurrences of fungus.

3.3 Energy Management in households

Several companies have already been working on implementing the energy saving methods discussed in section 3.1 and 3.2. This section will discuss some of these implementations in relation to the problem statement from section 1.2. First, several smart radiator thermostats and their energy saving features will be discussed, then lighting and general electricity saving devices will be presented. Finally, a connected home hub and its benefits are examined.

3.3.1 Danfoss Link CC

Danfoss developed the Danfoss One ecosystem that consists of a central controller unit, smart thermostats and room temperature sensors [25]. Danfoss claims that replacing your existing manual radiator thermostats with their smart thermostat may yield savings of up to 30% on the households heating consumption [26]. These savings can be achieved by reducing the temperature of the household during periods where the residents are either not at home or at sleep. To do this, the Danfoss Link relies on a static schedule that each resident must have adjusted to their daily day through an application installed on their smartphone. While the users themselves decide the setpoint temperature when they are not home, Danfoss recommends not going below 5 degrees from the setpoint when they are at home [27].

Another way that the Danfoss One ecosystem tries to decrease the amount of energy spent on heating is by sensing if the room is currently being aired out, i.e. is there a window open. It does this by checking if the temperature has fallen more than 0.5° Celsius over three minutes [28]. The system turns off the heating for an hour if it senses that a window

has been opened. This method is reactive as it does not respond to the open window before the temperature starts to drop.

An issue with the Danfoss One ecosystem is that it heavily relies on a static schedule set by the users, it does not track user habits or their location to regulate the temperature of the house [29]. This means that if a resident returns home earlier than what their schedule is set to, the temperature of the house would be cooler than desired. A reaction to this from the users side might be to enable the heating and setting the temperature higher than currently desired to quickly reheat the room. However, this means that the water temperature entering the radiator is so high that the heat it carries cannot properly be dispersed into the heating element of the radiator. As discussed in section 3.2, the water leaving the radiator will still be warm and still have heating capabilities in it as the water temperature and the air temperature have not reached an equilibrium.

3.3.2 Netatmo Smart Thermostat

The Netatmo Smart Thermostat is a system that consists of a hub and radiator thermostats. Similarly to the Danfoss system, Netatmo also regulate the heating based on a static schedule. This allows the hub to reduce the temperature during the night and when the residents are at work [30]. This may lead to the same issue as the Danfoss Link CC has where the user increases the radiator setpoint drastically to quickly increase the room temperature, thus also increasing the temperature of the returning water as discussed in section 3.2.

The Netatmo system differs from the Danfoss One in that it calculates the rooms thermal characteristics and uses outdoor conditions to learn how to efficiently heat the house. The system calculates the thermal characteristics by turning the radiators to maximum power in order to determine the maximum heating output, then turning off the valve and observe the time it takes for the temperature to fall again [31]. This allows the thermostat to start heating as close to the scheduled arrival time and thus not risk heating for too long before the residents arrive home. The parameters of the thermal characteristics are adjusted every two weeks to ensure that environment has not changed.

3.3.3 Tado Smart Thermostat

The last thermostat discussed is the Tado Smart Thermostat. The system consists of a gateway and smart thermostats. Tado differs from the other two in that it does not necessarily rely on a static schedule to regulate the temperature. Instead it may use the geolocation of the user's smartphones. When all residents are out of the house the thermostat automat-

ically turns off the heating to avoid unnecessary energy usage. This allows the system to adapt to the user's current needs rather than their needs being decided by their schedule. As such, if a resident arrives home earlier than expected, the house would be sufficiently heated. However, it may also regulate based on a static schedule or manual mode similar to the other thermostats [32]. In addition, Tado also have reactive open window detection and calculates the thermal characteristics of the house [33].

3.3.4 Smart Lighting

A solution to managing the energy usage of a household is by controlling the lighting. As discussed in section 3.1, lighting in an average Danish household take up 12% of the total electricity usage. Several companies have developed solutions that allows the residents to control the lights in the homes remotely from their smartphones. These companies include Philips, Ikea and Lifx. All of which allows the residents to schedule when their lights should turn on and off automatically, similarly to how the schedule on the thermostats discussed earlier works [34, 35, 36]. Both Philips Hue and Ikea TRÅDFRI sells motion sensors that can overwrite the schedule if the user happens to arrive home earlier than expect. However, this requires a motion sensor in every room, which may quickly become costly. In addition to a schedule and motion sensors, the Phillips Hue can also be controlled through geolocation based on the user's smartphone. This allows the system to reduce the energy usage while the residents are not home by turning off all of the lights [36].

3.3.5 Smart Plugs

As discussed in section 3.1, automatically cutting the power to electronic devices that are standby is not a novel way of saving energy in a household, timed plugs have been on the market for many years. In a pre-defined time frame these cut to power from the socket to the device or power strip that is connected to it. Generally, these are fine for devices that can be turned on and off based on a schedule, however, for many other devices, such as media centers and computers, this may not be a viable option, as turning off a computer unexpectedly may result in lost work. In recent years a digitized version of the timed switches have started to be developed. Belkin developed the Wemo Switch Smart Plug [37]. This allows users to control the smart plug through a smartphone application. Smart plugs can be used to devices that use electricity when in standby mode such as game consoles, DVRs and television.

In addition to be remote controlled, other smart plugs such as the NodOn Smart Plug can

also monitor the power usage of the device that is plugged in [38]. This can be beneficial to use on devices that are continuously running and where having a power source connected is important. Using a smart plug with a refrigerator can help ensuring that the refrigerator is working properly and is not drawing more electricity than what it requires. Further, it could help notifying the user when to defrost the freezer compartment to reduce power draw as discussed in section 3.1.

3.3.6 SmartThings

The SmartThings Hub, a connected home controller created by Samsung SmartThings, allows various devices to connect to it. The hub allows types of devices discussed earlier in this chapter, e.g. thermostats, lights, to communicate information with each other, perform actions based on triggers from other devices and let the user control them remotely through their smartphone [39]. While the SmartThings hub itself will not optimize the energy usage, it may act as a medium for two devices to communicate with each other. An example of this is using a Multipurpose Sensor [40] that can detect when a window is opened, alert the SmartThings hub which then can turn off the heating to reduce unnecessary energy usage. Another example to save electricity is to use a smart plug [41] that can be operated remotely and can monitor electricity usage. Paired with presence detection it allows the hub to turn off media centers, computers, etc. when the residents are not home to avoid using electricity when in standby mode.

While SmartThings is generally a closed system that only works with select partners, the partners they have and their services are very versatile. They include support for home assistants such as Google Home and Amazon Echo, light systems such as Phillips Hue and various sensors such as motion sensors, window/door sensors, smart plugs, etc. [42]. All of these can be used to reduce the energy consumption a household uses.

A major problem with the SmartThings Hub is that their architecture is centralized, meaning that if a user wants to control a device the request is routed through SmartThing's server rather than being send directly to the hub. This means that if the central server is offline then no one can control their devices. While this generally is not a problem if their servers are stable, SmartThings have had problems in the past with keeping their servers online [43], rendering remote controlling devices disabled.

A companion smartphone application can be used to control the various devices connected to the SmartThings Hub [44]. In order to pair the application with the user's personal SmartThings hub, they have to enter a 6 character code printed on the hub [45]. This allows

the hub and the application to establish a communication channel through a remote server.

3.4 Machine Learning in Home Energy Management Systems

There have been done a lot of research in the field of energy reducing measurements for households and buildings. One of the areas that has been emphasized is the development of home energy management systems, which is defined as a system allowing the consumer to control, understand, and optimize the energy consumption in the household[46]. Aman et al. [46] provides a overview of different solutions, such as PERSON, ALIS, and NOBEL which have been prominent for the research of the field. Furthermore, Aman et al. [46] also provide their own definition of what functionality a HEMS should have, and analyses the affordability towards the end-user, while taking other factors such as the utility companies, neighbors, and weather data into consideration. A visualization of how they believe the HEMS should be integrated into the society can be seen in figure 3.2.

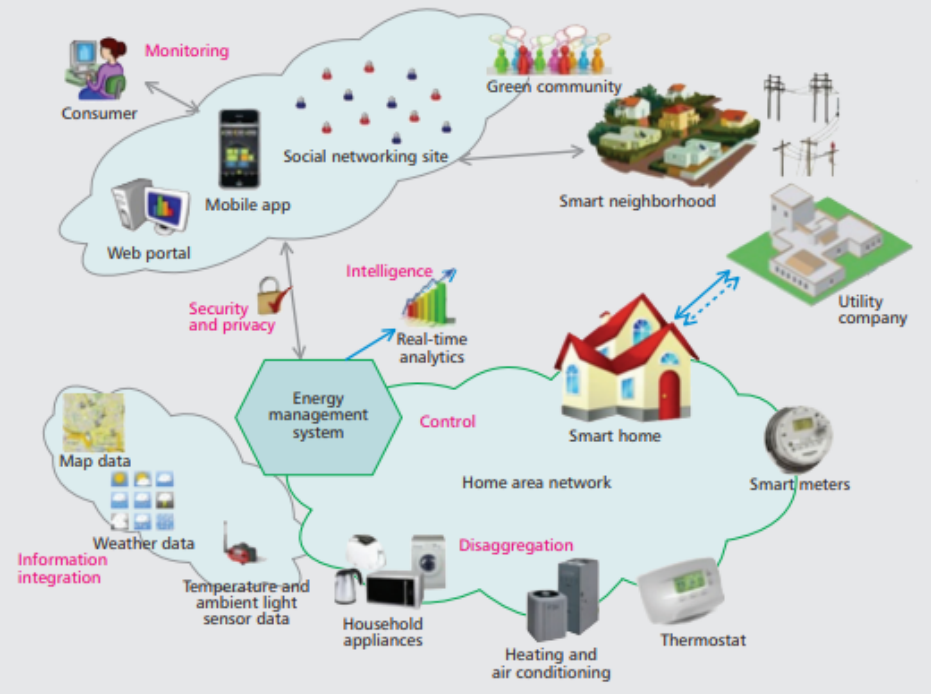


Figure 3.2: Integration of HEMSs within the infrastructure [46].

The different functions they define are monitoring, dis-aggregation, availability and accessibility, information integration, affordability, control, cyber-security and privacy, and intelligence and analytics. All which are important in order to provide a solution that empowers

the consumer. However, most paramount function to consider for this study is the intelligence aspect, which in this project is considered the machine learning aspect. Only one of the solutions presented by Aman et al. [46] is intelligent, where the others are lacking in this area. Aman et al. [46] concludes that a HEMS should be integrated with the home automation system that controls the rest of the house, and that the solutions presented are not future proof, as they only scratch the surface or optimizing energy consumption with the use of machine learning (intelligence and analytics).

Shakeri et al. [47] further supports the definition presented by Aman et al. [46]. Shakeri et al. [47] extends the definition by putting the research on HEMS' into two categories, namely predictive energy management and real-time energy management. The former category is where e.g. historical data is used to predict the electricity consumption, and the latter is where controllable appliances are postponed in order to optimize electricity at peak hours [47].

Shahriar and Rahman [48] also proposes a system which utilizes machine learning in order to help limit energy waste and being more eco-friendly. In their experiment they try to predict the amount of energy that the solar panels on a house can be expected to produce, in order to plan energy usage. Shahriar and Rahman [48] also uses machine learning to predict which devices and appliances that consumes energy while taking the temporal context into consideration, e.g. time of day and day of the week. They found that taking the context into consideration gave them a high accuracy rating, allowing them to efficiently manage the energy in the urban smart home [48]. However, in order to get the most out of their proposed solution, you have to fulfill the assumptions they present, which requires you to have solar panels, a hybrid car, an energy monitor sensor, and the system should also aggregate local weather and environmental data [48]. Furthermore, they do not present a end-user solution with an interface, which otherwise could be used to communicate and visualize energy usage for the user.

Kleiminger et al. [49] mention that monitoring occupancy in households and buildings is typically achieved through motion sensors and cameras, whereas they use quite a different method by combining motion sensors and energy consumption. They installed PIR sensors in homes as well as smart meters and smart plugs for measuring power consumption. The was also to use machine learning to figure out whether anyone was at home or not. They also used an Android application to gather data, which was installed on a tablet in the homes. This means that users had to explicitly change states from being home or away through the tablet. The reason for doing this was because they needed to label their gathered data in order to know whether users were home or away, since they used a supervised machine

learning approach. However, one cannot be certain that the users always remembered to change states whenever they left their homes or arrived at home. Their users also admitted to forgetting it sometimes. They only focused on electrical power consumption, which means heating is not considered.

Based on Koehler et al.'s [50] research in 2013, most users controlled their thermostat temperatures manually, however some households used thermostats that could be programmed to follow a static schedule. They mention that manually controlling the temperature can be effective by using eco-feedback technology. Eco-feedback technology is defined as using technology to provide feedback to the users based on their usage in order to give them an insight on how they can e.g. reduce their usage in order to spare the environment [51]. Koehler et al. [50] presents two ways of reducing energy usage in a household, either through either eco-feedback technology or through automation. Their solution uses a GPS to monitor the users location every 2 minutes, where eco-feedback was added through a smartphone application as well. They allowed the user's to see graphs of their monthly energy usage. They thereafter tried to predict whether the users will be at home or away in the next hour using the GPS data.

3.5 Presence Detection

The use of presence detection can be seen available in many different smart home solutions currently on the market. A typical reason for detecting presence is for automation, where a system can know whether users are at home or not, and therefore allows for automatic adjustment of e.g. heat when the users are home or away. This can create a comfortable home experience for the user, but also reduce the amount of energy used when people are not at home [52]. There are multiple ways to detect presence, such as GPS, WiFi, Bluetooth and more, and the purpose is to make a context-aware system [53]. This means for example that sensors in a smart home can adjust their behaviour depending on a user's location. This can be used to detect the activities of a user. The SmartThings Arrival Sensor allows users to automate other sensors by turning them on or off when e.g. a user arrives at home [54]. The sensor is small, which allows a user to carry it or it can be attached to something. Samsung also has another small key chain size presence sensor called SmartSense Presence Sensor. It uses ZigBee to communicate whether it is within or outside the range of the smart hub [55]. The Wink Motion Sensor is a bit different as it is used for security purposes to detect unwanted motion in your house [56]. The Tado thermostats discussed in section 3.3.3 come with an application that uses geographical location, which means that the thermostats can automatically turn up the heat when a user about to get home [57]. The aforementioned

products are quite different, which is why technologies for presence detection will be further investigated in this section.

3.5.1 Geofencing

The Global Positioning System (GPS) is used for acquiring geolocation information through the use of satellites. Using geolocation it is possible to setup geofences. A geofence is a geographical boundary that defines a virtual perimeter [58], it can trigger an event when the user enters or exits the virtual zone. Geofences have also existed for a long time and they make use of GPS coordinates. A geofence can for example be made on an application that uses Google Maps, where a circle can be drawn on the map to imitate a virtual fence. An application can then send messages to the user whenever he enters or exits this circle. Statler [58] mentions three important factors when using geofencing on mobile applications:

- **Permission** - users have to download a mobile application and give it permission to use the location of their mobile device.
- **Privacy** - locations visited before entering a geofence should not be tracked.
- **Preference** - users should be able to choose their preferences for getting notifications.

The Tado thermostat discussed in section 3.3.3 is a perfect use case for geofence, because their mobile application dynamically controls the thermostats based on the location of the users. Users are also able to choose a balance between comfort and heating savings, which give them some choice of preference. However, it is not possible to use geofences for tracking users inside a home, but a geofence could be used to notify users to turn on Bluetooth when they arrive home, incase they have bluetooth beacons in their home. Bluetooth beacons will be explained in the next section. Geofencing can either work as *active* or *passive*. Active geofences are limited to only running while the mobile application is open, which allows the application to use high-accuracy GPS while draining battery quite fast. Passive geofences work by running in the smartphones background, which does not use high-accuracy GPS and allows the application to use much less battery compared to the active geofences [58].

3.5.2 Beacons

Locating a user can be done through the use of beacons, where these beacons are Bluetooth Low Energy (BLE) devices. These have become increasingly popular for indoor positioning, since GPS might not work very well indoors as material such as walls can decrease the effectiveness [59]. BLE devices are used in conjunction with another device, such as a

smartphone, where the position of the smartphone can be calculated based on the information from multiple BLE devices [59]. There are different algorithms that can be used for positioning, where one of them is trilateration. Trilateration requires information from three beacons with fixed positions, since it is necessary to know each beacons' location in order to compute the distance to the smartphone [59]. BLE devices are cheap and have a long battery life, which make them affordable when you need many beacons [53].

Apple made a BLE device in 2013 called iBeacon, which is used to transmit Bluetooth signal that other compatible devices (e.g. smartphones) can detect, and then certain actions can be performed on the smartphone [60]. An example of a use case, where such beacons are used, is the campaign "Pink Light", which was tested in South Korea. There were 500 pregnant women involved in the campaign, where each of them carried a BLE device that was used for activating a pink light in trains whenever one of the pregnant women entered the train. The light would only turn off when she got a seat [61]. This is only one use case, but there are many different ways beacons can be helpful. Blas and de Ipiña [59] mention four different ways of benefiting from beacons. They can be used in retail by sending smartphone notifications with special offers to users when they walk close to a beacon. They can also be used to provide content when for example walking in a museum. Patients in hospitals can be tracked using beacons in order to provide them with the best comfort possible. Beacons can also be used in suitcases at the airport in order to make users track their suitcase. Conte et al. [62] came up with Blue-Sentinel, which was a project where iBeacons were used to reduce power consumption in buildings. They made use of iBeacons to detect the number of users in rooms and track them inside the building. It would then be able to control some of the heating and electricity usage automatically based on users' location, but this was not done in the Blue-Sentinel project, since they only focused on tracking users.

3.5.3 Wi-Fi Tracking

Another way to track users is through Wi-Fi. Most people have a smartphone today, and many of them always have Wi-Fi running in the background. A way of tracking smartphones is by having an access point that detects the Wi-Fi signal transmitted from the smartphone. This is referred to as Wi-Fi Probe requests, where smartphones broadcast a signal on a regular basis [58, 63]. One of the simple location tracking techniques is to measure the signal strength of smartphones close to the access point [58]. This could for example be used to track if a store has many customers throughout the day. It is possible to track if customers are returning to the store later, since every devices has a unique Media Access Control (MAC) address. However, this can also cause some issues nowadays since Apple started to change

MAC addresses on iPhones on a random basis [58]. MAC randomization was also introduced as a feature of the new Android version called *Android Oreo* [64]. The real MAC address is only showed when the smartphones are connected to the network. Despite this problem, previous research can still can still provide insight on how users can be located using Wi-Fi. Boonsriwai and Apavatjrut [65] investigate how Wi-Fi localization can be used, where they mention techniques that use access points and the current location of a Wi-Fi enabled device to figure out where the device is located. However, this requires multiple access points in order to provide accurate results. Considering a usual household, then they are likely to only have one router or access point, which makes it difficult to use Wi-Fi localization with installing multiple access points. Simply checking if the Wi-Fi device is connected to a specific router or access point might help reveal if the user is at home, but if the user lives in a building with multiple apartments, then the user can be at one of the neighbors while still being connected to their own home network.

3.5.4 Passive Infrared Sensor

A type of sensor that is often used in smart homes is a Passive Infrared Sensor (PIR). These sensors are used in different smart home products such as in motion sensors for security purposes or for triggering other sensors. PIR sensors are popular for controlling lights whenever they detect motion, and it has the benefit of not being as obtrusive as e.g. cameras utilizing computer vision, since it only detects movement [66]. A PIR sensor measures temperature or heat in a certain angle, which makes it possible to detect whenever a human is within the angle of the PIR [67]. This allows for automatically turning other sensors on or off based on human movement. Other methods, such as using a camera, can also be more expensive and cause a privacy concern, where PIR sensors are cheap and might not have such privacy concerns [66]. There are certain disadvantages with PIR sensors as well. Agarwal et al. [66] mention a problem when two persons are in the same room, and then one of the persons leaves while the other one keeps sitting at a desk writing. If the person at the desk does not move his body enough then the PIR sensor cannot detect any movement.

3.6 Machine Learning Processing Architecture

As a part of this project concerns how machine learning can be used to automate reducing the energy consumption of the household, it is highly relevant to consider where the processing of the machine learning model should be located. This section will present three types of architectures; purely cloud based, local processing and a hybrid. Advantages and

disadvantages of each will be discussed in relation to the problem statement in section 1.2. The findings will be used to help find a solution to perform machine learning processing based on factors such as user privacy, scalability and system complexity.

3.6.1 Cloud

Several major cloud companies, such as Amazon [68], Google [69], Microsoft [70] and IBM [71], have developed and deployed Software as a Solutions (SaaS) that allows both companies and individuals to make use of machine learning in their projects. These services are not limited to machine learning, but also includes speech recognition, speech-to-text and natural language understanding [72]. A cloud based system allows a single remote component to handle all the processing from data preparation to model training and predictions. The advantages of this of a cloud based SaaS is the flexibility of the system. It allows for offloading of scalability and development to an external partner to avoid having to consider up-keep of servers and development of complex algorithms. In addition, you do not have to worry about a lack of computational power in the future, as this is only a concern for the owner of the physical computational unit. An example of a smart home product that makes use of cloud based processing is the series of Amazon Alexa devices. Any query the user gives Alexa is sent to a cloud service for interpretation [73].

A cloud solution does come with some disadvantages. A major concern is ownership and sharing of user data. The most optimal place to store the data in order to continuously retrain the model is with the machine learning SaaS provider. However, this opens up for issues with user privacy. It has been stated many times in the past that the new gold is data [74, 75, 76] as it may describe users very efficiently and precisely. Some users may not be comfortable with having their data stored and shared with external services, especially with companies such as Google who make their living of user data [77, 78]. In addition, using an external services adds another dependency for the product and possibly adds a single point of failure to the system. If the services shuts down then another solution would have to be developed and an update pushed to the smart home hubs.

3.6.2 Local

A local processing method means that the smart hub or another device within the house performs the machine learning training and prediction. The benefits of this architecture is that it is not dependent on an external service and as such, down-time in the Internet connection does not mean that the machine learning capabilities will be unavailable. While

removing the dependency of an Internet connection may be an advantageous feature, it does come with disadvantages as it increases the requirements for the smart hub in terms of computational power and storage as it has to perform all computations and store all of the data locally. At some point the model may become so complex because it is based on many instances that it may not be feasible to train the model due to the computational power needed. In addition, using a local processing method also puts puts further emphasis on the development team, as it requires complex algorithms to be developed properly and correctly.

Recently there has been put a focus on smartphones with neural processing engines (NPE) and even dedicated chipsets called neural processing units (NPU) that can support on-device machine learning. Having these on the device allows the CPU to offload machine learning computations to the NPU as they are designed to specialize in fast and efficient model training and prediction [79]. In September 2017 Huawei announced the Kirin 970, a mobile CPU with a dedicated NPU that, according to Huawei themselves, are 25 times faster and 50 times more efficient at machine learning computations than a 2016 multipurpose chipset [80]. Another example of technologies that allows for on-device machine learning capabilities are the Apple A11 Bionic chipset that includes a neural engine [81]. When the Apple iPhone moved from authenticating users through fingerprint to authenticating users through facial recognition, they needed a chipset that could quickly accept or reject authentication attempts. The A11 Bionic allows the phone to build a machine learning model using features from the user's face that can be used to prediction during the authentication process [81].

3.6.3 Hybrid

A hybrid processing architecture shares the computations over multiple devices or services. It may train the model in the cloud using a SaaS, then download the model and perform the prediction locally on the hub using the remotely trained model. This architecture allows more flexibility in terms of computational requirements than what is required in the local processing architecture. This is due to the heavy computations being performed off-site. Generally, a hybrid processing architecture inherits the advantages from both the cloud and local architectures, while mitigating the computational requirements of the hub in the local architecture.

4 Analysis

The purpose of this chapter is to analyze how some of the topics investigated in chapter 3 can help to answer the problem statement. The results of the analysis will provide a requirement specification for a prototype, which then will be designed and implemented in later chapters. A high level diagram will be shown to give an overview of what a prototype for this project will consist of. Three scenarios will be presented in order to create use cases based on the functionality presented in the scenarios. The use cases will be used to define the required services that the system will consist of. These services are then used to define key elements needed for the prototype. Thereafter, requirement specifications will be made based on what has been analyzed in this chapter.

The investigation of the different subjects presented in the previous chapter has provided a general idea of which components will be needed for a prototype. Figure 4.1 is a visualization of the proposed system. As presented in section 3.4, figure 3.2 presents how a HEMS should be implemented in order to take the entire environment into consideration. A focus will be put on the development of a HEMS which takes several of the mentioned aspects discussed in 3 into account. In figure 4.1 a smartphone application allows the user to monitor and control the smart devices. Furthermore, the hub communicates with a server that fetches electricity prices from a third party source and stores these in a database. The hub should be compatible with smart devices and regularly collect data from these to store in a database. This data will be used to train machine learning models in order to automatically reduce the energy spent in a household.

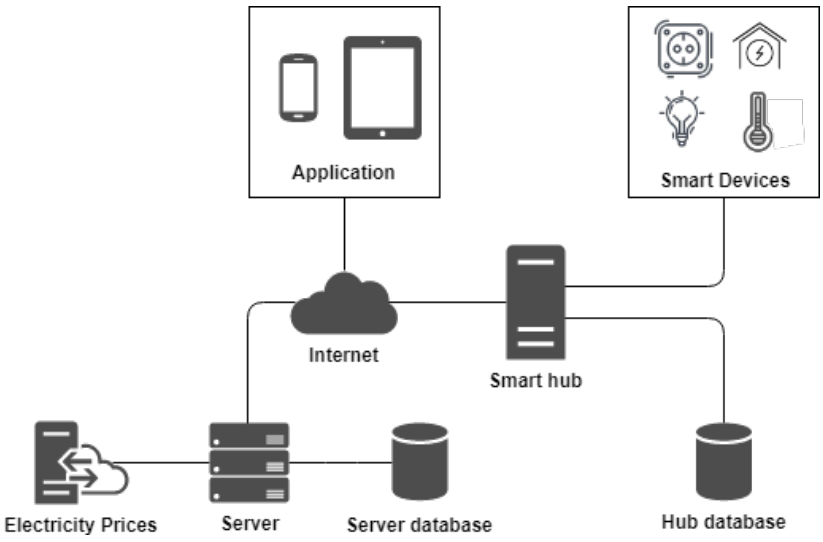


Figure 4.1: The components in the proposed system and how they communicate with each other.

4.1 Scenarios

Three scenarios will be used to present the system. Each scenario will be followed by a high-level sequence diagram that explains how the various actors are involved in the given scenario. Before each scenario their purpose and the involved participants will be presented. Furthermore, any equipment that is assumed that the individuals own will also be presented.

4.1.1 Scenario 1 - Existing Solutions

This scenario will be used to describe how existing consumer solutions manages energy consumption and how they take user presence into account.

- **Purpose:** Scenario that describes a system which helps the residents use less energy
- **Individuals:** Alice and her two kids
- **Assumed Equipment:** A smartphone, smart thermostats, and a smart hub.

”Alice usually works from 09:00-17:00 from Monday to Friday. She lives in a 3 room apartment, 20 minutes away from her work place. Alice has to drop off her kids at their school before she can go to her work. Alice wakes up at 06:00 to shower and wakes up her two kids afterwards. Alice checks how the temperature of the apartment is as she finds it to be a little cold. According to the heating

system that Alice has bought and installed in the apartment, it is 21.5° Celsius. After eating breakfast and getting ready, Alice and her two kids leave the home. During the day Alice receives a phone call from her kids' school, where she is told that one of her kids is feeling sick and needs to go home. Alice picks up the kid and they head home. When they arrive at the apartment it is cold as the heating system could not foresee her coming home early, as it heats based on a schedule set by Alice to fit her and her kids' daily schedule. In addition, Alice realized that the light in the kids room is turned on and has been so throughout the day. Alice turns up the heating through the schedule as the system indicates that the temperature of the apartment is currently 18.5° Celsius. The next day Alice's kid feels better and is ready to go to school again. As Alice arrives to her workplace, she remembers that she has to change the schedule again today, because if she does not, the system would start heating the apartment a couple of hours before she arrives home. She changes the schedule through the mobile application on her smartphone."

As presented in the scenario above, Alice changes the degrees through the mobile applications interface. In figure 4.2, a sequence diagram is presented. The sequence diagram shows the interactions with the system. Alice starts by inputting changes to the system through the application. The hub acknowledges the request and provides the user with a successful notification through the application. The loop in figure 4.2 depicts the process the system iterates through several times, as it needs to check if the actual time is equal to the scheduled time, where the system is supposed to start heating the apartment or household.

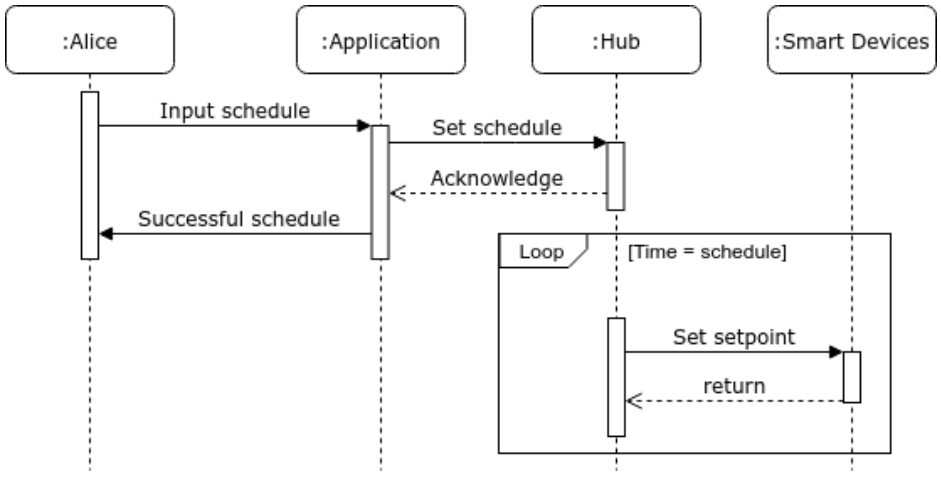


Figure 4.2: Sequence diagram of the different actors in Scenario 1 - Existing Solutions

4.1.2 Scenario 2 - Basic System Usage and Data Gathering

The previous scenario discussed how current consumer solutions handles energy consumption while the residents are away. This scenario will discuss the first half of the proposed solution, which includes a reasoning for using the system, setting it up and a period where it gathers data.

- **Purpose:** To provide insight into how the system registration process works, how it is setup, and how it gathers data.
- **Individuals:** Bob, his three kids and his wife.
- **Assumed Equipment:** A smartphone with an Internet connection, smart plugs, and smart radiator valves.

> ”Bob lives in a house along with his wife and their three kids. Recently Bob has noticed that their electricity and heating bills have increased. Wanting to reduce them, Bob invests in the proposed system which includes a hub, smart plugs, smart radiator thermostats and a smartphone application. After plugging in the hub it tells him to download the smartphone application and use it to scan the QR code on the outside of the hub. This allows the hub and the application to establish a connection through a remote server. He uses the smart plugs for computers and TVs, installs smart thermostats to all radiators and downloads the application to the rest of the smartphones in the household. With the application he can assign smart thermostats and smart plugs to individual users and rooms so a device, e.g. a smart thermostat, is only turned on if the owner of that room is home. After a day has passed, Bob can see how much energy that has been used throughout the day in the application, and see the current electricity prices. Bob has to wash some clothes before he goes to bed, and as he starts the washing machine his smartphone notifies him the current electricity prices being high. The system detects that the Bob can save some money if he waits an hour to start the machine, which causes the system to prompt him with suggestion that the system automatically starts the machine in an hour. Bob accepts and the system starts the washing machine an hour later. After three weeks the system notifies Bob that enough data has been gathered for the machine learning prediction to start taking effect.”

As discussed in the scenario Bob first sets up the system by scanning the QR code on the outside of the hub using the application he downloaded to his smartphone. This allows

the hub and the smartphone application to communicate together. This in turn starts the hub logging data from any connected smart devices and storing it in its database. This is illustrated in figure 4.3.

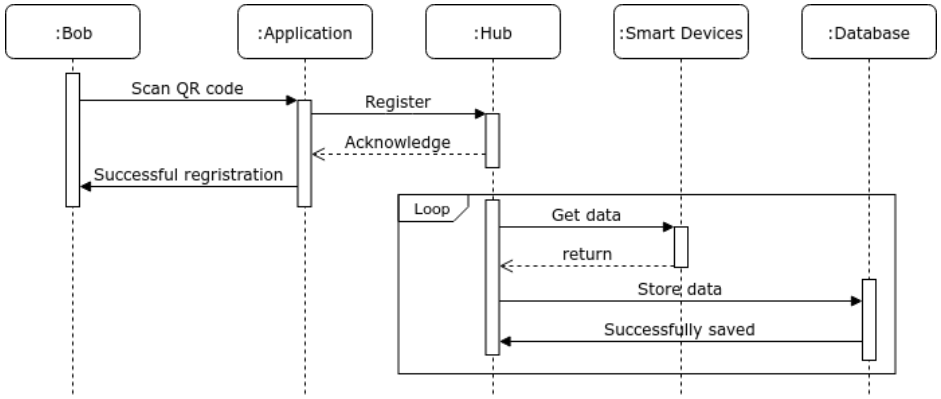


Figure 4.3: Sequence diagram of the different actors in Scenario 2 - Basic System Usage and Data Gathering during setup and data collection.

After he has registered the app, it allows him to monitor the energy consumption. This includes how much energy the house is currently using as well as historical data. In addition, the application also allows him to control the house. For example, he may notice that a radiator is currently heating, however no one from the household is home, the application then allows him to turn off the radiator. This is illustrated in figure 4.4.

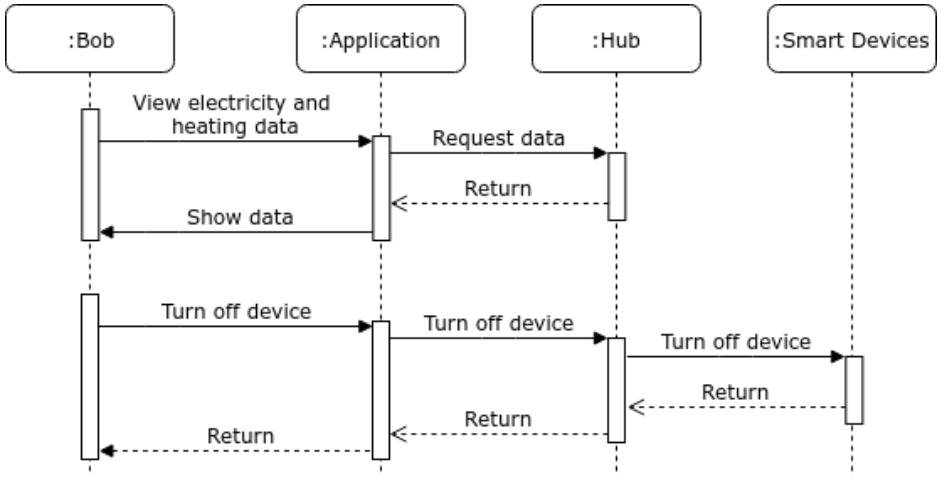


Figure 4.4: Sequence diagram of the different actors in Scenario 2 - Basic System Usage and Data Gathering during system usage.

4.1.3 Scenario 3 - Applied Machine Learning

The two previous scenarios have been presented based on how some of the current solutions work and how data can be gathered in order to explore another scenario that makes use of machine learning. The idea behind the third scenario is to intelligently control some of the devices connected to a user's home, based on the user's behavioural patterns.

- **Purpose:** Scenario where machine learning has been applied to the system.
- **Individuals:** Charlie.
- **Assumed Equipment:** A hub with connected smart devices, i.e. lights, thermostats and smart plugs.

> "Charlie wakes up at 07:00 and turns on the lights as he has to get ready for work. He goes to the kitchen to make breakfast and brew coffee. Charlie turns on the TV since he likes watching the news while eating breakfast. Charlie picks up his smartphone and checks if he has received any mails regarding his work. While holding the smartphone, Charlie enters the application that allows him to view data about heat and electricity consumption in his apartment. Charlie can choose to view the data in different intervals, such as hourly, daily, weekly, monthly, and yearly. It is also possible for Charlie to control the connected smart devices, and see information about them in the application. Charlie can see that his smart thermostats started to heat up the apartment 30 minutes before his alarm clock woke him. Charlie finishes breakfast and puts the TV on standby mode. Charlie has to be at work at 09:00, so he turns off the lights and leaves for work. Many of the electronic devices Charlie uses are not always turned off completely. Some of them, like the TV, are only in a standby mode, which still uses power. The HEMS that Charlie bought, in order to help him reduce his energy consumption and leave a smaller carbon dioxide footprint, intelligently detects that Charlie is not at home. The system therefore automatically turns off all the electronic devices and turns down the heat on the radiator through the smart thermostat in order to save energy. Charlie has a quite static work schedule as he usually stops working at 17:00. The system predicts that Charlie will be home at 18:00 and prematurely starts heating the apartment to Charlies desired temperature setpoint."

Figure 4.5 is a sequence diagram that illustrates parts of the system mentioned in scenario 3. The hub could e.g. be programmed to check if anyone was home every 30 minutes. The hub

would need to gather current data such as the status of lights and whether energy usage is detected through smart plugs. The hub will then run this data through a machine learning model in order to receive a prediction on whether the user is at home or away. In the above scenario, it predicts that Charlie is away, since he left for work, and the hub therefore turns off devices connected to the smart plugs.

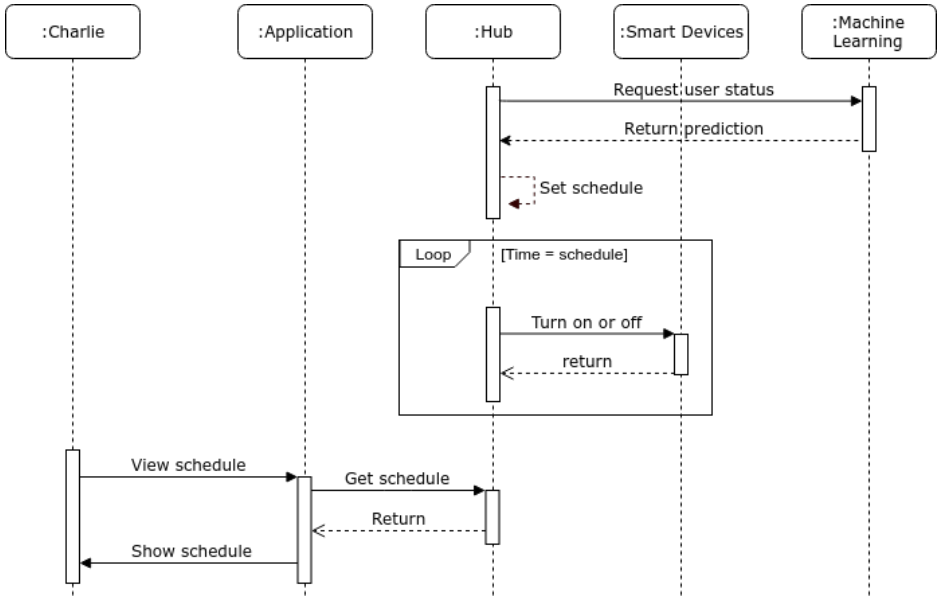


Figure 4.5: Sequence diagram of the different actors in Scenario 3 - Applied Machine Learning.

4.2 Use Cases

Based on the two latter scenarios it is possible to extract use cases to define the required functionality of the proposed system and the user interaction. All user interaction with the system will be handled through the smartphone application, this includes registering the application to the hub, registering and controlling smart devices and seeing historical energy consumption.

First, each entity will have a system context diagram presented to show their relevant terminators and how they communicate together. The terminators in the system context diagrams are derived from figure 4.1. These will be used to further define use cases for each of the entities. These use case diagrams will be used to further define the services the system should provide for the user.

4.2.1 Hub

Based on scenario 2 in section 4.1.2 and scenario 3 in section 4.1.3 it is possible to design a system context diagram for the hub. The diagram can be seen in figure 4.6. The system context diagram for the hub has four terminators. Each of these either provide or request data from the hub.

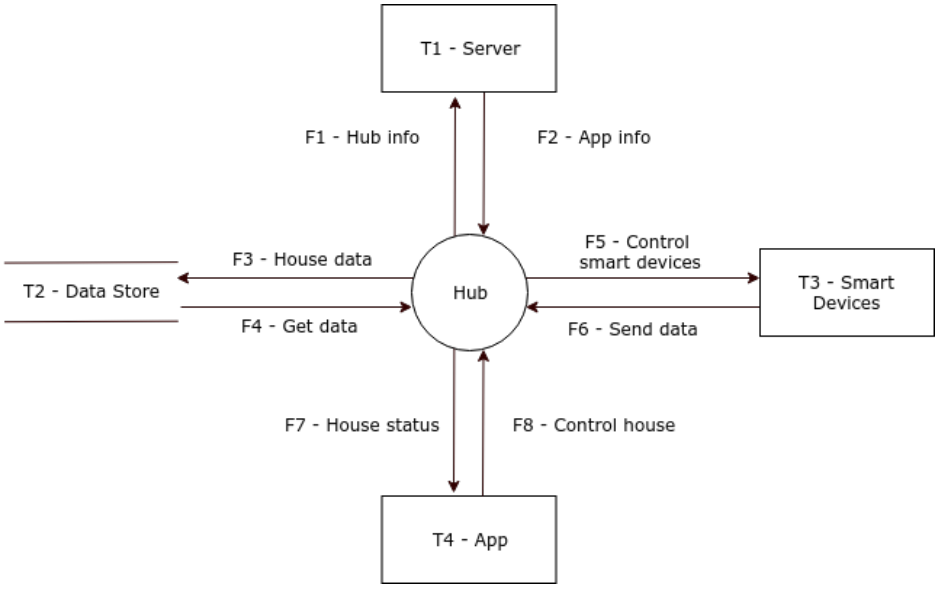


Figure 4.6: System context diagram for the hub and its four terminators; the application, its data store, the smart devices and the server.

The terminators and the flows between them and the hub are described in table 4.1 and table 4.2. Each of the terminators in figure 4.6 can be seen in 4.1. In addition, there are 8 flows between the hub in the middle and the four terminators. The flows describe data exchanges between the system and the terminators.

Table 4.1: An overview of the terminators in the system context diagram presented in figure 4.6

No.	Terminator	Description
T1	Server	Facilitates communication between the hubs and applications.
T2	Data Store	Contains data from the smart devices to be used for machine learning training.
T3	Smart Devices	Devices that collect data about the house they are put in, e.g. smart plugs, smart thermostats, etc.
T4	Application	Allows the user to monitor and control their house.

Table 4.2: An overview of the data flows in the system context diagram presented in figure 4.6

No.	Flow	Description
F1	Hub info	Information about the hub that is needed by the application.
F2	App info	Information about how the hub can communicate with the application.
F3	House data	Saves the data from the smart devices and user status to the data store.
F4	Get data	Request house data for machine learning and house monitoring.
F5	Control smart devices	Send commands to the smart devices.
F6	Send data	Get data from the data store.
F7	House status	Send data to the application to allow the user to monitor their house.
F8	Control house	Receive commands from the application to change the state of one or more smart devices.

Based on the terminators and flows in the system context diagram in figure 4.6 it is possible to define relevant use cases that the hub should provide for other entities. First, the hub should be able to register smart devices to itself. This will allow the application to monitor the status of the smart devices, i.e. to see if they are turned on or off and to see their energy usage in terms of e.g. Watts used in electricity or the setpoint of smart thermostats. In

addition, the application should also be able to control the smart devices. This use case will also allow the automated part of the system to turn off devices whenever possible.

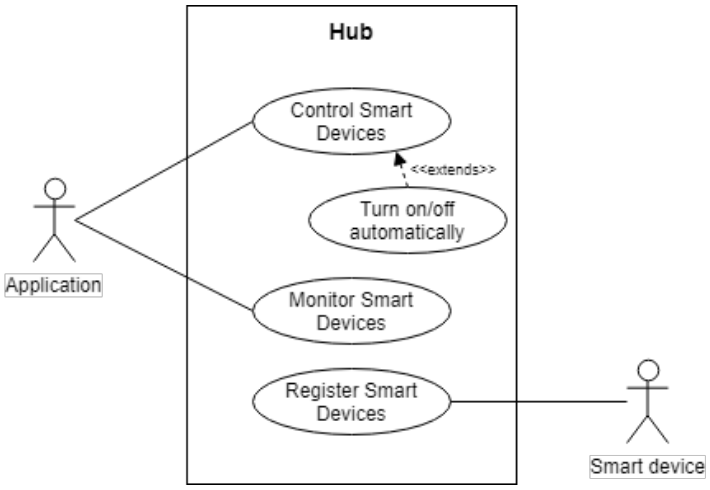


Figure 4.7: Use case diagram of how the application interacts with the hub.

4.2.2 Application

While the application is the primary way for the user to interact with the system, it only has two terminators in its system context diagram. First, it communicates with the server in order to establish a connection to the hub. With this connection it can request to see the status of the smart devices within the house and control them. Figure 4.8 shows the system context diagram for the application.

Figure 4.8: System context diagram for the application and its two terminators; the hub and the server.

The tables 4.3 and 4.4 present the terminators and flows from figure 4.8 along with a description of each. As there is a general overlap between figure 4.6 and figure 4.8, many of the flows and terminators refer back the tables relevant to the hub.

Table 4.3: An overview of the terminators in the system context diagram presented in figure 4.8

No.	Terminator	Description
T1	Hub	Contains endpoints which allows the application to control the connected smart devices
T2	Server	Same as T1 in table 4.1

Table 4.4: An overview of the data flows in the system context diagram presented in figure 4.8

No.	Flow	Description
F1	Control house	Same as F8 in table 4.2
F2	House status	Same as F7 in table 4.2
F3	Register app	Application information sent to the server when registering the application to a hub.
F4	Hub info	Same as F1 in table 4.2

An accommodating use case diagram can be made for the system context diagram seen in figure 4.8. The use cases in figure 4.9 have all been extracted from the scenarios described in section 4.1, however, a majority of them are based on scenario 3 in section 4.1.3. The use cases are all ways that the user can interact with the system through the application, as such the only actor in the use case diagram is the user. The user has the ability to control the smart devices connected to the system. As information about the various smart devices registered to the hub are routinely collected, the user is able to see a visualization of the energy usage. Furthermore, the application can be registered to the hub by scanning the QR code physically located on the hub. As the system presented in scenario 3 predicts when the user wakes up, leaves for work, comes home from work, and goes to bed, the user should be able to view and edit the predicted schedule the system provides. As the system also uses changes in electricity prices in order to help them reduce their energy consumption, the user should be able to view the current and forecasted electricity prices. These prices are also used to alert the user in case they increase their electricity consumption during expensive hours as discussed in section 3.1.

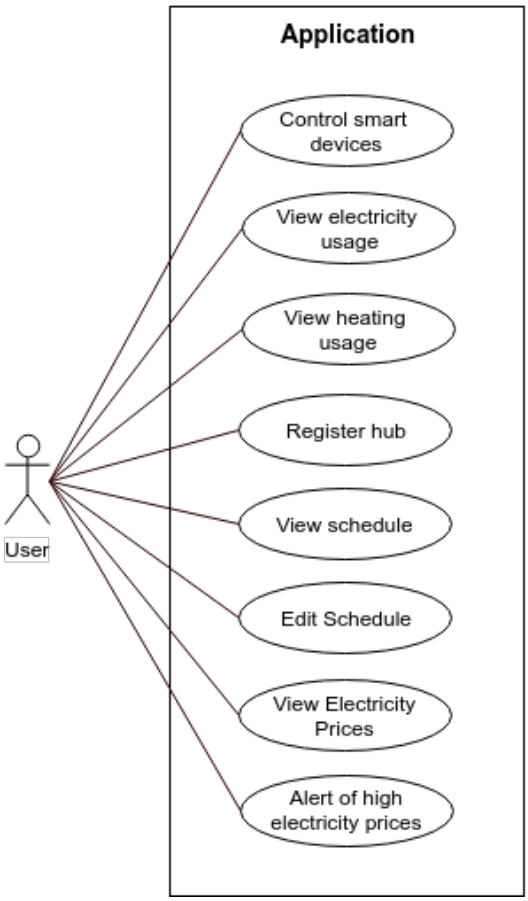


Figure 4.9: Use case diagram of how the user interacts with smartphone application.

4.2.3 Server

As previously mentioned in section 4.2.1 the server handles establishing the communication between the various components of the system. This includes registering the application to the hub. Furthermore, the server fetches information from a terminator called Electricity Prices, these will be used to warn the residents of high electricity usage during expensive hours as presented in section 4.1.2. In the future it will be possible to substitute the API that offers electricity prices with a smart meter as discussed in section 3.4. Figure 4.10 shows the system context diagram for the server. Similarly to the hub, the server have four terminators, though only with seven flows.

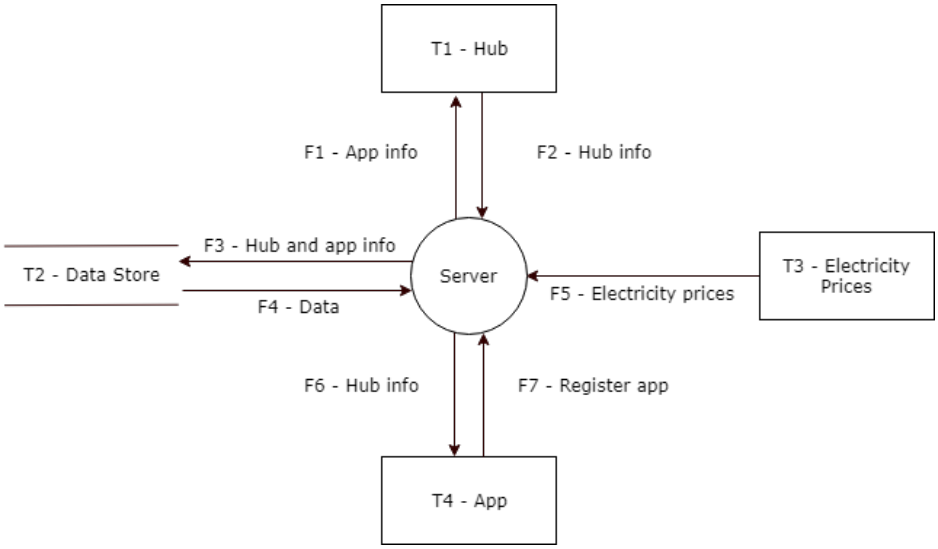


Figure 4.10: System context diagram for the server and its four terminators; the hub, its data store, the application and the source for electricity prices.

Table 4.5 presents the terminators from figure 4.10 alongside a description of their purpose. Afterwards, table 4.6 gives an overview of the data flows between the server and the terminators.

Table 4.5: An overview of the terminators in the system context diagram presented in figure 4.10

No.	Terminator	Description
T1	Hub	Contains endpoints which allow the server to register hubs.
T2	Data Store	The data store allows the server to store information about hubs and apps.
T3	Electricity Prices	Contains daily electricity prices
T4	App	Same as T4 in table 4.1

Table 4.6: An overview of the data flows in the system context diagram presented in figure 4.10

No.	Flow	Description
F1	App info	Same as F2 in table 4.2
F2	Hub info	Same as F1 in table 4.2
F3	Hub and app info	Information about both the hub and application is stored in a data store.
F4	Data	The server can get information about the hub and application from the data store.
F5	Electricity prices	Information about electricity prices
F6	Hub info	Same as F1 in table 4.2
F7	Register app	Same as F4 in table 4.4

The use cases derived from the system context diagram in figure 4.10 can be seen in figure 4.11. As the server is only a communication medium between two entities it does not have many use cases. First, the server allows both the hub and application to register their information to it. Furthermore, it also allows hubs to request the current electricity prices. This is to avoid having the hubs themselves to perform an API request at the electricity price service. The server will do it once and will allow a copy of the data to be sent by the hubs.

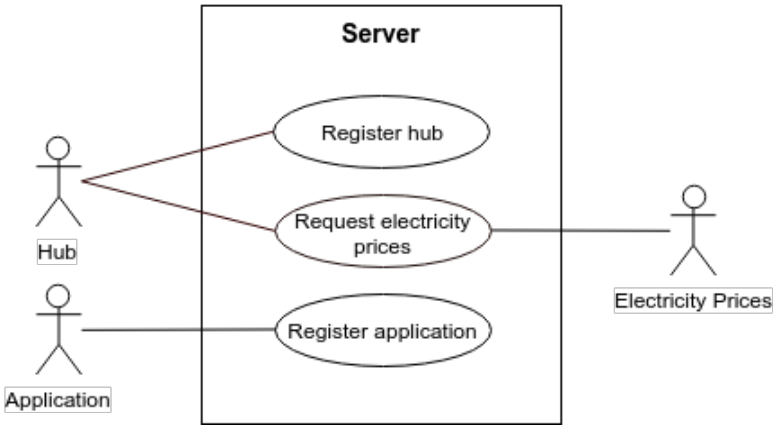


Figure 4.11: Use case diagram of how the hub and application interacts with the server.

4.3 Services

In this section the services that the system should provide are defined. The majority of the services can be seen in scenario 3, where some were more defined in the use case diagrams presented in figure 4.7, 4.9, and 4.11. There are a total of five different services which the system provides. Each of the services are focused on solving the problem statement and sub-questions defined in section 1.2. These services provide useful information for the user, while empowering them and giving them control over their energy usage. After each service the key elements required for them to function will be described. The key elements will then be presented and analyzed in order to extract requirements from them.

4.3.1 Service 1 - Control & Monitor Smart Devices

The currently existing products mentioned in section 3.3 all have a smartphone application that allows the users to control and check the status of the products. The scenarios presented in section 4.1 also require a way to control and monitor the smart devices connected to the hub. While this is not a novel feature to add, it is essential in order to allow users to not only control their home, but also overwrite any decision made by the system. Further, it is necessary to collect data as the intention is to automate tasks using machine learning. This data can be gathered from smart devices and from e.g. the user's smartphone. It can then be used to find patterns in the user's tasks. In addition, if it is required for the application to report the home and away status of the user, then the architecture has to be able to support it. This means supporting one or more technologies from section 3.5. This service will be used to define requirements that will help answering sub-question 2 and 3 in section 1.2.

As several different technologies can be used to setup personal area networks, it is important that the proposed solution supports these. The technologies include, but are not necessarily limited to, Z-Wave, Zigbee and Bluetooth [82]. This allows the proposed solution to work and operate with a wide range of devices and use cases, such as those presented in figure 3.2 in section 3.4.

Based on the two previous paragraphs it is possible to define key elements that the service requires. These include getting data from the smart devices, getting the user's home/away status and a processing architecture to support it. The key elements required are the following.

- Smart electricity and heating devices
- User home/away status
- Processing architecture

4.3.2 Service 2 - Adaptive Heating

In section 3.3 several solutions are presented, and are currently available for consumers. The Danfoss Link CC, Netatmo, and Tado focus on reducing the amount of energy spent on heating, whereas Samsung SmartThings is a more general system, which can incorporate different smart devices.

In table 4.7 each of the systems are being evaluated based on Openness, Smart Heating, Smart Electricity, Machine Learning, and Presence detection. Each of these features have been selected based on what has been presented in section 3.1, 3.2, and 3.4. In order for the system to fit the definition of a HEMS, it needs to be smart, connected, allow user control, available and accessible. As Danfoss Link CC, Netatmo, Tado, and SmartThings all allow the user control, and provides availability and accessibility these are not shown in table 4.7.

Table 4.7: Comparison of the existing heating solutions presented in section 3.3.

	Open	Smart Heating	Smart Electricity	Machine Learning	Presence Detection
Danfoss Link CC	✗	✓	✗	✗	✗
Netatmo	✗	✓	✗	(✓)	✗
Tado	✗	✓	✗	(✓)	✓
Samsung SmartThings	(✓)	✓	✓	(✓)	(✓)

As it can be seen in table 4.7 none of the systems are open, meaning that if you buy the smart hub or gateway, you can only use smart devices that are compatible with it, which in the most cases are the manufacturers own devices. Instead of creating a closed system, where only specific devices can be added to the eco-system, it should be open as it allows the user to freely choose between the smart devices that are available on the market.

All of the presented solutions from section 3.3 provide smart heating of some sort. Danfoss Link CC and Netatmo only provide a static schedule for the user, where Tado also utilizes the user's location through their smartphone to see if the users have arrived home before the scheduled time. This is useful as it overcomes one of the problems with using a static schedule. However, a problem with detecting user presence based on their geolocation is that they may be reported as being home, while they are not, e.g. visiting their neighbor. This is due to the accuracy limitations of the GPS.

When it comes to machine learning, Danfoss Link CC does not utilize any algorithms in order to make predictions for the user as it is purely rule-based using the static schedule set by the residents. Netatmo might use machine learning in order to decide when the thermostat should start heating up the apartment. Tado also uses outdoor temperature and weather forecasts in order to use the natural heat from the outdoor environment, which helps reduce the amount of energy spent on heating up the household. However, neither Tado nor Netatmo discloses any information regarding their usage of machine learning. One of the downside to using machine learning is that the system would require a certain amount of data in order to make reliable and usable predictions.

As already mentioned, Tado is the only solution that uses presence detection. Presence detection is very useful for such a solution, as they can detect if the residents are home. Tado uses geolocation in order to track the users, however, presence detection sensors that can be placed inside the household are also available. Presence detection can also be a valuable feature for a machine learning algorithm. Even if presence detection technologies are useful for such solutions, the user should still be taken into consideration.

Furthermore, as presented in section 3.2 there are several good heating practices that the system could benefit from. None of the systems presented in section 3.3 and table 4.7 uses the first (distribute the heating load over multiple radiators) or second practice (reduce the temperature of the return water) presented in section 3.2. If the system should be able to check the returning water in the radiator, a temperature sensor could be used and placed on the pipe where the returning water runs through. The system could make use of window and door sensors in order to detect if the doors and windows are open. Based on this, the system could tell the user to close the door to rooms which are not used, and the system could benefit from the windows sensors telling the smart thermostats that they should stop heating if a window is opened.

This service has three key elements. First, it has to be able to control the smart devices connected to the smart hub. Second, the system has to be able to predict the user's schedule, this includes when they leave for work in the morning, get home in the afternoon and when they go to bed in the evening. This allows the system to automatically turn down the heating while the user is either away from home or asleep. Third, the architecture regarding machine learning has to be considered, since this can require many resources. As such, the key elements are:

- Smart electricity and heating devices
- Machine learning

- Processing architecture

4.3.3 Service 3 - Adaptive Electricity

Presented in section 3.3, two different smart devices which can help the user gain more control over the electricity usage in their household are presented, namely smart lighting and smart electricity plugs. All of the lighting from section 3.3 nearly provide the same amount of utility for the user. Each of them provide a schedule for the user to set, in order to automatically turn on and off the lights in the household. However, the same problem presented in the previous section is also relevant here, as the schedule is static. Hence the system does not know if the residents leave the household before scheduled time. In order to avoid this, Philips Hue and Ikea TRÅDFRI sells a PIR sensor separately, which can help prevent this. However, that means that the user needs to go out and spend more money on the system in order for the system to become somewhat intelligent.

Instead of buying a motion sensor it could be more beneficial to utilize one of the smart plugs presented in section 3.3.5. They provide control over the devices that are using the smart plug. The two solutions presented in section 3.3.5 can be controlled by the user through the smartphone, allowing them to completely turn off the power socket, powering down standby devices, which can consume a lot of energy, as presented in section 3.1. Furthermore, they provide an overview of the amount of energy used through the smart plugs.

The smart plugs could also help reduce the amount of electricity usage spent on entertainment, as this is one of the largest areas where the average danish family spent electricity on, which was seen in figure 3.1 in section 3.1. The smart plugs could be used together with the gaming console, TV or computer to turn them off if no one is using them. The smart plugs could also be connected with the appliances in the kitchen.

However, the functionality of the smart plugs are limited. They do not provide much automation for the users, as they still have to actively turn off the smart plug through the application. Instead, the process could be automated through the use of machine learning. If the system predicts that the user is not home, but the devices connected to the smart plug consume a lot of energy, it could prompt the user asking them if it should be turned off. In conjunction to this, the combination of smart lighting solutions and smart plugs could be used to predict whether or not the lights should be turned off as well. A full-fledged system should predict if the user is home based on electricity activities and some sort of user tracking, and notify the user if the smart devices should be turned off. With the aforementioned in mind, smart plugs and smart lighting will be used in the system.

The required key elements for this service are the same as the service in section 4.3.2. The required key elements are:

- Smart electricity and heating devices
- Machine learning
- Processing architecture

4.3.4 Service 4 - Energy Visualization

The system should support a combination of the three strategies presented in section 3.1. The information strategy can be personalized by showing information about the residents' energy usage. The system could display which appliances in each room that have a high energy usage as well as the total energy usage of the room. The aforementioned would utilize the informational and psychological strategy (see section 3.1) as it provides information, knowledge, and motivation for the user. As presented in section 1.1 there seems to be a general consensus that people are willing to change behaviour with regards to their energy usage if they can save money. The information and visualization of the energy usage in the household could enlighten, motivate, and help the residents to reduce consumed energy, as it can help them identify how possible savings can be made.

This service only have a single required key element. As it is about visualizing the energy consumption to the user, it requires the data to be collected from the smart devices. As such, the key element required is:

- Smart electricity and heating devices

4.3.5 Service 5 - Price notifier

Many of the topics presented in figure 3.2 are already present in the solutions presented in section 3.3, 4.3.2, and 4.3.2. However, one of the elements that these solutions do not take into consideration, is the upcoming smart meters that are to be implemented in every household in Denmark. In the future, every household in Denmark has a smart meter, which allows the Danish electricity suppliers to remotely read the amount of energy spent in a household. The old static energy prices are going to be changed to a price that changes based on e.g. the load in a specific set of time, as mentioned in section 3.2. The system presented in scenario 2 and 3 in section 4.1 takes the changing energy prices into consideration and indicates this to the user.

In a completely automated system, the user could input a time for when he needs certain

tasks to be done, e.g. the dishwasher or washing machine, and then the system should start the tasks autonomously, when the possibility to save the largest amount of money is present.

The data has to be collected from the smart plugs that measure the electricity consumption of appliances in the household. The relevant key element required for this service is therefore:

- Smart electricity and heating devices

4.4 Key Element 1 - Smart Electricity & Heating Devices

As mentioned throughout chapter 3, HEMS' and smart homes require smart devices that can communicate with the hub in order to be controlled. The scenarios presented in section 4.1 assume that the system supports smart devices such as lights, smart plugs, and smart thermostats. The devices have to be able to be controlled remotely by an application. The electrical devices used in this project will not directly have smart features, which is why smart plugs will be used.

Some of the research projects mentioned in 3.4 also use smart plugs and thermostats. Koehler et al. [50] proposed a way to reduce energy consumption by controlling thermostats through the use of eco-feedback or automation. They only mentioned thermostats, since they try to predict whether the user will be home in the next hour, because then they can adjust the thermostat accordingly. Kleiminger et al. [49] measured power consumption by using smart plugs. This was used to predict whether anyone was home or not. Our project will use both thermostats and smart plugs, since the problem statements regards both reducing electricity and heating usage. Lights are also included, because if the system detect that no one is home, then the lights could be turned off to reduce electricity usage.

4.5 Key Element 2 - User Home/Away Status

The first scenario presented in section 4.1 is based on how current heating solutions work today. These ask the user to create a schedule with specific timeslots for when the user is at home or away. The third scenario in section 4.1 is different in the way that the system intelligently can adjust itself based on the user's behaviour. The scenario reveals that the system can detect if the user is at home or not in order to turn lights and electronic devices on and off. It is also able to predict when the user arrives at home in order to adjust the temperature. This section contains an analysis on which techniques can be used to figure out if the user is at home. The techniques presented in chapter 3 section 3.5 will be evaluated with the purpose of choosing one of them for a prototype. Four techniques were

presented: geofencing, beacons, Wi-Fi, and PIR sensors. A focus will be put on advantages and disadvantages related to GPS, extra equipment, and privacy concerns for each technique.

Table 4.8: Comparison of the different methods to detect if the user is home or not.

	Geofencing	Beacons	Wi-Fi	PIR
GPS	✓	✗	✗	✗
Requires extra equipment	✗	✓	(✓)	✓
Privacy concerns	High	Medium	Medium	Medium

Table 4.8 shows the results on the analysis of presence detection techniques, where each of them will be discussed below in relation to this specific project.

Geofencing would provide access to GPS, which might be beneficial for heating especially. Using geofencing would make it possible to detect when users were within a certain radius of their home. This relates to the way Tado thermostats work, which was presented in section 3.3.3. However, this could cause some privacy concerns for the users, as it would potentially be possible to make multiple geofences and analyze where the user is located outside of the home. An advantage for the user is that it does not require any extra equipment apart from a smartphone for the user. The user would of course need a smartphone in order to download the application. Another note regarding GPS precision is that the recommended radius of a geofence should be minimum 100 meters [83]. This means that using geofencing could provide false positives on whether the user really is at home or not. A user could be at a neighbors place, if they lived close enough.

As mentioned in section 3.5.2, BLE devices can be great for indoor tracking, since GPS is not always able to penetrate the walls in a building. It would not be possible to get the GPS position of the user using BLE devices as beacons, making it impossible to determine the users location outside the home. BLE beacons could be needed if it was important to know the user's exact location in his home, but this is not necessarily needed for this project, since the scenarios presented in section 4.1 only require the system to know if the user is home or not. We assume that using BLE beacons for detecting if users are home is less of a privacy concern compared to geofence, since BLE beacons would only reveal if users are home or not while Geofence also can position users outside their homes. However, BLE beacons would require physical beacons, which the users would have to buy. This makes it more expensive for the user, and it would be required for them to have the beacon on them at all times, such as in a keychain.

Wi-Fi can also be used for indoor tracking as long as there are multiple access points that can be used for triangulating the user's position. GPS is not an option with Wi-Fi, since it requires a router or access point to connect to. This is not a problem for our project, since we only need to know if the user is at home or not, not the exact position of the user. We found two ways to use Wi-Fi for detecting if the user is home. One is by using Wi-Fi Probe requests, which was presented in section 3.5.3, but this would potentially give us problems since the smartphones MAC address might get randomized. Another way is by leveraging the smartphone application, which was mentioned in the two first scenarios in section 4.1, by having the application check if the smartphone is connected to the user's Wi-Fi network. The application would need to check if the smartphone is connected to the user's network by looking at the name of the network (SSID). This is possible on Android applications [84], and it would not require any new equipment for the user, if we assume they have a Wi-Fi router. Wi-Fi can cause the same problem as geofence, e.g. the user is at the next door neighbor while still being connected to their own Wi-Fi network. Wi-Fi is less of a privacy concern compared to geofence, since it is only possible to get information about the user if he is close enough to the Wi-Fi network.

PIR sensors are localized and can be used to detect movement, which can reveal if a user is home. Kleiminger et al. [49] used PIR sensors for detecting occupancy, where they logged data from the sensors every second of the day. They stored the value from their PIR sensors as 0 (absence) or 1 (presence). If a PIR sensor was triggered, then the next 30 seconds would be logged as presence. However, using this method alone could cause problems, since user's can potentially be sitting still without triggering a PIR sensor, which also was mentioned in section 3.5.4. It might be a good idea to combine PIR sensors with another method in order to achieve more precision. A user would require to buy multiple PIR sensors for every room, which could get expensive. In regards to privacy, PIR sensors can reveal whenever there is activity in a sensor's view, but it might not seem as intrusive as other methods.

Based on the analysis of different presence detection techniques, it has been decided that Wi-Fi should be used to detect user presence in this project. Two methods using Wi-Fi were previously presented, where our focus will be on having a smartphone application check if it is connected to a specific SSID instead of using probe requests. This would allow us to detect if a user is currently connected to the home Wi-Fi and reveal if the user is at home. There are of course use cases where this alone might not be feasible, e.g. the user would not appear to be home if the Wi-Fi was disabled on the smartphone, or if the smartphone was turned off. Another disadvantage is that the home Wi-Fi had a reach all the way to the neighbor, which could make it look like the user was at home when he was actually is at the

his neighbor's. It would be a good idea to also use PIR sensor in combination with Wi-Fi, because then we would have more information to determine whether the user actually was home. However, we only are going to build a proof of concept prototype, which is why only Wi-Fi will be used for the prototype.

4.6 Key Element 3 - Processing Architecture

As discussed in section 3.6, the computations can generally be performed in three ways; in the cloud using an SaaS, locally on a device within the household or by using a hybrid between the two. In this section the architectures will be analyzed to make a decision on how the machine learning architecture should be designed. They will be analyzed based on following five factors; computational power and storage, data ownership, privacy and trust, and development and dependencies.

4.6.1 Computational Power & Storage

The first factor is how the processing architecture affects the hardware design choices of the hub. Sending the data collected to a cloud operator would void this concern, as the concern of adequate computational power and database space would be outsourced to the cloud operator. Performing the data processing locally on the hub means that the hardware has to be designed to be viable not only for today's use cases but also for the future's. This increases the requirements of the chipset within the hub to ensure that it can train a model based on several years worth of data. Further, it also increase the requirements for on-device storage, as it has to be able to store all the data collected from the devices.

Another possible way to perform computations is on another device within the household. Either by splitting the computations between the hub and the device or by having the other device perform the computations entirely. This would reduce the requirements for computational power of the hub, however, it would require extra equipment to share the computations with. As discussed in section 3.6.2, some smartphones today are being developed with extra chipsets which sole purpose is to increase the efficiency of on-device machine learning computations. These chipsets would allow splitting the computations between the hub and the user's smartphones by e.g. training the model on the smartphone, serializing it and sending it to the hub and then perform prediction on the hub. This would move the computational power and storage requirements from the hub to smartphone.

4.6.2 Data Ownership, Privacy & Trust

Another concern about using a cloud operator's SaaS for machine learning capabilities is the data ownership. This concern is even more relevant with the introduction of with the European Union's General Data Protection Regulation (GDPR) which took effect in May 2018 [85, p. 65]. GDPR is a regulation designed to protect the personal data of citizens within the European Union [85, p. 32]. In order to train a model on a remote service, said service has to have access to the user's data. In some cases this might not only mean transfer of the data, but also a transfer of ownership. This issue might make it difficult for the user to request their data deleted from the SaaS, as they are not the ones who sent their data to the SaaS. While the services discussed in section 3.6.1 all do not retain any ownership of the data [86, 87, 88, 89], it is an important consideration to make if a cloud service is to be used. As such, in terms of this concern, the most optimal processing architecture is performing it locally. This ensures that the data never leaves any of the user's devices, and is not shared with a third party such as a cloud operator and no intellectual property rights are surrounded.

In addition to data ownership, data privacy and trust is an important consideration to make when comparing cloud computing to local processing. Using a SaaS in the cloud requires the user to not only trust the service they are using, but also any external third-party that the given service uses. This is because using a SaaS in the cloud requires the service to share user data with any external third-party. As discussed briefly in section 3.6.1, some might not be willing to share their user data with companies such as Google who are known for making money of user data. While performing the processing locally does alleviate some privacy issues, it does not completely solve them. The user still have to trust that their data will only be stored locally and not sent to remote server. Storing the data locally allows the owner to control it themselves [72].

4.6.3 Development & Dependencies

The last two concerns of the processing architecture are the development of the processing algorithm and possible dependencies each method may bring. Generally, the benefits of using a cloud operator is the ease of implementation with the downsides of bringing in additional dependencies to the system. Relying on external entities could possibly include a single point of failure. The importance of network reliability increases by relying on a cloud operator. This is important for both the local hub requesting a computation performed as well as the internet connection to the cloud operator. If either of the two is offline then the machine learning capabilities may be limited or in some implementations, e.g. both

training and prediction is performed in the cloud, unavailable. In addition, changes that the cloud operators performs on the SaaS might impact how it was implemented, thus requiring additional time to continue operating. Performing the computations locally will reduce the impact of a device losing it's Internet connection, however, it puts further emphasis on the development team to properly and correctly implement the chosen algorithm.

4.6.4 Selected Architecture

As presented above, each architecture has their own strengths and weaknesses that may have an influence on the evaluation of the prototype and answer to the problem statement from section 1.2. The main benefits of performing the processing locally are the high data privacy as it never leaves the network and the possibility for customization of the implementation at the cost of higher requirements for the processing power and storage capabilities, which in turns means a more expensive hub. The advantages for performing it in the cloud is the ease of development, lesser requirements for the hub, a simpler architecture, but at the expense of added dependencies, a single point of failure and data sharing with a third party processor. Based on this it was decided to perform the processing in the cloud, more specifically to use an established SaaS.

4.7 Key Element 4 - Machine Learning

As presented in section 3.4, a lot of the comparisons that Aman et al. [46] made of existing designed HEMS indicated that the majority of them were not intelligent according to their definition. Furthermore, the solutions presented in section 3.3, which have been analyzed in section 4.3.2 and 4.3.3, does not necessarily use any type of machine learning. However, as presented in section 3.4 there are newer solutions that utilize machine learning to predict different factors such as whether or not anyone is home based on the power consumption, predicting the amount of energy produced by solar panels, and predicting if the residents of a home is home or away the next hour.

4.7.1 Supervised learning

In machine learning, an algorithm is trying to solve a problem related to supervised learning, unsupervised learning, or reinforcement learning. This project will only focus on supervised learning, since that what will be used for the prototype. For supervised learning it is necessary to use sample data that contains input data (the independent variable) and the desired output variable (the dependent variable), as this is needed for training a model [90]. The

algorithm relies on the input data, also called features, which are used when trying to get a prediction from the trained model. Depending on the problem you are trying to solve, a supervised learning approach can either be used to solve a *regression* or a *classification* problem [90]. Regression is used when the output is a continuous value while classification is when the output is a discrete value (i.e. a label) [90].

Scenario three in section 4.1 alongside the two services in section 4.3.2 and 4.3.3 all revealed that the system needs to automatically adjust the smart devices connected to the hub based on the user's pattern. Machine learning will utilize this to automatically turn on/off electrical devices and adjust thermostats in order to reduce energy consumption. As mentioned in scenario three in section 4.1.3, the system should be able to control devices based on whether the user is home, away, or sleeping. As presented in section 3.4, Kleiminger et al. [49], other researchers tried to predict whether the users were home or away based on the power consumption and PIR sensors. To label their data they used a tablet which the users had to interact with every time they left or came home. We do not think this is viable, because their research also showed that users forgot to use the tablet sometimes and that resulted in wrong data. Section 4.5 therefore concludes that we will use Wi-Fi on a user's smartphone to determine whether he is home or away. This data should thereafter be used to extract meaningful features for our machine learning algorithms.

4.7.2 Algorithms, Relevant Data & Features

Machine learning models are trained using training data and evaluated with test data to determine the accuracy of the model. It is possible to train multiple different models using different algorithms to solve the same problem, this is why it is hard to choose one single algorithm for a problem prior to training the models [90]. A way to choose an algorithm is to train multiple different models and choose the one that scores higher with the test data. Another challenging part of creating good models is choosing appropriate features for a model. This usually needs to be done in an early stage of a project, since it is necessary to know what data should be gathered [90]. Choosing the right features can require a domain expert within the field of the problem you are trying to solve, because the independent variable has to be descriptive of the dependent variable and thus be relevant for the problem at hand.

Time is a feature we deem necessary for our problem, as we have to predict when the user comes home, leaves home, or is asleep. There is hopefully a pattern for what time users e.g. go to work and come home. *Weekday* is also relevant, because the users might have a somewhat static work schedule from Monday to Friday, and they might have weekdays where they work

from home or leave for work later than other days. The weekday is therefore presumed to be useful. Section 3.4 introduced Shahriar and Rahman's [48] research, where time and day of week was used for predicting which device or appliance was using energy. As mentioned in section 4.7.1, data from the smart plugs can be used can reveal if a user is currently at home using any kind of electronic equipment, e.g. a computer or a TV. Kleiminger et al. [49] used power consumption as a feature when predicting whether users were at home. Figure 3.1 in section 3.1 showed that 39% of average electricity usage for a Danish family is from entertainment, the power consumption of these devices may reveal if the user e.g. currently is at home using a computer or watching TV. The state of smart lights will also be considered as a feature since we assume that people attempt to turn off the lights when they leave their home or go to bed, and turn them on when they wake up or come home from work.

Setpoint temperature on thermostats may also be used as data for the model as we assume that users turn down the heat when leaving home for work and then turn it back up when they come home again. Some might also turn down the heat of the radiator before they go to sleep. The setpoint temperature might therefore follow a pattern that can be relevant for an machine learning algorithm. However, as discussed in section 3.3 there are many products that advertise automatically large savings in heating bills because people may not turn down the temperature on the radiator while they are out of the house or asleep. If the users does not change the setpoint of the thermostat regularly during data gathering, then this data may not have large enough variance to be considered informative for the model, as the setpoint might always be set on e.g. 22° Celsius.

4.7.3 Prediction Goals

Two different ways of using machine learning for this project has been identified. As such, two different models will be trained and evaluated in order to find the an optimal solution to reduce the energy consumption of the household. In order to do so, the system should automatically turn off devices when possible. Using a classification model it may be possible to find the user's current status, e.g. are they at home, asleep or away from the home. By performing a prediction every time the smart hub samples data it is possible to predict the user's status with the same resolution as the data sampling. However, doing so might require a lot of computations. If data sampling was performed once every minute it would require 1440 predictions a day. To avoid this, predicting the user status should be limited to certain times a day, e.g. once every 30 minutes. However, as the identified times to automatically reduce the energy consumption is when the user is away from the home or asleep, it may be beneficial to limit predicting the user's status to timeslots around these transitions, i.e. when

they wake up in the morning, leave for work, comes back home and go to bed. Calculating these timeslots could be done in different ways, e.g. taking the average time the user leaves their home and start performing regularly predicting an hour before to see if they have left the home. However, this method may be sensitive to outliers. Another way could be to use a regression algorithm as discussed in section 4.7.1. The regression algorithm should predict at what time during the day the user might change their status. Doing so would require two machine learning models, a regression algorithm to forecast a timeslot when the user's status might change and a classification algorithms for real-time predictions to see if the status change has happened.

As presented in section 3.3.1, Danfoss' solution has a static scedule, where users manually input when they are home or away, while section 3.4 presented how researchers tried to figure out how to detect if users were home or away. The goals in this project is to avoid such a static schedule and use machine learning to figure out when the users is home or away. More specifically, it would be beneficial to know when the user leaves home, comes home, goes to sleep, and wakes up, because we assume that the timespan between the timestamps of these actions could be used to reduce energy consumption.

The first model should attempt to forecast at what time the user changes their status, e.g. when do they wake up in the morning or when they come home from work in the afternoon. This will allow the model to forecast the user's daily schedule. A disadvantage of this model is that it may require extensive data preprocessing. This is to extract features from the collected data to train the model with. From this point on in the report this model will be referred to as the *Forecasting Model*. Based on this, four status changes that should be predicted have been identified.

1. **Asleep to awake:** When the user wakes up in the morning to prepare for work.
2. **Awake to left for work (away):** When is leaving their home for work.
3. **Away to arrived home:** When the user arrives back home after work in the afternoon.
4. **Home to went to sleep:** When the user goes to bed in the evening.

The second model should attempt to predict what the user is currently doing in real-time rather than trying to forecast their status changes. Based on the statuses defined for the Forecasting Model, this model should continuously attempt to predict what the user is currently doing, i.e. home, away or asleep. From this point on in the report this model will be referred to as the *Real-time Model*. The labels that should be predicted are the following.

1. **Home:** Is the user currently at home?
2. **Away:** Is the user currently away?
3. **Asleep:** Is the user currently asleep?

Using the Forecasting Model and the Real-time Model together may be an efficient way to predict when the system should automatically turn off the smart devices and reduce the setpoint of the thermostats. It is only required to perform a prediction using the Forecasting Model once a day. Using this predicting the system knows when it has to start performing predictions using the Real-time Model.

4.8 Requirement Specifications

With the various aspects of the problem statement and it's sub-questions examined in chapter 3 and further discussed in this chapter, it is possible to specify the requirements for the system. The requirements for a system concerns the goals, objectives, features and constrains that the end product should have [91]. They do this by describing the parts of the system in detail alongside a motivation for the requirement. The requirements will be prioritized using the MoSCoW Method. MoSCoW is an acronym for **M**ust have, **S**hould have, **C**ould have and **W**on't have [92]. The method allows product owners, project managers, etc. to assign an importance to each requirement using the predefined schema. A major advantage of the MoSCoW method is the flexibility. It allows the development team to move onto requirements prioritized with "should" after all the "must" requirements have been implemented. For this project only requirements prioritized with "must" will be implemented to develop a minimum viable product for the evaluation. The requirements prioritized with "must" will be used to define the requirements of a prototype that can be used to answer the problem statement in section 1.2. Table 4.9 shows how the four priorities in the MoSCoW method and how they are interpreted in this project. Afterwards, in section 4.8.1, 4.8.2, 4.8.3, and 4.8.4 are the requirements for the hub, application, server and cloud respectively shown. Functional requirements will be referred to as FR and non-functional as NFR.

Table 4.9: Descriptions of the four priority categories and their usage in this project.

Priority	**Description**
Must	Required to be implemented for the evaluation.
Should	Not required. May be implemented, however, wont affect the evaluation.
Could	Not required. May be implemented if resources permits it, however wont affect the evaluation.
Won't	Will not be implemented.

4.8.1 Hub

Table 4.10: Table of hub's the functional requirements

No.	**Requirement**	**Motivation**	**Priority**
Hub_FR1	The hub has to be able to manually turn on/off the smart lights, smart plugs, & smart thermostats.	Needed in order to provide control for the user. See section 4.1.3 and 4.2.2.	Must
Hub_FR2	The hub has to be able to manually adjust the setpoint on the smart thermostats.	Needed in order to provide control for the user. See section 4.1.3 and 4.2.2.	Must
Hub_FR3	The hub has to be able to detect electricity usage higher than the threshold defined in Hub_NFR4.	In order to notify the user of high electricity usage. See section 4.1.3.	Must
Hub_FR4	When the hub detects high electricity usage based on Hub_FR3 it has to be able to notify the application.	In order to notify the user of high electricity usage. See section 4.1.3.	Must
Hub_FR5	The hub has to be able to request a prediction from the trained model in the cloud-based environment.	Needed in order to provide predictions for the user. See section 4.1.3 and 4.2.1.	Must

Continued from previous page

No.	Requirement	Motivation	Priority
Hub_FR6	The hub has to be able to collect data (on or off, setpoint temperature, power consumption) from the connected smart devices.	Used in the machine learning algorithm as features. See section 4.2.1.	Must
Hub_FR7	The hub has to be able to recognize a holiday.	In order to treat the days differently. See section 4.1.3.	Could
Hub_FR8	A PIR sensor has to be used to detect user presence.	In order to be more precise, when collecting information about the users home/away status. See section 4.5.	Could

Table 4.11: Table of the hub's non-functional requirements

No.	Requirement	Motivation	Priority
Hub_NFR1	The hub has to be able to connect to the Internet	In order to provide remote communication between the hub and the application. See section 4.1.2 and 4.1.3	Must
Hub_NFR2	The hub has to be able to support Z-Wave smart devices	In order to be compatible with Z-wave smart devices, it should communicate through Z-wave. See section 4.1.3.	Must
Hub_NFR3	The hub has to be able to fetch electrical prices from the server	In order to mimic smart meters. See section 4.9	Must

Continued from previous page

No.	**Requirement**	**Motivation**	**Priority**
Hub_NFR4	The threshold for high electricity usage has to be calculated as a percentage of the average power consumption when the user is home. The percentage is defined in App_FR12.	In order to support Hub_FR4	Must
Hub_NFR5	The hub has to be able to collect time, date and weekday.	Used in the machine learning algorithm as features. See section 4.2.1.	Must
Hub_NFR6	The hub has to be able to save data from Hub_NFR6 to a cloud database.	Needed in order to use the data later, for processing. See section 4.1.3.	Must
Hub_NFR7	The hub has to be able to receive the user home/away status from the application.	Used for the machine learning. See section 4.1.3 and 4.2.2.	Must
Hub_NFR8	The hub has to be able to support Zigbee smart devices	In order to allow the user to connect any device to the system. See section 4.1.3.	Should
Hub_NFR9	The hub has to be able to support Bluetooth smart devices	In order to allow the user to connect any device to the system. See section 4.1.3.	Should
Hub_NFR10	The hub has to be able to support Wi-Fi smart devices	In order to allow the user to connect any device to the system. See section 4.1.3.	Should
Hub_NFR11	The hub has to comply with the GDPR.	In order to avoid fines, compliance to GDPR is important. See section 4.6.	Should

4.8.2 Android Application

Table 4.12: Table of the application's functional requirements

No.	Requirement	Motivation	Priority
App_FR1	The user has to be able to monitor the status (on/off) of the smart lights and smart plugs, power consumption of smart plugs, and setpoint temperature of the smart thermostat.	In order to allow the user to see information about the smart lights, plugs, and thermostats, it should be presented in the application. See section 4.1.3 and 4.2.2.	Must
App_FR2	The user has to be able to view a visualization of historical electricity & heating usage since initial setup of the system.	In order to provide information to the user, a visualization of historical data should be available through the application. See section 4.1.3 and 4.2.2.	Must
App_FR3	The application has to be able to show notifications based on information provided from Hub_FR4.	In order to provide the user about notification about change in electricity price, it receives information from the hub. See section 4.2.2, and 4.	Must
App_FR4	The user has to be able to assign/unassign smart lights, smart plugs, & smart thermostats to a room.	As a room can consist of several light, plugs, and thermostats, the user should be able to add several to a single room. See section 4.1.2 and 4.2.2.	Should

Continued from previous page

No.	**Requirement**	**Motivation**	**Priority**
App_FR5	The user has to be able to add/delete smart devices, which are supported by the communication technologies in Hub_NFR2, Hub_NFR8, Hub_NFR9, and Hub_NFR10.	The user should have the freedom to connect any smart device they buy. See section 4.2.2.	Should
App_FR6	The application has to be able to adjust the setpoint temperature of the smart thermostats through Hub_FR2	The user should have control over the smart devices that are connected to the system. See section 4.2.2.	Must
App_FR7	The application has to be able to turn on/off the smart lights, smart plugs, & smart thermostats through Hub_FR1.	The user should be able to turn on/off the smart devices through the application. See section 4.2.2.	Must
App_FR8	The application has to be able to show the current electricity price.	The user should be able to see the current electricity price. See section 4.2.2.	Should
App_FR9	The application has to be able to fetch the data that are specified in Hub_NFR6 from the cloud database.	In order to show it to the user. See section 4.2.2.	Must
App_FR10	The application has to be able to send the user's home/away status to the hub.	In order to allow the system to turn of device when the user is not home. See section 4.1.3.	Must
App_FR11	The user has to be able to view/edit the predictions from Cloud_FR1.	In order to allow the user to have control over the system. See section 4.1.3.	Should

Continued from previous page

No.	Requirement	Motivation	Priority
App_FR12	The user has to be able to define a threshold in percentages for high electricity usage.	In order to allow the user to define his personal reference.	Should
App_FR13	The application has to be able to visualize the data from Hub_FR6 in different intervals (hourly, daily, weekly, yearly).	The user should be able to filter the visualized data. See section 4.2.2.	Should

Table 4.13: Table of the application's non-functional requirements

No.	Requirement	Motivation	Priority
App_NFR1	The application has to be able to check if it is connected to the home's Wi-Fi SSID.	In order to let the system know if the user is home or not. See section 4.1.3 and 4.5	Must
App_NFR2	The application has to fetch the hub's IP from the server using the hub's unique ID.	In order to allow the application to communicate with the hub. See section 4.1.3.	Must
App_NFR3	The application has to be able to fetch electricity and heating usage from the hub from the initial setup till the last collected instance.	Used for data visualization. See section 4.2.2.	Must
App_NFR4	The application has to be able to scan a QR code on the hub.	The hub's unique ID is stored in the QR code, which is needed to fulfill App_NFR2. See section 4.1.3 and 4.2.2.	Must

Continued from previous page

No.	Requirement	Motivation	Priority
App_NFR5	The application has to comply with the GDPR.	In order to avoid fines, compliance to GDPR is important. See section 4.6.	Should

4.8.3 Server

Table 4.14: Table of the server's functional requirements

No.	Requirement	Motivation	Priority
Server_FR1	The server has to be able to receive the IP address from the hub.	Used in the registration and setup process. See section 4.1.2.	Must
Server_FR2	The server has to store received IP addresses in a database.	In order to allow the application to fetch the IP, as it is used for communication between the application and hub. See section 4.2.3.	Must
Server_FR3	The server has to be able to send a hub IP when requested.	In order to support the communication between the hub and the application. See section 4.2.3.	Must
Server_FR4	The server has to collect electrical prices from an external source.	In order to allow the application to display notifications, based on the electricity price. See section 4.2.2 and 4.1.3.	Must

Table 4.15: Table of the server's non-functional requirements

No.	Requirement	Motivation	Priority
Server_NFR1	The server should comply with the GDPR.	In order to avoid fines, compliance to GDPR is important. See section 4.6.	Should

4.8.4 Cloud-based Environment

Table 4.16: Table of the cloud's functional requirements

No.	Requirement	Motivation	Priority
Cloud_FR1	The machine learning model should predict the four status changes: Wakes up, leaves for work, comes back home, goes to sleep.	In order to be able to turn off devices when nobody is home, or using them. See section 4.1.3. See section 4.7.1.	Must
Cloud_FR2	The machine learning model should predict the user's current status: Home, Away or sleeping	In order to be able to turn off devices when nobody is home, or using them. See section 4.1.3. See section 4.7.1.	Must

Table 4.17: Table of the cloud's non-functional requirements

No.	Requirement	Motivation	Priority
Cloud_NFR1	Machine learning models have to be trained in a cloud-based environment that supports supervised learning algorithms	Section 4.6 concluded that a cloud-based environment had to be used for machine learning, and section 4.7.1 specifies that the environment has to support supervised machine learning algorithms.	Must

Continued from previous page

No.	**Requirement**	**Motivation**	**Priority**
Cloud_NFR2	The following data has to be used for feature extraction for training a model: date, time, weekday, power consumption, temperature setpoint, light status.	Used in the Machine Learning algorithm. See section 4.7.	Must

5 Design

This chapter will present the design of the prototype that will be used to answer the problem statement defined in section 1.2. The architecture of the prototype and the system will be presented along with the services that have been presented in section 4.3 in chapter 4. The components the system will consist of, will further be elaborated upon with argumentation for why these technologies, frameworks, or libraries are used. Furthermore, the requirements that are specified as a 'must' for each of the prototypes' components (hub, server, application, cloud) presented in table 4.10, 4.11, 4.12, 4.13, 4.14, 4.15, 4.16 and 4.17 will be designed, as the prototype must fulfill these in order to provide an answer to the problem statement defined for this project.

Below in figure 5.1 a detailed presentation of the architecture of the prototype and its components can be seen. The connected smart devices should be able to communicate with the hub through Z-wave (Hub_NFR2). As it can be seen in the figure, the hub is a Raspberry Pi. Other small computers could have been used as well, but as the group had prior experience with it, while already having obtained one, it was decided that it would be used as the hub. Furthermore, as presented in Hub_NFR6 it should be able to save the data to a cloud DB, which can be seen in the lower right corner in the figure.

Furthermore, the hub should be able to establish a connection to the Internet, as it can be seen in Hub_NFR1. Through the connection the hub should be able to communicate with the dedicated Android application, which should provide all of the functionality marked with a 'Must' in table 4.12. The system should also be intelligent in order to make predictions which can help the user reduce his energy consumption, and in order to fulfill Cloud_NFR1. The decision on why Microsoft Azure Machine Learning Studio will be used can be found in section 5.6.

In order to allow the system to send notifications to the application, a notification system should be implemented. In figure 5.1 this is visualized as Firebase, which allows the fulfillment of Hub_FR4 and App_FR3. Firebase will be elaborated upon in section 5.1.1 below. As for the server, a Node.js web server will be used in order to accomplish the objectives presented in table 4.14. The server will also have its own MongoDB database. More about the server will be presented in section 5.2.

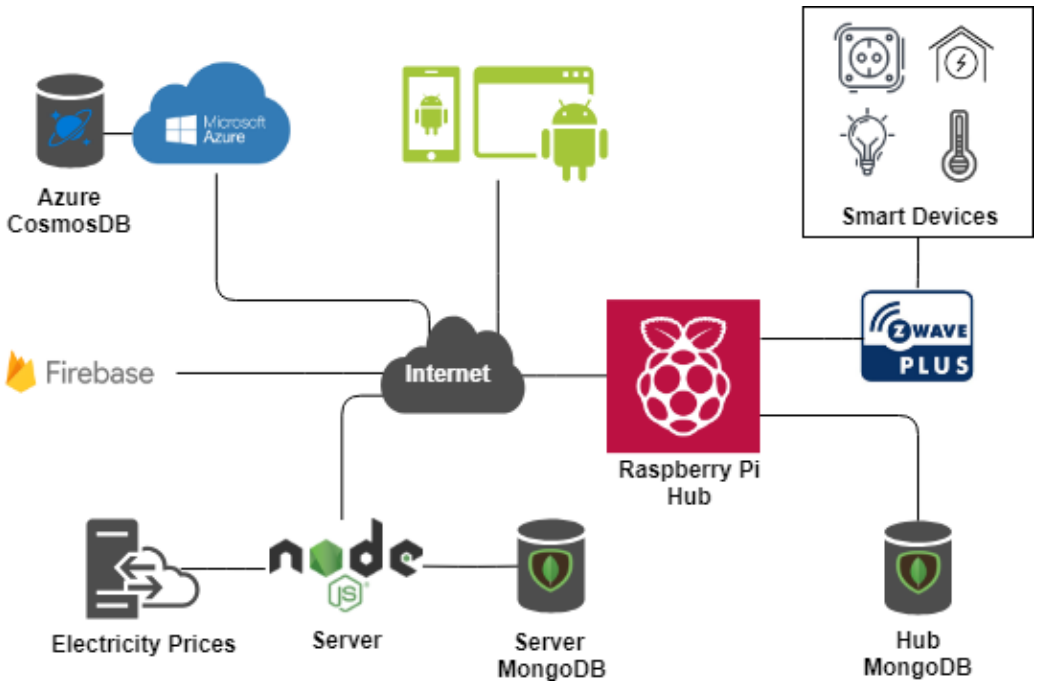


Figure 5.1: Free flow diagram of the components of the system.

5.1 Hub & Application

The following sections involve design decisions regarding the hub and the application presented in figure 5.1. The hub and application will be presented together in these sections as many of the requirements in sections 4.8.1 and 4.8.2 are related. The initial setup of the hub and application will be explained in order for the two entities to be able to communicate. The smart devices and hub also have to be able to communicate with each other, which will be discussed as well. The price of electricity will be utilized as a tool for affecting the users' behaviour in order to get them to use electronic devices when the price is lower. Data visualizations will also be designed as users can be physiologically affected by visually seeing their energy usage [20].

5.1.1 Hub & Application Setup

This section will cover how the user can initially setup the hub, which makes it possible to interact with the smart devices through the application. As specified in section 4.8 App_NFR4, the application has to be able to scan a QR code printed on the hub, since this is needed for the application to identify the hub with. If the user is located in a remote

location from the hub and wants to control their home, then the application has to contact the hub through its public IP address. The public IP address is the usually dynamically assigned by the user's Internet Service Provider (ISP). To avoid using a static IP address, our server will store the user's public IP address for the application to fetch and thereafter communicate with the hub. Figure 5.2 is a sequence diagram that shows how the user can setup a hub and application.

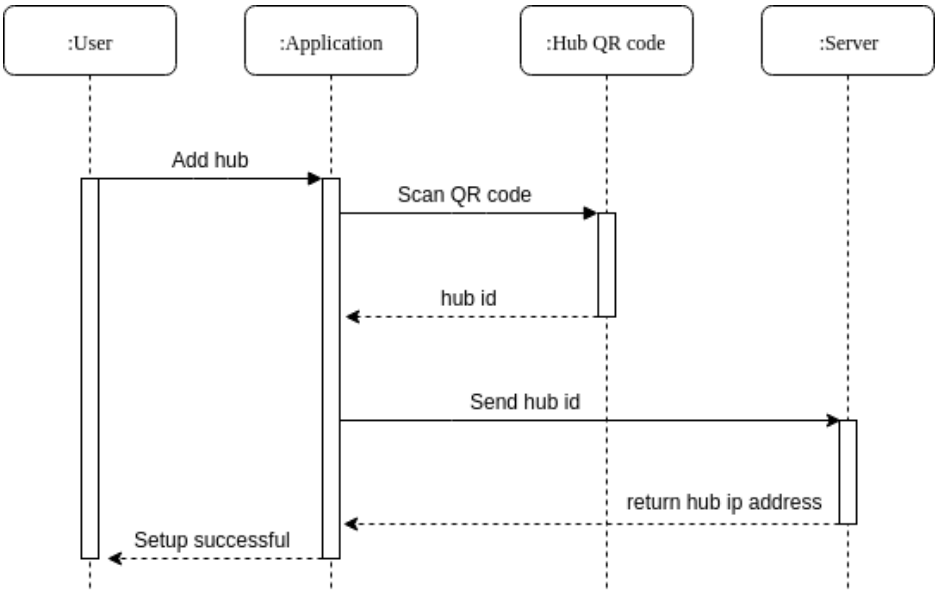


Figure 5.2: Sequence diagram showing how the user can register a smartphone application to the hub through the server.

First of all, the user has to install the application. Thereafter, the user should be able to choose to add a hub to the application. The application starts the camera and illustrates that the user should scan the QR code on the hub. The application will then decode the QR code into text, which is the ID of the hub. The application then exchanges the ID for the hub's current IP at the server. The user will then be told that the setup has been successful. This means that the application now has the IP address of the user's hub, which allows the user to control or monitor smart devices through the application, since the application now can communicate with the hub.

As specified in Hub_FR4, the hub has to be able to send notifications to the application. As seen in figure 5.1, one of the entities is named Firebase. Firebase offers many services that allows application developers to focus on developing their application rather than building the infrastructure it requires. These services includes analytics of usage and crashes, cloud

messaging and remote configurations. However, only Cloud Messaging is used for this project. Firebase Cloud Messaging allows developers to send data to their applications from a remote server through HTTP requests. The application developer does not have to keep track of device IP addresses, device IDs, etc. as this is handled by the Firebase library and back end. Furthermore, Firebase also takes care of delivering and handling the messages in a battery efficient way [93]. When the application is installed onto the smartphone it is granted a unique token. This token is used when the developer wants to send a message to the device, it essentially tells the Firebase back end where to send the message.

5.1.2 Communication Between Hub & Smart Devices

Multiple of the requirements mentioned in section 4.8.1 relate to which communication technology the hub should support in order to communicate with the smart devices. Only one of those requirements, namely Hub_NFR2, was chosen to be implemented for the prototype. It requires that the hub should be able to communicate over the Z-Wave protocol. Z-Wave is a wireless protocol that often is used for smart home devices as it uses very low energy [94]. Adding Z-Wave support to the hub is imperative and can be achieved using a library called OpenZWave. OpenZWave is a library that can be used for applications in order to communicate with Z-Wave devices through a USB-stick, where the USB-stick is a Z-Wave controller [95]. OpenZWave's control panel can be used to manage devices that are added to the Z-Wave network. However, it would require that we developed an entire solution to get data from the smart devices. We have therefore chosen to use Home Assistant for the prototype, an open-source platform for controlling and automating devices [96], maintained by other developers, thus saving us time developing such a solution. Home Assistant can easily be installed on a Raspberry Pi (RPi). It automatically installs OpenZWave on setup, which means that a USB Z-Wave controller needs to be plugged into the RPi in order to be able to communicate with devices. The advantages of using Home Assistant is that it has a built-in API that allows other software to consume the endpoints to control and get data from the connected Z-Wave devices, which saves a considerable amount of time compared to using only OpenZWave. The API is also the only feature of Home Assistant that will be used for this project.

Interacting with smart devices will be as simple as making an HTTP request to the appropriate API endpoint. An example of how data can be gathered from smart devices is by calling the /api/states/ endpoint, which will return a list of all smart devices and their data in JSON format (see Listing 5.1).

Listing 5.1: An example of JSON data return when calling the /api/states/ endpoint of Home Assistant.

```
"attributes": {
        "friendly_name": "NodOn MSP-3-1-XX Micro Smart Plug Power",
        "node_id": 2,
        "power_consumption": 52,
        "unit_of_measurement": "W",
        ...
    },
    "entity_id": "sensor.nodon_msp31xx_micro_smart_plug_power",
    "state": "on"
    ...
}
```

The smart devices can also be controlled individually by calling /api/states/<entity_id>/, where entity_id is the unique identity for a smart device. This means that smart devices such as smart lights, smart plugs, and smart thermostat can be turn on/off and adjusted through the API, which fulfills requirement Hub_FR1 and Hub_FR2.

5.1.3 Price Notifier

The price notifier should be designed so it automatically fetches the electricity prices through the server each day in order to ensure that the correct prices are always used. The user should be able to receive notifications about high electricity prices when he is performing electricity demanding actions (App_FR3. If the user receives such a notification, then the system could suggest a cheaper timeslot for the action (e.g. 2 hours later). The sequence of what happens when the user wants to check the electricity prices can be seen in figure 5.3 below.

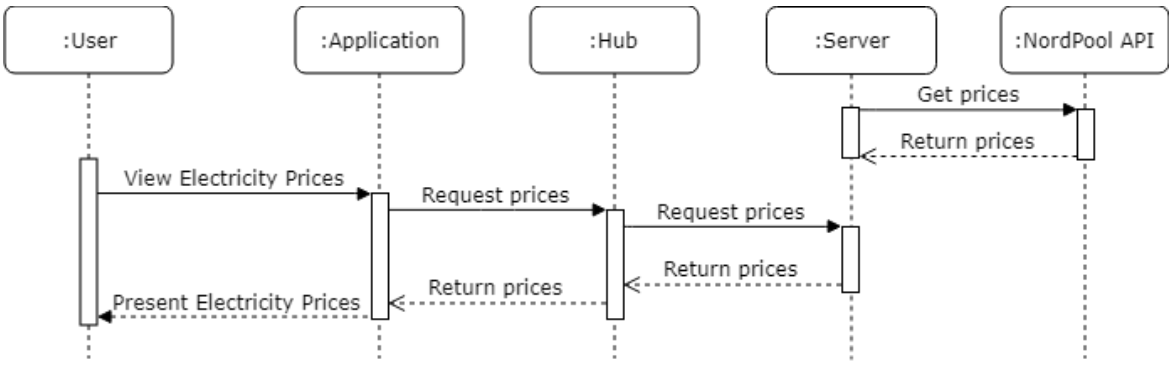


Figure 5.3: Sequence diagram of how the price notifier fetches electricity prices from Nord Pool

As seen in figure 5.3, the server starts of by fetching the electricity prices from the Nord Pool API once a day. The application should communicate with the hub, since the hub fetches the electricity prices from the server (Hub_NFR3), and then sends them to the application, allowing the Price Notifier to function. In order to send the user notifications through the application (Hub_FR4), the Firebase development platform should be used, as it already should be used to perform other actions in the system.

The complete system should also fetch the electricity prices from the server (Server_FR4, while taking the user's context and location into consideration [97]. In order to avoid showing wrong prices, the user should either input where he lives or allow the application to take the location through the smartphone, however, as mentioned in section 1.3, the prototype will only be developed for Denmark, and hence this iteration of the prototype will not do so.

5.1.4 Data Visualization

In order to fulfill App_FR1, the application should allow the user to view the amount of electricity and heating usage, since the system was setup. To further extend on this, the application should allow the users to decide at what interval they would like to see their usage. The intervals that should be presented for the user should be hourly, daily, weekly, monthly, yearly (App_FR13).

Figure 5.4 shows a sequence diagram for how the application fetches the historical data. The application starts of by requesting the data from the server, which requests the collected historical data from the clouds database.

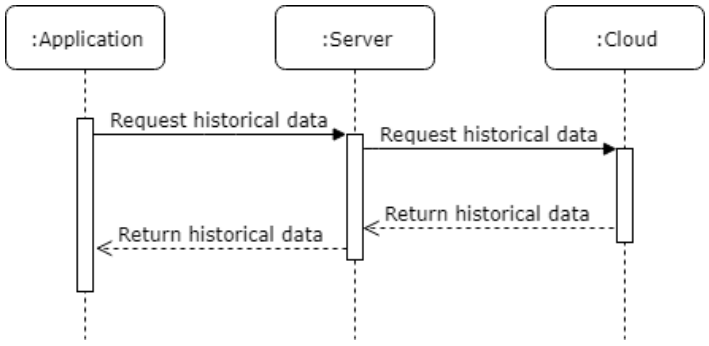


Figure 5.4: Sequence diagram showing how the application can request data about the energy usage from the cloud.

A library for Android OS will be used for creating visualizations for the application. The decision on which library will be used can be found in chapter 6.

5.2 Server

As discussed in the start of this chapter a remote server is required for the various components of the system to properly communicate with each other. During this process the server must be able to receive an IP address from a hub as specified in Server_NFR1 and be able to forward it when requested as specified in Server_FR3. A web server will therefore be used, which exposes one or more endpoints. It was decided to use a Node.js web server for this project. The reason for this is that Home Assistant's APIs returns JSON formatted responses, which makes it quick to parse and react to the responses. In addition, as exposing an endpoint is not a computational heavy operation using Node.js is beneficial as it does not have the same overhead as e.g. Spring for Java does because of the reliance of the Java Virtual Machine. The LAMP stack could have been used instead of Node.js, however as we have prior experience with Node.js and JavaScript, it allowed us to quickly start prototyping. Using LAMP would require us to learn PHP and MySQL first. A downside to using Node.js is that any heavy computations may block the main thread and thus halt all operations. In order to avoid this all operations should be optimized for efficiency, e.g. no large database requests. If any computational heavy requests has to be made they should be performed on a separate child thread.

5.2.1 Registration of Hub & Application

As discussed earlier in section 5.1.1, the server plays a vital role in establishing a communication tunnel between the application and the hub. This communication tunnel allows the hub to send notifications to the application and for the application to see the status of the smart devices in the household. For the server to facilitate the communication it needs to store two things from the hub; the hub's unique ID and it's public IP address as specified in Server_FR1. From the application it only requires a valid Firebase token for sending notifications to the application. These three should be stored locally on the server so the hubs and applications can request them when required as specified by Server_FR2.

5.2.2 Electricity Prices Fetching

As the prototype should be innovative, future-orientated and fulfill Hub_FR4 and App_FR3, it should take the changing electricity prices into consideration, as the implementation of smart meters in every household in the future has a high chance of becoming reality. As the majority of the households in Denmark does not have a smart meter installed yet, it is not possible to utilize them in the project. However, in order to mimic the changing electricity

prices, a website that provides the changing electricity prices through an open API will be utilized. There are several websites which provides information about current, and forecast electricity prices. However, it has been found that one of the websites, namely Nord Pool, offers an API that provides a wide variety of APIs for different contexts. Access to the API and its documentation is gained through signing in through the website (see [97]). Nord Pool's API allows the system to fetch data about current and forecasted electricity prices. These will be used to alert the user when the electricity prices are high, and he is starting a high electricity consuming task.

However, there are some limitations to the Nord Pool API, which for instance is the amount of times you can make calls through the API. This would become a concern if each hub should fetch the electricity prices directly from the Nord Pool API. In order to remedy this, the server will fetch the electricity prices once, and then the hub will make a request to get the prices from the server. This means that the server only needs to make one call to the Nord Pool API a day, as it fetches all of the electriticy prices for the whole day, in order to make sure that the electricity prices are correct. Making the hubs call the for the prices are also a cheaper, flexible and scalable way. If each hub should call the Nord Pool API, it could create a lot of unwanted traffic. Furthermore, Nord Pool also writes in their Terms of Agreement, that they have the right to change the rate or amount of calls that the user are able to perform. Server_FR4 will be fulfilled by fetching the electricity prices from Nord Pool.

5.3 Data Storage

Smart devices and user patterns have to be gathered if any kind of machine learning should be used for automation. Hub_NFR6 specifies that the hub has to store data in a database. This section will be used to evaluate which type of database should be used for the prototype. The two types of databases considered will be SQL and NoSQL databases.

SQL is a relational database management system that works well with structured where there are relationships between the data [98]. SQL databases also rely on predefined schemas, which means that tables have to be designed and created before any data can be stored. Using a relational database can be beneficial when considering how simple it can be, and data integrity through use of constrains [98]. Time-consumption spent on designing tables is a drawback in SQL, and it can be more difficult to scale because of the complexity of the database [98].

A NoSQL database is non-relational and can be of different types, such as key-value or

document based [98]. NoSQL supports different formats of data, such as JSON and XML. Unlike SQL databases, it is not necessary to design a NoSQL database beforehand, since it is a schema-less database that allows developers to add data to the database without spending time predefining the schemas. A benefit of using NoSQL is ease of scalability, which also is one of the reasons NoSQL became popular [98]. Reliability can be a disadvantage of NoSQL, as two developers could access a newly updated databases at the same time, however, one of them would see an updated version of the database while the other would see the old version [98].

For this prototype, we have decided to choose NoSQL over SQL. First of all, the group members all have previous experiences in working with NoSQL, and its schema-less design allows us to start working on the prototype much faster than if we had to learn SQL and predefine the necessary tables. The fact that NoSQL is easier to scale is also appealing to us, because we assume that people will use a lot more smart devices in the future, which in the end will result in a huge amount of data that can be gathered. As mentioned in section 5.1.2, the data received from smart devices is already in JSON format, which also makes it easy to store in a NoSQL database as it supports JSON as a format. Furthermore, as there is only one entity that will be looking into the database, as seen in figure 5.1, the reliability problem will not cause any disturbances. Hence a NoSQL will be the database type for the hub, server and the cloud-based environment.

5.4 Key Element 1 - Smart Electricity & Heating Devices

As already mentioned in section 4.4, the prototype will consist of three types of smart devices. As seen in Hub_NFR2 the prototype should be able to communicate with Z-Wave devices, as all of the devices that will be used with the prototype all use Z-Wave as their communication technology. The complete system should however, be able to communicate with all of the communication technologies presented in Hub_NFR8, Hub_NFR10, and Hub_NFR9 in table 4.11.

The three different sensors and actuators are smart plugs, smart lights, and smart thermostats. The system will have two smart plugs, one smart thermostat, and two smart light bulbs. As it can be seen in figure 5.1, the smart devices connect and communicate directly with the hub, which also means that if the user makes changes from the dedicated application, the commands goes through the hub. Below in figure 5.5 a sequence diagram for the smart devices, the hub, and the application can be seen.

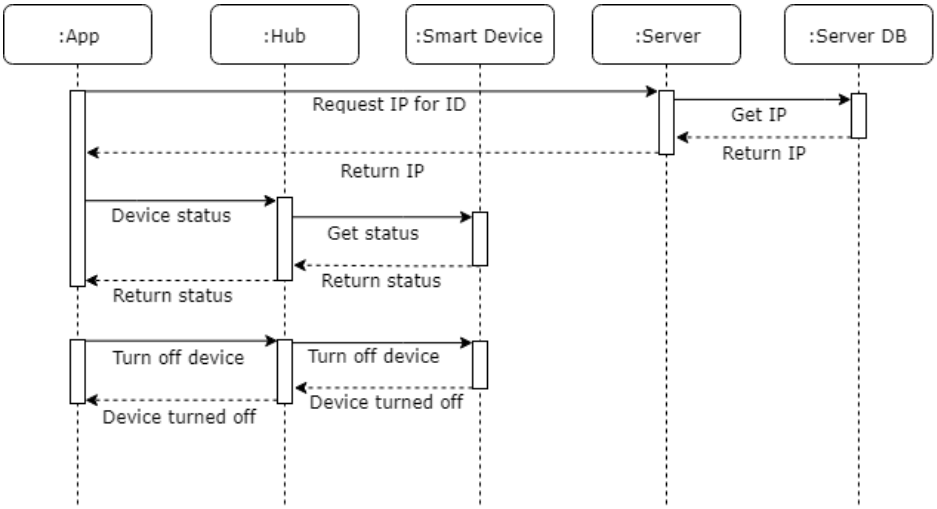


Figure 5.5: Sequence diagram showing how the system allows the user to monitor and control smart devices.

Before the user are able to make changes to the connected smart devices, it needs to make a request for the hubs IP in order to be able to communicate with it. Afterwards, the application can communicate with the smart devices through the hub. As it can be seen figure 5.5 the application first has to ask the hub about the smart devices status (on/off/setpoint temperature). After receiving the status, the application is able to send commands to the hub, which relays the commands to the smart device.

Furthermore, the system calls the smart devices in order to collect data which will be used for machine learning. This part will be further elaborated upon in section 5.7 below. The data that should be collected by the smart devices can be seen in section 4.7 and Hub_FR6.

5.5 Key Element 2 - User Home/Away Status

According to Hub_NFR7, the hub needs to be able to receive the user's home or away status from the smartphone application. Section 5.1.1 introduced how a smartphone and a hub should be set up in order to establish a communication channel between the two, and it was concluded that the smartphone should be able to communicate with the hub through the hub's public IP address. This means that the hub can run a web server with an endpoint that is accessible by the smartphone application. This is essential as App_FR1 requires that the smartphone checks if it is connected to the appropriate Wi-Fi SSID, which the smartphone thereafter has to report that to the hub according to App_FR10.

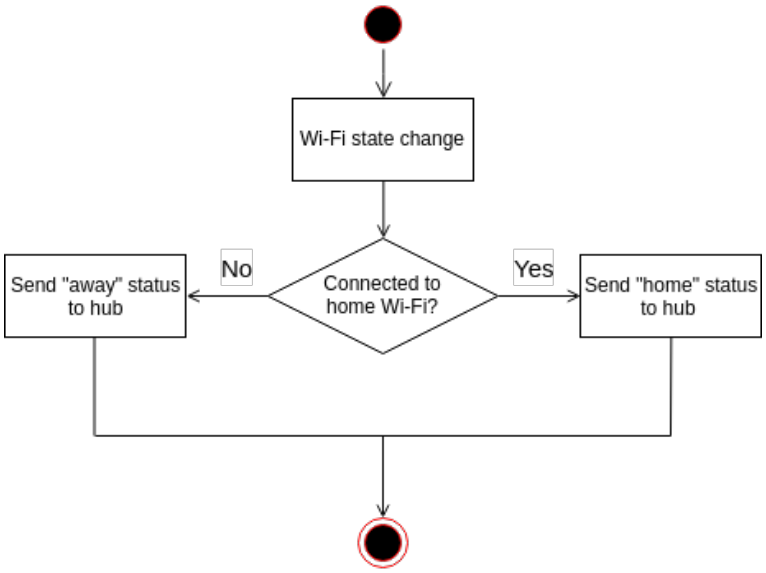


Figure 5.6: Activity diagram of how the applications reacts to a change in the Wi-Fi state.

Figure 5.6 shows an activity diagram of what should happen whenever the user's Wi-Fi state changes. A Wi-Fi state change is for example when the users comes home and the smartphone automatically changes from using cellular data to Wi-Fi. Whenever the state changes, the smartphone application checks if it is connected to the user's home Wi-Fi network. If the smartphone is connected to the home Wi-Fi SSID, it should notify the hub that the user is home. Otherwise, it should notify that the user is away.

5.6 Key Element 3 - Processing Architecture

The advantages and disadvantages of different processing architectures were discussed in section 4.6. This helped to decide whether to perform the machine learning computations locally or use a cloud service. Based on the analysis it was decided to use a cloud service in section 4.6.4. This is also specified in Cloud_NFR1. Several companies offer machine learning as a service, however, it was decided to narrow it down to the four big companies within SaaS; Amazon Web Services (AWS), Google App Engine, IBM Watson, and Microsoft Azure. These web services will be evaluated and the best fit for this project will be used. The evaluation and design choice is based on Gartner's Magic Quadrant [99] as well as personal experiences with the services.

Google have primarily focused on developing their infrastructure rather than their feature

set. This means that they have great performance and response times, but may lack in service offerings. Their machine learning is primarily based on TensorFlow, an open source software library for neural networks. The primary interface for interaction is through a command line, which may require expertise to learn how to use efficiently. IBM is in the same category as Google, where they have focused on developing their infrastructure, however, they have also invested resources into specific services such as speech-to-text and visual recognition.

AWS is generally seen as the market leader for IaaS, an advantage that increases their trust in terms of up-time (important to avoid a single point of failure), security and performance. They have an extensive portfolio of services that can work well together with their machine learning offerings, however, this means it requires expertise and prior knowledge of the services to properly implement them. A lot of their machine learning processes are automated. This is good as it removes the workload from the user, but it also reduces customization possibilities. Compared to the Google and AWS, Microsoft's primary focus has been on offering cloud services distributed through Microsoft Azure. A part of Azure is their Machine Learning Studio that is based on a Node-RED like interface that allows the user to quickly drag and drop processes. This means that using Azure makes it easy to start training models and use them for predictions. In addition, Azure supports a large variety of machine learning algorithms, which makes it a good fit for testing multiple models to see which performs the best.

Based on the rich amount of services, ease of use, comprehensive data preprocessing, and amount of algorithms, it was decided to use Microsoft Azure Machine Learning Studio for this project. Furthermore, the group have previous experience with it, which makes it possible to quickly setup. In addition, through an Azure Cosmos DB it is possible to save data directly to Azure, which fulfills requirement Hub_NFR6. Azure Cosmos DB is a multi-model database, that allows the user to chose what database type to use [100]. The only supported NoSQL database model in Azure is a NoSQL MongoDB database, similarly to the one decided to be used in section 5.3. Due to this it has been decided to use MongoDB with Azure Cosmos. Storing the data in a Cosmos DB would outsource the data handling to Microsoft Azure rather than having to worry about stability and performance of yet another server.

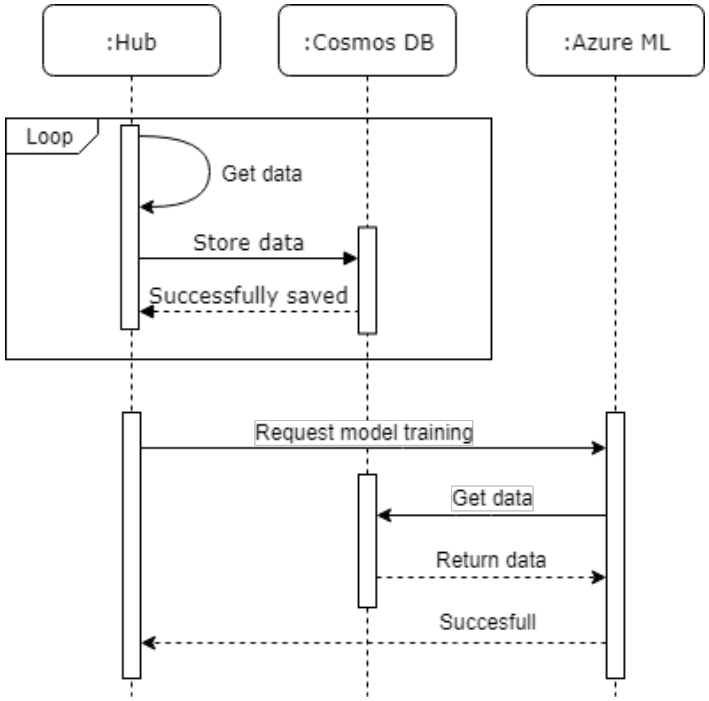


Figure 5.7: Sequence diagram of how model training would work in a cloud-based environment using Azure.

Figure 5.7 shows how the hub continuously gathers data from the sensors and saves it to the Cosmos DB stored on Microsoft Azure's servers. As machine learning requires large amounts of data in order to produce accurate predictions, the system should only request the Azure Machine Learning Studio to train a model when it has data enough data to do so. When Azure can provide solid predictions, it should fetch the data specified in Cloud_NFR2 that is stored in the Cosmos DB, train a model, and store the model locally for later use. When a prediction is required, the hub sends an HTTP request to Azure with the current instance in the body alongside data about which previous trained model to use. Azure then performs the prediction and sends the results back to the hub.

5.7 Key Element 4 - Machine Learning

As discussed in section 4.7 machine learning capabilities will be added to the prototype. The requirements for these capabilities were presented in section 4.8.4 in table 4.16 and 4.17. As discussed in section 4.7.3, two models should be trained and evaluated. One model (the Forecasting Model) should predict a timespan where the user might change their status. In this timespan the other model (the Real-time Model) should continuously predict

the user's current status based on the data gathered. Requirement Cloud_FR1 specifies a model to forecast the user's status and requirement Cloud_FR2 specifies a real-time model to predict the user's current status. As the data collected from requirement Hub_FR6 and Cloud_NFR2 may not be directly compatible with the algorithms, some data preprocessing is required to extract this. When the preprocessing is done, an evaluation of the different algorithms supported by Azure can be performed.

Later in this section the relevant algorithms for both for the Forecasting and Real-time Model will be discussed. While the algorithm evaluation is technically part of the project analysis, it was not possible to select algorithms in the Analysis chapter as the requirements for the models were not specified yet. When the algorithms discussed below in sections 5.7.2 and 5.7.4 have been evaluated and the best fit has been chosen, the requirement specification in section 4.8 in the Analysis chapter should be revisited and a requirement for the algorithm should be added as a part of an iterative process.

As all the regression algorithms discussed in section 5.7.4 used by the Forecast Model return both a mean and a standard deviation, it is possible to define the timespan discussed earlier in this section. The timespan can be defined as $[StartTime, EndTime]$, where $StartTime$ is defined as $Mean - (1 \times StandardDeviation)$ and $EndTime$ defined as $Mean + (1 \times StandardDeviation)$. Figure 5.8 presents how the two models would predict during the day. At midnight the Forecasting Model predicts the timespans in which the the user wakes up, leaves the home, comes back home, and goes to sleep. When any of these timespans starts, the Real-time Model begins to predict the user's current status, e.g. predicting if the user has woken up yet.

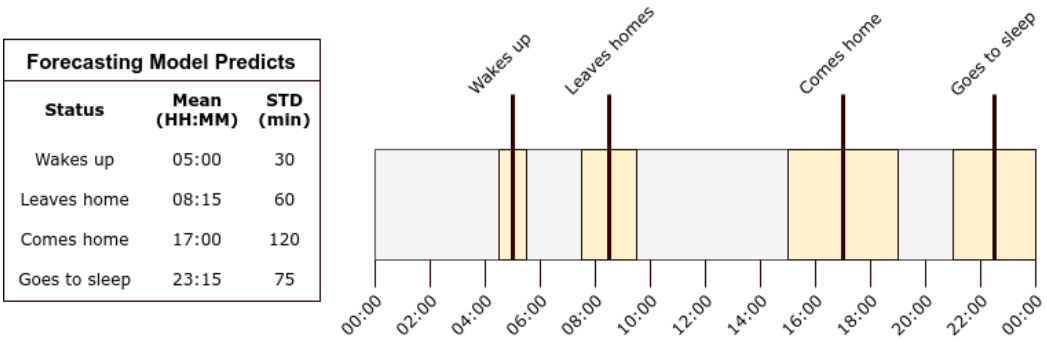


Figure 5.8: Example of how a daily prediction should be performed. The yellow areas are the timespans where the Real-time Model should perform prediction. The timespans are computed using the mean and standard deviation (STD) of the Forecasting Model's prediction.

When the Real-time Model predicts that the user has changed his status, the system turns on or off the devices in the house based on what his newly predicted status, i.e. turn on when he wakes up or arrives home and turn off when he leave the hour or goes to sleep.

5.7.1 Data Preprocessing For The Real-time Model

One of the machine learning models discussed in section 4.7.3 regarded performing real-time predictions in order to guess what the user is currently doing. Three prediction labels were identified, home, away and asleep. The model should use the data gathered from sensors with minimal data processing in order to see how this can be used to interpret the user's context. Figure 5.9 shows how the data from the sensors and the user's smartphone should be processed in order to generate instances to train the Real-time Model with.

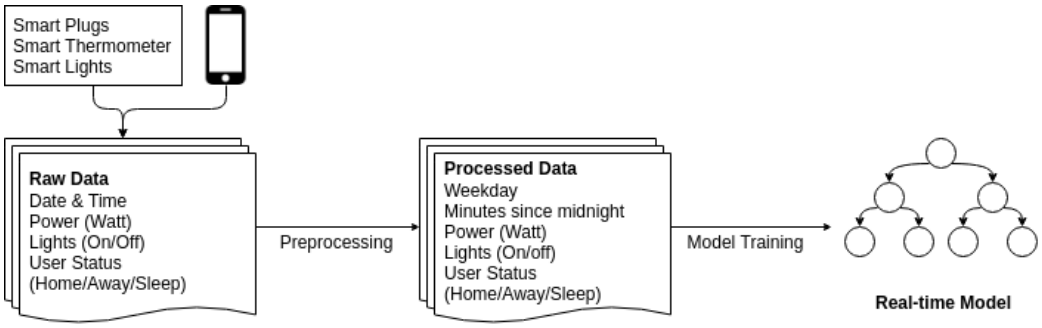


Figure 5.9: The flow of the data from gathering to processing and model training.

By using time since midnight rather than a timestamp allows us to treat time as a ratio value, as such a regression algorithm can be used to train the Forecasting Model. It is important to note that it should be the minutes passed since midnight of the day of prediction. This means that if the label is after midnight, e.g. the user goes to bed after midnight, then the label continues to count from the value 1440 (24 hours x 60 minutes = 1440 minutes in a day) rather than starting over from zero. This is important because to a human the times 23:30 and 00:10 are only 40 minutes apart, however, the machine learning would interpret them as the number 1410 minutes since midnight and 10 minutes since midnight respectively. This causes the model to calculate a difference of 1400 and interprets them as not related, while the real difference is only 40 minutes which means they are very related. With minutes since midnight calculated it is possible to generate the instances using the data from the sensors, the weekday and the time of day.

5.7.2 Classification Algorithms To Evaluate

Section 4.7.2 discussed how several algorithms should used to train multiple machine learning models and each of those models should be evaluated in order to find the best model. As specified by requirement Cloud_NFR1 the models has to be based on supervised algorithms. As mentioned in section 5.6, it has been decided to use Azure Machine Learning Platform. Azure supports four multiclass classification algorithms, all of which will be trained and evaluated.

1. Multiclass Logistic Regression (MLR) is still a classification algorithm despite having regression in its name. Logistic regression uses a logistic function to find the probability of an instance belonging to one of two classes [101].

2. Multiclass Decision Forest (MDF) ensembles many decision trees and uses them to vote on which label to predict. Each tree outputs a histogram of labels, the model then sums the histograms with weights. The more confident a histogram is in a specific label, the higher weight it is assigned [102]. Decision trees are flexible to noisy features.

3. Multiclass Decision Jungle (MDJ) are an extension to decision forest. It also ensembles many decision trees, however, in a decision forest there is only one path to each leaf node. A decision jungle may merge some leaf nodes together and as such allow multiple paths to a leaf node [103]. By merging leaf nodes the decision trees becomes simpler and thus increases performance.

4. Multiclass Neural Network (MNN) consists of three components; an input node that takes some data, one or more hidden layers that tries to weight the data to fit it to the model and an output node that returns the prediction. Depending on the amount of hidden layers an NNR may be computational heavy [104].

5.7.3 Data Preprocessing For The Forecasting Model

As discussed in section 4.7.1 and 4.7.2 different kinds of data have to be collected. This data will be used to extract features and labels to train a models with. For the Forecasting Model, the data will be used to extract features in order to find the specific time the user's status changes, e.g. from being asleep to waking up in the morning, as specified in Cloud_NFR2. This time will then be saved as the label as discussed in section 4.7.3. After the feature extraction three data points should be generated for each instance.

1. **Day of the week:** People have different schedules on different weekdays.
2. **User status:** What is the user's new status, i.e. waking up, leaving for work, arriving back home and going to bed.
3. **Time since midnight:** At what time in the day did the change in user status happen.

Both day of the week and time of the day is easily acquired as the data processing are the same as the Real-time Model described in section 5.7.1. However, getting the four possible user status' requires the application data and sensor data. To see when the user leaves their house or arrives back can be found during the data logging, as the user's smartphone reports when they leave or arrive to the hub as per App_FR10. However, figuring out when the user is going to sleep or waking up is more troublesome. As per Hub_FR6 the hub should collect the status of the lights, the power consumption and the temperature setpoint of the thermostats. Based on this data it is possible to find the time when the user has gone to bed and wakes up. If there is high power consumption, the lights are turned on and the user is home then he is most likely not asleep. However, if all lights are turned off and there is a low power consumption then he is most likely asleep. This means that when the lights turn off and the power consumption is reduced, the user has most likely gone to bed. Similarly the other way around, when the house starts to consume energy in the morning the user has most likely woken up. Figure 5.10 shows a decision tree of how the user status was classified based on their home/away status, power consumption and if the lights are turn on or off.

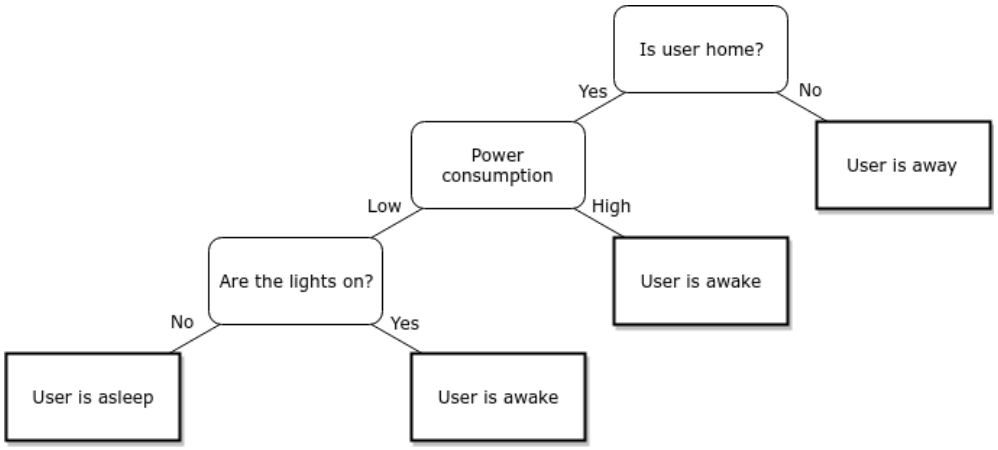


Figure 5.10: Decision tree to classify the user status during preprocessing.

The first decision box in figure 5.10 checks to see if the user is home or not. This is gathered through the Wi-Fi connectivity of the user's smartphone as described in App_NFR1 and

App_FR10. If the user is away then the status is set to away and the decision tree stops. However, if the user is at home then the decision tree considers the power consumption of the smart plugs. If a device in the house has a high power consumption, e.g. TV or computer, then the status is set to awake as the user is most likely using it. If the power consumption is low then the decision tree considers the lights. If the lights are turned on then the user is set to awake and if they are turned off then the status is set to asleep. Figure 5.11 shows how the data coming in from the smart devices and smartphone are processed to generate the features needed which is in turn used to train the Forecasting Model with a supervised machine learning regression algorithm.

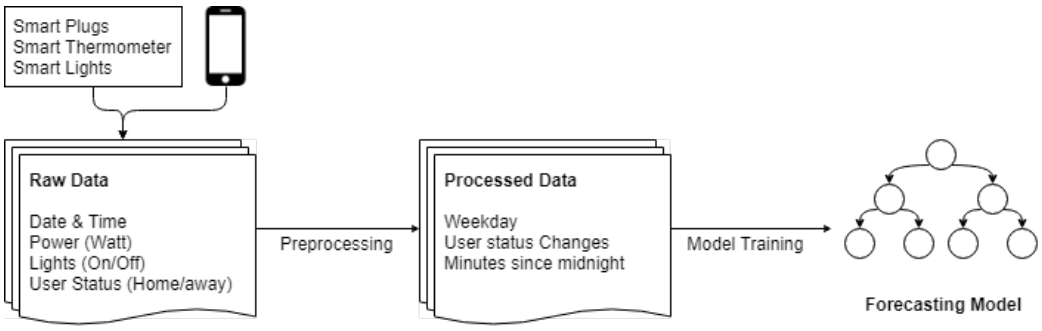


Figure 5.11: The flow of the data from gathering to processing and model training.

5.7.4 Regression Algorithms To Evaluate

As discussed in section 4.7.2 several different algorithms should be used to train multiple models with and each should be evaluated. The model with the best evaluation results will be used for the final implementation of the Forecasting Model. Requirement Cloud_NFR1 specifies that the models must be based on supervised learning algorithms, meaning that each training instance describes an output (or label). Furthermore, as discussed in the previous section, the label of the model should predict is the amount of minutes that has passed since midnight. As this is a number the algorithm used should be based on regression. Using classification in this case would require a label for each single minute of the day, which would decrease the accuracy of the model. As discussed in section 5.6, it has been decided to use Azure Machine Learning Platform. Azure supports several different regression algorithms that may be relevant for this project. The algorithms offered by Azure that will be evaluated for this project is the following three:

1. Bayesian Linear Regression (BLR) is an algorithm based on the Bayes' Theorem. It describes probabilities based on knowledge and likelihood of previous events. It can be

useful for data that is poorly distributed, i.e. data sets with many outliers [105].

2. Decision Forest Regression (DFR) ensembles multiple decision trees together into a single model. The output of each decision tree is an Gaussian distribution. During prediction it accumulates the trees together in order to find a distribution that matches the distribution of the model [106].

3. Boosted Decision Tree Regression (BDTR) uses multiple decision trees added together to improve the prediction accuracy similarly to decision forest regression. The trees make use of the residual in order to make up for errors of prior trees. Low level of entropy and highly related features generally perform well with decision trees [107].

5.8 Application User Interface

As it can be seen in figure 5.1 there is a dedicated Android application for the system. It was decided to only create a Android application, due to the free development platform (Google Play) and prior experience. The Android application should allow the user to perform all of the actions that can be seen in the application use case diagram, presented in figure 4.9. Furthermore, the functionality and usability presented in table 4.12 should also be designed and incorporated into the application. The other systems presented in 4.7 all have a dedicated application allowing the user to remotely control and monitor the connected devices through the applications interface. As the majority (88%) of the population of Denmark owns a smartphone [108], the incentive to use it as a means of controlling and monitoring (App_FR1 and App_FR6) the system was deemed to be appropriate.

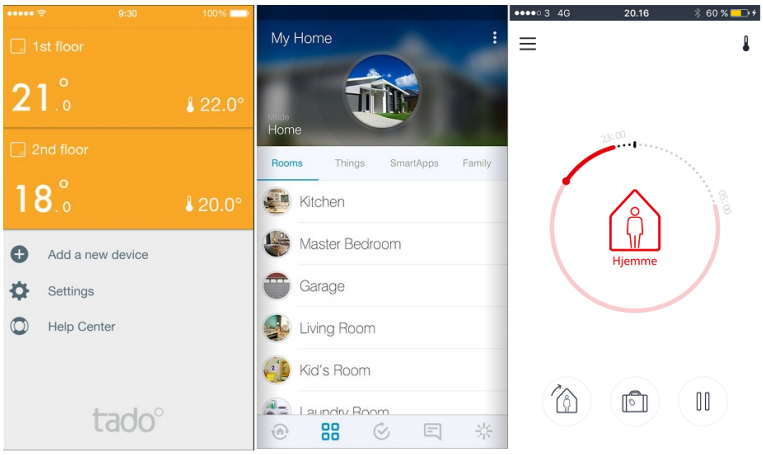


Figure 5.12: The user interface in the Tado (left) [109], SmartThings (middle) [110], and Danfoss (right) applications

As it can be seen in figure 5.12 there are many different ways the interface of the application can be designed, however, as this is a prototype only the core functionality will be emphasized. The layout of the main menu in the application should show all of the functionality and tasks that the user should be able to perform. Below in figure 5.13 a sketch of the user interface (UI) can be seen. From here, the user should be able to easily and quickly go to the desired event. It should be made clear for the user, what each of the buttons or menus presented in figure 5.13 does or contains. For now only text will be used to describe the buttons, however, in the future a icon corresponding to that menu should be implemented, in order to give a more pleasing look and feel.

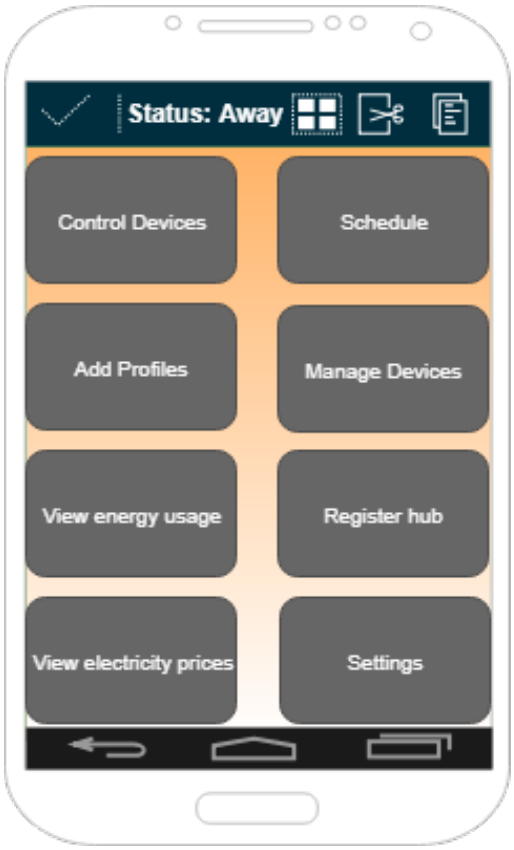


Figure 5.13: A sketched version of how the UI should look in the application

As it can be seen in figure 5.13 all of the functionality presented in figure 4.9 in section 4.2.2 are visualized, however, as this prototype is a conceptual design of the final application, only the requirements with priority "must" in table 4.12 will be implemented. A visualization of some sketched menus can be seen in figure 5.14 below. The application is used as a mean for

the user to control, monitor, add and delete the devices, which are presented in App_FR1, App_FR2, App_FR5, App_FR4, and App_FR6.

Figure 5.14: A sketch of how some of the menus should look. Starting from the left there is the "View energy usage", in the middle there is "Control devices", and to the right it is "Schedule"

In the final system, the applications UI should more pleasing to look at, with more graphics, and better functionality, while keeping the learning curve of the application to a minimum. Several methods for testing the usability of the application should be performed in order to insure that it is easy to navigate for the user. A lot of the test can be found in the book written by Rogers et al. [111].

6 Implementation

The purpose of this chapter is to present how the necessary requirements from the previous chapter have been implemented. All of the requirements will not be implemented for the prototype. Only the ones that were prioritized as *must* in section 4.8 will be implemented, since these are deemed the most critical requirements. The chapter will cover how the hub and application are registered in order to link a specific smartphone to a specific hub. The implementation of how Wi-Fi on a smartphone is used to detect user presence will be explained, along with how the hub gathers data from smart devices. Machine learning will be implemented in order to predict the user's status, i.e whether he is going to bed, waking up, leaving home, or coming home. The price of electricity will be used to implement a notification service, and data visualizations will be created with the purpose of influencing the user psychologically. The full codebase can be accessed through Github on https://github.com/nicfol/MasterThesis. The relevant branches are: "Server", "hubNodeServer" and "androidApp".

6.1 Hub & Application Setup

As discussed in section 5.1.1, both the hub and the application has to register themselves to the server before the server can support them in establishing a communication tunnel between the application and its hub. When the setup is complete the application will have access to the data gathered as well as being able to control the smart devices.

6.1.1 Send Hub IP To Server

First, the hub needs to send its global IP so that the server can send it to the application when requested to. For this implementation, and in order to fulfill Server_NFR1, it was decided to use a Bash script that would be executed by a Cronjob. Bash was used as it can natively run on most Linux distributions including Raspbian, a Linux distribution that runs on the Raspberry Pi. Bash is a universal and mature language that allows many different actions, including sending HTTP requests, file and string manipulations. In addition, as Bash is a shell script it can easily be scheduled to run regularly with a Cronjob. Cron is a service that allows the user of an operating system to schedule commands to run on a schedule, e.g. schedule a backup of a specific directory or reboot the computer each night

[112].

Listing 6.1: Bash code to send the hub's ip to the server.

```
1 #!/bin/bash
2 HUBID="1001"    # Unique hub ID
3 IP=$(curl -s http://whatismyip.akamai.com)  # Fetch the hub's global IP from akamai
4
5 # Send post request to the server with the ID and IP
6 curl -X POST -H 'Content-Type: application/json' \
7 -d '{"_id": "'$HUBID'", "ip": "'$IP'"}' \
8 http:// ( ... ) /config/setIp
```

For this project the code in listing 6.1 was executed every 15 minutes. On line 2 and 3 the hub's unique ID is hard coded to match the QR code placed on the face and a cURL command to to fetch the hub's global IP from a service provided by Akamai through an HTTP Get request. cURL is a command line tool that allows transferring data between computers [113]. Line 6-8 then sends an HTTP Post request to the server. This request contains a header specifying the content type, the body object in JSON that contains the two variables from line 1-2 and the endpoint at the server.

As mentioned in section 5.2, it was decided to use a Node.js web server to handle incoming requests for the remote server. A benefit of using Node.js is the Node Package Manager (NPM), a large-scale repository of JavaScript libraries (known as packages) that allows developers to reuse code written by others. Several of these packages are used for the implementation of this project. One of these is Express, a lightweight framework for handling HTTP routing [114]. In listing 6.2 the router for handling HTTP Post requests to the endpoint "/setIp/", this is defined on line 1. If the parameters "_id" and "ip" are specified in the HTTP body when a search query for the id it created on line 6 and an update query on line 7. As decided in section 5.3 and section 5.6 a NoSQL database in form of a MongoDB schema should be used. To perform the database operations a package called Mongodb (referred to as MongoClient in the code) will be used, which also fulfills Server_FR2. This package handles establishing a connection to the database as well as performing the queries requested. In this case the query is to update on object in the database on line 11. This takes a query to find the object matching the hub id and updates the ip object within with the updated ip.

Listing 6.2: JavaScript for the Node.js server to receive the IP from the hub in listing 6.1.

```
router.post('/setIp', function (request, response, callback) {
    // Connect to the database using the Mongodb package
    MongoClient.connect(mongoUrl, function (err, db) {
        if (err) throw err;
        // Check if the HTTP body contains both _id and ip
        if(request.body._id != null && request.body.ip != null) {
            var query = {_id: request.body._id};
            var newIp = {$set: {ip: request.body.ip}};
            // Save the IP to the database using the query above
            var dbo = db.db(dbName);
            dbo.collection(collectionNameHubs).updateOne(query, newIp, function (err, res) {
                if (err) { throw err; } // Send error if required
            });
        } else {
            res.send("Invalid parameters")
        }
    });
});
```

6.1.2 Register Application To Server

The above allows the smartphone application to communicate with the server as it can fetch the server's IP, however, as the hub should be able send notifications to the application it has to be able to communicate with the smartphone as well. In order to do so it was decided in section 5.2.1 to use Firebase Cloud Messaging. This allows us to focus on the development of our system instead of a supporting infrastructure. A Firebase token allows the hub to send information to the application through the Firebase back end. Figure 6.1 shows how the Firebase library triggers the onTokenRefresh, which in turn forwards the updated token to an endpoint on the server. The figure and relevant code is described in detail below.

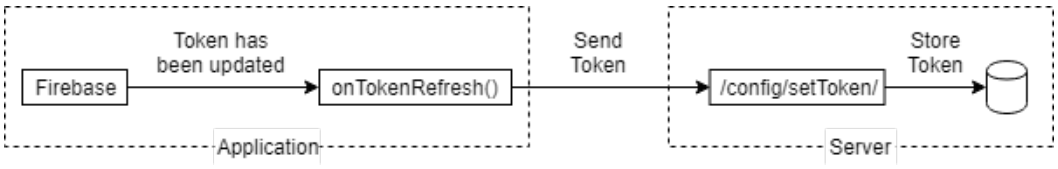


Figure 6.1: The registration process from the application to the server that stores the Firebase token.

When a token is update a message is sent from the Firebase library to the application which

triggers a service defined in the Android manifest. The manifest contains information about the application such as the application name, services, activities, etc. [115]. A service allows an application that is not currently active to perform long-running operations, e.g. media playback, file I/O, etc. [116]. In this case it is use to perform networking I/O to upload the Firebase token. Listing 6.3 shows how the service is defined with a filter so that the application only receives intents specified with the action on line 3. Furthermore, line 2 specifies the package name of the Java class to run when the service is triggered. This class has to extend "FirebaseInstanceIdService" in order to properly react to the service trigger.

Listing 6.3: XML code in the Android Manifest that defines the Firebase Instance service.

```
1 <!-- Class to call when token is updated -->
2 <service android:name=".utils.firebaseMsgService.MyFirebaseInstanceIDService">
3     <intent-filter>
4         <action android:name="com.google.firebase.INSTANCE_ID_EVENT" />
5     </intent-filter>
6 </service>
```

Listing 6.4 shows how the applications sends an updated Firebase token to the server. As mentioned earlier, this class has to extend "FirebaseInstanceIdService" in order for us to override the method onTokenRefresh() on line 1-2. This method is called when the application is notified of an updated token from the Firebase back end. Overriding this method allows us to perform any action we would like with the token even without the application being active, in this case it will be uploaded to the server and stored in its database. First thing to do in the overwritten method is to request the newly updated token from the Firebase instance, each application that uses a Firebase service is tied to a unique instance. In order to send an HTTP request from the smartphone a library called Volley is used [117]. A major benefit of using Volley is that it performs threading and scheduling automatically. The Volley HTTP request is defined on Line 9 in listing 6.4, this methods takes four arguments; the HTTP method which in this case is an POST request, an endpoint and two anonymous methods. The two latter handles the response from the server. Furthermore, as this is a POST request we need to put something in the body of the request for the server to react to. This is perform on lines 21-22 where the token is inserted into a Map that contains two strings, the key (token) and a value (refreshedToken). Finally, on line 25 the request is added to the Volley queue. The RequestQueue handles caching, responses and parsing it back to the main thread.

Listing 6.4: Java code in the Android application to send the refreshed token to the server.

```
@Override
public void onTokenRefresh() {
    // Request the token from the Firebase instance
    String refreshedToken = FirebaseInstanceId.getInstance().getToken();
    RequestQueue queue = Volley.newRequestQueue(context);
    String url = "http:///( ... )/config/setToken"

    // Construct the HTTP request using Volley
    StringRequest updateTokenRequest = new StringRequest(Request.Method.POST, url,
            new Response.Listener<String>() {
                @Override
                public void onResponse(String response) { ... }
            }, new Response.ErrorListener() {
                @Override
                public void onErrorResponse(VolleyError error) { ... }
            }) {
        @Override
        protected Map<String, String> getParams() // HTTP Parameters
        {
            Map<String, String> params = new HashMap<String, String>();
            params.put("token", refreshedToken);    // Insert the token as a parameter
            return params;
        }
    };
    queue.add(updateTokenRequest);  // Add the request to the queue to be sent
}
```

6.1.3 Application Request Server IP

Similarly to the router shown in 6.2, the server receives the updated Firebase Token and stores it in its database. Now the server has access to both the Hub's global IP and the smartphone application's most recent Firebase token. These allows the user to start the initial setup as described in 5.1.1. First, the user has to scan the QR code on the face of the hub. When the scan is successful the application will send a request to the server exchanging the hub ID for the hub's IP, which fulfills Server_FR3. Figure 6.2 shows a flowchart of the logic for the listings discussed in this section.

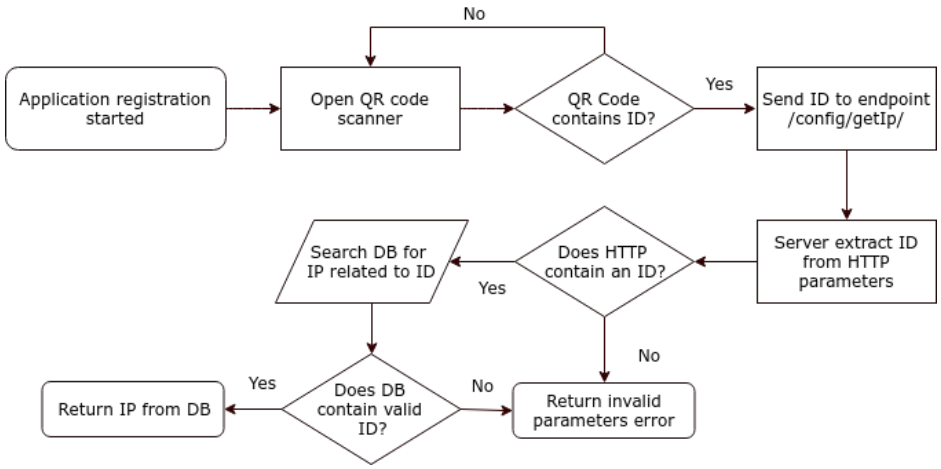


Figure 6.2: Flowchart of how the Android application scans the QR code, extracts the ID and sends it to the server.

To perform the QR code scanning within the application a library called ZXing was used. The library opens a new activity by using an IntentIntegrator with the camera viewfinder, this allows the user to point their camera to the QR code and for the application to scan it and extract the results. When the user is finished scanning the QR code the previous activity is relaunched and the method onActivityResult() is called. Similarly to the method from listing 6.4, this is also overwritten, which allows us to perform any action desired with the results from the QR code scan. The results from the QR code scanning is stored in an Intent object called data. The unique hub id stored in the physical QR code can be extracted by performing the operations on line 3. After on line 4 and 5 the results are first checked if the QR scanning was successful and if it could extract any information from the QR code respectively. If neither is null then we can send an HTTP request to the end described in listing 6.6 using Volley.

Listing 6.5: Java code in the Android application that is run after the QR code scanning has been performed.

```
1  @Override
2  public void onActivityResult(int requestCode, int resultCode, Intent data) {
3      IntentResult result = IntentIntegrator.parseActivityResult(requestCode, resultCode, data);
4      if(result != null) {
5          if(result.getContents() != null) {
6              // ... Send hub ID to server to receive the hub IP.
7          } else {
8              Log.d("QR Scan: ", "Failed"); // If the scan failed
9          }
10     } else {
11         Log.d(TAG, "QR Scan results = null"); // If the scan succeeded but no data was sent
12     }
13     // Execute the original non-overwritten method
14     super.onActivityResult(requestCode, resultCode, data);
15 }
```

Listing 6.6 shows how the router handles the HTTP request from 6.5, where App_NFR2 also is fulfilled. Similarly to listing 6.2 the server saves the contents of the HTTP request to its database, however in this case the endpoint specifies an HTTP Get request rather than an HTTP Post request. Due to this the parameters were put directly into the URI of the endpoint. In this case the endpoint is specified as "/getIp/:hubId" where hubId is replaced with the id of the hub when the hub sends the HTTP request, e.g. "/getIp/1001" where 1001 is the hub's unique id. After the result has been parsed a check if the :hubid parameter is extracted is specified on line 5, if it is then a search query in the database using the findOne operation with the id is performed on line 8. If it finds the object it returns the ip on line 12, however if it does not find any object it returns an error message on line 14.

Listing 6.6: JavaScript code in the Node.js server that receives the ID from the application and returns the related IP.

```
router.get('/getIp/:hubId', function(req, res, next) {
    MongoClient.connect(mongoUrl, function (err, db) {
        if (err) { throw err; }

        if(req.params.hubId != null) {
            var dbo = db.db(dbName);
            // Find document in the database
            dbo.collection(collectionNameHubs)
              .findOne({_id : req.params.hubId}, function (err, result) {
                if (err) { throw err; }
                if(result != null) {
                    var ipObj = {"ip": result.ip};
                    res.send(ipObj);
                } else {
                    res.send("Invalid hub id");
                }
            });
        }
    });
});
```

The previous allows the application to upload its Firebase token to the server and the server to save it to its database. Essentially allowing the server to send notifications to the application. This will be used to notify the user of high electricity usage as discussed in section 5.1.3.

6.2 Wi-Fi State Update

As discussed in section 5.5 the application should be able to report when the user enters or leaves the home, which also supports requirement App_FR10 and App_NFR1. To do this it was decided to use the SSID of the user's home Wi-Fi hotspot. First, it requires the user to explicitly tell the application which Wi-Fi should be treated as their home. This will be used to compare against other SSIDs they connect to in order to see if they are at home or not. Furthermore, it also requires the user to have Wi-Fi enabled on their smartphone at all times. During setup the user's home SSID is stored in a persistent storage build into Android called Shared Preferences. Shared Preferences is a non-volatile data storage that allows developers to save data in the form of key-value pairs [118].

With the user's home SSID saved to the Shared Preferences it is possible to compare it to the Wi-Fi SSIDs that they connect to. To do this a explicit broadcast receiver is used. A broadcast receiver allows the Android OS to send messages to applications based on events, e.g. plugging in a headset or when the phone receives an SMS [119]. In this case it is used whenever the Wi-Fi state has changed as discussed in section 5.5. Figure 6.3 shows the flow when the Wi-Fi state changes. The flow chart shows the logic of the listings in this section.

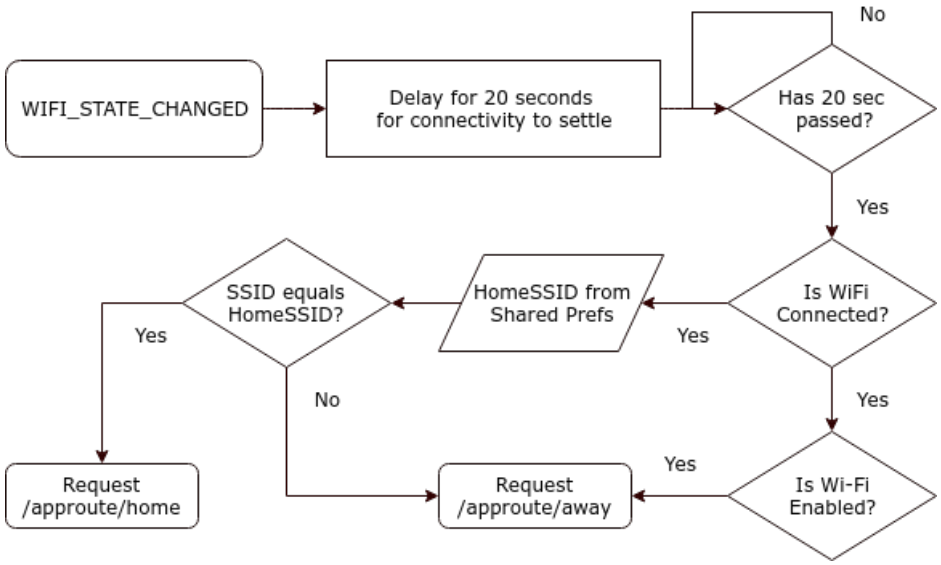


Figure 6.3: Flowchart of how the Android application sends the user status to the hub.

Listing 6.7 shows the desired Java class to be called (line 1) when a broadcast with the events WIFI_STATE_CHANGED (line 3) and STATE_CHANGE (line 4) are sent from the OS to the application. The first is used to find the status of the smartphones Wi-Fi, e.g. enabling, disabled, etc. The second event is used to get the connectivity information of the Wi-Fi when it is connected to a hotspot.

Listing 6.7: XML code in the Android application manifest that subscribes to the Wi-Fi state changes.

```
1 <receiver android:name="com.nicfol.thesis.utils.broadcastReceiver.WifiBroadcastReceiver">
2     <intent-filter>
3         <action android:name="android.net.wifi.WIFI_STATE_CHANGED"/>
4         <action android:name="android.net.wifi.STATE_CHANGE"/>
5     </intent-filter>
6 </receiver>
```

The Java class defined on line 1 in listing 6.7 are discussed below in listings 6.8, 6.9 and 6.10. The class "WifiBroadcastReceiver" extends the BroadcastReceiver from the Android OS. Similarly to how listings 6.4 and 6.5 overwrote a method, the same thing happens on lines 1-2 in listing 6.8. This method is called whenever a broadcast with the filters specified in the Android manifest are sent to the application. The thing this method does is delay all commands for 20 seconds in order for the Wi-Fi state and connectivity to settle. This is done with a Handler on lines 4-5 and then executed on line 8. Within the execution we use the ConnectivityManager in Android to get information about the type of Internet connection. This is used to see if the smartphone is connected to a Wi-Fi hotspot or not. The if statement on line 12 is executed if it is connected to a Wi-Fi hotspot and the else if statement in listing 6.10 is run if it is not.

Listing 6.8: Java code in the Android application that is execute on a Wi-Fi state change

```
1  @Override
2  public void onReceive(final Context context, final Intent intent) {
3  // Delay by 20 seconds to ensure a stable connection
4      final Handler handlerWaitForConnectivityToSettle = new Handler();
5      handlerWaitForConnectivityToSettle.postDelayed(new Runnable() {
6          @Override
7          public void run() {
8              ConnectivityManager connectivityManager = (ConnectivityManager)
                ↪  context.getSystemService(Context.CONNECTIVITY_SERVICE);
9              NetworkInfo networkInfo = connectivityManager.getActiveNetworkInfo();
10
11             if (networkInfo != null && networkInfo.getType() ==
                ↪  ConnectivityManager.TYPE_WIFI && networkInfo.isConnected()) {
12
13         ( ... )
14
15     }, 20000);
16 }
```

If the smartphone is connected to a Wi-Fi the code in listing 6.9 is responsible for extracting the SSID on line 1 and fetching the home SSID stored in Shared Preferences. When extracting the home SSID from shared preferences it is enclosed in quotations marks. A regular expression is used on line 3 and 9 in order to remove the leading and trailing quotation marks. Next the two SSIDs are compared in the if-statement on line 11 and calls a method that sends an HTTP request to the appropriate endpoint at the server. If the SSID matches

the home SSID then it sends an HTTP request to the hub stating that the user arrived home. Similarly, if it does not match then it tells the server that the user is not home.

Listing 6.9: Java code in the Android application that is execute on a Wi-Fi state change

```
1  WifiInfo wifiInfo = intent.getParcelableExtra(WifiManager.EXTRA_WIFI_INFO);
2  if(wifiInfo != null) {
3      String ssidFromCurrentConnectedWifi = wifiInfo.getSSID().replaceAll("^\"|\"$", "");
4
5      SharedPreferences sharedPreferences =
       ↪  PreferenceManager.getDefaultSharedPreferences(context);
6      String homeSsidFromSharedPrefs =
       ↪  sharedPreferences.getString(String.valueOf(R.string.HOME_SSID_KEY), null);
7
8      if (homeSsidFromSharedPrefs != null) {
9          homeSsidFromSharedPrefs = homeSsidFromSharedPrefs.replaceAll("^\"|\"$", "");
10
11         if(ssidFromCurrentConnectedWifi.equalsIgnoreCase(homeSsidFromSharedPrefs)) {
12             UpdateUserLocationToHub.updateLocationToServer(true, context);
13         } else {
14             UpdateUserLocationToHub.updateLocationToServer(false, context);
15         }
16     }
17 }
```

The above listing only triggers if the smartphone connects to a Wi-fi hotspot, however whenever the smartphone disconnects from a hotspot is an important event to check for. This event is used to tell if the user has left their home. On line 1 in listing 6.10 the else-if statement checks if the smartphone's Internet connect is not through Wi-Fi. If the connection is based on the cellular network then the if statement on line 3 checks if Wi-Fi is enabled. This check is to see if the Wi-Fi was disabled by the user, OS or another application. If it is still enabled then the applications reports that the user has left their home to the hub on line 4.

Listing 6.10: Java code in the Android application that is execute on a Wi-Fi state change

```
} else if (networkInfo != null && networkInfo.getType() != ConnectivityManager.TYPE_WIFI) {
    WifiManager wifiManager = (WifiManager)
     ↪  context.getApplicationContext().getSystemService(Context.WIFI_SERVICE);
    if (wifiManager != null && wifiManager.isWifiEnabled()) {
        UpdateUserLocationToHub.updateLocationToServer(false, context);
    }
}
```

The method updateLocationToServer() called in listings 6.9 and 6.10 calls an endpoint at the hub to report the current user status. The router that is responsible to taking care of this endpoint is shown in listing 6.11. This endpoint is designed similarly to the endpoint in listing 6.6, where the parameter is a part of the URL. If the applications sends an HTTP request to "/getIp/home" then the hub saves that the user is currently home. However, if it sends the request to "/getIp/away" then the hub saves that the user has left their home.

Listing 6.11: JavaScript code in the hub that receives the user status from the Android application.

```
router.get('/:status', function(req, res, next) {
    if (req.params.status.toLowerCase().localeCompare("away")) {
        isUserHome = true;
        res.send(isUserHome);
    } else if (req.params.status.toLowerCase().localeCompare("home")) {
        isUserHome = false;
        res.send(isUserHome);
    } else{
        res.send('Error');
    }
});
```

6.3 Data Gathering For Machine Learning Features

As mentioned in section 5.4, the hub had to be able to collect data from smart devices in order to fulfill Hub_FR6 and Hub_NFR5. Acquiring the data was as easy as calling certain API endpoints on RPi, since Home Assistant already has an API that can be consumed. A Z-Wave dongle was used to be able to communicate with the smart devices, which was required by Hub_NFR2. Section 5.2 described that the server would be running Node.js as a webserver, and we decided to also install a Node.js server locally on the RPi, because this

would make it possible to automatically call Home Assistants endpoint on a regular interval. A Node.js web server was scheduled to run on the hub at startup, which contained a cronjob that ran a function to gather data from smart devices every minute.

Listing 6.12: JavaScript in the hub that defines the cronjob that logs data once a minute.

```
1  let timeZone = "Europe/Copenhagen";
2  let job = new CronJob('1 * * * * *', function() { // create cronjob and run every min
3    getUserData.getUserData(); // run the getUserData function
4    console.log("Cronjob is running!");
5  }, function () {
6          console.log('Cronjob stopped'); // run if cronjob stops
7      },
8      true, // run right away (no delay)
9      timeZone
10 );
11 job.start(); // start the cronjob
```

Listing 6.12 shows a snippet of the Node.js server. Line 2 creates the actual cronjob, where the parameters can be adjusted depending on how often you want the cronjob to run. The cronjob was scheduled to run every minute, since that was specified in section 5.4. Everything between line 2-5 will be executed when the cronjob runs, and this is where we run a function called getUserData(). This function is what actually gathers the data, and it will be explained later in this section. The second parameter of the cronjob takes a function as an argument, which is executed when the cronjob stops. Lines 5-7 show that an anonymous functions is passed as an argument, which just prints out that the cronjob has stopped. Line 11 starts the cronjob, and it is run when the RPi has booted. An overview of getUserData() can be seen in figure 6.4, afterwards the figure and the method will be presented and discussed.

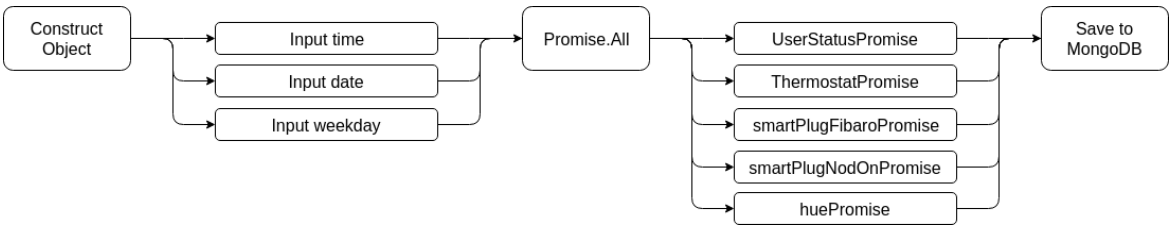


Figure 6.4: Overview of how getUserData() generates a data object to save to the MongoDB

The getUserData() function mentioned in the previous paragraph is the function that gathers

all the data by calling Home Assistant endpoints. The Home Assistant API can only be accessed if you use an appropriate headers in your calls, and only JSON is accepted. A header can be seen in listing 6.13, which contains *x-ha-access* with 1234 as a password. This password is configured in the Home Assistant configuration files. The *content-type* is also set to json, since that is what the Home Assistant API expects. This header is needed when calling the API, which will be described in the next paragraphs.

Listing 6.13: JSON object that defines the HTTP headers sent with the HTTP request to Home Assistant.

```
let hassHeaders = {
    headers: {
        'x-ha-access': '1234',
        'Content-Type': 'application/json' } };
```

The data needed was on/off status of smart devices, setpoint temperature of thermostat, power consumption from smart plugs, date, time, and week day. Getting this data was achieved by use of Promises, and the reason for using Promises is because they are easy to read and are asynchronous. Promises can have four states: fulfilled, rejected, pending, and settled. We've only utilized fulfilled and rejected for now. A promise has been made for every smart device, an example of one of the smartplugs can be seen in listing 6.14.

Listing 6.14: A JavaScript promise for gathering data from a smart plug.

```
smartPlugFibaroPromise = new Promise(function (resolve, reject) { // create promise
  let endpoint = piIp + ":8123/api/states/sensor.fibaro_system_fgwpef_wall_plug_gen5_power";
  request(endpoint, hassHeaders, function (error, response, body) {
    if (error) {
      reject(error) // return error if rejected
    } else {
      let jsonBody = JSON.parse(body); // parse JSON string to javascript object
      if(jsonBody.state > 0){ // check if smartplug is on or off
        let isOn = true;
      } else{
        let isOn = false;
      }
      var data =[
        {"isTurnedOn" : isOn},
        {"powerusage" : jsonBody.attributes.power_consumption}];
      resolve(data); //resolve promise with the data object
    }  });  });
```

The first line just creates the promise, where the second line shows the endpoint that is called for this particular smartplug. A simple /GET request will return the data about from the smartplug, but it requires that we include the header that was previously mentioned in this section. Line 3 shows that we use the *Request* library, which is used for making HTTP requests. In our example, the request function takes three arguments: endpoint, hassheaders, and a callback function. Endpoint contains the Home Assistant endpoint, hassHeaders contains the header, and the callback function is where we control whether the promise gets fulfilled or rejected. Lines 3-6 check if there was an error performing the HTTP request with the endpoint and hassHeaders as arguments, where it rejects the promise if there is an error. If there is no error, then it continues and allows us to receive the data from the request. Line 13-15 is where the data from the smartplug is stored as an object, and then line 16 resolves (fulfills) the promise with the smartplug data as an argument. The data is then accessible when we run the promise, which will be explained later in this section. As previously mentioned, this kind of promise runs for every smart device, but the home/away status of the user also has to be logged, which also is achieved through a promise that was similar to the ones for the smart devices, which also fulfills Hub_NFR7.

Listing 6.15: A JavaScript promise for getting the user's status (home/away).

```
1 userStatusPromise = new Promise(function (resolve, reject) {
2   let endpoint = "http://localhost:80/approute/status";
3   request(endpoint, function (error, response, body) {
4     // Get user data status from response and resolve
5     . . .
6   });
7 });
```

Listing 6.15 shows a promise where the endpoint mentioned in listing 6.11 is used on line 2. This endpoint is used to access the status of whether the user is home or away at the current time. The rest of the promise contains code to resolve the promise with the status or home or away, much like previously shown in listing 6.14.

The previous listings have shown how promises have been created that can access data from the different Home Assistant endpoints, but the promises have to be executed as well. Listing 6.16 shows that the Promise.all() function is used to run all the different promises created, which return the aggregated result of all the promises passed as arguments on lines 1-2. Then an empty object is created on line 5 as soon as all promises have been executed, where the data from the promises is stored (lines 6-12). The function then return the data stored in

the dataObj, which makes the the data accessible in the result variable in the next function (continues on listing 6.17).

Listing 6.16: The Promise.all() function is used to run all the promises in order to gather data.

```
1  Promise.all([   userStatusPromise, thermostatPromise,
2                  smartPlugFibaroPromise, smartPlugNodonPromise, huePromise])
3      .then(function (result) {
4          // Create empty object and store results from promises in object
5          let dataObj = { };
6          dataObj.isUserHome = result[0];
7          dataObj.setpoint = result[1];
8          dataObj.powersocket1 = result[2];
9          dataObj.powersocket2 = result[3];
10         dataObj.lights[0].light1isReachable = result[4].light1.reachable;
11         dataObj.lights[1].light2isReachable = result[4].light2.reachable;
12         dataObj.lights[2].light3isReachable = result[4].light3.reachable;
13
14         return dataObj; // return the dataObj in order to save to DB
15     }).then(function (result) {
16         // continued on next listing
17         . . .
```

The two previous listings have shown the use of promise chaining, which is when *.then* is used right after a promise (See lines 3 and 15 in listing 6.16). This means that when the first promise is finished, then continue to the next promise, which allows us to run asynchronous promises in a chain. Listing 6.17 continues the promise chain from the previous listing, and this is where the data is actually saved to the Azure Cosmos DB, which fulfills Hub_NFR6. Line 6 uses the MongoClient driver for Node.js to connect to the database using the connection URL for the database specified on line 4. The rest of the lines choose the appropriate collection in the database and update the data document.

Listing 6.17: End of the promise all from listing 6.16. This saves the data object to the DB if the promise is fulfilled.

```
// code from previous listing
}).then(function (result) {
    // ConnectionString to our Azure Cosmos DB (hidden since code is on Github)
    let mongoUrl = "mongodb://...";

    MongoClient.connect(mongoUrl) // connect to DB
        .then(function (db) {
            let dbName = "thesis";
            let collectionName = "dataColl";
            let query = {_id: "dataObject"};
            let dataToInput = {\$push: {data : result}};
            // update document with id "dataObject" with the data resolve in prev promise
            db.db(dbName).collection(collectionName).updateOne(query, dataToInput)
                .then(function (result) {
                    console.log("1 object pushed to " + collectionName + " in " + dbName);
                })
                .catch(function (err) {  console.log("DB Error: " + err);  });
            })
                .catch(function (err) { console.log("DB Error: " + err); });
            }).catch(function (err) {
                console.log("Promise.all failed");
                console.log(err);
            });
}
```

6.4 Price Notifier

As discussed in section 5.1.3, the price notifier located on the hub collects electricity prices every day to ensure that they are valid and up to date. This data will be gathered through an endpoint on Nord Pool's server. Through this endpoint it is possible to get data divided into various intervals, e.g. hourly, daily, weekly, etc. For this project the hourly prices will be used. This allows the prototype to alert the user when they have a high electricity consumption while the prices are high, and fulfills Hub_FR4 and App_FR3. Listing 6.18 shows how a promise is made, similarly to listing 6.14. An NPM package called Nord Pool will be used to fetch the prices. This package wraps a NPM Request method which makes it easier to pass the parameters as an object on line 3. The respond is saved to the results array, which allows us to traverse it through a for loop on lines 6-11 and save the prices to the prices array within the priceObj object on line 10.

Listing 6.18: JavaScript promise that fetches the electricty prices from Nord Pool's endpoint and resolves it as priceObj.

```
1  nordPoolPromise = new Promise(function (resolve, reject) {
2      // Request hourly Nord Pool prices
3      prices.hourly({area: 'DK2', currency: 'DKK'}, function (error, results) {
4          if (error) { reject(error); }
5
6          let priceObj = { "_id":  "", "prices" : [] }; // Object to save the DB
7          for (let i=0; i<results.length; i++) {
8              // Save the date
9              let date = results[i].date.tz('Europe/Copenhagen').format("YYYYMMDD");
10             if(i === 0) { priceObj._id = date; } // Define date if it's the first entry
11
12             priceObj.prices.push(results[i].value); // Push the data into the prices array
13         }
14         resolve(priceObj);
15     });
16 });
```

The object from listing 6.18 is saved to a collection within the servers MongoDB database. This object is exposed through an endpoint. This allows the hubs to call the endpoint and get a response with the daily prices. When the hub collects data from the smart plugs it compares the usage to the threshold specified by the user. If it is higher than the threshold then the hub sends a notification to the user's smartphone through Firebase. Do to this it sends an HTTP Post request to Firebase with the body elements on lines 8-14 in listing 6.19. The application receives this through the listings discussed in section 6.1.2.

Listing 6.19: HTTP options to send an notification through Firebase Cloud Messaging

```
1  var options = { method: 'POST', // HTTP Method
2      url: 'https://fcm.googleapis.com/fcm/send', // Endpoint
3      headers: { // HTTP Headers to send
4          Content-Type: 'application/json',
5          Authorization: 'key= ( API bearer key from Firebase )'
6      }, body: { // HTTP Body to send
7          to: '( Firebase instance key for devices to notify )',
8          collapse_key: 'type_a', // Collapse multiple notifications?
9          notification: {
10             body: 'One or more devices are consuming more than your specified threshold',
11             title: 'High Power Consumption!' }    },
12     json: true  };
```

6.5 Data Visualization

As seen in table 4.12, and more specifically App_NFR3, App_FR2, and App_FR9, the application should allow the user to view a visualization of the data that has been collected since the system has been setup. As it was decided in chapter 5 the application that should be incorporated into the system should be a Android Application, however, as there is no native functionality for creating or generating charts in a Android application, a library was the optimal way to go. There are many different open-source libraries available for one to use. It was chosen to use the library called MPAndroidChart [120]. The library is well documented and comes with a lot of different charts that can be implemented, which also has running examples attached to them. Other libraries offers the same functionality, however, as this seemed to be more widely used, it was chosen to be used.

Seen below in listing 6.20 we use the setData method in line 1. Next we create an ArrayList, which will be used as a container for the datapoints, that will be presented in the line chart. On line 6-9 a for-loop is created, iterating through based on the defined count. Before exiting the for-loop (line 8) new data values are added as entries in the defined ArrayList. This process is done each time a new line should be displayed on the graph. In line 11 we declare two LineDataSet, as these are the data sets that will be used.

Listing 6.20: Using the setData method in the MPAndroidChart library

```
1  private void setData(int count, float range) {
2  //Creating a new ArrayList
3  ArrayList<Entry> yVals1 = new ArrayList<Entry>();
4  ( ... ) // Collect data from the server's endpoint
5
6  for (int i = 0; i < count; i++) {
7      // Iterate through the data and insert into ArrayList
8      yVals1.add(new Entry(i, objFromEndpoint));
9  }
10
11 LineDataSet set1, set2;
```

In listing 6.21 we are still inside the setData method presented in listing 6.20, but here the data is given a type and a name in line 2. Line 3-11 defines how the data presenting the line should look like. See the documentation in MPAndroidChart [120] for descriptions on how you can customize the layout and look of the data and line.

Listing 6.21: Layout options for the graph for the data sets to be visualized.

```
1         // create a dataset and give it a type
2             set1 = new LineDataSet(yVals1, "Smart Plug 1");
3             set1.setAxisDependency(AxisDependency.LEFT);
4             set1.setColor(ColorTemplate.getHoloBlue());
5             set1.setCircleColor(Color.GREEN);
6             set1.setLineWidth(2f);
7             set1.setCircleRadius(3f);
8             set1.setFillAlpha(255);
9             set1.setFillColor(ColorTemplate.getHoloBlue());
10            set1.setHighLightColor(Color.rgb(244, 117, 117));
11            set1.setDrawCircleHole(false);
12
13
14            // create a data object with the datasets
15            LineData data = new LineData(set1, set2);
16            data.setValueTextColor(Color.WHITE);
17            data.setValueTextSize(12f);
18
19            // set data
20            mChart.setData(data);;
21
22        }
```

In line 15 in listing 6.21 above we define a new LineData object, which contains the two data sets that have just been defined above. In line 20 we call the setData method, and define that our newly declared object, called data (line 15) should be displayed in the chart.

Below in figure 6.5 a visualization of the final visualization activity in the android application can be seen. In the implementation, the legend are hard coded, however, it should be done so the name of the label of the two data sets are changed accordingly to the smart plugs name.

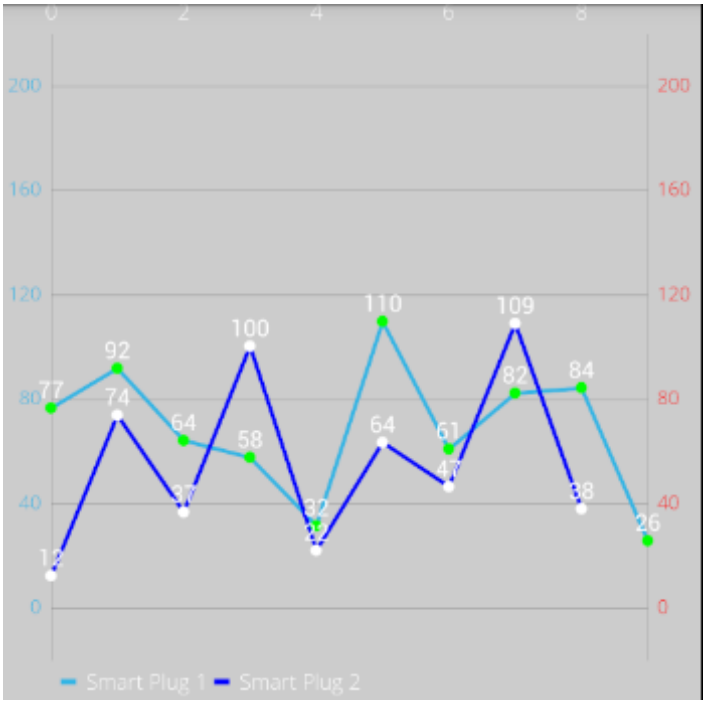


Figure 6.5: How the final visualization activity in the application ended up looking like

As presented in chapter 5, section 5.8 the data visualization should have several buttons allowing the user to set the interval of how the data should be shown, however, as that requirement (App_FR13) is only a should, it was not implemented.

6.6 Machine Learning

The platform that will be used for performing machine learning is Microsoft Azure Machine Learning Studio, which was presented in section 5.6. The platform is free to use as a developer with certain limitations, but it is possible to use a Pay-As-You-Go subscription, where you only charge for what you use. Our implementation did not require a subscription as the prototype was developed, but it would be needed if the prototype turned into a real product. The next sections will explain how we use Azure Machine Learning Studio, where it also can be found that Cloud_FR1 and Cloud_NFR2 were fulfilled.

6.6.1 Model For Forecasting User Status

The data that was to be used for machine learning had to be made ready in Azure. Figure 6.6 shows how the data was prepared. The first module, *ThesisDataset.csv*, was a comma-separated values (CSV) file that contained all the necessary data, where the columns *MinutesSinceMidnight* and *DayStatus* had been added to the dataset based on the gathered data.

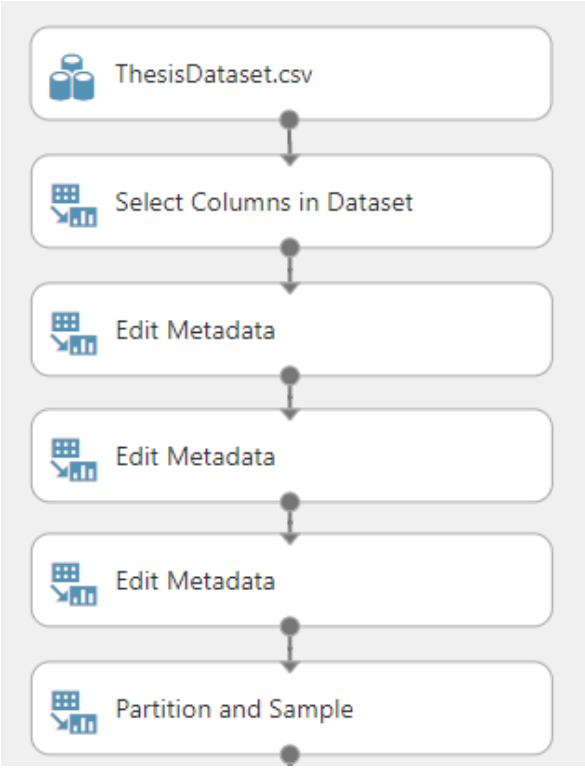
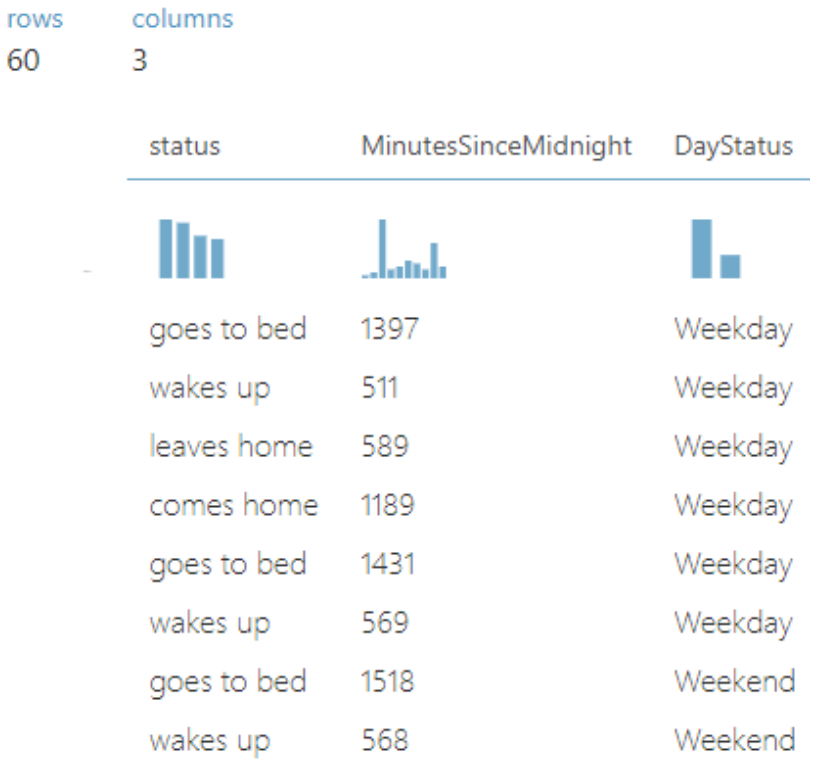


Figure 6.6: The dataset is first imported to Azure ML, where the columns UserStatus, MinutesSinceMidnight, and DayStatus were selected. UserStatus and DayStatus were selected as features, and MinutesSinceMidnight as label.

The next step is to select all the necessary data from the dataset, which was achieved by using *Select Columns in Dataset* that allowed us to choose specific columns from the dataset. The chosen columns were *UserStatus*, *MinutesSinceMidnight*, and *DayStatus*, which was decided in section 5.7.3. Figure 6.7 shows what the dataset looked like after selecting the columns.



rows	columns
60	3

status	MinutesSinceMidnight	DayStatus
goes to bed	1397	Weekday
wakes up	511	Weekday
leaves home	589	Weekday
comes home	1189	Weekday
goes to bed	1431	Weekday
wakes up	569	Weekday
goes to bed	1518	Weekend
wakes up	568	Weekend

Figure 6.7: A snippet from the dataset after the relevant columns had been selected. The snippet only shows 8 rows, where the dataset contains 60 rows in total

UserStatus and DayStatus were features and MinutesSinceMidnight was the variable we were trying to predict. Figure 6.6 shows three *Edit Metadata* modules, and these were used to specify which columns were features and which one was the target variable for prediction.

The last module seen on figure 6.6 is *Partition and Sample*, which was used to make partitions of the dataset in order to perform cross validation, which will be explained in the next section.

As mentioned in section 5.7.4, multiple regression algorithms were to be tested in Azure Machine Learning Studio before choosing a specific one. Figure 6.8, which is a continuation of figure 6.6, shows how multiple algorithms were tested at the same time.

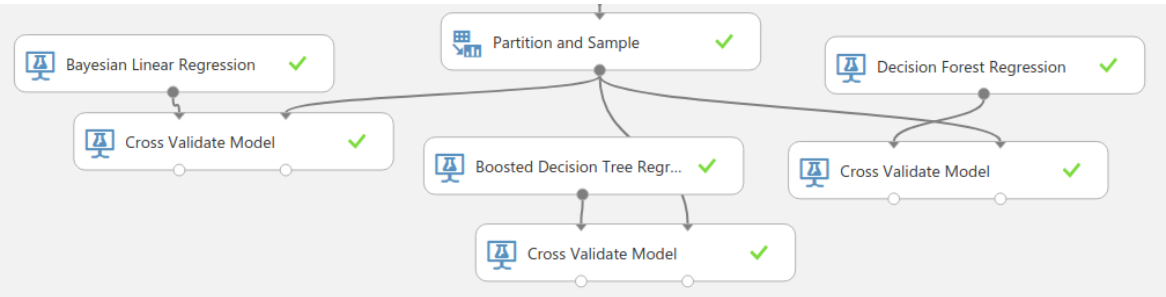


Figure 6.8: The partition and sample module is used to select a k-folds of 4, where multiple regression algorithms were used to train 3 different models.

Figure 6.8 shows that *Partition and Sample* is connected to multiple *Cross Validate Models*, where a type of regression algorithm is connected to each *Cross Validate Model*. The algorithms used were Bayesian Linear Regression, Decision Forest Regression, and Boosted Decision Tree Regression, since these were chosen in section 5.7.4. The *Cross Validate Model* is used to perform cross validation, which is used to make sure that the whole dataset is used for both training and testing. A model is often trained using approximately 80% of the data for training and 20% for testing, but the accuracy of the model can be mislead if many of the rows in the 20% of the testing data is noise. Cross validation help solve this problem, since the prediction accuracy can be seen as more reliable. The *Partition and Sample* module is also used to specify a number of folds, referred to as *k-folds*, where k-folds was set to 4. This resulted in four equally sized subsets of the dataset, where the training and testing data was different for each subset, and then the accuracy of all four folds was averaged into an overall accuracy score.

Figure 6.8 showed how multiple models were trained using different algorithms in order to find the best one, but this will be discussed in chapter 7. When a specific model was chosen, it was necessary to make a web service in Azure which made it possible to actually use the model and receive predictions.

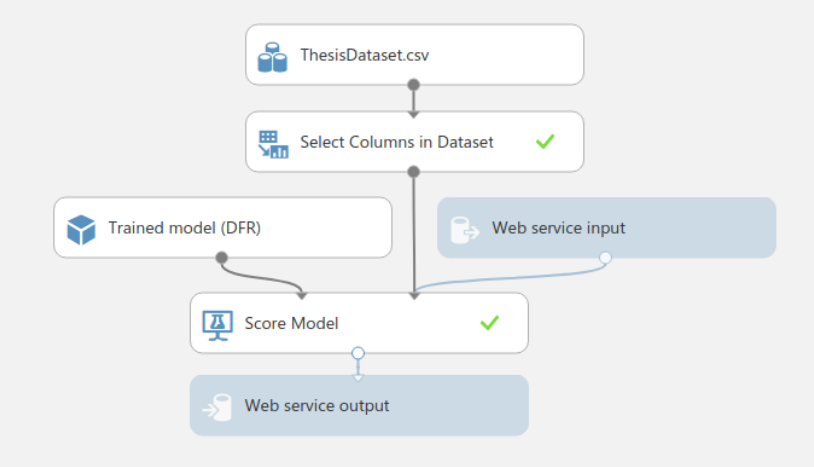


Figure 6.9: Web service in Azure that uses the trained Forecasting Model. This allows for HTTP calls to be made to the web service in order to receive predictions.

Figure 6.9 shows how a predictive web service was made. The *Web service input* module allowed us to make requests to the web service with JSON data as input. The input would then go through the model and the actual prediction was return through the *Web service output* module as a JSON response. A sample of the input used for a POST request to the web service can be seen in listing 6.22.

Listing 6.22: JSON data used as input for the Forecasting Model running on the web service presented in figure 6.9

```
{
  "Inputs": {
    "input1": {
      "ColumnNames": [
        "status",
        "MinutesSinceMidnight",
        "DayStatus"
      ],
      "Values": [
        [
          "wakes up",
          "0",
          "Weekday"
        ] ] } },
  "GlobalParameters": {}
}
```

Lines 5-7 in listing 6.22 contain the names for the features and label, and the actual values provided as input are seen in lines 11-13. This example tries to predict when the user will wake up (line 11) on a weekday (line 13). Line 12 contains the value 0, and this is only because the web service requires the any value for the target variable is sent to, but the value has no effect on the actual prediction. Listing 6.23 shows the result return from the web service.

Listing 6.23: The results received from the web service.

```
1  {
2      "Results": {
3          "output1": {
4              "type": "table",
5              "value": {
6                  "ColumnNames": [
7                      "status",
8                      "MinutesSinceMidnight",
9                      "DayStatus",
10                     "Scored Label Mean",
11                     "Scored Label Standard Deviation"
12                 ],
13                 "Values": [
14                     [
15                         "wakes up",
16                         "0",
17                         "Weekday",
18                         "533.803382034632",
19                         "29.1417167192072"
20                 ] ] } } } }
```

The results in listing 6.23 contain the same kind of structure which previously was presented in listing 6.22, where the actual prediction value can be seen on line 18. The returned value is 533.80, which means that the user will wake up 533.80 minutes after 00:00, which is approximately at 08:53.

6.6.2 Model For Real-Time Classification

The model for performing a prediction based on current data does not need the same kind of pre-processing of data as the model in section 6.6.1, because the raw data will be used for training the model. The features used were: time since midnight, current day, power consumption (smart plug 1), power consumption (smart plug 2), light status (light 1), light

status (light 2). The labels that were used for classification were home, away, and sleep. The model was trained the same way as presented in figure 6.8. There were three labels, which is why Multiclass classification algorithms in Azure were used. The algorithms were Multiclass Logistic Regression, Multiclass Decision Forest, Multiclass Decision Jungle, and Multiclass Neural Network. The evaluation of the algorithms will be discussed in chapter 7.

A web service in Azure was also made in order to be able to use the model for the prototype. The web service for this model was created just like the one presented in figure 6.9 in the previous section, where the classification model was used instead. An example of a request made to the web service can be seen in listing 6.24.

Listing 6.24: Input used for the Real-time Model.

```
"input1": [
  {
    "currentDay": "Tuesday",
    "powersocket1/1/powerusage": 0,
    "powersocket2/1/powerusage": 0,
    "lights/0/light1isReachable": 0,
    "lights/1/light2isReachable": 0,
    "UserStatus": "",
    "MinutesSinceMidnight": 600
  }
]
```

In the above example the current day is set to Tuesday and minutes since midnight is set to 600 (10:00 in the morning), where power consumption and light status is set to 0. The results from the prediction model can be seen in listing 6.25.

Listing 6.25: Output received from the web service running the Real-time Model.

```
"output1": [
    {
        "Scored Probabilities for Class \"Away\"": "1",
        "Scored Probabilities for Class \"Home\"": "0",
        "Scored Probabilities for Class \"Sleep\"": "0",
        "Scored Labels": "Away"
    }
]
```

The results in the above listing show that the algorithm predicted that the users status was

away, where line 2 indicates that it is 100% certain as the probability is 1.

6.7 Requirements Fulfillment

As it can be seen throughout this chapter, requirements have been fulfilled and incorporated into the system. An overview of the requirements prioritized with a 'Must' in the tables presented in section 4.8 are presented in table 6.1. Table 6.1 also provides an overview of in which section(s) they have been fulfilled, and what service(s) they have used in. The services are as followed: "Service 1 - Control and Monitor Smart Devices", "Service 2 - Adaptive Heating", "Service 3 - Adaptive Electricity", "Service 4 - Energy Visualization" and "Service 5 - Price Notifier"

Table 6.1: Functional requirements and their implementation description

Req.	**Fulfilled in sections(s)**	**Used in service(s)**
Hub_FR1	6.1	1, 2, 3
Hub_FR2	6.1	1, 2
Hub_FR3	6.4	5
Hub_FR4	6.4	5
Hub_FR5	6.6	2, 3
Hub_FR6	6.3	2, 3, 4, 5
Hub_NFR1	6.1	1, 2, 3, 4, 5
Hub_NFR2	6.3	1, 2, 3, 4, 5
Hub_NFR3	6.4	4, 5
Hub_NFR4	6.4	4, 5
Hub_NFR5	6.3	2, 3, 4, 5
Hub_NFR6	6.3	2, 3, 4, 5
Hub_NFR7	6.3	2, 3
App_FR1	6.1	1
App_FR2	6.5	4
App_FR3	6.4	5

Req.	**Fulfilled in sections(s)**	**Used in service(s)**
App_FR6	6.1	1, 2
App_FR7	6.1	1, 2, 3
App_FR9	6.5	4, 5
App_FR10	6.2	2, 3, 5
App_NFR1	6.2	2, 3, 5
App_NFR2	6.1.2	1, 2, 3, 4, 5
App_NFR3	6.5	4
App_NFR4	6.1.2	1, 2, 3, 4, 5
Server_FR1	6.1.1	1, 2, 3, 4, 5
Server_FR2	6.1.1	1, 2, 3, 4, 5
Server_FR3	6.1.3	1, 2, 3, 4, 5
Server_FR4	6.4	4, 5
Cloud_FR1	6.6	2, 3
Cloud_FR2	6.6	4, 5
Cloud_NFR1	6.6	2, 3
Cloud_NFR2	6.6	2, 3

7 Test & Evaluation

In this chapter the test environment and setup that was in the data collection period will be presented. The data collected during the data gathering period will be analyzed and evaluated. Furthermore, the different machine learning algorithms will be evaluated and an algorithm for each model will be chosen. A test was performed in order to show how the system will react in a real world setting.

7.1 Test Environment and Setup

The test was conducted in Copenhagen in Denmark in April to May. The system that was setup includes two smart plugs, one smart thermostat, two smart light bulbs, the hub, and the android application. The system was setup in an 28 m^2 apartment on Amager. A floor plan of the apartment can be seen below in figure 7.1, where it also is possible to see the connected smart devices and where they were placed and how they are connected to the hub, which communicate with the server, Azure, and the application.

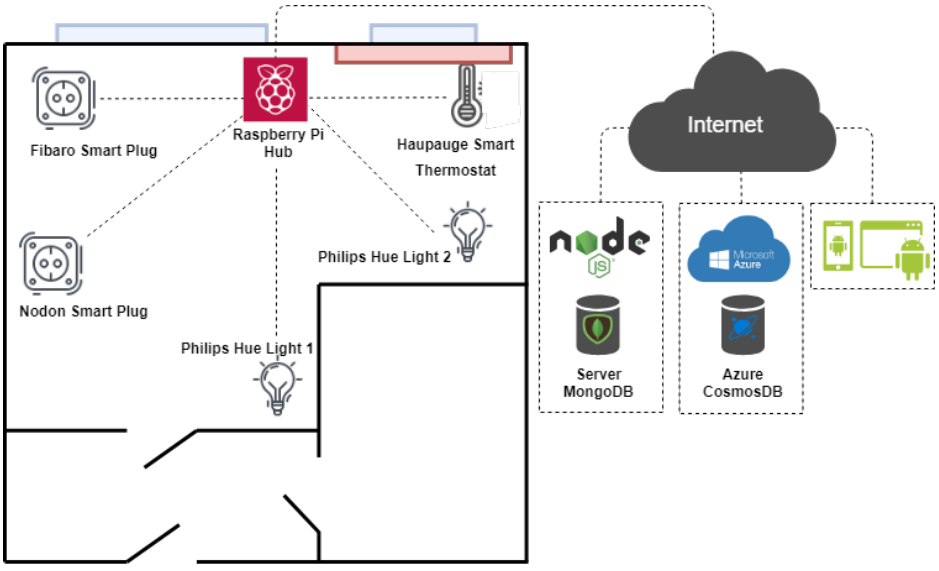


Figure 7.1: Top-down view of the apartment, the connected devices, and how the overall architecture are. The blue rectangles are windows, and the red one is a radiator.

As it can be seen in figure 7.1 the Fibaro Smart Plug is connected to the power outlet that

supplies the residential computer with electricity. The Nodon Smart Plug is connected to the power outlet that provides electricity for the TV. As the apartment only have one radiator, the smart thermostat was setup on that. The smart lights are placed in the two living room lamps. See table 7.1 for an overview of the smart devices.

Table 7.1: Overview of the smart devices used, what they were connected to, and their measuring units

Device name	**Connected to**	**Measuring Units**
Fibaro smart plug	Desktop Computer	Watt
Nodon smart plug	Television & Console	Watt
Haupauge Smart Thermostat	Radiator	Degree Celsius
Philips Hue Light 1	Living Room Lamp 1	On/Off
Philips Hue Light 2	Living Room Lamp 2	On/Off

7.2 Data Collected

The data collection period started on the 25th of April and and ended on the 30th of May, meaning that it lasted for 36 days. As presented in chapter 6, the prototype collected data from the devices once every minute, which results in 1440 data points for each day. Each of these data points contains information from the connected smart devices and the android application presented in section 7.1 above.

As mentioned in section 4.7.2 and Hub_FR6, and App_NFR1, the data that has been collected is the device status, setpoint temperature, power measurement, time, date, weekday, and the users home/away status. The collected data and the units of measurement can be seen in table 7.2 below.

Table 7.2: Overview of the different features, where they are from, and their unit of measurement

Feature name	Data from device	Measuring Units
Device Status	Philips Hue Light 1 & 2	On/Off
Setpoint temperature	Smart Thermostat	Degree Celsius
Power consumption	Fibaro & Nodon smart Plug	Watt
Time	The hub	HH:MM
Date	The hub	DD/MM/YYYY
Weekday	The hub	Day
Home/Away status	Android Application	Home/Away

7.2.1 Data Presentation

As mentioned in section 5.7.3 and 4.7.3, the data had to be preprocessed in order to find the four statuses presented in section 4.7.3. In table 7.3 the different statuses are presented along with their day, their averaged value in minutes since midnight, the standard deviation (σ), and the accepted timespan ($\pm 1\sigma$). The accepted timespan indicates values that falls into the category of being a valid data point, whereas values that falls out of that timespan are outliers. The timespan was chosen to be $\pm 1\sigma$, as it still counts for 68.2% of the data. If $\pm 2\sigma$ was chosen, only six outliers would have been excluded, however, by selecting $\pm 1\sigma$ the outliers increase to 14. A table for indicating the accepted timespan with $\pm 1\sigma$ can be seen, as a table showing $\pm 2\sigma$ would look the same, just with a much greater accepted timespan.

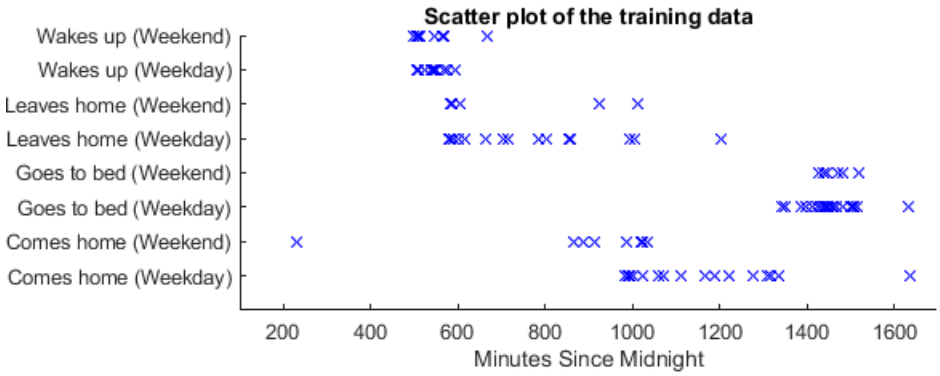


Figure 7.2: A scatterplot of the data points collected for each status, and their position relative to each other

Table 7.3: This table shows the statistical values for a specific status, the day, the average time, the σ, and the accepted timespan

Status	**Day**	**Average time (Minutes since midnight)**	**σ (Minutes since midnight)**	**Accepted Timespan ($\pm 1\sigma$)**
Goes to bed	Weekday	1451 (00:11)	59	1393 - 1511 (23:13 - 01:11)
Wakes up	Weekday	530 (08:50)	25	505 - 555 (08:25 - 09:15)
Leaves home	Weekday	733 (12:13)	180	554 - 913 (09:13 - 15:12)
Comes home	Weekday	1167 (19:27)	167	1000 - 1334 (16:40 - 22:14)
Goes to bed	Weekend	1463 (00:23)	30	1433 - 1494 (23:53 - 00:54)
Wakes up	Weekend	540 (09:00)	51	489 - 591 (08:09 - 09:51)
Leaves home	Weekend	682 (11:22)	177	505 - 860 (08:25 - 14:20)
Comes home	Weekend	869 (14:29)	267	602 - 1136 (10:02 - 18:56)

A visual presentation of the data that has been collected and used can be seen in figure 7.2. Each of the statuses for weekdays and weekend are presented on the y-axis, and minutes since midnight are presented on the x-axis. The data are presented in table 7.4.

With the timespan provided in table 7.3 it is possible to further analyze on the data that has been collected. The number of occurrences of each status, their average value, number of outliers, and the average value without outliers for the weekdays can be seen below in table 7.4. An example of an outlier can be seen on the 23rd of May, where the user leaves the household at 16:32. In the other instances, the user leaves the home between 09:49 and 14:18.

Table 7.4: Occurrences, averaged result, no. of outliers, and averaged value without outliers, of the different status' in the data collection period, for the weekdays

User status	**Occurrences**	**Averaged result (HH:MM)**	**No. of outliers**	**Average without outliers (HH:MM)**
Wakes up	23	08:52	3	08:42
Leaves home	19	12:23	3	11:18
Comes home	18	17:34	4	19:23
Goes to bed	24	00:11	4	00:48

As it can be seen in table 7.4, the test subject on average wakes up at 08:52 and 08:42 without the outliers. While the difference with or without the outliers are not that high, the difference between when the user leaves home with and without outliers are more than 60 minutes. The difference in time between status 'Comes home' with and without outliers are almost two hours, which is quite high. Lastly the difference between the result with and without the outliers in the status 'Goes to bed' are 37 minutes.

Table 7.5: Occurrences, averaged result, no. of outliers, and averaged value without outliers, of the different status' in the data collection period, for the weekends

Use status	**Occurrences**	**Averaged result (HH:MM)**	**No. of Outliers**	**Average without Outliers (HH:MM)**
Wakes up	10	09:08	1	08:51
Leaves home	8	11:55	2	09:45
Comes home	8	14:02	1	16:05
Goes to bed	8	00:34	2	00:25

Table 7.5 presents the same data as table 7.4 above for the weekend instances. In the weekends the user wakes up at 09:08 with the outlier included, but without the outlier, it changes to 08:51. On average the user leaves the home at 11:55 with the outliers, and 09:45

without them, which is a more than two hours difference. Likewise the difference between the user coming home with and without outliers is two hours as well. The difference between the status 'Goes to bed' with and without outliers are only 10 minutes. The outliers can still be useful as they still represents the user's behaviour and will be discussed in chapter 8.

In figure 7.3 a visual presentation of how the power consumption was throughout a day. The data presented is from the 15th of May, and this presentation is how the majority of the data looks like. As it can be seen in figure 7.3, the x-axis is time of the day in hours, starting from 00:00 and going to 23:59, and the y-axis is the power consumption in Watt. The power consumption presented in figure 7.3 is the sum of the data collected from the Fibaro and Nodon smart plugs. The blue line displays the sum of the two smart plugs, while the red line indicates the Wi-Fi status of the user. The red line in figure 7.3, 7.4, and 7.6 represents the user's home/away status. If the user was home it is set to the highest total power consumption and if he was away it was set to zero.

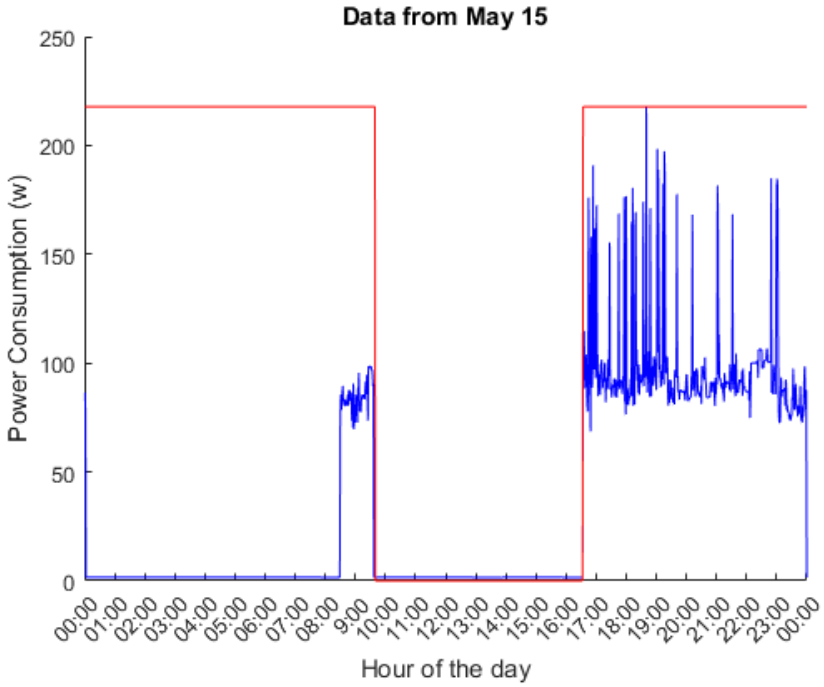


Figure 7.3: How the energy consumption of the majority of the normal days looked like. The blue line indicates power consumption. The red line indicates whether or not the user is home (high) or away (low), where the shown values goes from 0 to the maximum value of the power consumption measured.

In figure 7.3 the test subject wakes up at around 08:30 as there is an increase in the power

consumption, which again decreases at approximately 09:30 as the user leaves the home, this is indicated by the red line as it is set to zero. At around 16:30 there is a large spike in power consumption, which indicates that the user arrived home and turned on both of the devices (computer and TV) that were connected to the smart plugs. The remaining time there is a lot of small spikes, which is due to the user turning on or off either of the devices connected to the Smart Plugs. Figure 7.3 clearly depicts a pattern of the user, and more examples of graphs which present the same can be found in Appendix C. The graphs are from the 26$^{\text{th}}$ of May to the 30$^{\text{th}}$ of May.

Figure 7.4 is a zoomed version of figure 7.3, where there has been put a emphasis on the data where the user arrives home. There is a large spike in the power consumption, which is due to the test subject arriving home and turning on the TV and the computer at approximately the same time.

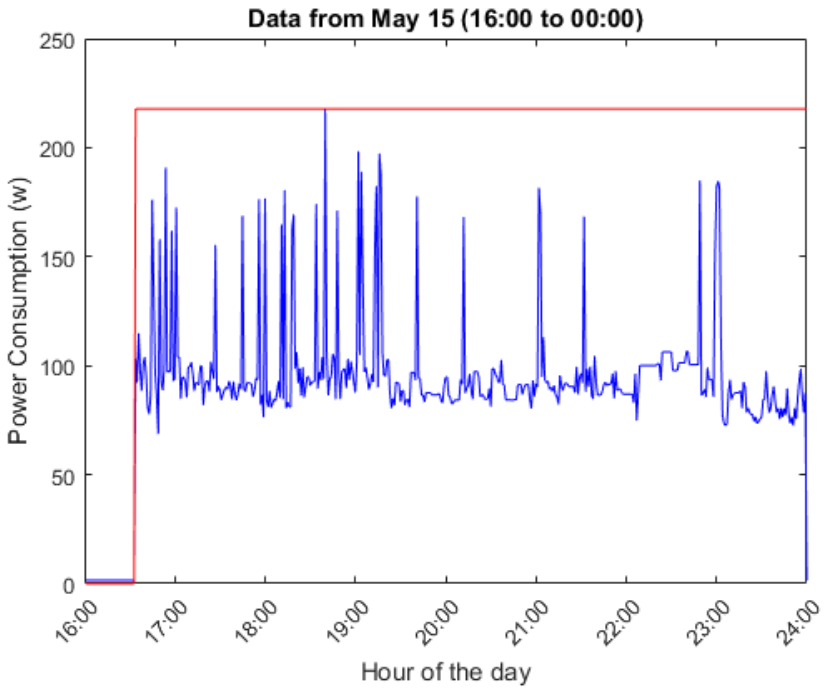


Figure 7.4: Data from 15$^{\text{th}}$ of May, zoomed in from 16:00 to 00:00. The blue line indicates power consumption. The red line indicates whether or not the user is home (high) or away (low), where the shown values goes from 0 to the maximum value of the power consumption measured.

Figure 7.5 was analyzed in order to understand the peaks of power consumption with the help of the user's notes, since he was asked to write down what he was doing on the computer

during a day. The data is from the 1st of June. The y-axis is power consumption for the computer, and the x-axis is time. As it can be seen in figure 7.5 there are two peaks; one from 21:20 to 21:21 and a second one from 21:25 to 21:26. The peaks in the power consumption can be explained as the user perofrming heavy computational tasks on the computer. In the first instance the user compiled an application in Android Studio, which explains the increase in power consumption. At the next peak the user started a Virtual Machine which explains the increase in power consumption.

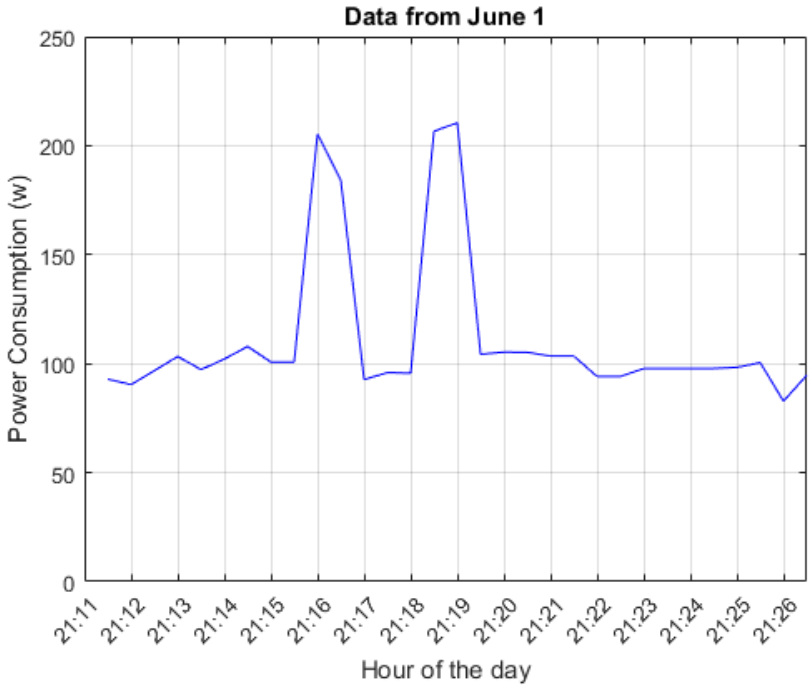


Figure 7.5: How the peak energy consumption looks, and the cause. The blue line indicates power consumption.

However, there are also days where the pattern shown above is disturbed or changed. These changes can be seen below in figure 7.6. The majorities of these are during the weekend, where the user wakes up later and leaves the home later, if at all. An instance of where the user changes his schedule in the weekday can be seen in figure 7.6, where the data presented is from the 9th of May. The test subject wakes up at around 08:30, but does not leave the household until around 11:00, which is later than the other days. Furthermore, the user comes home at around 18:30, which is later than the user's usual pattern. The user then leaves the household again at around 20:00, which is unusual. The user comes home at 03:13 the next day, but this is not seen in the graph.

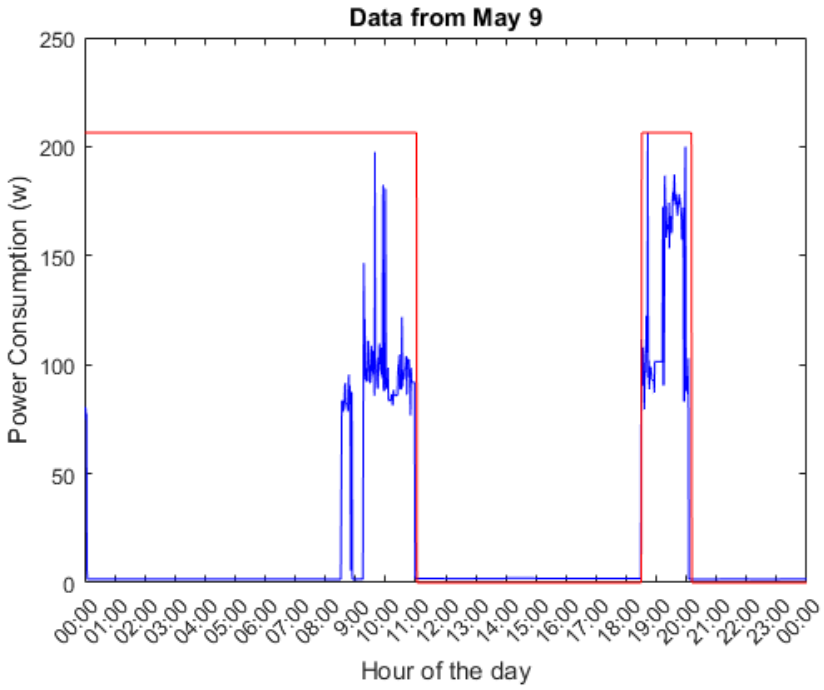


Figure 7.6: How the energy consumption is throughout an outlier day. The blue line indicates power consumption. The red line indicates whether or not the user is home (high) or away (low), where the shown values goes from 0 to the maximum value of the power consumption measured.

The day presented in figure 7.6 is a good presentation of how some of the outlier days look like. Nonetheless, as the data collection period only consisted of 36 days, these can affect the machine learning algorithm quite a bit. This will further be discussed in chapter 8.

The data presented in figure 7.7 is from the period between 25th of April to 20th of May. In figure 7.7 a presentation of the collected setpoint temperature data can be seen. Figure 7.7 presents data that covers several days, which is done in order to give a better understanding of why the setpoint temperature did not contribute to the feature extraction presented in section 5.7.3.

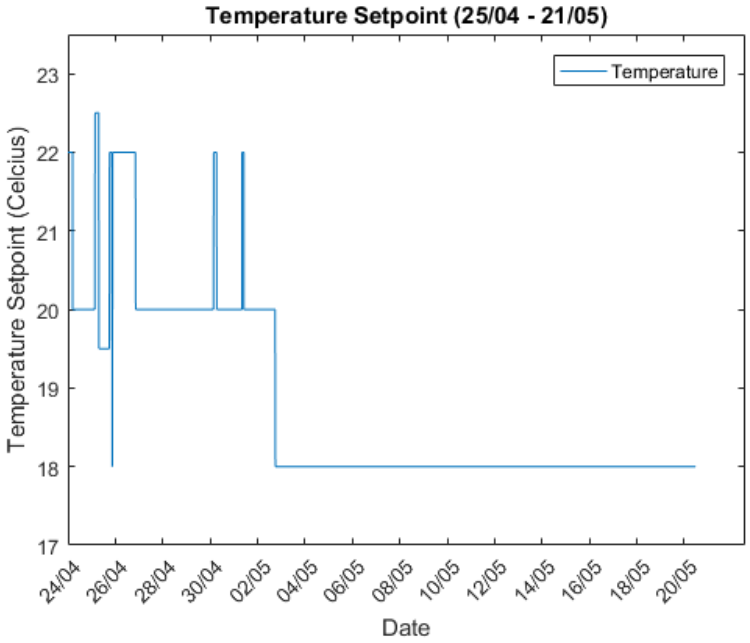


Figure 7.7: How the setpoint temperature changed throughout the data collection period

Figure 7.7 presents data from 26 days. The setpoint temperature only changes between the 25th of April and the 4th of May, and then it stays at 18° Celsius throughout the rest if the days. As the changes only covers a very small amount of the data collected, it would be hard to indicate whether or not any energy reductions could have been achieved. This will further be discussed in chapter 8.

7.3 Forecasting Model Evaluation

The various algorithms discussed in section 5.7.4 were evaluated and tested. The test method is discussed in Appendix D. The data set containing the evaluation results, mentioned in Appendix E, consists of five different metrics. These metrics all describe how far the model predictions were from the true value, that is the correct value. The five metrics results can be seen in figure 7.8, whereas an explanation of the metrics can be seen in Appendix E. The four first metrics (Mean Absolute Error, Root-Squared Mean Error, Relative Absolute Error and Relative Squared Error) should be equal to zero for a perfect fit between the model and data, since they all measure the size of the errors. The last metric is the Coefficient of Determination (R^2), it shows how two variables correlate to each other. The R^2 returns a value between 0 and 1, where 0 means that there is no correlation and 1 means that there is

a perfect correlation. Thus, the higher R^2 the better. The results of the model training can be seen in figure 7.8.

Algorithm	Mean Absolute Error	Root Mean Squared Error	Relative Absolute Error	Relative Squared Error	Coefficient of Determination
Bayesian Linear Regression	107.117931	155.133034	0.317647	0.195207	0.804793
Boosted Decision Tree Regression	95.552782	145.75375	0.281974	0.16621	0.83379
Decision Forest Regression	98.849701	147.096629	0.291893	0.170758	0.829242

Figure 7.8: Evaluation results for each algorithm. The explanation of what the metrics mean can be seen in Appendix E. The model with highest coefficient of determination was chosen, which was BDTR.

Based on the results in figure 7.8, the BDTR scored the best across all metrics. It received the lowest values in the RMSE and RSE, both of which punishes errors harsher than the others. In addition, it also had the highest R^2, showing a fit of 83%. Due to this the BDTR model was chosen for the prototype. It is worth noting that the DFR scored very closely to the BDTR, as such it may be worth reevaluating the models when more data have been collected.

7.4 Real-time Model Evaluation

The second model was a classification model developed in section 6.6.2. This model was used to get live predictions on whether the user was at home, away, or asleep. The input data was fetched from the smart sensors directly and used as input, along with time and day. The algorithms used were mentioned in section 5.7.2, namely MLR, MDF, MDJ, and MNN. The results of the algorithms can be seen in table 7.6.

Table 7.6: Evaluation results of the classification algorithms. The values have been rounded down to three decimal digits. The original numbers can be seen in Appendix F

Algorithm	**Overall accuracy**	**Average accuracy**
MLR	0.907	0.938
MDF	0.965	0.977
MDJ	0.951	0.967
MNN	0.922	0.948

The overall accuracy metric presented in table 7.6 shows how often the model predicted the correct outcome, where e.g. 0.90 would mean that it predicted correctly 90% of the time. The algorithm that performed best was MDF, and the model was therefore trained using this algorithm.

7.5 Energy Consumption Reduction

We decided to run an experiment using predictions from the BDTR model. This was to see if there would be a difference in power consumption if the system had automatically controlled the smart devices based on the prediction results. For example, if the algorithm predicts that the user will leave home at 09:30 on a weekday, then lights and smart plugs would be turned off automatically. Ideally, the system should predict different results for every day in the week, however, our model only used weekdays and weekend as features, because we did not have enough data to use individual days as features. This means that predictions for when a user comes home will be identical from Monday to Friday, since these are considered weekdays. The same applies to the other user's status changes (goes to bed, wakes up, leaves home). Nonetheless, an attempt was made to analyze the power consumption data from the 22^{nd} to the 30^{th} of May where the user was using the system, and compared it to the power consumption that would have been used if the system was controlling smart devices automatically based on predictions. Most of the data from the 24^{th} and the 25^{th} of May was missing, and it was therefore excluded. The first step of the experiment was to get predictions for each user status depending on whether it is a weekday or weekend. Table 7.7 shows a list of the predictions received from the Forecasting Model using UserStatus and DayStatus as input.

Table 7.7: This table shows what the model predicts based on user status and day, which returns minutes since midnight that is converted to time (hour and minute)

User status	**Day**	**Minutes since midnight (Prediction)**	**Time (HH:MM)**
Goes to bed	Weekday	1452	00:13
Wakes up	Weekday	533	08:53
Leaves home	Weekday	744	12:24
Comes home	Weekday	1174	19:34
Goes to bed	Weekend	1474	00:34
Wakes up	Weekend	548	09:08
Leaves home	Weekend	715	11:55
Comes home	Weekend	843	14:03

After the timestamps, presented in 7.7, were found, multiple line charts were made showing the user's total power consumption during each day between the 22nd and 30th of May compared to power consumption if the system automatically turned off devices on the forecasted timestamps. Two graphs will be shown for 22nd and 23rd of May, the rest of the days can be found in Appendix C.

Figure 7.9 is based on 22nd of May (Tuesday), which shows that the model predicts that the user will go to bed at 00:13, but he went to bed at around 00:20. The model predicted that the user woke up at 08:53, but he actually got up at around 08:30. During the night, the blue line goes down to 0 Watt, since this is to illustrate that the system automatically turned off devices that were on standby during the night. At 12:24 it predicts that the user leaves the home at 12:24 and comes back home at 19:34. The user actually left at 10:02 and came home at 17:47. The blue line is only set to 0 if the user's status changes with the timespans defined in Appendix G.

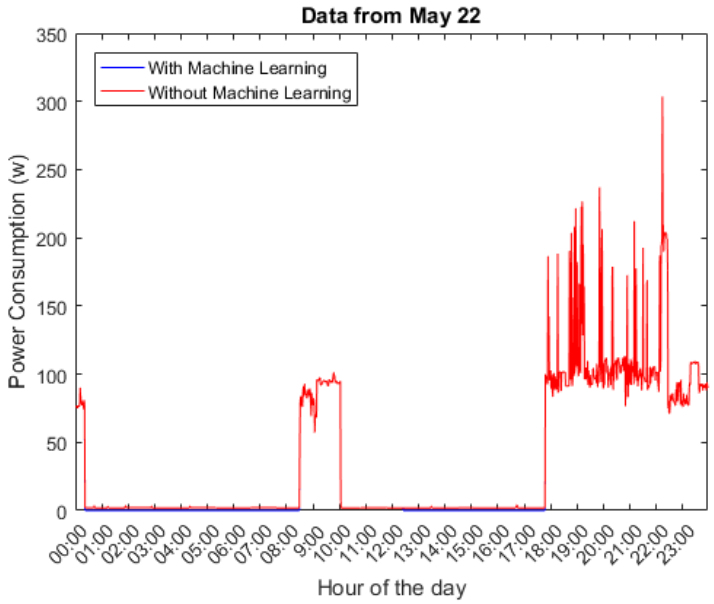


Figure 7.9: Comparing the actual data with the prediction provided by the algorithm. The red line is power consumption without machine learning, and he blue is with machine learning. The data are from the 22nd of May

Power consumption from the 23rd of May (Wednesday) can be seen in figure 7.10. The prediction for when the user went to bed is only 15 minutes earlier than what happened in reality, and the prediction for waking up was only 7 minutes before he actually woke up. This results in the Real-time Model being able to change user status at the appropriate times. The power consumption fluctuated between approximately 100 to 200 Watt to around 16:10, where the user actually still was home. From 16:10 to around 20:40 the total power consumption was between 90-100 Watt, but the data gathered during this period shows that the user was not home. If the Forecasting Model had been more precise, it would be able to turn off the smart plug, reducing the energy consumption between 16:10 and 20:40.

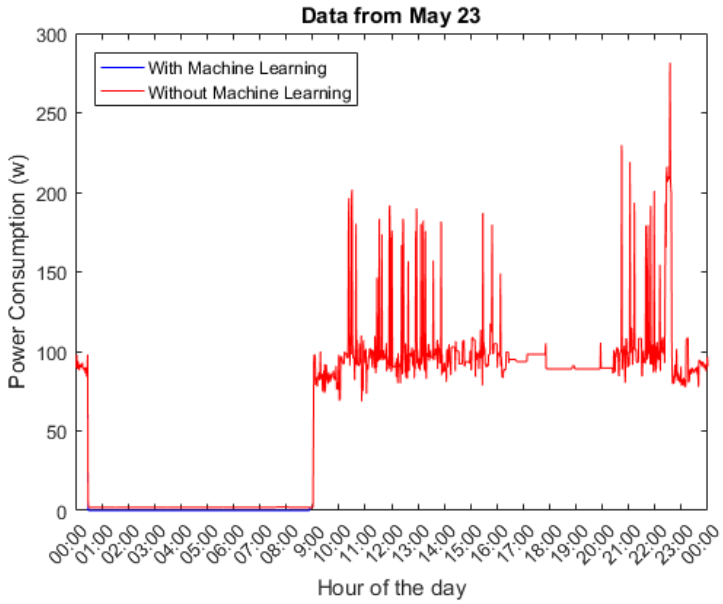


Figure 7.10: Comparing the actual data with the prediction provided by the algorithm. The red line is power consumption without machine learning, and he blue is with machine learning. The data are from the 23rd of May (Weekday)

The data from 22nd to 30th of May has been analyzed in order to find whether or not energy could have been reduced. The average power consumption for each of those days was calculated with and without the machine learning models. The total power consumption was set to 0 whenever the model predicted the user was away or asleep. The results can be seen in table 7.8.

Table 7.8: Power consumption from 22nd of May to 30th of May with and without ML (averaged). Difference in percentage is also showed.

Date	**Watt without ML (average)**	**Watt with ML (average)**	**Difference in percentage**
22nd May	36.08	34.97	-3.07%
23rd May	64.15	63.47	-1.07%
26th May	37.24	36.23	-2.72%
27th May	46.58	36.26	-22.15%
28th May	63.78	63.19	-0.93%
29th May	45.51	44.57	-2.07%
30th May	17.00	13.77	-18.97%
Average	**44.33**	**41.78**	**-7.28%**

Similarly to the presentation of the power consumption, the same was done for the setpoint data of the smart thermostat. As discussed in section 7.2.1, the setpoint data from the smart thermostat was not used for the feature extraction. Figure 7.7 shows how the user only changed the setpoint for the first 10 days, afterwards the setpoint was set to 18° Celsius for the remainder of the test period. Due to this it was assumed that the pattern the user displayed for the first 10 days would continue in the final seven days that the prototype was evaluated in. Using the assumptions on the data logged about the user's home/away/sleep status allowed us to estimate the energy usage during the period from 22nd to 30th of May and compare this to how the energy consumption would be with the machine learning model applied. As such, the assumptions for calculating the energy used on heating in this period were:

1. The user would continue to adjust the radiator thermostat
2. During the daytime the setpoint would be 22° Celsius
3. During the nighttime and when the user was away the setpoint would be 20° Celsius

The setpoint of the thermostat was adjusted to 17° Celsius when the user was away from home. This setpoint was chosen based on the system presented in section 3.3.1. When the user was asleep the system adjusted the setpoint to 20° Celsius and 22° Celsius when he was awake, these were chosen based on the habits the user showed in figure 7.7. With this the average setpoint was calculated with and without the results from the machine learning models. Table 7.9 shows the average setpoint both without and with the models as well as the difference between the two in percentage.

Table 7.9: Setpoint from 22nd of May to 30th of May with and without ML (averaged). Difference in percentage is also showed.

Date	**Average setpoint without ML (°C)**	**Average setpoint with ML (°C)**	**Difference in percentage**
22nd May	20.63	19.66	-4.70%
23rd May	20.91	19.78	-5.39%
26th May	20.84	19.72	-5.38%
27th May	20.89	19.88	-4.83%
28th May	21.25	21.25	0%
29th May	20.80	19.89	-4.36%
30th May	20.35	18.84	-7.44%
Average	**20.81**	**19.86**	**-4.59%**

The estimation generally showed a reduction in the daily energy usage of between 4.36% and 7.44%. The results show an average reduction of 4.59% in the energy usage between the assumed real world and the assumed real world with the machine learning models applied. A representation of the difference between average in table 7.8 and table 7.9 can be seen in Appendix H.

7.6 Requirements Testing

The fulfilled requirements presented in table 6.1 in chapter 6 have not been tested directly, however, one could argue that the prototype has been made ready for the end-users to use. However, as none of the software testing methods such as unit testing, integration testing, system testing, and acceptance testing, have been conducted, we cannot explicitly that the system should be made available for the end-users [121].

If the prototype should have been tested, we would have gone through unit testing, testing each of the implemented functionality, such as the data visualization. Next the integration testing would have had been done, in order to make sure that the different integrated functionality would work together. An example of this could be seen as integration testing the communication between the application and the hub. When all of the different units have been tested and passed the integration test, the system testing would be done. Here the complete prototype would be tested, in order to make sure that all of the functionality and the requirements have been fulfilled. The last step would be to put the prototype through an acceptance test, where the prototype are tested with regards to user needs and requirements conducted.

8 Discussion

This chapter will discuss the findings presented in the previous chapter and some of the aspects of the project which could have had implications towards the developed system. All of the discussed aspects will be seen in relation to the problem statement and its sub-questions presented in section 1.2. First of all, the test environment will be discussed with regards to how it might have had an effect on the results. Second, the collected data will be discussed. Third, locating the whereabouts of the user will be discussed, as there are several ways this can be done. Fourth, we discuss the two different models with regards this project. Finally we present and discuss the implications of the architecture that was decided upon, while also discussing how it otherwise could have been designed.

8.1 Test Environment

As presented in section 7.1 the test setup was conducted in an apartment which is 28 m^2, and as it was shown in figure 7.1 all of the smart devices were set up in a single room. Instead of selecting a test setup where all of the devices were placed in a single room, it might have been more beneficial to include more smart sensors in an apartment that has more rooms. This could also have been a great way of providing the machine learning algorithm with data with more variance in it. This could make the predictions more versatile as several other statuses, e.g. cooking or cleaning, could be defined. This will be further discussed below in section 8.3.

Furthermore, as only five smart devices were used, the test environment allowed for up to 13 smart devices in total, the amount of energy reduced can only be a small indicator of how much of an effect the system has had. If more sensors had been used in a larger test environment with more rooms, additional energy saving practices presented in chapter 3 could have been used, and in return provide a bigger energy reduction. While the test environment may give a decent example of how a room in a complete smart home could look like, it does not provide a satisfactory example of the complete solution. For a more generalized test environment, it is suggested that the test environment should have at least two rooms, as it has a high possibility to increase the measured energy reduction, increase features for the machine learning algorithm, and increase the possibility for more specified labels of what the test subject might be doing.

8.1.1 Electricity & Heating Data Analysis

The electricity and heating data, which was collected throughout the data gathering period, was used to answer the sub-question two, which was regarding using machine learning to control heating and electricity in an intelligent way. The sub-question can be found in section 1.2.

Section 7.2.1 presents the frequency of which the power consumption from the Nodon and Fibaro smart plugs were collected. The frequency was set to be in 1 minute intervals, however, the chosen frequency presented some challenges. These challenges can be seen in the data and in figure 7.4 in section 7.2.1. A sampling frequency higher than 1 minute would be beneficial, because there are power consumption peaks in our data that can be hard to explain. However, we provide a foundation for saying that the possibility of the user performing some heavy computational tasks are quite possible, as it was presented in figure 7.5 in section 7.2.1. Setting the sampling frequency to 30 seconds could help explain the peaks, however, by increasing the frequency, the number of data also increases. So instead of having 1440 data samples for one day with 1 minute frequency, using a data sampling rate of 30 seconds would increase it to 2880 data samples a day and make the data more fine-grained. However, increasing the sampling rate would also mean that more data needed to be stored, which in return could mean that multiple databases would be required in the end, as one might not be enough.

A part of this project was supposed to evolve into a comparison of how the thermostat had been used before and after implementing our system. The previous chapter, section 7.2.1, mentioned how little variation there was in setpoint temperature in the user's home. This was seen in figure 7.7, where the setpoint temperature only was adjusted approximately 10 of the days in the beginning of the data gathering period. The reason for the small amount of interaction with the thermostat was because of the weather. The outdoor temperature increased to a point where the user did not need to turn up his temperature setpoint anymore. This led to the exclusion of setpoint temperature data in the machine learning models. See Appendix I for a graph of the outdoor temperature.

Furthermore, if the data from the smart thermostat should have been used, it would be beneficial to gather the indoor temperature as well, in order to see if the radiator was heating or not. As the only data that can be collect from the Hauppauge Smart Thermostat is the setpoint temperature, one should implement a sensor which can measure the actual indoor temperature. Other beneficial data to collect could be to measure the temperature of the returning water in the radiator, as it could help the system figure out if it could heat up the household more efficiently.

8.2 User Location

As mentioned in section 6.2, the Android application was responsible for updating whether the user was home or away. This worked well for this prototype, however, relying only on a smartphone might not be the best choice for a real product. There is no other way to figure out if the users are home if the smartphone is turned off. PIR sensors could have been added to the rooms in the test environment order to detect if there was any activity in the house. Adding a PIR sensor in front of the door, could help determine if the user was leaving or entering the house. The activity from the PIR sensor could have been used as a feature for the ML model. However, when using such a sensor, you have to take house pets into consideration, as they might trigger the sensor. In order to avoid misleading data the PIR sensor could be combined with a camera and computer vision software in order to determine whether or not it was the user or the pet that caused the motion that the sensor detected. A filter could be setup to exclude the animal pets, so the only motion detected would come from the residents. However, the aforementioned solution could feel more intrusive than using a PIR sensor, as the users would be recorded with a camera.

Other possibilities than using Wi-Fi tracking on the smartphone, could be to use geolocation, however, this still poses the same problem as using Wi-Fi tracking. Still, using geolocation might be more beneficial, as it is possible to set up so called geofences, which can detect if the user enters or exits them and react accordingly. If the system detects that the user might be coming home before the scheduled time, it could e.g. start heating up the household ahead of time. However, this also poses problems with regards to the user entering a geofence without going home. Since the recommended minimum radius for a geofence is 100 meters, any activities close to the home may report that the user is home when they are not.

8.3 Machine Learning

The results presented in table 7.3 in section 7.2.1, indicate that the features were not good enough as the accepted timespans are very broad. As mentioned in section 7.3, the instances for the day status features were either specified as weekday or weekend, but this is too general as all weekdays do not necessarily have the same kind of schedule. If users meet late on a specific day, or maybe have a day off each week. An example from this project is that the user usually came home quite late on Wednesdays (19:08) compared to e.g. Tuesdays (16:54). The reason for this was because the user was attending an event every Wednesday afternoon. The prediction model treats these days as the same, since they are both weekdays, and therefore causes the model to provide skewed predictions for when the users comes home.

In a real scenario, the model should be trained using individual days instead of just using weekend and weekdays. Furthermore, in order to avoid excluding the given outliers presented in section 7.2.1, more statuses about the users could be implemented. The outliers can be used to further specify if there are any days the user has a specific patterns or habits, which might only occur two or three times every second month. However, a lot more data would be needed.

8.3.1 Sleep Status

As we did not have any sensors that specifically could indicate whether or not the user were asleep or not, we had to go through the data manually, and decide when the highest possibility of the user going to bed were. We looked at power consumption, home/away status, whether or not any lights were turned on, the previous status. Based on this we assumed that the user would be going to bed if the power consumption was low, lights turned off, and the last status was home.

The sleep label was found by analyzing the collected data, the method is illustrated in figure 5.10. If the user was at home with all lights turned off and the total power consumption was very low, then the label would be set to sleep. We could have added a pressure sensor under the user's bed in order to be able to know exactly when the user goes to bed. However, there was a pattern in the data showing that the user always turned on the TV when he went to bed, and it would be turned on for up to an hour. This means that a pressure sensor under the bed could only have helped us by figuring out if the user was physically lying in the bed. The use of PIR sensors could also help determine if the user was sleeping. Cook et al. [122] used motion sensors to detect activity in the bed, which they then could use as labels for the user being asleep.

8.3.2 Energy Reduction

The results from the Forecasting Model, showed in section 7.3, will be discussed with the aim of figuring out if the energy consumption could have been reduced using the models. Figure 7.9, figure 7.10 and the figures in Appendix C reveal that the model was quite good at predicting when the user woke up. Appendix G also indicates that the σ is only 29 minutes for the weekdays, and 15 minutes for the weekends which provides a small accepted timespan, meaning that the predictions are quite precise. Table 7.4 also shows that there were only two outliers for predicting when the user wakes up on weekdays. Predicting when the user wakes up can be beneficial for reducing energy, since the system could be

configured to automatically turn off smart plugs and lower the heat while the user was sleeping. As mentioned in section 7.2.1, we did not have the opportunity to make a real world test with focus on reducing energy on heating, since the outside temperature was so high (See Appendix I for temperature data), which resulted in the user not interacting with the thermostat. However, as mentioned in section 3.2 and 3.3.1, it can be cheaper to slowly heat a radiator compared to just setting it to the maximum power in order to heat a room quickly. The Forecasting Model could have been utilized to start heating the radiator slowly before the user usually wakes up, which could potentially save some energy.

Figure 7.9 shows that the Forecasting Model predicted *away* from 12:24 to 19:34, where the user really was *away* from 10:02 to 17:47. This means there were a few hours difference between the prediction and reality, however, if the model was a bit more accurate it could have turned off devices for 7 hours while the user was away. The accuracy of the Forecasting Model is highly affected by the feature called DayStatus, since we are using *weekday* and *weekend* instead of individual days. With more data gathered it would be possible to retrain the model using individual days as features instead of weekday/weekend.

The system could prompt the user if it detected energy usage while he was away. An example of where this could have been used can be seen in figure 7.10. The user was away from 16:10 to 20:40 while a smart plug was using between 90-100 Watt a minute. The system could have been used to detect this and asked the user if the smart plug should be turned off through a push notification.

The results presented in section 7.5 (table 7.9 and 7.9) showed a reduction in average power consumption and setpoint usage if the models would have been used. For most of the days the power could have been reduced by between 0.93% to 5.92%. The Forecasting Model was good at predicting when the user went to bed and woke up again, which provides a good timespan for turning off devices. The figures in Appendix C shows that the model was worse at predicting when the user left home and came back, but it usually predicted a partial timespan correct for when he left and came back. One of the days (27^{th} of May) showed a potential reduction of 22.15%, however, this reduction is not an actual reduction as the user arrives home before the system predicts, meaning that the devices would be turned off while the user would be using them. In this case the system could instead have had sent a notification asking whether or not the user were home. Furthermore, as the reductions are based on the assumptions presented in section 7.5 it is impossible to say if this reduction would have been observed, if the outside temperature had not increased.

Furthermore, the 23^{rd} of May showed a reduction of 1.07%, however, here the actual energy reduction could have reached 18.01%, which was because the user had forgotten to turn of

the computer while he was away for 4.5 hours. We assume that not all users would prefer that their computer automatically shutdown, since there might be a reason for leaving it on, but turning off other electrical devices in a household might not be as intrusive.

The two examples above each present energy reductions which are flawed mostly due to the predictions from the Forecasting Model. The predictions are wrong because of the lack of data, as we had to categorize the days into two categories, namely weekday and weekend. If more data would have been collected, it could have allowed us to use the specific days, where the users patterns would be more apparent.

Similarly to the power consumption, it was also possible to see a reduction in the setpoint. Table 7.9 showed a comparison between the daily energy usage with and without the model applied. The average daily reduction was assumed to be 4.59%.

These numbers are computed based on the assumption that the user makes an effort to adjust the setpoint of the thermostat multiple times a day. If the user forgets to turn it down before leaving the house or going to sleep the radiator would continue to heat. With the machine learning was deployed the system would automatically turn down the radiator to achieve a reduction in the energy consumption. This scenario is not considered as the assumption was that the user would set the setpoint to 20° Celsius every time he went to bed or left the house. This means the achieved reduction is an absolute worst case scenario when comparing the user to the machine learning. However, the reduction of 4.59% is mainly achieved by reducing the setpoint an additional 3° Celsius from the user's 20° Celsius to 17° Celsius. However, the reduction of 4.59% is mainly achieved by reducing the setpoint even more than what the user would normally do. Danfoss recommends that the lowest temperature for a room actively used is 5° Celsius below the comfort temperature (See section 3.3.1) when no one is home. As the user's comfort temperature is 22° Celsius (see figure 7.7) the setpoint was decided to be 17° Celsius. If the reduction was calculated with 20° Celsius (the user's preferred away/sleeping setpoint) instead no reduction would have been achieved.

Figure 7.7 shows how sporadic the user was at changing the setpoint, often leaving it at the same setpoint multiple days in a row. Based on this it can be assumed that the user is likely to not change it multiple times a day. Another factor that may have an affect on the reduction is the size of the test environment. This project only included a single thermostat as seen in figure 7.1. However, if the household had more thermostats it may be possible to assume that the user would not adjust the setpoint of multiple thermostats as they did not even adjust one thermostats.

Going through the data, we found that on the 30^{th} of May (Wednesday) the models predicted

that the user was not home in the accepted timespan presented in table G.1. The user came home at 20:32, however, as the system could not predict this, the smart thermostat was still turned down to 17° Celsius until 23:30, where the system performed another Real-time prediction, which then sets the setpoint temperature 20° Celsius as it predicts that the user is going to bed. This prediction was wrong, as the user actually was home, indicating a flawed reduction.

8.4 Prototype Architecture

In order to answer sub-question 1 in section 1.2 the architecture was considered and research through the State of the Art and Analysis chapters. Through the Analysis chapter it was decided to use a cloud service to perform the machine learning, this was further narrowed down during the design phase, where Azure Machine Learning Platform was decided to be used. While Azure was decided to be the platform for enabling the machine learning capabilities, it may have brought some issues to the architecture of the prototype. It allowed us to focus on developing the core functionality of the prototype while offloading the implementation of the seven algorithms that was decided to be evaluated. This centralized system's primary disadvantage is the dependency to a third party data processor. Essentially it adds a single point of failure to the intelligent part of the prototype. If Azure's servers were to experience downtime, the service would be unavailable and thus temporarily removing the machine learning capabilities of the prototype. While the chances of Azure experiencing major downtime is limited as it is run by Microsoft who have a vast content delivery network, the possibility is still present.

It would be possible to implement a backup solution that performs the machine learning locally if the cloud environment suddenly does not work. However, this would increase the requirements for the hub in terms of computational power and storage as discussed in section 4.6. An alternative to performing the machine learning computations on the hub was to use the user's smartphone, as smartphones computational power has increased the last decade. Furthermore, many new smartphones include dedicated chips for performing machine learning computations. This architecture paired with the current one, where the smartphone talks directly to the hub, would avoid the system encountering problems similarly to those Samsung SmartThings have encountered in the past with keeping their servers online as discussed in section 3.3.6. If their servers went offline then no user would be able to send commands to their household as all commands were routed through Samsung's servers to the smart hubs.

9 Conclusion

The aim of this project was to develop a home energy management system, where machine learning was used to reduce energy consumption. Based on this the following problem statement was defined:

How can a smart energy management system be developed in order to reduce the amount of energy spent on heating and electricity in a household?

Sub-questions were thereafter defined in order to narrow the project down to specific subject related to the main problem statement. The first sub-question was defined as follows:

How can the architecture be designed?

The home energy management system consisted of a smart hub, two smart plugs, two smart lights and a smart radiator thermostat. An Android application was used to control the system. To facilitate communication and perform external API requests, a remote server was developed. Azure Machine Learning Platform was used implement machine learning capabilities. The architecture allowed for a scalable and flexible deployment as all communication between the hub and application was sent directly between the devices.

The second sub-question was focused on how machine learning could be used for reducing energy used. The sub-question was defined as follows:

How can electricity and heating be controlled in an intelligent way to reduce the energy consumption?

Two different machine learning models were trained using the data gathered from the smart sensors in the household and from the smartphone carried by the user. A regression model was trained in order to predict when the user would come home, leave home, go to bed, and wake up. A classification model was trained with the purpose of running real-time within the timespans predicted by the regression model. The accuracy of both models could have been improved if more data had been gathered, but the results showed that energy could be reduce using the proposed approach.

The last sub-question was related to the users location, since it is necessary to know whether the user is at home or away when automating the system. The third sub-question was defined as follows:

How can the location of the user be used?

The proposed system only used Wi-Fi to determine if the user was home or away, which worked well in this project. However, it would have had been beneficial to implement several different tracking devices, such as passive infrared sensors, motion sensors, or pressure sensors. Relying on a single technology to determine the presence of a user can be fragile. If the technology stops working there is no other way of telling if the user is present in the household. Using several technologies and sensors would avoid this problem, not having a single-point of failure.

The report presented a possible future-proof architecture for reducing the energy consumption used on electricity and heating using machine learning. Data for training the machine learning models was collected over 26 days. The models were deployed and tested over 10 days at the end of May in Copenhagen, Denmark. The results showed an average potential reduction of 7.28% in the electricity consumption and 4.59% of the heating consumption. However, the reduction in heating was based heavily on assumptions with little statistical foundation and may therefore be flawed. Further testing of the heating consumption should be performed during steadier outdoor weather conditions. In addition, performing a test while the outdoor temperature was lower could provide better results.

10 Future Works

In this chapter we present some of the aspects that could be further investigated. We present some of the things that were not the focus in this project, but still could be researched. First, we start by presenting what we recommend and what one should consider when expanding the project to a household with more than one user. Next we provide a foundation for a machine learning feature that could be beneficial to use. As this project used supervised machine learning, we present other possibilities and opportunities which can be researched in order to further improve the intelligence of the system. Lastly, we present how different profiles can be used in such a system.

10.1 Multiple Users

This project was only focused on providing a system for one user, which was mentioned in section 1.3. Instead of only tracking one user through the Wi-Fi connection, several users would have had to be tracked in order for the system to know if anyone was present in the household. This would of course mean that the machine learning algorithms would have to be adjusted so that every user has an effect on the output. One way of reducing energy would be to just predict whenever no one is at home, or if the users are asleep, which is what we have focused on in this project. But with more smart devices it could be possible to automate certain areas in the home depending on which user is at home. For example, one user might show a pattern of using certain devices while being at home, and none of the other users are using those devices, so they could be turned off when this particular user is not at home. The same can be said about thermostat temperature, since different users might have their own room.

10.2 Calendar Integration

A feature that could be interesting to investigate for further development is integrating digital calendars into the system. Some users might use a calendar every day, which could help provide more information on when the user e.g. is at work or going to regular sports practices. It would probably be hard to make the system figure out what each event in the calendar means, e.g. an event with the title "Dinner with Bob and Alice" does not reveal anything about where this event takes place. However, most digital calendars allow users to

specify a location of the event, which could be helpful in order to know if the user will be at home at that time or not.

10.3 Reinforced Learning

As this project already uses machine learning in order to predict the users schedule, in regards to when he comes home, leaves home, wakes up, and goes to bed, through supervised algorithms, the next step could be to look at the feasibility of reinforced learning. Reinforced learning could be beneficial for a system such as the one developed in this project, as the feedback from the user can be used to tell the system whether or not it predicted correct or did something wrong, and thereby learn by its mistakes. The system could take the users input into consideration. This could be done through notifications sent to the application, where the user would have to make a decision on whether or not, e.g. the system should turn down the heat, turn of devices.

10.4 Different Profiles for Different Seasons

As it was found in this project, seasons can have a great impact on the machine learning outcome, as it e.g. affects the data. For example, the user might not interact with the thermostat during the summer. Here it could be beneficial to classify the data into different seasons, as there is a difference in the data collected. For this project, the user did not interact with the smart thermostat connected to the system, because the outdoor temperature increased to a point where the radiator was not used. This require the system to be tested through each of the four seasons. We believe that the system would greatly benefit from it, even though it would be time consuming.

Bibliography

[1] Hannah Ritchie and Max Roser. Energy production & changing energy sources, 2018. URL https://ourworldindata.org/energy-production-and-changing-energy-sources. (Accessed on 01-06-2018).

[2] Access to electricity (% in population) | data, No date. URL https://data.worldbank.org/indicator/EG.ELC.ACCS.ZS?end=2014&name_desc=true&start=1990&view=chart. (Accessed on 27-02-2018).

[3] E. A. Weiss. Biographies: Eloge: Arthur lee samuel (1901-90). *IEEE Annals of the History of Computing*, 14(3):55, 1992. ISSN 1058-6180. doi: 10.1109/85.150082.

[4] John Markoff. Computer wins on 'jeopardy!': Trivial, it's not, 2011. URL https://www.nytimes.com/2011/02/17/science/17jeopardy-watson.html. (Accessed on 02-06-2018).

[5] Jon Russell. Computer wins on 'jeopardy!': Trivial, it's not, 2011. URL https://techcrunch.com/2016/03/15/google-ai-beats-go-world-champion-again-to-complete-historic-4-1-series-victory/. (Accessed on 02-06-2018).

[6] Shahzeen Z Attari, Michael L DeKay, Cliff I Davidson, and Wändi Bruine De Bruin. Public perceptions of energy consumption and savings. *Proceedings of the National Academy of sciences*, 107(37):16054–16059, 2010.

[7] Kirsten Gram-Hanssen. Residential heat comfort practices: understanding users. *Building Research & Information*, 38(2):175–186, 2010. doi: 10.1080/09613210903541527. URL https://doi.org/10.1080/09613210903541527.

[8] Danish Energy Agency. Regulation and planning of district heating in denmark. Technical report, Danish Energy Agency, 2017. URL https://ens.dk/sites/ens.dk/files/Globalcooperation/regulation_and_planning_of_district_heating_in_denmark.pdf.

[9] Jose Miguel, David Mauricio, and Glen Rodriguez. A review of software quality models for the evaluation of software products. *International Journal of Software Engineering & Applications (IJSEA)*, 5:31–54, 11 2014.

[10] Malte Nachreiner and Ellen Matthies. Enhancing informational strategies for supporting residential electricity saving: Identifying potential and household characteristics in germany. *Energy Research & Social Science*, 11:276–287, 2016.

[11] Linda Steg. Promoting household energy conservation. *Energy Policy*, 36(12): 4449–4453, December 2008. URL https://ideas.repec.org/a/eee/enepol/v36y2008i12p4449-4453.html.

[12] Elregning: 7 spareråd | forbrugerrådet tænk, No date. URL https://taenk.dk/test-og-forbrugerliv/privatoekonomi-og-aftaler/elregningen-7-sparetips. (Accessed on 28-02-2018).

[13] Sparetips - elforbundet, No date. URL https://www.elforbundet.dk/spar-penge-el/. (Accessed on 28-02-2018).

[14] Sådan gør du din elregning mindre, No date. URL https://www.bolius.dk/saadan-goer-du-din-elregning-mindre-17338/. (Accessed on 28-02-2018).

[15] 5 nemme tips - sådan sparer du på el i din bolig, No date. URL https://www.energihjem.dk/5-nemme-tips-saadan-sparer-du-paa-el-i-din-bolig-2/. (Accessed on 28-02-2018).

[16] Sonni T. Johnsen. Spar på strømmen - gode tips til at spare på strømmen, No date. URL https://ewii.com/privat/energi/energiraadgivning/energispareraad/el-sparetips. (Accessed on 28-02-2018).

[17] Dit elforbrug | energistyrelsen, No date. URL https://sparenergi.dk/forbruger/el/dit-elforbrug. (Accessed on 28-02-2018).

[18] Payam Nejat, Fatemeh Jomehzadeh, Mohammad Mahdi Taheri, Mohammad Gohari, and Muhd Zaimi Abd Majid. A global review of energy consumption, co2 emissions and policy in the residential sector (with an overview of the top ten co2 emitting countries). *Renewable and sustainable energy reviews*, 43:843–862, 2015.

[19] DanskEnergi. 5 skarpe om fleksafregning | dansk energi, No date. URL https://www.danskenergi.dk/nyheder/5-skarpe-om-fleksafregning. (Accessed on 17-04-2018).

[20] Julie Goodhew, Sabine Pahl, Tim Auburn, and Steve Goodhew. Making heat visible: promoting energy conservation behaviors through thermal imaging. *Environment and Behavior*, 47(10):1059–1088, 2015.

[21] Energivejledning - høje taastrup fjernvarme a.m.b.a, No date. URL http://www.htf.dk/energivejledning. (Accessed on 28-02-2018).

[22] Hvidebaek-fjernvarme, No date. URL http://www.varmevaerk.dk/media/2024467/Hvidebaek-Fjernvarme-motivation.pdf. (Accessed on 28-02-2018).

[23] Gode energivaner - en gevinst for din pengepung og for miljøet, No date. URL https://eniig.dk/privat/el/godt-at-vide-om-el-og-energi/gode-energivaner/. (Accessed on 28-02-2018).

[24] Energirigtige vaner - energistyrelsen, No date. URL https://sparenergi.dk/forbruger/varme/dit-varmeforbrug/energirigtige-vaner. (Accessed on 28-02-2018).

[25] Danfoss. Danfoss link™, 2018. URL http://products.danfoss.com/productrange/heatingsolutions/danfoss-link/#/. (Accessed on 27-02-2018).

[26] Danfoss. Save up to 30% on your heating bill, 2018. URL http://smartheating.danfoss.com/en/savings/. (Accessed on 27-02-2018).

[27] Danfoss. Getting started with danfoss smart heating is easy, 2018. URL http://smartheating.danfoss.com/en/support/. (Accessed on 27-02-2018).

[28] *Danfoss Link™ Connect Thermostat*. Danfoss, October 2016. URL http://heating.danfoss.com/PCMPDF/VDLTA202-connect%20B2C.pdf. (Accessed on 18-11-2017).

[29] Danfoss. Danfoss link™ app, 2018. URL http://smartheating.danfoss.com/en/a-solution-for-every-home/danfoss-link/the-link-app/. (Accessed on 27-02-2018).

[30] Netatmo. Netatmo thermostat, 2017. URL https://www.netatmo.com/en-US/product/energy/thermostat. (Accessed on 18-11-2017).

[31] Netatmo. How does anticipation or the auto-adapt function work?, 2017. URL https://www.netatmo.com/en-GB/helpcenter/energy/3/heating-and-consumption-control/13/how-does-anticipation-or-the-auto-adapt-function-work/69. (Accessed on 18-11-2017).

[32] Tado. Smart radiator thermostat - starter kit, 2017. URL https://www.tado.com/dk/products/smart-radiator-starter-kit. (Accessed on 18-11-2017).

[33] Tado. Smarter heating with your smart climate assistant, 2017. URL https://www.tado.com/dk/thermostat-heating. (Accessed on 18-11-2017).

[34] Lifx. Schedules in the lifx app, 2018. URL https://support.lifx.com/hc/en-us/articles/202472110-Schedules-in-the-LIFX-app. (Accessed on 18-03-2018).

[35] Ikea. Timers, 2018. URL http://www.ikea.com/gb/en/customer-service/smart-lighting-support/faq-smart-lighting/timers/. (Accessed on 18-03-2018).

[36] Philips. Set your lights to welcome you home, 2018. URL https://www2.meethue.com/en-us/light-your-home-smarter/daddys-home-light. (Accessed on 18-03-2018).

[37] Belkin. Wemo® switch smart plug, 2017. URL http://www.belkin.com/us/p/P-F7C027/. (Accessed on 06-03-2018).

[38] Nodon. The z-wave® micro smart plug, 2017. URL https://nodon.fr/en/nodon/z-wave-micro-smart-plug/. (Accessed on 06-03-2018).

[39] SmartThings. Smartthings is the easy way to turn your home into a smart home., 2017. URL https://www.smartthings.com/uses. (Accessed on 06-03-2018).

[40] SmartThings. Samsung smartthings multipurpose sensor, 2017. URL https://shop.smartthings.com/products/samsung-smartthings-multipurpose-sensor/574. (Accessed on 06-03-2018).

[41] SmartThings. Samsung smartthings outlet, 2017. URL https://shop.smartthings.com/products/samsung-smartthings-outlet. (Accessed on 06-03-2018).

[42] SmartThings. Smartthings works with a wide range of connected devices., 2017. URL https://www.smartthings.com/products. (Accessed on 06-03-2018).

[43] SmartThings. Platform status, 2018. URL https://status.smartthings.com/. (Accessed on 06-03-2018).

[44] SmartThings. Samsung connect is now smartthings, No date. URL https://support.smartthings.com/hc/en-us/articles/205380634-Setting-up-an-account-the-Hub-and-Things. (Accessed on 02-04-2018).

[45] SmartThings. Setting up an account, the hub, and things, 2017. URL https://support.smartthings.com/hc/en-us/articles/205380634-Setting-up-an-account-the-Hub-and-Things. (Accessed on 02-04-2018).

[46] Saima Aman, Yogesh Simmhan, and Viktor K Prasanna. Energy management systems: state of the art and emerging trends. *IEEE Communications Magazine*, 51 (1):114–119, 2013.

[47] Mohammad Shakeri, Mohsen Shayestegan, SM Salim Reza, Iskandar Yahya, Badariah Bais, Md Akhtaruzzaman, Kamaruzzaman Sopian, and Nowshad Amin. Implementation of a novel home energy management system (hems) architecture with solar photovoltaic system as supplementary source. *Renewable Energy*, 125:108–120, 2018.

[48] Md Sumon Shahriar and M Sabbir Rahman. Urban sensing and smart home energy optimisations: A machine learning approach. In *Proceedings of the 2015 International Workshop on Internet of Things towards Applications*, pages 19–22. ACM, 2015.

[49] Wilhelm Kleiminger, Christian Beckel, Thorsten Staake, and Silvia Santini. Occupancy detection from electricity consumption data. In *Proceedings of the 5th ACM Workshop on Embedded Systems For Energy-Efficient Buildings*, pages 1–8. ACM, 2013.

[50] Christian Koehler, Brian D. Ziebart, Jennifer Mankoff, and Anind K. Dey. Therml: Occupancy prediction for thermostat control. In *Proceedings of the 2013 ACM*

International Joint Conference on Pervasive and Ubiquitous Computing, UbiComp '13, pages 103–112, New York, NY, USA, 2013. ACM. ISBN 978-1-4503-1770-2. doi: 10.1145/2493432.2493441. URL http://doi.acm.org/10.1145/2493432.2493441.

[51] Jon Froehlich, Leah Findlater, and James Landay. The design of eco-feedback technology. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '10, pages 1999–2008, New York, NY, USA, 2010. ACM. ISBN 978-1-60558-929-9. doi: 10.1145/1753326.1753629. URL http://doi.acm.org/10.1145/1753326.1753629.

[52] Wilhelm Kleiminger, Christian Beckel, and Silvia Santini. Household occupancy monitoring using electricity meters. In *Proceedings of the 2015 ACM International Joint Conference on Pervasive and Ubiquitous Computing*, UbiComp '15, pages 975–986, New York, NY, USA, 2015. ACM. ISBN 978-1-4503-3574-4. doi: 10.1145/2750858.2807538. URL http://doi.acm.org/10.1145/2750858.2807538.

[53] F. Palumbo, P. Barsocchi, S. Chessa, and J. C. Augusto. A stigmergic approach to indoor localization using bluetooth low energy beacons. In *2015 12th IEEE International Conference on Advanced Video and Signal Based Surveillance (AVSS)*, pages 1–6, Aug 2015. doi: 10.1109/AVSS.2015.7301734.

[54] Smartthings - samsung smartthings arrival sensor, 2017. URL https://www.smartthings.com/us/products/samsung-smartthings-arrival-sensor. (Accessed on 27-02-2018).

[55] Smartsense presence sensor - smartthings support, 2017. URL https://support.smartthings.com/hc/en-us/articles/200903140-SmartSense-Presence-Sensor. (Accessed on 27-02-2018).

[56] Wink | wink motion sensor, No date. URL https://www.wink.com/products/wink-motion-sensor/. (Accessed on 27-02-2018).

[57] Smart thermostat for heating systems and radiators | tado, No date. URL https://www.tado.com/dk/thermostat-heating. (Accessed on 27-02-2018).

[58] Stephen Statler. *Geofencing: Everything You Need to Know*, pages 307–316. Apress, Berkeley, CA, 2016. ISBN 978-1-4842-1889-1. doi: 10.1007/978-1-4842-1889-1_17. URL https://doi.org/10.1007/978-1-4842-1889-1_17.

[59] A. De Blas and D. López de Ipiña. Improving trilateration for indoors localization using ble beacons. In *2017 2nd International Multidisciplinary Conference on Computer and Energy Science (SpliTech)*, pages 1–6, July 2017.

[60] Apple. Determining the proximity to an ibeacon, No date. URL https://developer.apple.com/documentation/corelocation/determining_the_proximity_to_an_ibeacon. (Accessed on 27-02-2018).

[61] BBC. South korea trials wireless seat alerts for pregnant train travellers - bbc news, 2016. URL http://www.bbc.com/news/technology-36443920. (Accessed on 27-02-2018).

[62] Giorgio Conte, Massimo De Marchi, Alessandro A. Nacci, Vincenzo Rana, and Donatella Sciuto. Bluesentinel: A first approach using ibeacon for an energy efficient occupancy detection system. In *Proceedings of the 1st ACM Conference on Embedded Systems for Energy-Efficient Buildings*, BuildSys '14, pages 11–19, New York, NY, USA, 2014. ACM. ISBN 978-1-4503-3144-9. doi: 10.1145/2676061.2674078. URL http://doi.acm.org/10.1145/2676061.2674078.

[63] A. Di Luzio, A. Mei, and J. Stefa. Mind your probes: De-anonymization of large crowds through smartphone wifi probe requests. In *IEEE INFOCOM 2016 - The 35th Annual IEEE International Conference on Computer Communications*, pages 1–9, April 2016. doi: 10.1109/INFOCOM.2016.7524459.

[64] Giles Hogben. Android developers blog: Changes to device identifiers in android o, 2017. URL https://android-developers.googleblog.com/2017/04/changes-to-device-identifiers-in.html. (Accessed on 18-03-2018).

[65] S. Boonsriwai and A. Apavatjrut. Indoor wifi localization on mobile devices. In *2013 10th International Conference on Electrical Engineering/Electronics, Computer, Telecommunications and Information Technology*, pages 1–5, May 2013. doi: 10.1109/ECTICon.2013.6559592.

[66] Yuvraj Agarwal, Bharathan Balaji, Rajesh Gupta, Jacob Lyles, Michael Wei, and Thomas Weng. Occupancy-driven energy management for smart building automation. In *Proceedings of the 2Nd ACM Workshop on Embedded Sensing Systems for Energy-Efficiency in Building*, BuildSys '10, pages 1–6, New York, NY, USA, 2010. ACM. ISBN 978-1-4503-0458-0. doi: 10.1145/1878431.1878433. URL http://doi.acm.org/10.1145/1878431.1878433.

[67] Adb El-Latif Mowad Mohamed, Ahmed Fathy, and Ahmed Hafez. Smart home automated control system using android application and microcontroller. *International Journal of Scientific & Engineering Research*, 5(5):935–939, 2014.

[68] Amazon machine learning - predictive analysis with aws, No date. URL https://aws.amazon.com/aml/. (Accessed on 22-03-2018).

[69] Cloud ai, No date. URL https://cloud.google.com/products/machine-learning/. (Accessed on 22-03-2018).

[70] Azure machine learning studio documentation, No date. URL https://docs.microsoft.com/en-gb/azure/machine-learning/studio/. (Accessed on 22-03-2018).

[71] Unlock the power of cognitive with ibm watson, No date. URL https://www.ibm.com/cloud/ai. (Accessed on 22-03-2018).

[72] Reza Rawassizadeh, Timothy J Pierson, Ronald Peterson, and David Kotz. Nocloud: Exploring network disconnection through on-device data analysis. *IEEE Pervasive Computing*, 17(1):64–74, 2018.

[73] Amazon. Alexa and alexa device FAQs, 2018. URL https://www.amazon.com/gp/help/customer/display.html?nodeId=201602230. (Accessed on 26-11-2018).

[74] The Economist. The world's most valuable resource is no longer oil, but data. *Economist.com*, 2017. URL https://www.economist.com/news/leaders/21721656-data-economy-demands-new-approach-antitrust-rules-worlds-most-valuable-reso (Accessed on 22-03-2018).

[75] Credit Suisse. Mark cuban: "data is the new gold". *Credit-Suisse.com*, 2017. URL https://www.credit-suisse.com/corporate/en/articles/news-and-expertise/mark-cuban-data-is-the-new-gold-201706.html. (Accessed on 22-03-2018).

[76] CNBC. Ibm ceo ginni rometty says 80% of the world's data is where the 'real gold' is. *CNBC.com*, 2017. URL https://www.cnbc.com/2017/06/20/ibm-ceo-says-80-percent-of-the-worlds-data-is-where-the-real-gold-is.html. (Accessed on 22-03-2018).

[77] Alphabet. Alphabet announces fourth quarter and fiscal year 2017 results, 2018. URL https://abc.xyz/investor/pdf/2017Q4_alphabet_earnings_release.pdf.

[78] Google. Google cloud platform terms of service, 2018. URL https://cloud.google.com/terms/. (Accessed on 01-04-2018).

[79] Qualcomm. Introducing qualcomm zeroth processors: Brain-inspired computing, 2013. URL https://www.qualcomm.com/news/onq/2013/10/10/introducing-qualcomm-zeroth-processors-brain-inspired-computing. (Accessed on 26-11-2018).

[80] HUAWEI. Huawei reveals the future of mobile ai at ifa 2017, 2017. URL https://consumer.huawei.com/en/press/news/2017/ifa2017-kirin970/. (Accessed on 26-11-2018).

[81] Apple. The future is here: iphone x, 2017. URL https://www.apple.com/newsroom/2017/09/the-future-is-here-iphone-x/. (Accessed on 26-11-2018).

[82] Anzar Mahmood, Nadeem Javaid, and Sohail Razzaq. A review of wireless communications for smart grid. *Renewable and Sustainable Energy Reviews*, 41:248 – 260, 2015. ISSN 1364-0321. doi: https://doi.org/10.1016/j.rser.2014.08.036. URL http://www.sciencedirect.com/science/article/pii/S1364032114007126.

[83] Android. Creating and monitoring geofences | android developers, No date. URL https://developer.android.com/training/location/geofencing.html. (Accessed on 02-04-2018).

[84] Android. Wifiinfo | android developers, No date. URL https://developer.android.com/reference/android/net/wifi/WifiInfo.html. (Accessed on 03-04-2018).

[85] The European Union. Regulation (eu) 2016/679 of the european parliament and of the council, 2016. URL http://eur-lex.europa.eu/legal-content/EN/TXT/PDF/?uri=CELEX:32016R0679&from=EN. (Accessed on 20-04-2018).

[86] Data privacy - amazon web service, 2018. URL https://aws.amazon.com/compliance/data-privacy-faq/. (Accessed on 01-04-2018).

[87] Google cloud platform terms of service, No date. URL https://cloud.google.com/terms/. (Accessed on 01-04-2018).

[88] Microsoft azure agreement, 2014. URL https://azure.microsoft.com/da-dk/support/legal/subscription-agreement-nov-2014/. (Accessed on 01-04-2018).

[89] Ibm watson - data privacy, No date. URL https://www.ibm.com/watson/data-privacy/. (Accessed on 01-04-2018).

[90] Yalin Başztanlar and Mustafa Özuysal. *Introduction to Machine Learning*, pages 105–128. Humana Press, Totowa, NJ, 2014. ISBN 978-1-62703-748-8. doi: 10.1007/978-1-62703-748-8_7. URL https://doi.org/10.1007/978-1-62703-748-8_7.

[91] Bashar Nuseibeh and Steve Easterbrook. Requirements engineering: a roadmap. In *Proceedings of the Conference on the Future of Software Engineering*, pages 35–46. ACM, 2000.

[92] Dave Ensor and Ian Stevenson. *Oracle Design: The Definitive Guide*. "O'Reilly Media, Inc.", 1997.

[93] Firebase. Firebase cloud messaging, 2018. URL https://firebase.google.com/docs/cloud-messaging/. (Accessed on 09-05-2018).

[94] Z-Wave. Z-wave | z-wave smart home products are the #1 choice for smart homes., No date. URL http://www.z-wave.com/about. (Accessed on 26-04-2018).

[95] OpenZWave. Ozw utilities, No date. URL http://www.openzwave.net/. (Accessed on 26-04-2018).

[96] Home Assistant. Home assistant, No date. URL https://www.home-assistant.io/. (Accessed on 28-04-2018).

[97] NordPool. Api | nord pool, No date. URL https://www.nordpoolgroup.com/TAS/api/. (Accessed on 26-04-2018).

[98] Dayne Hemmes, Hiram Medero, and Harrison Mitchell. Comparison of nosql and sql databases in the cloud. *Sais 2014 Proceedings*, 12:1 – 8, 2014. ISSN 1364-0321. URL http://www.sciencedirect.com/science/article/pii/S1364032114007126.

[99] Lydia Leong, Raj Bala, Craig Lowery, and Dennis Smith. Magic quadrant for cloud infrastructure as a service, worldwide, 2017. URL https://www.gartner.com/doc/reprints?id=1-2G2O5FC&ct=150519&st=sb. (Accessed on 10-05-2018).

[100] Microsoft. Welcome to azure cosmos db, 2018. URL https://docs.microsoft.com/en-us/azure/cosmos-db/introduction. (Accessed on 10-05-2018).

[101] Microsoft. Multiclass logistic regression, 2018. URL https://docs.microsoft.com/en-us/azure/machine-learning/studio-module-reference/multiclass-logistic-regression. (Accessed on 21-05-2018).

[102] Microsoft. Multiclass decision forest, 2018. URL https://docs.microsoft.com/en-us/azure/machine-learning/studio-module-reference/multiclass-decision-forest. (Accessed on 21-05-2018).

[103] Microsoft. Multiclass decision jungle, 2018. URL https://docs.microsoft.com/en-us/azure/machine-learning/studio-module-reference/multiclass-decision-jungle. (Accessed on 21-05-2018).

[104] Microsoft. Multiclass neural network, 2018. URL https://docs.microsoft.com/en-us/azure/machine-learning/studio-module-reference/multiclass-neural-network. (Accessed on 21-05-2018).

[105] Microsoft. Bayesian linear regression - azure machine learning studio | microsoft docs, 2018. URL https://msdn.microsoft.com/en-us/library/azure/dn906022.aspx. (Accessed on 19-05-2018).

[106] Microsoft. Decision forest regression - azure machine learning studio | microsoft docs, 2017. URL https://msdn.microsoft.com/en-us/library/azure/dn905862.aspx. (Accessed on 19-05-2018).

[107] Microsoft. Boosted decision tree regression - azure machine learning studio | microsoft docs, 2018. URL

https://msdn.microsoft.com/en-us/library/azure/dn905801.aspx. (Accessed on 19-05-2018).

[108] Kantar Gallup, Danske Medier, and Krea Kom. Mobile life 2017, 2017. URL https://danskemedier.dk/wp-content/uploads/pressemeddelelse-mobile-life-final.pdf. (Accessed on 10-05-2018).

[109] Tado. Multi-zone control - do it your way!, 2017. URL https://www.pinterest.com/pin/488077678351820815/. (Accessed on 06-06-2018).

[110] Expert Reviews. Samsun smartthings review - hands on with a connected home in a box, 2017. URL http://www.expertreviews.co.uk/accessories/gadgets/1403849/samsung-smartthings-review-hands-on-with-a-connected-home-in-a-box. (Accessed on 06-06-2018).

[111] Yvonne Rogers, Helen Sharp, and Jenny Preece. *Interaction design: beyond human-computer interaction*. John Wiley & Sons, 2011.

[112] David Both. How to use cron in linux, 2017. URL https://opensource.com/article/17/11/how-use-cron-linux. (Accessed on 15-05-2018).

[113] cURL. command line tool and library for transferring data with urls, 2018. URL https://curl.haxx.se/. (Accessed on 15-05-2018).

[114] Express. Express - node.js web application framework, 2017. URL http://expressjs.com/. (Accessed on 15-05-2018).

[115] Google. App manifest overview, 2018. URL https://developer.android.com/guide/topics/manifest/manifest-intro. (Accessed on 15-05-2018).

[116] Google. Services overview, 2018. URL https://developer.android.com/guide/components/services. (Accessed on 15-05-2018).

[117] Google. Volley overview, 2018. URL https://developer.android.com/training/volley/. (Accessed on 15-05-2018).

[118] Google. Data and file storage overview, 2018. URL https://developer.android.com/guide/topics/data/data-storage. (Accessed on 15-05-2018).

[119] Google. Broadcasts overview, 2018. URL https://developer.android.com/guide/components/broadcasts. (Accessed on 19-05-2018).

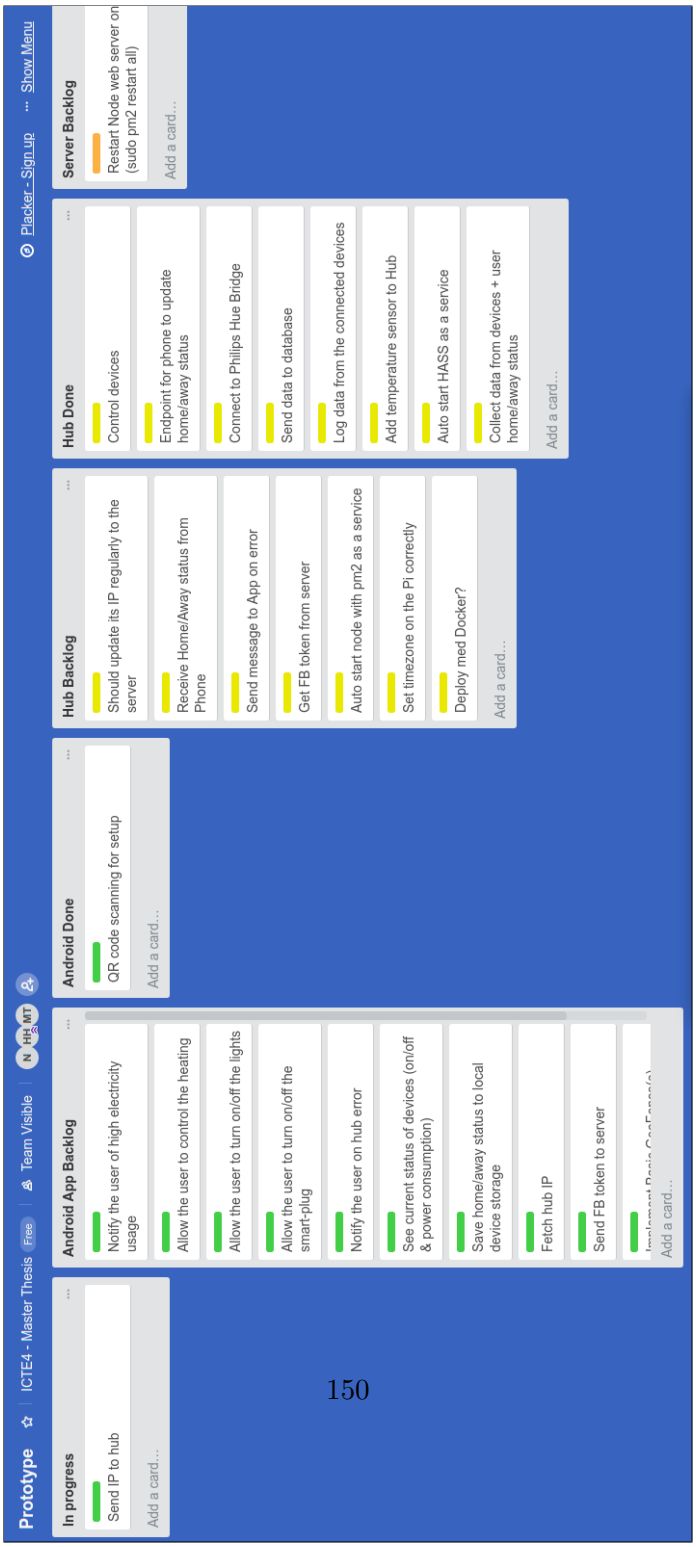
[120] Philipp Jahoda. Mpandroidchart, 2018. URL https://github.com/PhilJay/MPAndroidChart. (Accessed on 15-05-2018).

[121] SoftwareTestingFundamentals. Acceptance testing - software testing fundamentals, 2018. URL http://softwaretestingfundamentals.com/acceptance-testing/. (Accessed on 01-06-2018).

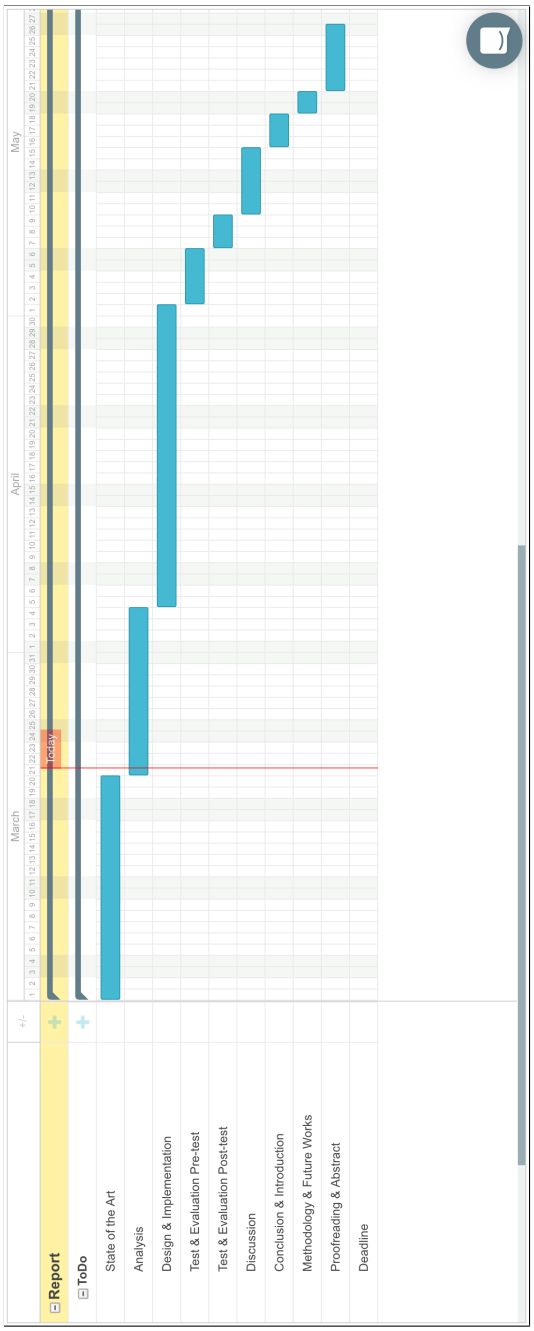
[122] D. J. Cook, A. S. Crandall, B. L. Thomas, and N. C. Krishnan. Casas: A smart home in a box. *Computer*, 46(7):62–69, July 2013. ISSN 0018-9162. doi: 10.1109/MC.2012.328.

[123] Weather Undergournd. Weather history for ekch - may, 2018, 2018. URL https://www.wunderground.com/history/airport/EKCH/2018/5/21/CustomHistory.html?dayend=30&monthend=5&yearend=2018&req_city=&req_state=&req_statename=&reqdb.zip=&reqdb.magic=&reqdb.wmo=. (Accessed on 03-06-2018).

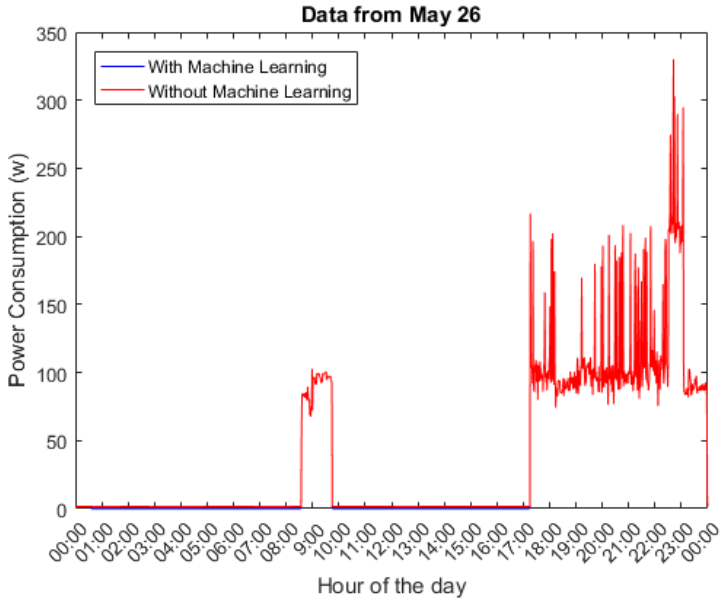
Appendix A

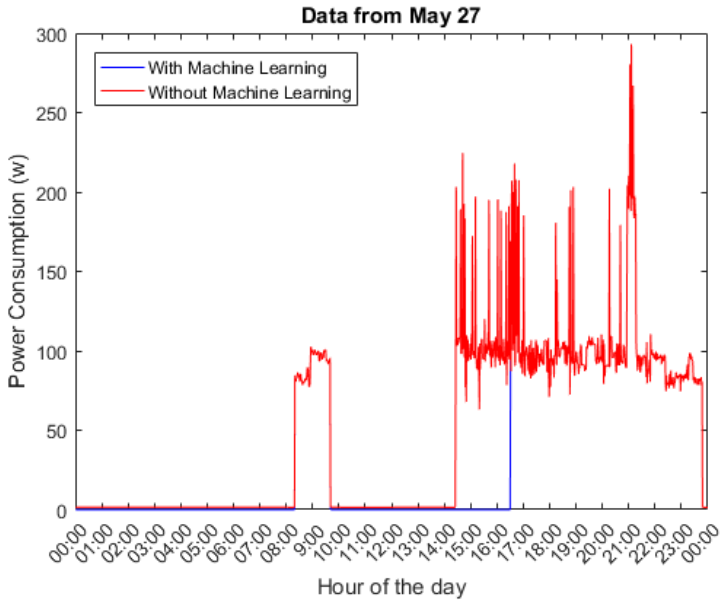

Prototype
ICTE4 - Master Thesis
Free
Team Visible
Placker - Sign up
Show Menu
In progress
Send IP to hub
Add a card...
Android App Backlog
Notify the user of high electricity usage
Allow the user to control the heating
Allow the user to turn on/off the lights
Allow the user to turn on/off the smart-plug
Notify the user on hub error
See current status of devices (on/off & power consumption)
Save home/away status to local device storage
Fetch hub IP
Send FB token to server
Add a card...
Android Done
QR code scanning for setup
Add a card...
Hub Backlog
Should update its IP regularly to the server
Receive Home/Away status from Phone
Send message to App on error
Get FB token from server
Auto start node with pm2 as a service
Set timezone on the Pi correctly
Deploy med Docker?
Add a card...
Hub Done
Control devices
Endpoint for phone to update home/away status
Connect to Philips Hue Bridge
Send data to database
Log data from the connected devices
Add temperature sensor to Hub
Auto start HASS as a service
Collect data from devices + user home/away status
Add a card...
Server Backlog
Restart Node web server on (sudo pm2 restart all)
Add a card...

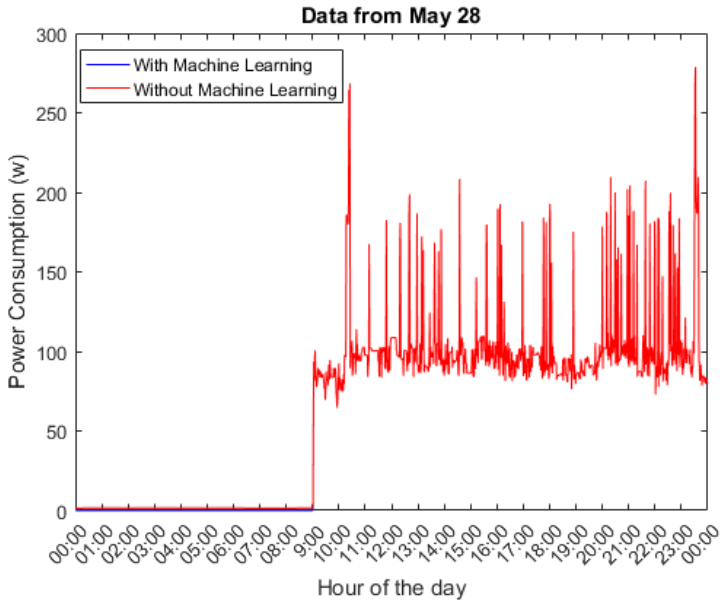
Appendix B

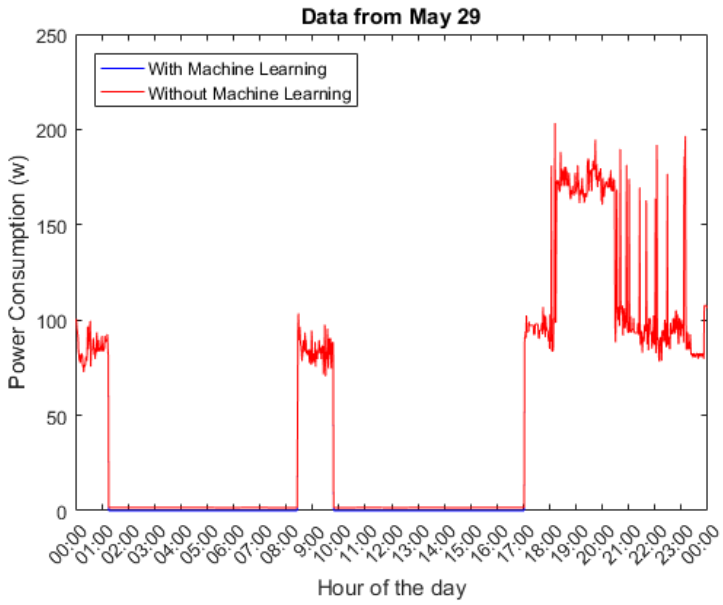

March
April
May
Today
Report
ToDo
State of the Art
Analysis
Design & Implementation
Test & Evaluation Pre-test
Test & Evaluation Post-test
Discussion
Conclusion & Introduction
Methodology & Future Works
Proofreading & Abstract
Deadline

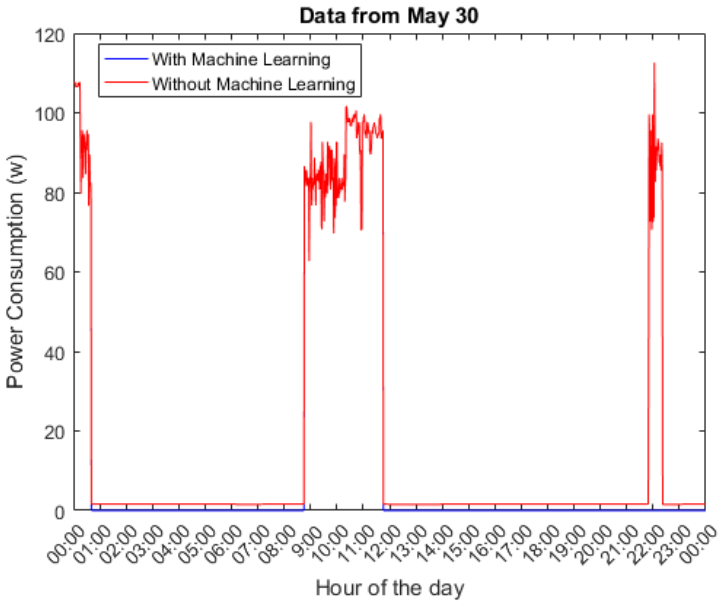
Appendix C


Data from May 26
With Machine Learning
Without Machine Learning
Power Consumption (w)
Hour of the day


Data from May 27
With Machine Learning
Without Machine Learning
Power Consumption (w)
Hour of the day


Data from May 28
With Machine Learning
Without Machine Learning
Power Consumption (w)
300
250
200
150
100
50
0
00:00
01:00
02:00
03:00
04:00
05:00
06:00
07:00
08:00
9:00
10:00
11:00
12:00
13:00
14:00
15:00
16:00
17:00
18:00
19:00
20:00
21:00
22:00
23:00
00:00
Hour of the day


Data from May 29
With Machine Learning
Without Machine Learning
Power Consumption (w)
250
200
150
100
50
0
00:00
01:00
02:00
03:00
04:00
05:00
06:00
07:00
08:00
9:00
10:00
11:00
12:00
13:00
14:00
15:00
16:00
17:00
18:00
19:00
20:00
21:00
22:00
23:00
00:00
Hour of the day


Data from May 30
With Machine Learning
Without Machine Learning
Power Consumption (w)
120
100
80
60
40
20
0
00:00
01:00
02:00
03:00
04:00
05:00
06:00
07:00
08:00
9:00
10:00
11:00
12:00
13:00
14:00
15:00
16:00
17:00
18:00
19:00
20:00
21:00
22:00
23:00
00:00
Hour of the day

Appendix D

Multiple regression models were tested in order to figure out which one performed best as mentioned in 5.7.4. Figure D.1 is similar to figure 6.8, where every algorithm is trained using a *Cross Validate Model* module.

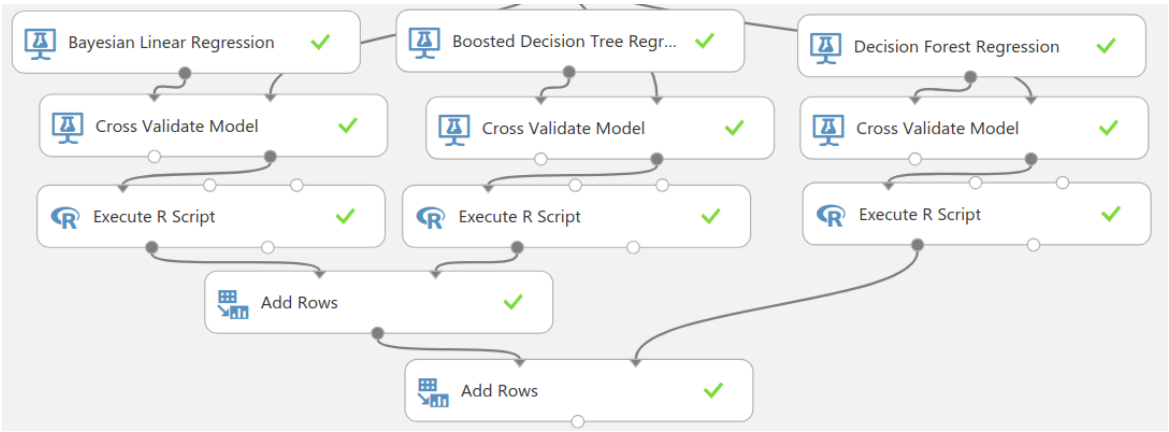


Figure D.1: A snippet of Azure Machine Learning, where models were trained.

In order to evaluate all the algorithms at the same time, we added an R script to each algorithm in order to extract the evaluation results from them individually, because Azure Machine Learning does only have a module for comparing two algorithms at the same time. We used this approach since we had 5 algorithms to test. The *Add Row* module is just used to append the results from the algorithms to a data set. The scripts used can be seen below.

Listing D.1: The code inside the R Script module for the Boosted Decision Tree Regression model, which extracts the evaluation results from the model.

```
# Select the incoming dataset from input 1, and save it in 'dataset'
dataset <- maml.mapInputPort(1)
# Add column with algorithm name
data.set <- data.frame(Algorithm='Boosted Decision Tree Regression')
# Select columns 4 to 8 in row 5 (which are the evaluation metrics for the cross validation mean)
data.set = cbind(data.set, dataset[5,4:8]);
# Select data.set and make sure it is the output of this module
maml.mapOutputPort("data.set");
```

Listing D.2: The code inside the R Script module for the Bayesial Linear Regression model, which extracts the evaluation results from the model.

```
# Select the incoming dataset from input 1, and save it in 'dataset'
dataset <- maml.mapInputPort(1)
# Add column with algorithm name
data.set <- data.frame(Algorithm='Bayesian Linear Regression')
# Select columns 4 to 8 in row 5 (which are the evaluation metrics for the cross validation mean)
data.set = cbind(data.set, dataset[5,5:9]);
# Select data.set and make sure it is the output of this module
maml.mapOutputPort("data.set");
```

Listing D.3: The code inside the R Script module for the Decision Forest Regression model, which extracts the evaluation results from the model.

```
# Select the incoming dataset from input 1, and save it in 'dataset'
dataset <- maml.mapInputPort(1)
# Add column with algorithm name
data.set <- data.frame(Algorithm='Decision Forest Regression')
# Select columns 4 to 8 in row 5 (which are the evaluation metrics for the cross validation mean)
data.set = cbind(data.set, dataset[5,5:9]);
# Select data.set and make sure it is the output of this module
maml.mapOutputPort("data.set");
```

Appendix E

The Mean Absolute Error (MAE) describes the average value of the error margin that is in the predictions. The absolute of the value is used in order to avoid negative errors (i.e. errors that predict a higher value than the true value) subtracting from the total sum before calculating the mean. The formula for calculating the MAE is shown in equation E.1.

$$MAE = \frac{1}{N}\sum_{i=1}^{N}|P_i - T_i| \tag{E.1}$$

Where

N = The total amount of true values
P_i = Prediction of instance i
T_i = True value of instance i

The Root Mean Squared Error (RSME) describes the relative error similarly to the MAE, however the RSE squares the errors prior to calculating the mean. This serves to punish errors disproportionately and as such the RSE is sensitive to outliers. The formula for calculating the RMSE is shown in equation E.2.

$$RMSE = \sqrt{\frac{1}{N}\sum_{i=1}^{N}(P_i - T_i)^2} \tag{E.2}$$

Where

N = The total amount of true values
P_i = Prediction of instance i
T_i = True value of instance i

Relative Absolute Error (RAE) gives an indication of how well the predicted value correlates to the true value. E.g. if the prediction is 3 and the true value is 30 then the RAE would be 3/30 = 0.1. The lower the RAE, the better the prediction is. Essentially, the relative error shows how large an impact the errors respective to the true value. The formula for calculating RAE is shown in equation E.3

$$RAE = \frac{\sum_{N}^{i=1} |P_i - T_i|}{\sum_{N}^{i=1} |T_i - M_{P_i}|} \tag{E.3}$$

Where

P_i = Prediction of instance i
T_i = True value of instance i
M_{P_i} = Mean of the true values

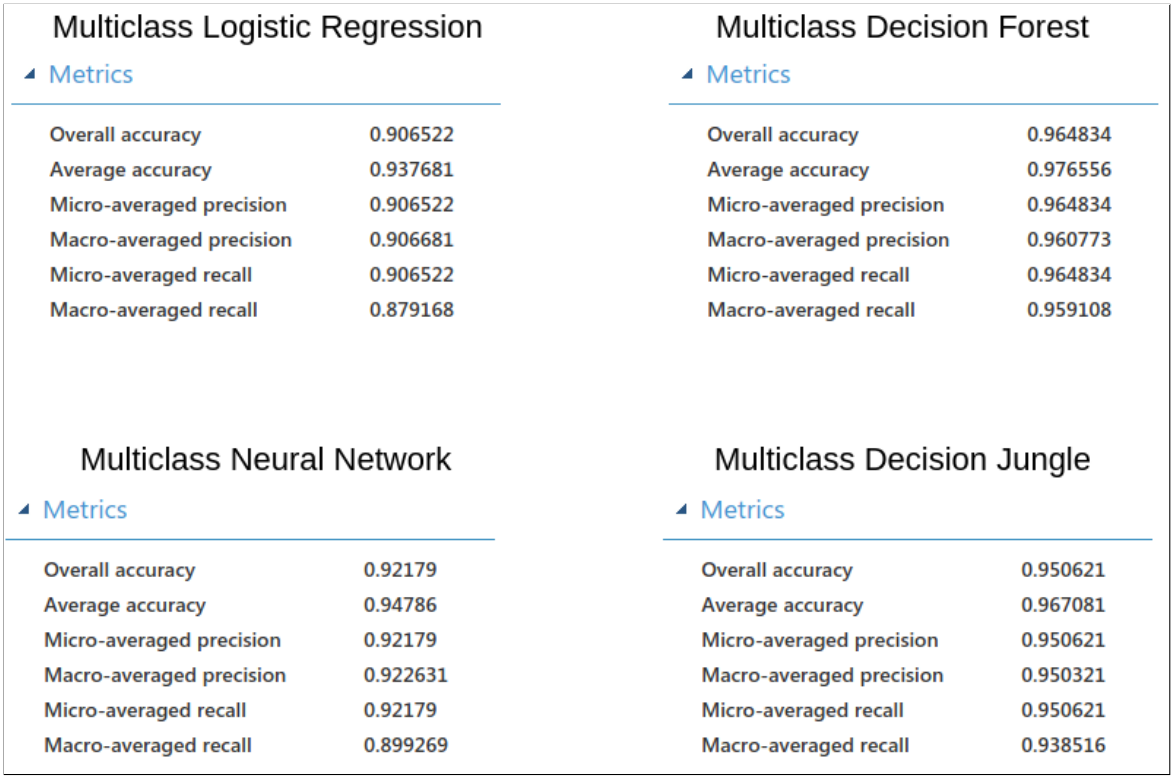
Relative Squared Error (RSE) computes the error relative to the true number similarly to how RAE does it. However, instead of taking the absolute it squares the errors similarly to a RMSE in order to punish errors disproportionately. The formula for RSE is shown in equation E.4

$$RSE = \frac{\sum_{N}^{i=1} (P_i - T_i)^2}{\sum_{N}^{i=1} (T_i - M_{P_i})^2} \tag{E.4}$$

Where

P_i = Prediction of instance i
T_i = True value of instance i
M_{P_i} = Mean of the true values

Appendix F

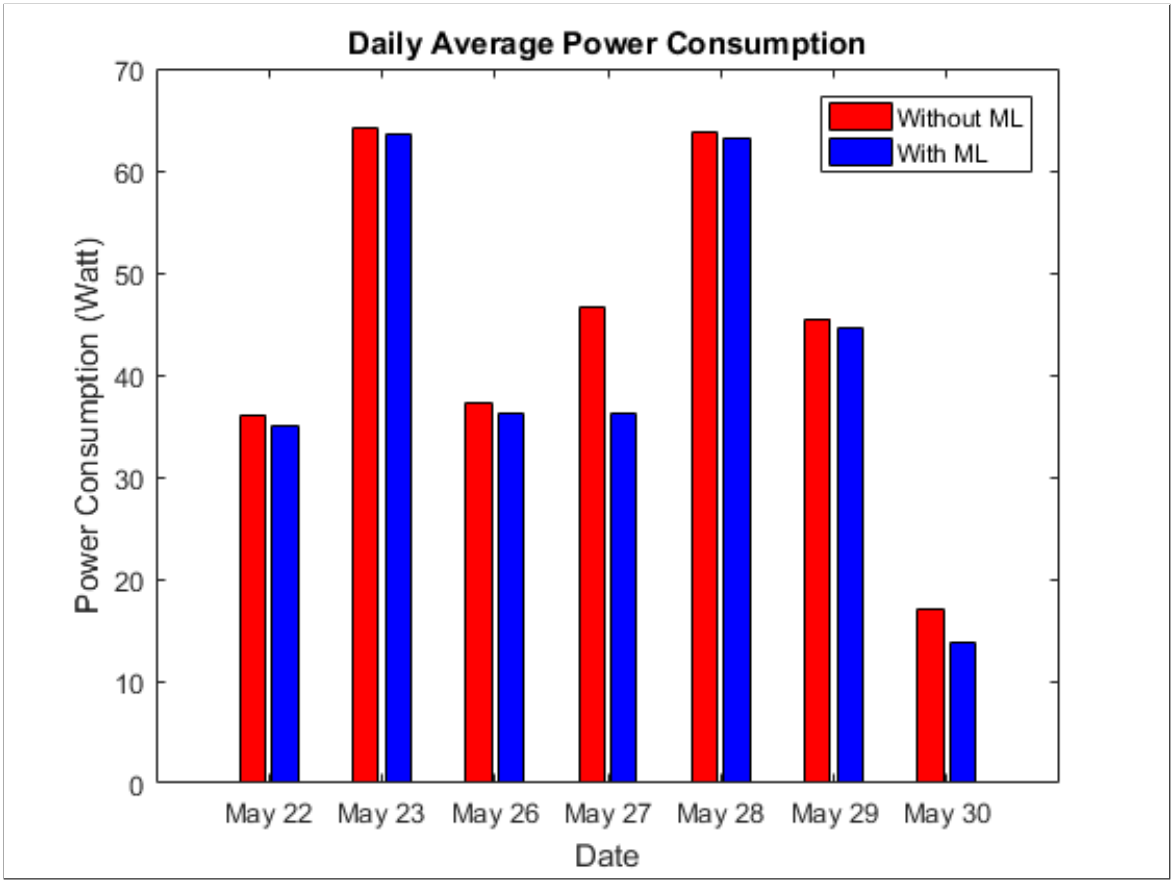

Multiclass Logistic Regression
Metrics
Overall accuracy 0.906522
Average accuracy 0.937681
Micro-averaged precision 0.906522
Macro-averaged precision 0.906681
Micro-averaged recall 0.906522
Macro-averaged recall 0.879168
Multiclass Decision Forest
Metrics
Overall accuracy 0.964834
Average accuracy 0.976556
Micro-averaged precision 0.964834
Macro-averaged precision 0.960773
Micro-averaged recall 0.964834
Macro-averaged recall 0.959108
Multiclass Neural Network
Metrics
Overall accuracy 0.92179
Average accuracy 0.94786
Micro-averaged precision 0.92179
Macro-averaged precision 0.922631
Micro-averaged recall 0.92179
Macro-averaged recall 0.899269
Multiclass Decision Jungle
Metrics
Overall accuracy 0.950621
Average accuracy 0.967081
Micro-averaged precision 0.950621
Macro-averaged precision 0.950321
Micro-averaged recall 0.950621
Macro-averaged recall 0.938516

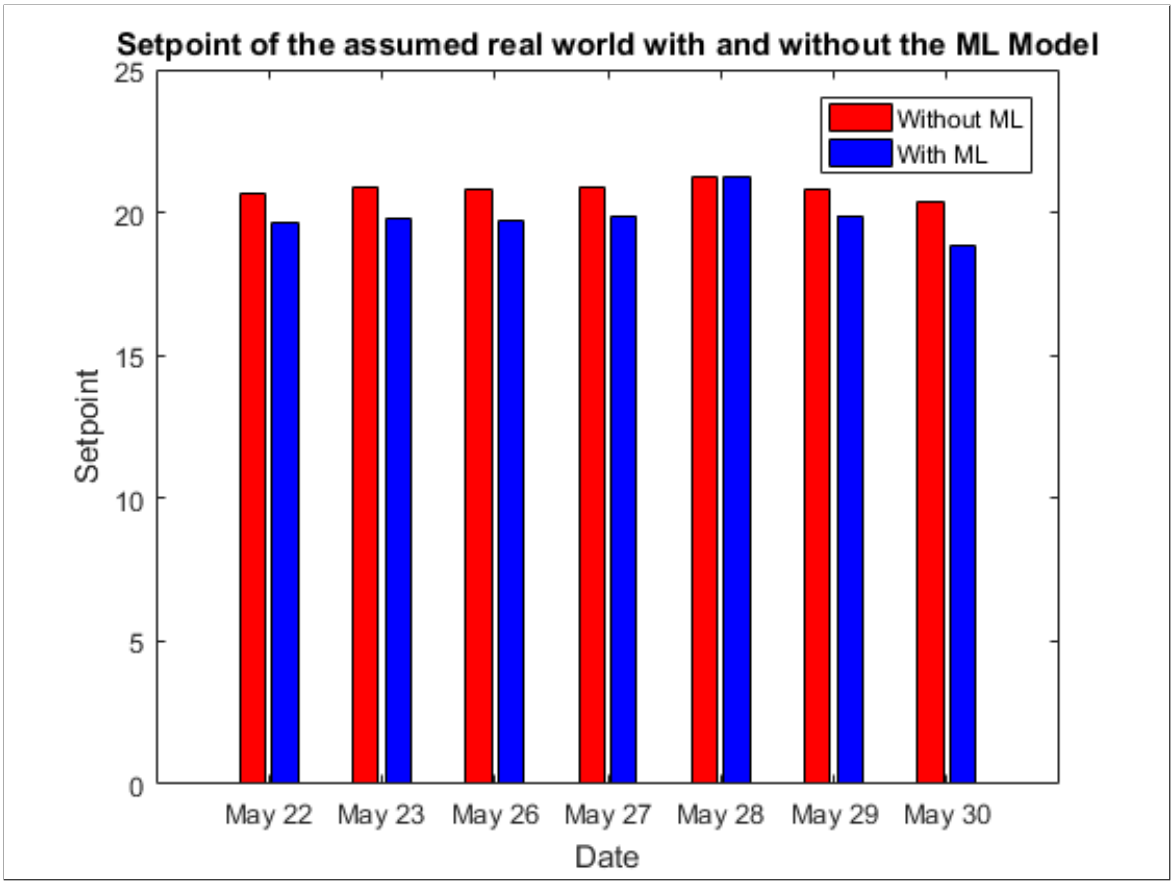
Appendix G

Table G.1: This table shows the statistical values for a specific status, the day, the average time, the σ, and the accepted interval. The data are from the 26th of April to the 21st of May (used for machine learning)

User status	**Day**	**Average time (Minutes since midnight)**	**σ (Minutes since midnight)**	**Accepted Interval ($\pm 1\sigma$ in HH:MM)**
Goes to bed	Weekday	1481	71	23:30 - 01:52
Wakes up	Weekday	534	29	08:25 - 09:23
Leaves home	Weekday	768	198	09:30 - 16:06
Comes home	Weekday	1111	114	16:36 - 20:25
Goes to bed	Weekend	1452	15	23:57 - 00:27
Wakes up	Weekend	556	59	08:16 - 10:14
Leaves home	Weekend	806	233	09:33 - 17:19
Comes home	Weekend	1104	115	16:29 - 20:19

Appendix H


Daily Average Power Consumption
Without ML
With ML
Power Consumption (Watt)
0
10
20
30
40
50
60
70
May 22
May 23
May 26
May 27
May 28
May 29
May 30
Date


Setpoint of the assumed real world with and without the ML Model
Without ML
With ML
Setpoint
25
20
15
10
5
0
May 22
May 23
May 26
May 27
May 28
May 29
May 30
Date

Appendix I

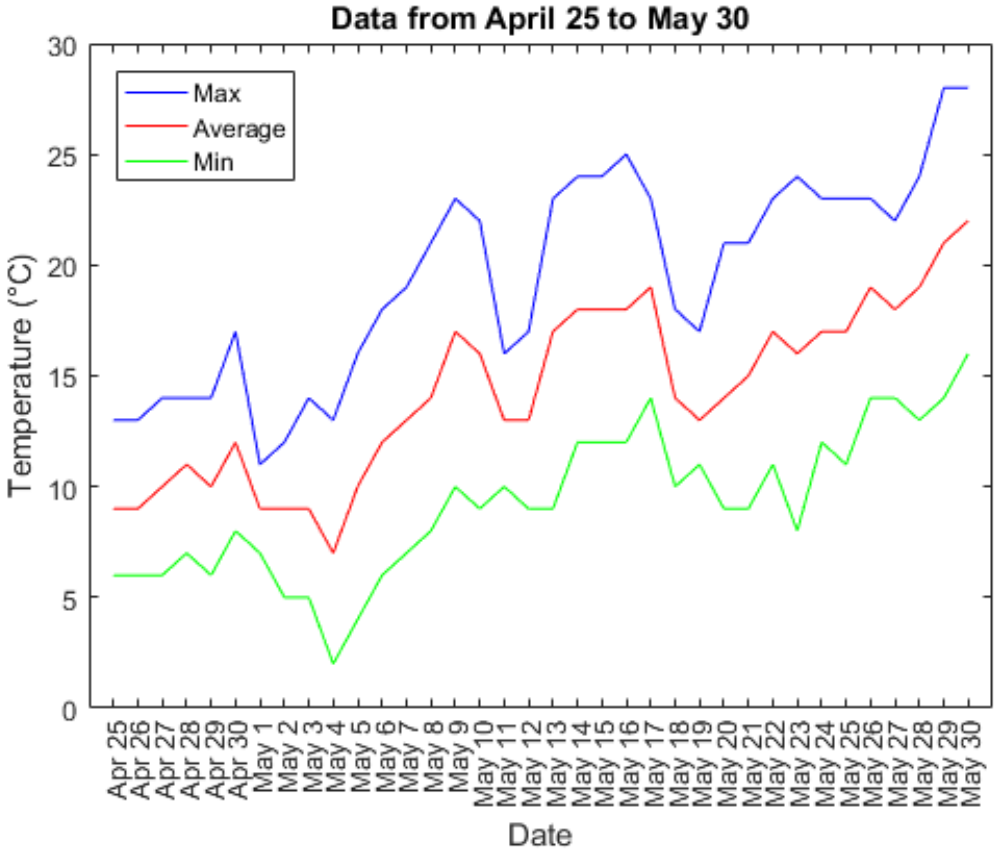


Figure I.1: Outside temperature of Copenhagen. Data from [123]