Master's thesis

NTNU
Norwegian University of Science and Technology
Faculty of Information Technology and Electrical Engineering
Department of Engineering Cybernetics

Cassandra Berdahl

Human-MPC Interface for Smart Houses

Development of a customized web application for a smart heating system controlled by a Model Predictive Control algorithm, with the intent of facilitating the average user to interact with and understand the smart control.

Master's thesis in Industrial Cybernetics
Supervisor: Sebastien Gros
June 2022

NTNU
Norwegian University of Science and Technology

Cassandra Berdahl

Human-MPC Interface for Smart Houses

Development of a customized web application for a smart heating system controlled by a Model Predictive Control algorithm, with the intent of facilitating the average user to interact with and understand the smart control.

Master's thesis in Industrial Cybernetics
Supervisor: Sebastien Gros
June 2022

Norwegian University of Science and Technology
Faculty of Information Technology and Electrical Engineering
Department of Engineering Cybernetics

NTNU
Norwegian University of
Science and Technology

Preface

This thesis is the final part of the Master's degree in Industrial Cybernetics and was written during the spring semester of 2022 at the Norwegian University of Science and Technology (NTNU) in Trondheim, Norway. The Master's degree belongs to the Faculty of Information and Electrical Engineering and the Department of Engineering Cybernetics.

This report is a continuation and supplement to a larger project (POWIOT), which investigate the implementation of an IoT-based smart house system with a Model Predictive Control (MPC) algorithm for controlling heat pumps. Prior to writing this thesis, the IoT structure and control algorithm has been implemented in the house. This thesis investigates the feasibility of implementing a JavaScript web application for the MPC scheme.

During the course of this project, I have gained a lot of experience with application development, which I initially had little experience with. Furthermore, I have also developed a better understanding of the usage of MPC algorithms in smart energy management in households and the challenges associated with implementing such a system.

The project is initiated and supervised by Professor Sebastian Gros at NTNU. I would like to show my appreciation by thanking my supervisor for allowing me to write this thesis, and for testing the software on the real-time system installed in his house. I would also like to thank him for valuable discussions about the control algorithm and for providing feedback on the report.

Cassandra Berdahl
6th of June 2022, Trondheim

Abstract

Smart house technologies are rapidly developing, with a focus on more advanced control strategies, such as Model Predictive Control (MPC), for optimizing heating systems to enable smart energy management. As digitization and the Internet of Things (IoT) become more prevalent, this installation has become more viable in households. The practical application, however, is still in its infancy due to challenges related to user adoption.

This thesis seeks to evaluate the feasibility of developing and implementing a human-MPC interface for an intelligent heating system controlled by an MPC algorithm. In the efforts to ease the adoption of more complex control strategies, a web application is developed with the intent of explaining concepts related to the MPC scheme. This will allow for monitoring, interacting, and potentially understanding the optimal control of the heating system.

The perceived challenges related to the implementation are having highly customized elements and graphical displays to explain the MPC behavior in a simple non-control-related language. The results obtained in this thesis show that developing a web interface is a viable implementation for this case study of a smart house located in Trondheim, Norway. The web application is developed using JavaScript and represents a full-stack application with a server and a client-side. This web interface provides customized functionalities for explaining and visualizing important concepts related to the predictive control algorithm. Eventually, the findings in this report serve as the basis for further development of the application and eventually completely integrate it with the existing smart house system.

Further work should focus on improving the software for data processing on both the server and client-side. Eventually, include more functionalities to further exploit the predictive capabilities of the MPC scheme and to accommodate more customized elements.

Sammendrag

Smarthusteknologier er i rask utvikling, der mer fokus er rettet mot avanserte kontrollstrategier som Modell Prediktiv Kontroll (MPC), for å optimalisere varmesystemer som kan muliggjøre smart energistyring i bygg. Ettersom digitalisering og Tingenes Internett (IoT) blir mer utbredt, har denne installasjonen blitt mer levedyktig i husholdninger. Den praktiske anvendelsen er dog fortsatt i sin spede begynnelse på grunn av utfordringer knyttet til brukeradopsjon.

Denne oppgaven vil evaluere muligheten for å utvikle og implementere et menneske-MPC grensesnitt for et intelligent varmesystem kontrollert av en MPC-algoritme. I arbeidet med å lette bruken av mer komplekse kontrollstrategier, utvikles en web applikasjon med den hensikt å forklare konsepter knyttet til MPC algoritmen. Dette vil tillate overvåking, samhandling og potensielt forståelse av den optimale kontrollen av varmesystemet.

Utfordringene knyttet til implementeringen er å utvikle svært tilpassede elementer og grafiske illustrasjoner for å forklare MPC på et enkelt ikke-kontrollrelatert språk. Resultatene oppnådd i denne oppgaven viser at å utvikle et webgrensesnitt er en levedyktig implementering for denne case-studien av et smarthus lokalisert i Trondheim, Norge. Web applikasjonen er utviklet med JavaScript og representerer en fullstack-applikasjon med en server og en klientside som gir tilpassede funksjoner for å forklare og visualisere viktige konsepter knyttet til kontrollalgoritmen. Funnene i denne rapporten vil til syvende og sist fungerer som et grunnlag for videreutvikling av smart hus applikasjonen og integreres fullstendig med det eksisterende smarthussystemet.

Videre arbeid bør fokusere på å forbedre programvaren for databehandling på både server- og klientsiden. Til slutt, inkludere flere funksjoner for ytterligere å utnytte de prediktive egenskapene til MPC algoritmen og for å imøtekomme mer tilpassede elementer.

List of Acronyms

Term	Definition
AMS	Advanced Metering System.
API	Application Programming Interface.
COP	Coefficient of Performance.
CSS	Cascading Style Sheets.
HAN	Home Area Network.
HMI	Human Machine Interface.
HP	Heat Pumps.
HTML	HyperText Markup Language.
HTTP	HyperText Transfer Protocol.
HVAC	Heating, Ventilation and Air Conditioning.
IoT	Internet of Things.
JS	JavaScript.
JSON	JavaScript Object Notation.
MET	Norwegian Metrological Institute.
MHE	Moving Horizon Estimation.
MPC	Model Predictive Control.
NLP	Non-Linear Programming.
PID	Proportional-Integral-Derivative.
REST	Representational State Transfer.
SCADA	Supervisory Control And Data Acquisition.
SYSID	System Identification.
UI	User Interface.
URL	Uniform Resource Locator.
VAT	Value Added Taxes.

List of Symbols

Symbol	Definition
C_{base}	BasePrice.
C_{diff}	Difference in cost.
$C_{\text{diff,high}}$	High difference in cost.
C_{grid}	Grid rent.
C_{spot}	Spot price.
$C_{\text{spot,avg}}$	Average of spot prices.
$C_{\text{spot,high}}$	High spot prices.
C_{24h}	24 hour cost.
Conv	Convection.
Fan	Fan level on heat pumps.
ΔFan	Difference in fan level.
k	Time step.
N	Prediction horizon.
P_{corr}	Power correction.
P_{HP}	Heat pump power.
$S_{\text{discomfort}}$	Slack variable for the reference temperature.
S_{min}	Slack variable for the minimum temperature.
T_{diff}	Difference between reference and room temperature.
$T_{\text{diff,high}}$	High difference between reference and room temperature.
T_{min}	Minimum temperature.
T_{ref}	Reference temperature.
T_{room}	Room temperature.
T_{target}	Target temperature.
T_{out}	Outside temperature.
T_{wall}	Wall temperature.
ΔT_{target}	Difference in target temperature
w	Weight.
w_{spot}	SpotGain weight.
$w_{\text{temp.above}}$	Temperature above weight.

Table of Contents

List of Figures

List of Tables

List of Code

Chapter 1

Introduction

The recent development within smart house solutions has accelerated the adoption of Internet of Things (IoT) devices and contributed to facilitating smart energy management in buildings. In terms of smart heating control, research papers published in the last decade have investigated the possibilities of more complex approaches such as optimal predictive control. The application of advanced building controls, such as Model Predictive Control (MPC), has been shown to enhance energy efficiency in buildings, by optimizing heating. Nevertheless, the implementation and adoption of this technology in the average household are still in the early stages, regardless of the abundance of research papers. This is related to the average user's limited perception of more advanced control strategies. Albeit MPC can provide several advantages in terms of predictive capabilities, thermal comfort, energy savings, and cost savings, the user tends to be hesitant regarding implementing complex systems in their households. As a result, more user-friendly approaches have to be investigated to sustain the development without compromising the understanding. [1]

1.1 Motivation

Modern alternatives to reduce energy consumption is necessary for the power grid to withstand the increased electrification. More specifically reduce the power peaks that contribute to overloading the capacity of the grid. The management of heat consumption and the implementation of effective heating strategies have become a key facet of energy management in residential buildings since, generally, household appliances are powered by electricity and contribute to a large proportion of the energy demand. [2]

Today, buildings account for approximately 40% of the energy demand in Norway, whereas households account for 30% [3]. Although more energy-efficient technologies such as heat pumps have become a common heating device in Norwegian households, the energy demand is expected to increase, whereas approximately 80% of the household heating comes from electricity [3]. Increased electrification can challenge the existing grid infrastructure. To avoid expensive grid investments new alternatives to handle the expected increase in electricity demand is essential for a sustainable energy transition. Peak power demand can incur high investment costs in the grid infrastructure, due to the constant need to upgrade the electrical components in the power grid.

The Norwegian government has made a proposal for directives considering the energy performance of buildings, which mainly contains provisions on energy requirements [4]. These directives are related to buildings' energy performance and energy efficiency which are stated as the following:

- All new buildings must have self-regulating equipment for temperature in each room / each heating zone. Existing buildings require the installation of such equipment when heat generators are replaced. The requirements only apply if it is technically and economically feasible.
- By 2025, all commercial buildings with heating and air conditioning systems over 290 kW must have automation and control systems if this is technically and economically feasible.

To comply with new regulations in building energy management, more advanced control strategies for regulating consumption will become essential. However, today, buildings tend to adopt classic control strategies with limited energy-saving capabilities. Albeit classic control strategies such as thermostats are easy to implement, they only provide simple regulation of temperature. One of the concern in regards to implementing modern control strategies in buildings are the requirement of expensive equipment and set-ups. This has sequentially led to the average user not being able to install such a system or understand it.

The MPC control method has been studied since the 1970s and has been extensively used in process control and is now becoming an interesting control methods for building energy management. MPC has been investigated for optimizing the control of energy-consuming units in households and typically relies on the spot prices to reduce the heating. Implementing optimal control in buildings has proven to reduce energy consumption and/or related electricity costs without compromising thermal comfort completely. Thus, optimal control is not only beneficial in terms of improving energy efficiency, another significant driving factor for the integration of MPC is the potential of reducing costs related to electricity. In December 2021, a new power price record was observed as a result of high power prices in major parts of Europe, partly due to increased

prices for gas, coal, and CO2 emissions [5]. In addition, the integration of intermittent renewable energy sources can increase the price volatility. With MPC, additional information such as weather forecasts and spot prices can be included in the controller design in order to predict and optimize the heating and automatically shift the consumption to lower-priced hours. As such the consumer will experience a lower electricity bill. However, modern optimal control strategies can be complicated for the average user to understand.

As a response to the challenges with implementation and understanding of more complex control strategies, the user perspectives have to be considered. The success of smart house solutions ultimately depends on how easily people can integrate them into their daily lives. A key element of any control implementation is the communication of data from the smart system to the user. In order for the average user to adopt smart house solutions, the user should be able to understand the system and interact with the devices installed.

A potential solution to integrate the user into the system is by developing more intuitive smart home applications to sustain the technological transition. The current development in smart home applications and services are expanding, however, with more modern alternatives to control strategies, these applications will require more functionalities, customization, and information to maintain a good user experience. This relates to concepts of how to understand and interact with higher complexity systems without compromising the understanding of the user. [6]

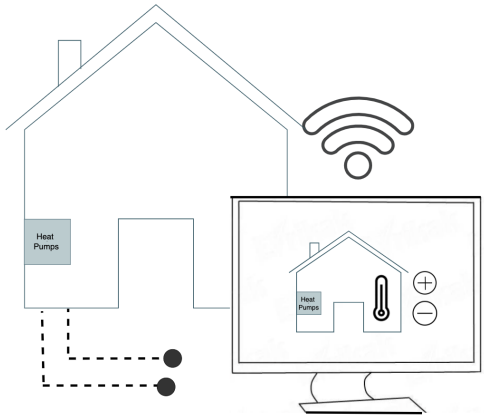


Figure 1.1: Illustration of a web interface for a smart control enabled by IoT.

With IoT, customizing your own software to fit the system has become more viable. The emergence of commercial IoT devices along with the installation of the Advanced Metering System (AMS) in the majority of Norwegian households, has introduced intelligence and real-time data

gathering and the potential for optimized electricity usage. With the development of home automation, digitization, and IoT, other technologies such as smart home application for including the user is necessary for a successful implementation of MPC solutions in residential homes. In figure 1.1, an illustration of a user interfaces for controlling heat pumps with IoT is presented.

Today, there are few smart home applications and services that support MPC, which contributes to maintaining the gap between the development of home automation technologies and user adoption. Thus, by investigating how smart home applications can accommodate modern control strategies and users, some of the problems related to adoption can potentially be resolved.

1.2 Problem statement

The main objective is to develop and implement a customized web application for a smart house control algorithm in order for the average user to interact with an advanced control strategy such as MPC. A simplified language is provided for the purpose of explaining control concepts related to the modern optimal control strategy. The MPC scheme is able to predict future temperature trajectories to perform optimal control of heat pumps.

The optimized output seeks to make a compromise between thermal comfort and monetary cost. One of the key challenges related to developing this human-MPC interface is to present the MPC data intuitively. This will encompass explaining concepts such as, why the thermal comfort level is not attained, explaining how different elements are influencing the MPC scheme and how the user can choose the outcome according to comfort and economic preferences. Furthermore, contribute to evaluating how a human-MPC interface can reduce the challenges related to the adoption of complex control strategies in the average household.

1.3 Structure of the thesis

This thesis contains a total of eight chapters, including the chapter 1, which presents the introduction. Chapter 2, describes the relevant theory for the project such as smart energy management in households, MPC, and the state of the art in smart houses and software. Chapter 3 describes the system overview with a focus on the MPC scheme. Chapter 4 presents the software tools used to develop the web application and chapter 5 describes an overview of the software implementation. Chapter 6 demonstrates the results obtained from developing the interactive web application components. Chapter 7 presents a discussion of mainly the software implementation and potential improvements. Finally, chapter 8 draws conclusions based on the discussion and presents suggestions for further development of the application.

1.4 Limitations

The web application presented in this thesis is developed to solely support controlling and monitoring the heat pumps regulating the temperature in a smart home located in Trondheim. Other IoT devices installed in the house are not covered in this thesis. The web application is designed specifically for the case study and will not work correctly for other smart homes.

Furthermore, the application is developed locally which refers to the whole application running on the local computer. This entails that to access the web interface, the user is required to have the software installed and running on their computer.

The thesis report is not concerned with providing a complete description of the control design of the smart house algorithm. The software implemented prior to writing this thesis is assumed to work correctly. Thus, further considerations for the house control algorithm are not investigated in this thesis. Instead, the thesis attempts to demonstrate how an easy understanding of the specific MPC schemes operating the heat pumps in the smart house can be gained by implementing a web-based software solution. Eventually, the development of the application has the goal of creating a foundation that can be used to further customize the environment for the smart house featured in this study.

Albeit this thesis focus on the user perspective, the web application has not been tested on average users. The user perspective can to some degree have been influenced by the author's knowledge of control concepts such as MPC and optimization in general. The assumption in this thesis is based on how the author views how the average user can understand modern optimal control strategies such as MPC.

Chapter 2

Theory

This chapter provides relevant background information for understanding concepts related to smart control of heating systems in buildings and home automation. In sections 2.1 and 2.2 the power consumption related to Norwegian households and the spot price market is explained. Furthermore, in section 2.3 the relevant theory behind MPC is provided. Section 2.4 presents the state of the art in smart home technologies with a focus on advanced control algorithms, such as MPC, and potentials for home automation software and services.

2.1 Power consumption

Considering that the power grid is designed to tolerate the highest power consumption, the increased electrification can result in more frequent upgrades of the grid infrastructure. The power grid components must have sufficient capacity to handle the highest peaks and the operation of upgrading these are considered expensive and should be avoided. The power consumption varies throughout the year, depending on the outside temperature. In terms of heating consumption, especially during colder months can contribute to peak demand periods. When numerous households consume energy simultaneously the capacity of the grid is challenged.

Figure 2.1 shows the hourly power consumption of appliances in the smart house investigated in this thesis during a day at the end of March 2021. Higher demands can be observed in the morning around 8:00-11:00 and late afternoon around 17:00-20:00. A similar trend is evident in average Norwegian households, where the majority of energy consumption comes from household heating.

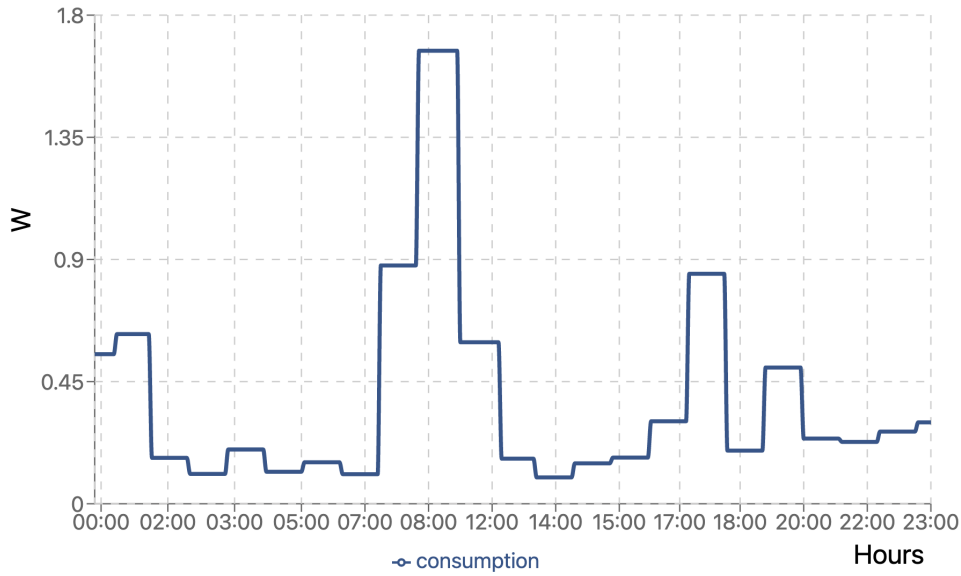


Figure 2.1: Measured consumption of home appliances [W] in the smart house during the 30th of March 2021.

Accordingly, introducing smart energy management can contribute to reducing the strain on the grid and perform load shifting, such that the user can even out their consumption and enhance their flexibility. Since the majority of the energy demand is related to electricity-generated heating there is a great potential for further reducing the consumption in the energy sector by optimizing heating in households. [2]

2.1.1 Heat pumps

The prevalence of heat pumps has increased, especially in recent years. In Norway, heat pumps have become one of the most common heating systems installed in households. While heat pumps are electric-driven, the energy efficiency is considered to be higher than other heating systems. Generally, heat pumps are considered energy efficient because they are capable of extracting heat from an environment that is cooler by using moderate amounts of energy. In order to measure the efficiency, the ratio between the electrical input and the heat output is calculated and referred to as the Coefficient of Performance (COP). On the other hand, electric-heated systems contribute to power peaks. With the increased use of electricity as an energy carrier, more energy-efficient technologies are required. Developing and implementing effective heat pump control techniques is essential to enable more energy-efficient and flexible use of electricity. [7]

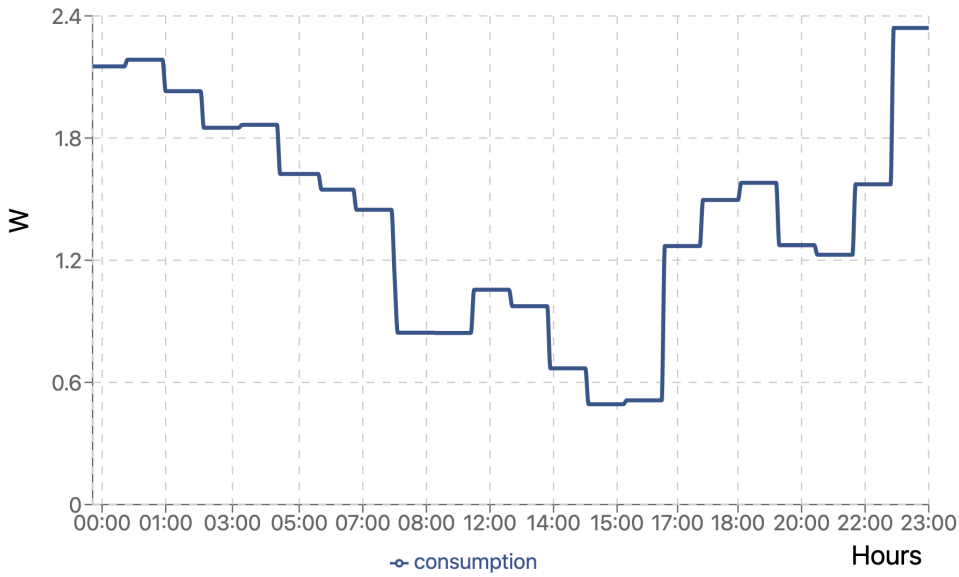


Figure 2.2: Measured consumption of heat pumps [W] in the smart house during the 30th of March 2021.

In figure 2.2, the heat pump consumption is optimized by implementing a modern control strategy into the controller design. To a large extent, the consumption is minimized during the peak hour demand. By optimizing the control of heat pumps, buildings can further reduce their energy consumption by improving their efficiency, thus there are several benefits of implementing energy and cost-saving actions into the controller design of heat pumps.

2.1.2 Demand response

Smart buildings with optimal heating control can exploit the electricity prices to perform demand-response. This term refers to balancing the consumer electricity consumption during peak demand periods. Moreover, with the increased usage of electricity from renewable energy sources, the grid can experience disturbances due to intermittent generation, and the supply and demand are reflected in the spot price market, further explained in section 2.2. As such, the consumers should become a flexible part of the power grid and participate in a demand-response program according to the spot prices. [8]

Reducing the electricity bill can be an important motivation for consumers to use energy in a more conscious manner. However, with the advancement in smart house technology, the system

can react automatically to the variations in the spot market. Considering heating systems, such as heat pumps, can have the ability to take advantage of lower electricity prices, e.g. increase the temperature to heat the thermal mass of the building in advance of peak spot prices, the heat pumps can avoid high electricity costs and contribute to enhancing the flexibility. Typically, the highest spot prices occur during peak hour demand, hence price-based demand response is able to reduce the consumption with the purpose of reducing the cost. [9]

2.2 Spot price market

This section provides information regarding the spot price market Nordpool and electricity costs for households in Norway. In this project, the smart control of heat pumps takes into account the optimization of the electrical consumption against the electricity spot market. As such, the consumption is adjusted according to electricity prices, and the supply and demand in the power grid are better utilized.

2.2.1 Nord Pool

The Norwegian power grid operates under the European power exchange market Nord Pool, which offers both day-ahead and intraday electricity markets across the Nordic regions. In this project, the day-ahead spot prices are used, where the spot price refers to the hourly price of electricity. The day-ahead market is an auction-based exchange of electricity.

In order to maintain a balance between production and consumption of power, Norway can import and export power between the connected countries and regions, depending on the power supply and demand. The power exchange flows from areas with low prices to areas with high prices.

Norway is divided into different price areas (electricity spot areas), and electricity prices are based on which area the consumer belongs to. Purchases and sales of electricity are made per area, where the spot prices are higher if there is an imbalance between the anticipated supply and demand in that region. The regions in Norway are divided into five geographical elspot areas, illustrated in figure 2.3. [11]

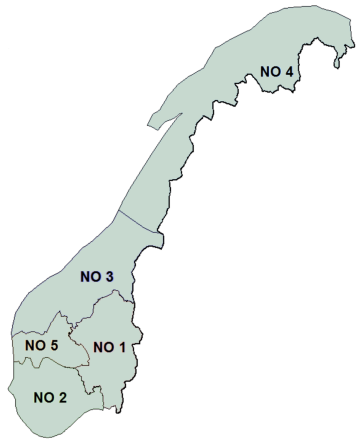


Figure 2.3: Illustration of the five geographical elspot areas in Norway [10].

2.2.2 Electricity cost

The electricity cost a private consumer pays is depending on the type of the power agreement with the power suppliers. There are three types of agreements, fixed-, variable- or spot-based power agreements. In this project, the power agreement is based on spot prices.

The electricity bill varies with both the consumption and prices - hour by hour. Having a spot price-based electricity agreement has proven to be the most affordable option for consumers, in particular, if the consumer uses electricity when the spot prices are low.

The private consumer is required to pay a monthly electricity bill that consists of two parts; the electricity price paid to the power supplier, and the grid rent paid to the network company, including taxes (VAT of 25%, statutory payment to the Energy Fund (Enova) of 0.125 NOK / kWh and consumption tax on electric power of 0.1926 NOK / kWh). The first part is measured in kWh, which is based on the spot prices from Nord Pool. This is registered by a home-installed AMS. In the second part, the grid rent is divided into a fixed part determined by the network company and a variable part determined by the power consumption in kWh. [12]

In 2022 there is proposed a new grid rent which will reward customers who cut the peaks in their electricity consumption, by price incentives in grid tariffs or other economic incentives. This will be formally introduced on the 1st of July, 2022. By introducing a power tariff to limit the congestion, the consumption is evened throughout the day. The goal is to provide better utilization of the power grid. These changes will affect the grid rent, such that the fixed part will change to depend on the maximum power consumption in kW and the variable part will depend on when, during the day and week, the power is used. [13]

Accordingly, the consumer pays for the actual energy use in kWh and the power in kW. This will primarily be beneficial for the power suppliers and the transmission grid operators, however, the consumers need to adopt a more conscious consumption behavior. By allowing a smart energy management system to account for this regulation, the consumers can better control their power consumption and avoid high electricity bills. There might be an increase or decrease in the grid load during different periods, reflected in the spot prices, hence this information can contribute to the smart energy management of heating. In regards to these proposed changes, reducing energy consumption when the spot prices are high can contribute to economic and operational advantages for the consumer and the transmission grid operator.

2.3 Model Predictive Control

The MPC principle is described as an advanced control strategy that optimizes an objective to control and predict future outputs of a dynamic process. By using the current measurement of the states, the MPC scheme calculates an optimal control sequence over a finite prediction horizon while satisfying a set of constraints. By predicting future outputs of the system, the algorithm is able to compute the optimal control input to drive the predicted output to the desired reference, where only the first step of the control input is implemented. Hence, for the next given time horizon, a new control input sequence is generated. [14]

$$\min_{x} \quad f(x) \tag{2.1}$$

$$\text{subject to} \qquad c_i(x) = 0 \quad i \in \mathcal{E} \tag{2.2}$$

$$c_i(x) \leq 0 \quad i \in \mathcal{I} \tag{2.3}$$

An MPC algorithm is based on solving an optimization problem at each time step to determine the optimal control action by minimizing the objective. In equation 2.1 a general cost function $f(x)$ is minimized, where x is the decision variable that minimizes the objective. The minimum needs to satisfy a set of constraints, described by equality constraints in equation 2.2, and inequality constraints in equation 2.3 (where $\mathcal{E}$ and $\mathcal{I}$ are sets of indices for equality and inequality constraints). An optimized-based algorithm is iterative methods that starts with an initial guess and seeks to improve the solution based on the objective.

The MPC can predict the dynamic evolution of the system and the changes in the temperature dynamics. However, to implement MPC in buildings, a good dynamic temperature model have to be established. A technique classified as gray-box modeling is most suitable and well adapted to perform optimization. A gray box model is essentially a combination of white and black box models. The model is formed from physics-based methods while the parameters are based on estimation algorithms. Such control algorithms use mathematical models, where thermal inertia is computed and used to avoid undershoot or overshoot of temperatures. [1]

A general output feedback MPC algorithm is demonstrated in table 2.1. Output feedback refers to the states being estimated by using available measurements of the states, instead of direct feedback from raw measurements, further detailed in section 3.2.2. Based on the currently estimated states and the dynamic model, the MPC is responsible for calculating, for each instance, the predictions of the dynamic evolution of the process. The prediction horizon changes due to the moving horizon and a new control problem are solved at time t + 1, discrete-time.

Table 2.1: Simplified output feedback algorithm for MPC [14].

Algorithm Output Feedback
for $t = 0, 1, 2, ...$ **do**
Compute an estimate of the current state $\hat{x_t}$ based on the measured data up until time t.
Solve a dynamic optimization problem on the prediction horizon from t to t+N with $\hat{x_t}$ as the initial condition.
Apply the first control move u_t from the solution above.
end for

The objective of an MPC scheme in buildings is commonly related to the thermal comfort of the user, however, other desired objectives can include minimizing energy use and monetary costs. Due to the predictive behavior of the control algorithm, additional information such as day-ahead spot prices and weather forecasts can be included, further explained in section 2.4.2.

Accounting for spot prices in the optimization is advantageous since the MPC can be used to optimally control the efficiency of energy use in a smart house and further contribute to performing local automated load shifting. Although minimizing consumption can contribute to minimizing the monetary costs, it is not necessarily the case since the electricity prices are volatile. As a result of the predictive behavior of the MPC, the control algorithm is able to make an informed decision on how to regulate the heating in a smarter way based on additional information such as spot prices and weather forecasts. [8]

2.4 State of the art of energy management in smart houses

Smart homes have been researched for nearly a couple of decades and the concept has gained widespread recognition in the society. In particular, research considering improving the energy efficiency of heating systems has gained significant attention. The approach to optimizing the operation of heating systems, such as heat pumps, requires more sophisticated control methods, posing challenges in terms of implementation. Nevertheless, advancements in home automation and the introduction of IoT have made it easier for homeowners to implement such systems to enable smart energy management.

2.4.1 IoT devices

The installation of AMS in Norwegian households has become a key enabler for IoT and the adoption of smart home technologies. An IoT device allows for sensors, actuators, and control strategies to be connected and transfer data through the Internet. Furthermore, allowing for optimized energy usage of appliances and improving energy efficiency by having a centralized system that controls for instance heating. In the context of controlling heating systems, sensors are typically low-powered devices that detect temperature conditions.

In regards to recent advancements in digitization and IoT, implementing smart home devices has become less expensive and easier to install. Smart houses can benefit from temperature control through IoT devices that utilize AMS. A Norwegian electricity company called Tibber offers commercial IoT devices and services which allow the consumer to monitor and control their real-time energy consumption. Tibber pulse is designed to fit all new AMS meters in Norway that have a Home Area Network (HAN) port. [15]

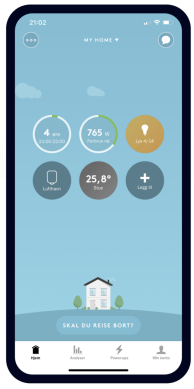


Figure 2.4: Tibber smartphone application [15].

Figure 2.5: Sensibo smartphone application [16].

In regards to heating systems, there are currently IoT services that allow control of heat pumps. These are often referred to as smart thermostats, such as Nest, Ecobee, and Honeywell Lyric that allow the consumer to control the temperature remotely [17]. For homes with installed air-to-air heat pumps, a company named Sensibo offers access to heat pump settings and control settings. In addition, the Sensibo device provides information about the measured humidity and temperature in the house and functions both as a sensor and a smart thermostat [16].

The information from commercial IoT services, such as Tibber and Sensibo, can be accessed through Application Programming Interfaces (APIs). They also offer to monitor and control the connected devices through their apps, shown in figure 2.4 and 2.5. The implementation of such devices is considered simple for the average user and can facilitate smart energy management. Tibber claims that by installing heat pumps that are compatible with Sensibo, it is possible to reduce both electricity costs and consumption through smarter heating, and in total reduce the power consumption by an average of 9.3% [18].

In addition to integrating IoT devices and software services, sophisticated control techniques can be implemented to control the smart appliances in the house to further improve the energy management in buildings. To further optimize the temperature control of heat pumps more advanced control strategies have to be considered.

2.4.2 Smart temperature control

Over the past decades, building heating controllers have seen significant advancements. Various articles suggest different control strategies, algorithms, and equipment to optimize the heating to enhance energy efficiency. The control methods vary from classic approaches such as thermostat controllers to modern predictive and optimization algorithms such as MPC. [19]

Today, thermostats are the most common type of temperature controller installed in residential buildings. The thermostats are still extensively used for regulating temperature in households because they are simple and cheap to install. However, thermostats are typically implemented without considering the thermal dynamics of the house and the user, which can result in inefficient regulation of the temperature.

A heating system's primary purpose is to provide thermal comfort for its occupants. However, optimal management of heating systems can be achieved where both thermal comfort and efficiency criteria are met. There are several studies related to improving the efficiency of Heating, Ventilation, and Air-Condition (HVAC) by implementing smart controller designs. In the following, a literature review has been conducted to evaluate the advantages and disadvantages of implementing more advanced control strategies for heating systems, with a focus on MPC schemes.

In [20, 21], reviews of several modeling techniques and control strategies are investigated for HVAC systems, where more advanced control strategies such as MPC are discussed in relation to the classic controllers such as thermostats and Proportional-Integral-Derivative (PID) controllers. Overall, the MPC outperforms other control strategies in terms of energy and cost savings while maintaining thermal comfort. The optimal control strategies rely on improving indoor comfort and reducing building energy consumption. However, the articles also highlight the challenges related to having the appropriate application of HVAC controller design, which is very dependent on establishing and good dynamic model of the building and heating system.

A relatively new study, [22], investigated an IoT-based architecture for MPC of HVAC systems in smart buildings. One of the interesting elements of this article is that it mentions approaches for integrating the user into the system by developing a user interface. This is one of the few reviewed literature that discusses this aspect in relation to MPC, where they consider the possibility of developing a device/dashboard dedicated to the user to allow monitoring of the environment and setting the control system mode. This highlights the importance of the user being included in the system in order to ease the adoption of such control strategies.

Other research projects, [23–25], have demonstrated that MPC can provide substantial cost savings, by including electricity prices and improve indoor comfort as compared to traditional control approaches. Smart temperature control can take advantage of the price signals in order to offer flexibility, as mentioned in 2.1.2. From the reviewed literature, implementing an MPC as the controller design for HVAC, both thermal comfort and electricity cost can be variables in the optimization and potentially contribute to active demand response and reduce the overall consumption. The benefits of smart temperature control such as MPC are illustrated in figure 2.6.

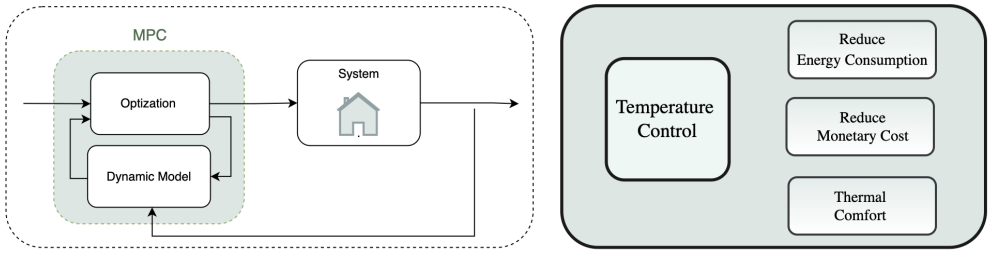


Figure 2.6: Potential benefits of implementing MPC into the controller design of a heating system.

However, the advancement in control strategies poses some challenges in relation to adoption. Most of the literature focus on buildings in general, however, the implementation of MPC in the average households can contribute to more challenges. In particular, this relates to the fact that these systems tend to require complex and expensive equipment which is hard to implement for

the average person. In addition, the implementation and understanding of such systems also require people with specialized knowledge. Furthermore, it is challenging to assess how the user can interact with higher complexity systems. Appropriate hardware and software are also considered important for developing compatible communication interfaces. Albeit MPC implementation traditionally has required an expensive and complicated set-up the smart house system in this project uses a combination of heat pumps, IoT, and MPC to optimize the temperature in the house, with the intent of having a simple and cost-effective implementation. This approach has proven to be a viable implementation in an average household. [1]

2.4.3 Smart house applications and software

The adoption of smart home solutions depends ultimately on how easily people can integrate them into their daily lives. The development of smart home applications such as web interfaces and smartphone apps has grown in popularity and can play a key role in facilitating the adoption of more complex controller designs, such as MPC.

However, currently, there are few studies or research projects that have investigated the possibility of developing a human-machine interface for MPC schemes. Albeit there are other approaches that relate to implement Supervisory Control and Data Acquisition (SCADA) systems, these systems are commonly used in process control, and power system operation and are currently gaining importance in smart building control. However, these are more applicable in commercial buildings than households. [1]

User perspectives

To solve the challenges with adoption, one must understand how users can comprehend and interact with more complex systems. The user perspective and experience is of paramount importance. In this project, the user interface is intended to work as a tool for the user to interact with the MPC scheme controlling heat pumps.

One of the primary barriers to adopting smart home technologies can be related to the lack of trust from the users in terms of the learning process, confidence in technology, and cost of the technology. In regards to MPC, the user might experience a limited perception when lacking an understanding of basic optimization concepts. Hence, providing an intuitive user interface can create a user-friendly experience without the need for in-depth knowledge. When the user does not understand the system they tend to fight it or try to control it in ways that are not ideal. Also taking into account that the user can also have a limited perception in terms of the usefulness and value of the technology. Another important factor is related to the security of IoT devices

and the collection of data. The user can have concerns about the leakage of sensitive personal information. [26]

Today, there are few people on average that have experience and understanding of modern optimal control methods, such as MPC. As such, the applications and services supporting MPC will rely on easy formulation and approaches that are understood by users in general. In this manner, the user can to a certain degree understands the system, by including simplified language, although the concept, in general, is complex.

With MPC there are concepts that can be communicated intuitively to the average person, further explained in section 3.4. These present some of the basic concepts behind control systems. Since most people are familiar with temperature control such as thermostats, regulating and setting reference temperature is considered easy to understand. However, understanding the concept of why the reference temperature is not reached or why the predictions change, is another problem.

According to the results from the article, [27], the reviews on opinions and perceptions of smart thermostats from users suggest that the users prioritize comfort over energy efficiency. The article also noted that users do not necessarily understand how the smart choices are made and interaction with the control can contribute to disturbing the potential savings.

However, developing user interfaces that can make the user aware of the benefits of maintaining a balance between comfort and energy savings/cost savings can be included. Thus, an increased level of technological literacy and better user interface design may reduce concerns and negative commentary about installation and usability.

Existing IoT applications and platforms

More and more people will integrate home automation solutions leveraging IoT into their homes, resulting in an increase in demand for smart house application development. A user interface can be developed as web applications, dashboards, or smartphone applications.

Some of the most commonly used smart home apps are Amazon Alexa, Apple home, and the Samsung Smart Homes app. The Samsung application called SmartThings can assist the user to regulate for instance the temperature remotely, as presented in figure 2.7. These smart home applications and services do not directly enable smart energy management in houses, rather they allow for simple remote control. The functionalities are typically limited to only adjusting temperature settings. However, by reviewing the state of the art in app development for smart homes it is possible to gain insight into different types of functionalities and user-interface elements that are important for achieving a good user experience.

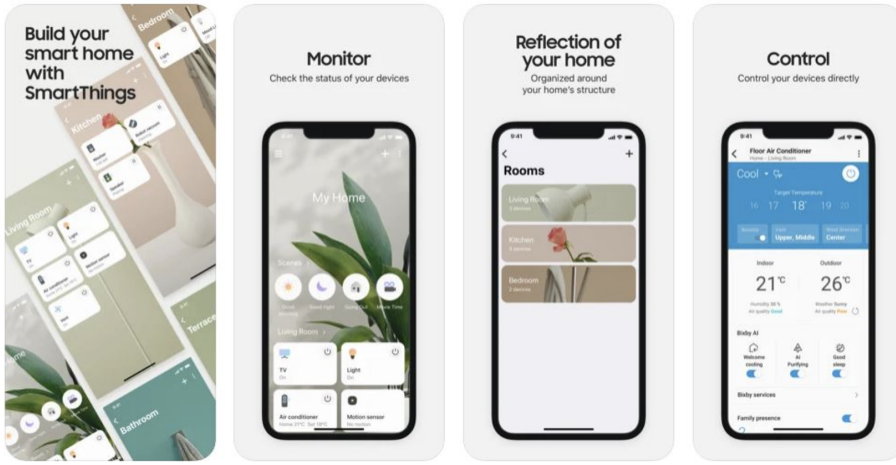


Figure 2.7: Smart home application (SmartThings) developed by Samsung [28].

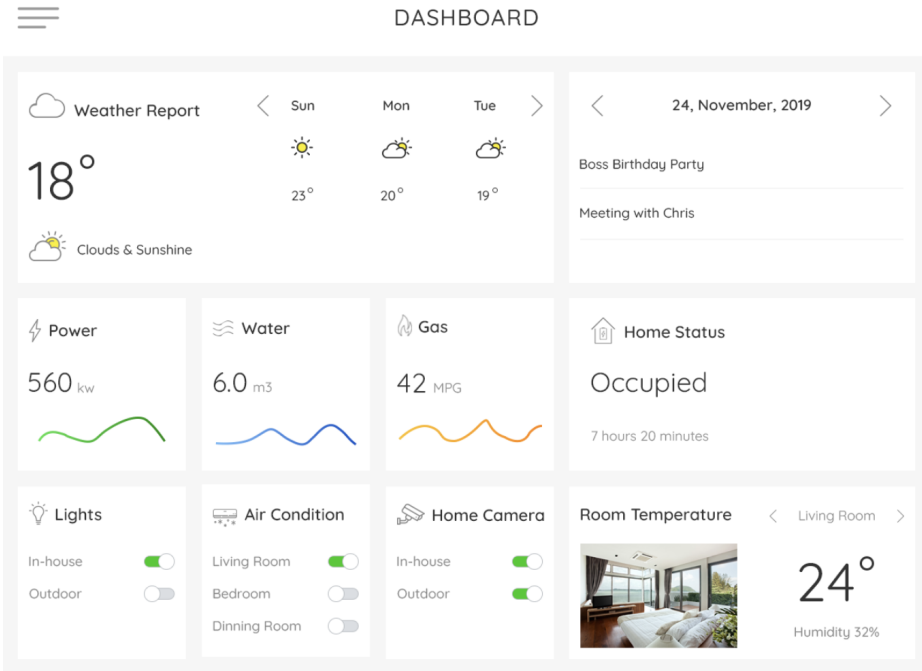


Figure 2.8: IoT smart home application dashboard [29].

Other alternatives to smartphone applications are dashboards and web applications. These user interfaces allow the user to a greater extent view of graphical displays due to typically bigger

screens. A dashboard/web application interface is presented in figure 2.8, which highlights the current room temperature and power consumption.

In addition to commercially available applications, there are platforms that allow people to create their own content, given that the platform supports the devices intended for the smart home. Currently, there are various IoT systems that can offer a platform for developing applications for smart homes. [30] These software services support the adoption of home automation by choosing a reliable IoT platform. Some of the most popular platforms are listed to the right. Such services typically provide generalized software to accommodate different sensors and actuators.

- Microsoft Azure IoT Suite
- Amazon Web Services
- Apple HomeKit
- Google Cloud IoT
- Android Things

Some of the existing platforms are to some degree constrained in terms of options. In addition, building a custom system on top of an existing platform can bring uncertainties such as updates and changes to the software. For more advanced algorithms such as MPC, the commercially available platforms and applications are not specifically designed for implementing such a system. As a result of this, a customized application is required in order to gain control of the system and support the MPC scheme.

Creating customized software

An alternative to utilizing the platforms is rather to utilize the APIs that various IoT services provide. These can further provide the necessary information to create specialized software for the smart house environment and the user can gain more control. Through APIs, it is possible to access information from the IoT devices and possibly develop a customized application for the smart house environment. Developing a custom IoT solution allows for originality and highly custom-built interfaces.By creating a custom solution, the compatibility, performance, and user experience can be increased. By developing a custom application designed to fit the environment, more functionalities, flexibility, and tailored features can be integrated. However, developing customized software for a smart house can be time-consuming and difficult. In the beginning, the application should be designed to manage only one device.

Since IoT enables smart home devices to be connected to the same ecosystem, the users can access the information retrieved from API and the data collection point through a user interface. This requires the data to be processed into a human-readable form, such as graphical displays.

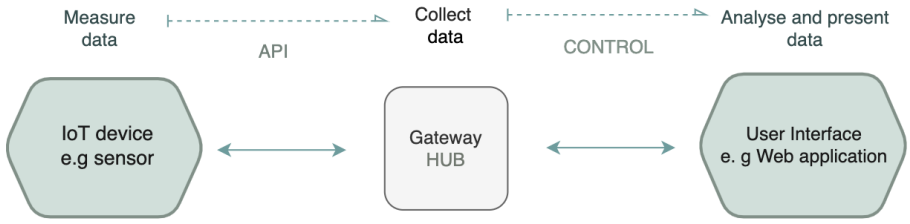


Figure 2.9: Flowchart of wireless data transfer from IoT devices to a user interface.

Home automation systems typically include a centralized entity that functions as the main controller. This main controller is commonly referred to as the hub and facilitates the process of data management by implementing control strategies to handle the data collected from the IoT devices transferring information through APIs. In addition, the output from the hub and the controller can be visualized through a user interface, as illustrated in figure 2.9. The data retrieved from the house hub does not directly provide any meaningful information to the user. However, this data can be directly used to control the appliance as well as transform the output into meaningful data through data processing and analyses. A control algorithm operating the hub performs actions according to the information received from the sensors and user. For instance, the hub can decide to change the temperature by increasing the power usage of the heat pumps. To further improve the user experience, a back-end server connected to a client app can be developed to organize and present the data received from the hub.

Chapter 3

Smart house system description

In this chapter, a general description of the smart house system is presented with a focus on the MPC scheme. The smart house is located in Trondheim and provides a real experimental setup. The first section 3.1 presents the components in the smart house. In section 3.2, information regarding the dynamic and physical model of the house is provided. Furthermore, section 3.3 describes the MPC scheme implemented in the smart house for controlling the heating system. Finally, section 3.4, introduces how the user can interact with the control algorithm.

3.1 Components

This section presents the components of the smart house system, including sensors, actuators, IoT devices, and API services. The components in this thesis are restricted to include the relevant components for the heating system and MPC. This entails heat pumps as the heating system, IoT devices such as Tibber and Sensibo, and API services for accessing data for forecasting. A complete overview of the system and components is provided in appendix A.

Heat pumps

The smart house as has four installed air-to-air heat pumps, where one heat pump in installed in each room. The four rooms are namely, *Main*, *Living*, *Livingdown*, and *Studio*. The heat pumps installed in the smart house include both an outdoor and indoor unit with a refrigerant as medium. Three indoor units main, livingdown, and studio, is connected to one outdoor unit, one indoor unit living, is connected to another outdoor unit, illustrated in figure 3.1.

The heat pump are responsible for generating heat, however, cooling mode cannot be triggered by the MPC code. The MPC scheme controls the heat pump settings, mainly referring to the target temperature, fan level and heating mode. The COP is assumed to be equal for all heat pumps and modeled as a function of the outdoor temperature. The individual heat pump power is not directly measured and need to be estimated in order for each room temperature to be controlled. The procedure of estimating the individual power is not included in this thesis.

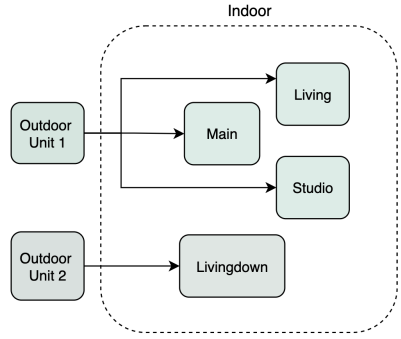


Figure 3.1: Illustration of the heat pumps system.

3.1.1 API services

The smart house system gathers data from several API services. These include the additional data received from online APIs such as weather forecasts and day-ahead spot prices. Various IoT devices are installed in the smart house, including 2 Tibber pulse devices, 4 Sensibo Sky devices, 3 Mill air sensors, and 3 IoT-enabled heating panels. In this thesis, the focus is aimed at the Tibber Pulse and Sensibo sky devices, which are related to the heating system in the house.

Norwegian Meteorological Institute's API service

In order to collect information about the outside temperature, data from the online Norwegian Metrological Institute (MET) is used. This API allows anyone to access weather information about the current weather, including forecasts and historical weather data. The MPC utilizes this API service to collect weather forecast data about the outdoor temperature.

Nord Pool's API service

In order to collect information about the day-ahead spot prices, data from the online Nord Pool API is used. The day-ahead spot prices are published around 12:00 AM. The MPC scheme utilizes the spot prices in the cost function in order to minimize the cost related to the energy consumption of the heat pumps.

Tibber Pulse

Tibber Pulse is an IoT device that provides information about power consumption through the AMS meters in the house. The power consumption is measured with two separate AMS, whereas one is dedicated to measuring the heat pump consumption, and the second measures other energy-

consuming devices in the house. The sampling rate for real-time power is two seconds and is accessed through the Tibber Pulse API.

Sensibo Sky

Sensibo Sky is an IoT device that provides information about the heat pump settings. The heat pump settings receive control actions from the MPC scheme. In addition, the Sensibo device works as a sensor providing information about the humidity and temperature. The sampling rate for updated measurements is 90 seconds, however, the temperature and heat pump settings from Sensibo Sky are measured at a 5 minutes interval and are accessed through the Sensibo Sky API.

Raspberry Pi and measurements

The Raspberry Pi installed in the house serves as the central hub and data collection point. A smart home hub is typically hardware that connects all smart devices and controls communication between them. This represents a relatively easy set-up where the aim is to have a system that is simple to install in average households, with the intent to optimize heating. The data collected from the heat pumps and online APIs are presented in table 3.1

Table 3.1: Data from APIs used in the MPC scheme.

Data	Accessed through
Heat pump states	Sensibo API
Weather forecast	MET API
Spot prices	Nordpool API

The smart house sensors provide measurements from APIs, presented in table 3.2. The control algorithm receives the data and measurements that allow it to control the heat pumps and perform a low-cost smart house modeling on the house.

Table 3.2: Measurements from APIs used in the MPC scheme.

Measurements	Measured by	Accessed through
Room temperature	Sensibo Sky	Sensibo API
Outside temperature	MET	MET API
Consumption	Tibber Pulse	Tibber API

3.2 Smart house model

In this section, an overview of the dynamic model is presented, nevertheless, a complete description is not provided due to the limitations of this thesis. A physical description of the system, including the states, inputs, and dynamics, is required to understand the MPC formulation.

In this project, System Identification (SYSID) is used to generate an approximate dynamic model of the temperatures in the house, where the parameters are generated based on measurements. A large amount of data has been collected on the house to perform an accurate identification of parameters. To solve numerical optimization problems a software tool, CasADi, has been used, which offers Non-Linear Programming (NLP) solvers.

3.2.1 Dynamic model

The smart control is based on a dynamic optimization, which takes into account the thermal inertia of the house. Generating a dynamic temperature model is important for the MPC scheme to work optimally and will also include estimating unknown parameters in the house. The unknown parameters are generated by SYSID, the current states are estimated and further supplied to the MPC scheme. The temperature dynamics are based on the heat transfer inside the house. Heat transfer encompasses the losses and gains of heat within a room environment and can be classified in three ways; conduction, convection, and radiation. A basic illustration of the modeling problem is presented in figure 3.2.

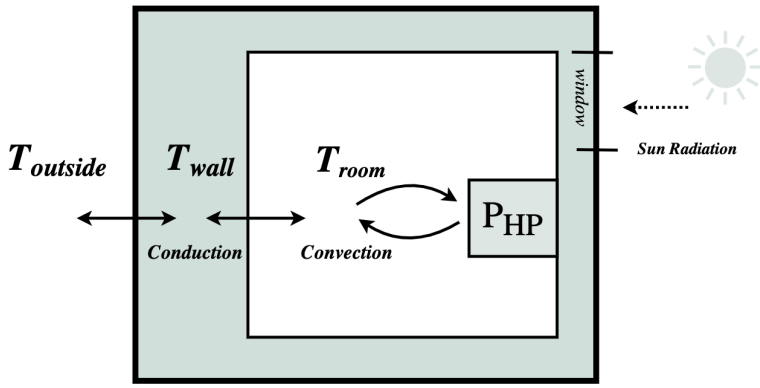


Figure 3.2: Simplified illustration of the modeling problem of a room environment, where T_{outside}, T_{wall}, and T_{room}, represents temperatures and P_{HP} is the heating power from the heat pump. The arrows indicate the flow of energy.

In buildings, the temperature difference between the inside and outside is important to consider. Due to the fact that heat transfer happens from hot areas to cold areas, the thermal comfort in the building might be affected by the conduction through the walls, as a result of temperature differences. Air movement causes heat transfer by convection in buildings as a result of air moving between different temperature zones. While radiation is typically induced by the sun. [31]

Establishing a thermal dynamic model of a building is necessary to estimate the energy consumption under different operating conditions. Other factors that need to be considered are windows, external heating, sun radiation, outside temperature, etc. The house model parameters include the thermal inertia and the losses of the rooms.

A simplified calculation to describe the heat transfer is demonstrated in equation 3.1. This equation models the heat transfer by conduction through the walls for a single volume in the house.

$$\begin{aligned} T_{\mathrm{wall,k+1}} &= \rho_{\mathrm{out}}(T_{\mathrm{out,k}} - T_{\mathrm{wall,k}}) + \rho_{\mathrm{room}}(T_{\mathrm{room,k}} - T_{\mathrm{wall,k}}) \\ T_{\mathrm{room,k+1}} &= \rho_{\mathrm{room}}(T_{\mathrm{wall,k}} - T_{\mathrm{room,k}}) + \rho_{\mathrm{dir}}(T_{\mathrm{out,k}} - T_{\mathrm{room,k}}) + P_{\mathrm{HP}}\mathrm{COP} \end{aligned} \tag{3.1}$$

Where $T_{\mathrm{wall,k+1}}$ and $T_{\mathrm{room,k+1}}$ represent the temperature of the wall and room, respectively, for the next iteration, in discrete time, k. The temperature inside the room, is influenced by the wall temperature and heat generated from the heat pumps P_{HP}, while the wall temperature, is influenced by the difference in the inside and outside temperature, T_{out}. The heat preserved by the wall is lost to the air around the wall and direct heat loss from the windows. The variables, ρ_{out} ρ_{room} and ρ_{dir} describe the unknown parameters estimated by SYSID which represents the gain and losses. In other words, these are the coefficients of heat transfer of the temperature differences. The effect of heat transfer by radiation is difficult to model, however for this model the radiation is considered as a positive disturbance in the system for simplicity.

3.2.2 Moving Horizon Estimation

For dynamic processes, estimating the current states can be more reliable than the direct measurements. The MHE uses the history of previous measurements to estimate the current states in the system based on solving an optimization problem. This optimization approach can work for nonlinear and constrained systems. The MHE cost function attempts to minimize the difference between the calculated trajectory and the household measurements. Contrary to the MPC, the MHE uses a backward time horizon. In this project, the MHE will estimate the room temperature, wall temperature, and heat pump power. At each iteration, the current states are estimated in

order to update the dynamic model, where the MHE is used to clean up the raw data and estimate the unmeasured states. The unknown variables are wall temperature, convection, and power correction. To collect the data, the MHE requests data from the Raspberry Pi interface. The currently estimated states are then supplied to the MPC. [14]

3.2.3 Physical description

In addition to defining the dynamic model of the smart house, the physical description of the MHE and MPC includes states and input as decision variables to generate the control input sequence to the heating system. The states are defined as discrete points in time for the entire prediction horizon, N, of 48 hours.

The MHE is responsible for estimating the states: T_{room}, T_{wall}, Conv, and P_{Corr}. Where P_{Corr} refers to the power correction on the estimated heat pump power and Conv refers to convection through the walls describing the gains or losses of heat in the rooms, which are not explained by the model and added to the room dynamic. These states are directly fed into the MPC scheme. In the optimization, additional states for controlling the heat pump settings include the target temperature, T_{target}, and fan level, Fan, in the MPC scheme.

Both states and inputs are included as constraints and cost terms in the optimization problem in order to control the dynamics further detailed in section 3.3. Variations in the ΔT_{target} and Δ Fan are the parameters used to steer the heating system, also referred to as the input. Further, the desired temperature settings, including the reference temperature, T_{ref}, and minimum temperature, T_{min}, are included to account for thermal comfort. Finally, two slack variables are introduced as $S_{\text{discomfort}}$, S_{min}, respectively in order to relax the MPC scheme, further explained in equation 3.7. All decision variables are listed in table 3.3.

Table 3.3: Physical description of MPC

	Notation	MPC decision variables
x	States	T_{room}, T_{wall}, Conv, P_{Corr}, T_{target}, Fan
u	Inputs	ΔT_{target} , Δ Fan
a	Actuator	HP
m	Measurements	HP settings, T_{room}
r	References	T_{ref}, T_{min}
s	Slack variables	S_{min}, $S_{\text{discomfort}}$

3.3 MPC description/implementation

This section describes the MPC algorithm that controls the heat pumps in the smart house, including the tuning factor, constraints, the cost function and a description of each cost term.

The MPC scheme utilize the estimated states and additional information from weather forecasts and day-ahead spot prices to predict and optimize the future temperature trajectories of the house. This information includes indoor and outdoor temperature, power consumption, and day-ahead spot prices, which is available from the Raspberry Pi, MHE and APIs. Furthermore, a minimum temperature and a reference temperature are specified by the user.

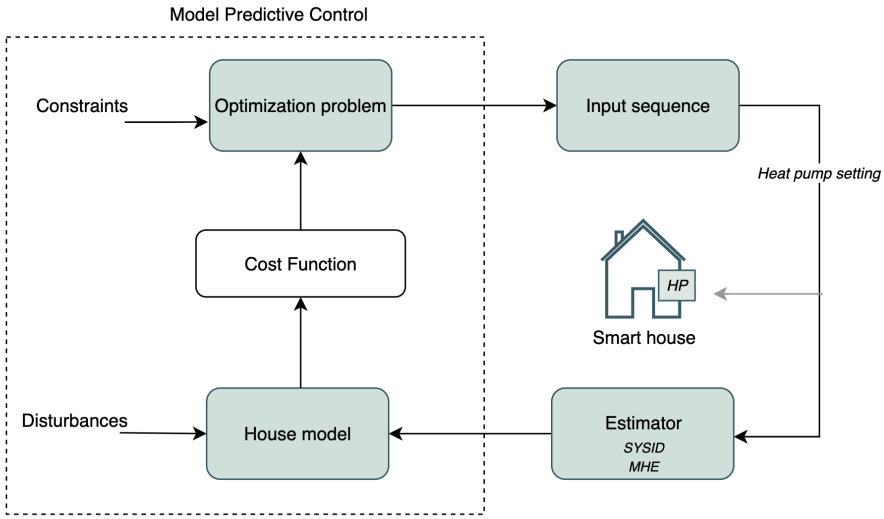


Figure 3.3: Schematic illustration of the MPC control in the smart house.

A simplified illustration of the closed-loop operation of the MPC is presented in figure 3.3. As mentioned in the section 3.2.2, the states are estimated with MHE based on the current measurements to provide the MPC with feedback for a closed-loop operation. Since the direct measurements are not used in the optimization, the feedback is referred to as output feedback. In essence, the MPC scheme is responsible for computing optimized settings for the heat pumps, where the optimal temperature trajectory serves as input to the system. The main objective of the MPC scheme is to perform optimal control of the temperature in the smart house in relation to the thermal comfort of the user and the spot prices. This is achieved by seeking to compromise between reduced monetary costs and increased comfort.

Commonly optimization problems have the main objective of driving the predicted output to the reference. In this case, the reference is the desired temperature of the user, however, the optimal control takes into account additional information in order to achieve the compromise between cost and comfort. Accordingly, the MPC scheme attempts to optimize a given performance objective rather than penalizing the distance from a reference.

3.3.1 Tuning factors

There are some parameters in the MPC scheme that have a considerable influence on the overall performance of the system and are required to be tuned such as the prediction horizon, the sampling interval, and weights. The MPC scheme complexity can increase when the prediction horizon increases or the sampling interval is shorter. The weights are responsible for giving certain preferences to cost terms in the cost function. Essentially, the weight determines how much the cost term should be penalized, further described in 3.3.3. All factors are tuned prior to writing this thesis and are therefore not extensively detailed.

- **MPC Horizon (N)**: The MPC scheme solves a finite horizon open-loop optimal control problem with a prediction horizon of 48 hours. The prediction horizon is denoted N and is typically selected in order to prevent discrepancy between open-loop and closed-loop profiles. The horizon is relatively long horizons which increases the computational speed, and subsequently increases the uncertainties in predictions.
- **Sampling interval (k)**: The MPC scheme has a 5 minute sampling interval corresponding to the measurements in the smart house. During the sampling intervals, the control actions stay constant. In total, the MPC computes 576 steps, k, during the prediction horizon.

3.3.2 MPC constraints

The MPC is exposed to a set of constraints, which include the constraints on the dynamic model. In the MPC scheme, there is a difference between hard constraints and soft constraints. Hard constraints refer to the constraints that require to be satisfied, typically referring to the model dynamics. While soft constraints can be violated by introducing slack variables which are included in the cost function as upper and lower bounds, typically referring to comfort violations.

The equality constraints for the temperature dynamics are equal for the MHE and the MPC. The dynamic temperature constraints are presented in equation 3.2, describing the next iteration for the wall temperature and the room temperature in discrete time.

$$\text{Dynamic Temp Constraint} \begin{cases} T_{\text{room,k+1}} & = T_{\text{room,k+1}} + \text{Conv}_{\text{k}} \\ T_{\text{wall,k+1}} & = T_{\text{wall,k+1}} \end{cases} \tag{3.2}$$

In order to control the dynamics, the inputs to the system (variation in T_{target}, Fan) and convection is included as equality constraints in the MPC. The dynamic input constraints are presented in equation 3.3. Where $T_{\text{target,k+1}}$, $\text{Fan}_{\text{k+1}}$, and $\text{Conv}_{\text{k+1}}$ describe the next iteration in discrete time. Here the convection is weighted by w_{conv} equal to 0.97.

$$\text{Dynamic Input Constraint} \begin{cases} T_{\text{target,k+1}} & = T_{\text{target,k}} + \Delta T_{\text{target,k}} \\ \text{Fan}_{\text{k+1}} & = \text{Fan}_{\text{k}} + \Delta \text{Fan}_{k} \\ \text{Conv}_{\text{k+1}} & = w_{\text{conv}} \cdot \text{Conv}_{\text{k}} \end{cases} \tag{3.3}$$

Heat pumps

Physical limitations for the total amount of power the heat pumps can draw are included as constraints in the MPC scheme. The rooms main, livingdown, studio, and living are denoted, M, LD, S, and L, respectively. Since three indoor units are connected to a single outdoor unit, the total power consumption for all three is limited to 3 kWh, denoted $P_{\text{M,LD,S}}$, while for the other one the limit is 2 kWh, denoted P_{L}. These enter the optimization as inequality constraints, shown in the equation 3.4. The power usage of each heat pump is not directly measured and has to be estimated, however, this is not detailed in this thesis.

$$\text{HP Power Constraint} \begin{cases} P_L & \leq 2 \quad [kW] \\ P_{M,LD,S} & \leq 3 \quad [kW] \end{cases} \tag{3.4}$$

Limit 24 hour electricity cost

The MPC scheme includes a constraint that applies to the monetary cost related to the total heat pump consumption. This constraint is implemented in order to restrict the electricity cost within 24 hours. The restriction is defined as a maximum value that the MPC scheme aims to keep the costs beneath. The calculation of the 24 hour costs is described in equation 3.5.

$$C_{24\mathrm{h}} = \frac{24}{N} \sum_{k=0}^{N-1} C_{\mathrm{spot,k}} + C_{\mathrm{grid}} \cdot P_{\mathrm{HP}} \frac{5}{60/100} \tag{3.5}$$

Where $C_{24\mathrm{h}}$ is the 24 hours cost predicted by the MPC over the prediction horizon. The spot prices denoted C_{spot}, are hourly values from Nord Pool accessed through their API. The VAT is included in the spot prices. The C_{grid} refers to the grid rent the private consumer pays to the grid companies, which is set to a 44 NOK/kWh according to the current pricing policy in Trondheim. The total spot price is further multiplied by the heat pump consumption, denoted P_{HP}. The scaling factor $\frac{24}{N} = 0.5$ aims at computing the average cost per 24h over the MPC horizon. The sampling interval of 5 minutes is converted to an hourly basis by dividing 60 to determine the kWh. The spot price conversion from øre/kWh to NOK/kWh requires dividing by 100.

$$C_{24\mathrm{h}} - \mathrm{L_{NOK}} \leq 0 \tag{3.6}$$

The inequality in equation 3.6 represents the difference between the calculated 24 hour cost and the defined limit, $\mathrm{L_{NOK}}$. This implies that the total predicted cost has to remain under this specified limit for the inequality to be respected.

Slack variables

In this MPC scheme, two slack variables are introduced, $S_{\mathrm{discomfort}}$ and S_{min} in equation 3.7. These slack variables are introduced to represent the violation of additional algebraic constraints and are responsible for compensating when the temperatures are below the reference temperature, T_{ref} and the minimum temperature, T_{min}. Occasionally, it may not be possible to meet the temperature demand. Therefore, the MPC problem is relaxed by these slack variables. The lower bounds are equal to (-inf) and the upper bounds are equal to 0.

$$\text{Slack Variables} \begin{cases} T_{\mathrm{ref,k}} - T_{\mathrm{room,k}} - S_{\mathrm{discomfort,k}} & \leqslant 0 \\ T_{\mathrm{min,k}} - T_{\mathrm{room,k}} - S_{\mathrm{min,k}} & \leqslant 0 \end{cases} \tag{3.7}$$

If the inequality is satisfied, hence the room temperature is higher or equal to the reference temperature, the slack variable is zero. However, when the room temperature is lower than the reference temperature the slack variable is responsible for maintaining the inequality. The same applies to the minimum temperature. The minimum temperature, T_{min}, refers to the temperature the MPC aims strictly aims to not heat under.

3.3.3 Cost function

Given the optimization problem, the resulting objective function to minimize is described by equation 3.8. The decision variables in this optimization are states, inputs, and disturbances that influence the house dynamic. This MPC scheme optimizes essentially for achieving a compromise between thermal comfort and electricity cost related to heat pump consumption.

$$\begin{aligned} J = \min \sum_{k=0}^{\mathrm{N}-1} & w_1 C_{\mathrm{cost,k}} + w_2 C^2_{\mathrm{temp.above,k}} + w_3 C^2_{\mathrm{discomfort,k}} + w_4 C^2_{\mathrm{min,k}} + C_{\mathrm{input,k}} \\ \text{s.t.} \quad & (3.2),\ (3.3),\ (3.4),\ (3.6),\ (3.7) \end{aligned} \tag{3.8}$$

In this thesis, the cost function is referred to as a multi-objective optimization, where there is more than one target included. A compromise has to be obtained in order for the cost function to be minimized while satisfying the constraints. The cost terms, $C_{\mathrm{discomfort,k}}$ and $C_{\mathrm{min,k}}$ are described by the slack variables in equation 3.7. The cost terms $C_{\mathrm{cost,k}}$, $C_{\mathrm{temp.above,k}}$, and $C_{\mathrm{input,k}}$ are described further below. Where k is the number of sampling intervals during the time horizon N equal to 48 hours. The weights denoted w_{i}, where $i = 1...8$, will define how the states and inputs are prioritized in the optimization problem. Considering larger weights on the state errors, the deviation is penalized more. The compromise is obtained by determining the weights for each cost term in order to provide priorities. The weights corresponding to the different cost terms are listed in table 3.4 and 3.5.

Table 3.4: The MPC weights for equation 3.8.

Weight	Name	Value
w_1	SpotGain	0.1
w_2	TempAbove	0.005
w_3	TempBelow	0.2
w_4	MinTemp	50

Table 3.5: The MPC weights for equation 3.13.

Weight	Name	Value
w_5	Δ Target	0.5
w_6	HUB	0.5
w_7	Δ Fan	1
w_8	Δ Temp	0

The economic aspect of the MPC scheme takes into account the spot prices in order to optimize the cost related to the consumption of the heat pumps, described in equation 3.9. Albeit the prediction horizon is set to 48 hours, the day-ahead spot prices are only available for the next 24 hours. In order to solve this problem, the spot prices after 24 hours are assumed to be constant for the remaining horizon and equal to the last available spot price.

$$C_{\mathrm{cost,k}} = (C_{\mathrm{spot,k}} + C_{\mathrm{grid}} - C_{\mathrm{base_i}})P_{\mathrm{HP}} \tag{3.9}$$

.

The BasePrice denoted $C_{\mathrm{base_i}}$ (where $i = 1, 2$), is introduced to the cost function in order for the MPC scheme to account for variations in the spot prices and optimize according to these variations by subtracting (either the minimum or average) from the spot price at time instant *k*. Accounting for variation refers to subtracting a baseline and only penalizing cost in relation to this base. The BasePrice is calculated as a scalar based on the historical data (day-before), day-ahead spot prices, and the grid rent. Including both historical and future spot prices provides a larger set of data and the variations are to a greater extent accounted for. There are three ways of calculating the BasePrice, depending on the selected BasePrice value. If the value is zero the BasePrice is not considered in the optimization. If the value is one, the minimum spot price is subtracted, described in equation 3.10 and if the value is two, the average is subtracted, described in 3.11. Where t is the time instant 24 hours back in time, implying that the summation is summing up the past 24 hours and the future 24 hours.

$$C_{\mathrm{base_1}} = \min \sum_{t=1}^{48} C_{\mathrm{spot,t}} + C_{\mathrm{grid}} \tag{3.10}$$

$$C_{\mathrm{base_2}} = \mathrm{mean} \sum_{t=1}^{48} C_{\mathrm{spot,t}} + C_{\mathrm{grid}} \tag{3.11}$$

The cost term $C_{\mathrm{temp.above}}$, evaluates the difference between the reference temperature and the measured temperature in the room, described in equation 3.12. The corresponding weight is set relatively low since the MPC should not penalize the deviation too much when the temperature is above the reference. This is done in order to allow the MPC to store heat in the thermal mass of the house.

$$C_{\mathrm{temp.above,k}} = T_{\mathrm{ref,k}} - T_{\mathrm{room,k}} \tag{3.12}$$

The cost term C_{input} refers mainly to cost terms penalizing the input parameters in the optimization, presented in equation 3.13. Albeit they are not the focus of this thesis, they are included to provide a complete description of the system.

$$C_{\text{input,k}} = w_5 C_{\Delta T_{\text{target}},k} + w_7 C_{\Delta \text{Fan}^2,\text{k}} + w_8 C^2_{\Delta T,\text{k}} \tag{3.13}$$

The target temperature and fan level serve as input to the system, where the first control step is sent to the heat pump settings. Both cost terms are designed to add cost when the ΔT_{target} and ΔFan are high in order to restrict the input to the heat pumps and limit large variations.

The first cost term, $C_{\Delta T_{\text{target}}}$, will restrict the target temperature to drastically change all the time, making sure the MPC scheme outputs a more consistent target temperature to the heat pumps, as described in equation 3.14 (also referred to as the Hubber penalty function).

$$C_{\Delta T_{\text{target}},k} = w_6^2 \quad \sqrt{1 + (\frac{\Delta T_{\text{target}}}{w_6})^2} - 1 \tag{3.14}$$

The w_6 refers to constant value of 0.5. In a way, this penalty function will promote larger deviations if zero is not possible to maintain. This will cause the target temperature to increase more when this is considered necessary.

The second cost term, $C_{\Delta \text{Fan}}$ will prompt the fan level to be as negative as possible and serve as input to the heat pumps, described in equation 3.15.

$$C_{\Delta \text{Fan,k}} = \Delta \text{Fan} \tag{3.15}$$

Lastly, the cost term $C_{\Delta \text{T}}$ is added in order to limit very high temperatures and prompt the difference between the target temperature and the room temperature to be as negative as possible, described in equation 3.16. However, the weight corresponding to this cost term is currently zero, thus, not included.

$$C_{\Delta T,\text{k}} = T_{\text{target}} - T_{\text{room}} \tag{3.16}$$

3.4 MPC interaction

In this section, user interaction is introduced to the MPC scheme. In particular, the aim is to demonstrate the elements the user can change to influence the optimization problem in the user interface, described in table 3.6.

Table 3.6: MPC interactive elements.

Element	Type	Interaction
Weight	SpotGain	Select priority
BasePrice	1 or 2	Select calculation
Constraint	Cost24h	Select limit of 24 h monetary cost
Temperature	Reference	Select reference
	Minimum	Select minimum

3.4.1 Temperature settings

In order for the user to adjust the room temperature, the first parameter that needs to be considered is the reference temperature. This temperature is decided by the user in terms of the ideal level of comfort and the room temperature is measured to adjust the difference. This difference can vary significantly under certain circumstances depending on the spot prices. Although, the MPC scheme attempts to reach the desired temperature other factors are influencing the system. This is the difference between the MPC scheme attempting to solely penalize the distance from a reference, rather it attempts to directly optimize a given performance objective. By allowing the user to decide the reference temperature and minimum temperature, the user can influence the optimization.

3.4.2 Limit 24 hour predicted cost

As mentioned in section 3.3.2, a constraint is implemented to restrict the electricity cost within 24 hours, where the calculation is defined by the spot prices and a defined value. This defined value can be changed by the user and will be the only constraint the user can interact with. Initially, this limit was set to 80 NOK, however, this can be changed by the user to lower the cost further. This limit can give the user the opportunity to have the MPC operating under controlled conditions.

3.4.3 Spot price weight

Taking into account the proper weights of the cost functions, the system is assumed to be optimized. The weights are responsible for giving a certain preference to the cost terms, such that the cost term is penalized in the overall cost function. Although the MPC scheme has several weighting elements and costs terms, only the relevant weights for the user interaction are implemented in the web application, namely the SpotGain weight, w_{spot}.

All weights in the cost function influence the temperature, however, tuning w_{spot} will scale the linear cost term related to the spot market and the user can directly choose how important cost savings are. Adjusting the value of the weights will directly influence the behavior of the optimization, however, selecting the priority of the w_{spot} is an intuitive approach for the user to regulate the temperature and have better control over the system. To some extent the user will need to understand the influence of changing the weights. Thus, the term weights, are transformed into more understandable concepts such as priority, saving money, and increasing comfort, further detailed in section 6.4.1.

3.4.4 BasePrice

As presented in equation 3.9, the BasePrice is subtracted from the current spot price, and is calculated depending on the minimum or average. By allowing the user to select the BasePrice calculation the user can decide how to heat in reference to a base. Accordingly, the the spot prices enter the cost function relative to the minimum or average. This user interaction is considered an intuitive approach to working with the spot market since it provides an opportunity to work directly with the spot market variations to regulate the temperature. This will allow the MPC scheme to a greater extent adjust the temperature in relation to the variations and achieve a higher thermal comfort overall.

Chapter 4

Software tools

This chapter describes the software tools used to develop a full-stack web application for monitoring and controlling the heating system in the smart house. A full-stack application refers to having a dedicated back-end and front-end. The first section, 4.1, introduce the web application development. The following sections 4.2 and 4.3 presents the software technologies and methods used for developing the server and client-side of the web application, respectively.

4.1 Web application

Smart home applications should provide the necessary and relevant information, with functionalities customized to the environment and the user, in order for the user to experience convenience. In particular, understanding advanced control strategies such as MPC will require a framework that is simple, flexible and customizable.

As mentioned in section 2.4.3, the available platforms are not able to offer the necessary functionalities for the case study presented in this thesis. Accordingly, a customized software is developed to provide a human-machine interface (HMI) for the MPC scheme. The application is developed to support the IoT devices in the smart house for controlling the heat pumps, for monitoring and controlling the temperature. More specifically, it is necessary for the user to understand how the heat pumps are controlled by the MPC scheme. Other IoT devices can be connected to the same application, however, this is beyond the scope of this thesis.

In this thesis the server-side and client-side is developed using JavaScript (JS), which is the most popular programming language for developing dynamic and interactive web application. A web

application is a software or program which is accessed through a web browser. The reason for choosing to develop a web application is due to its simplicity and ability to be easily integrated with the existing system. This approach allow for local storage between the control algorithm and the server, which makes the data processing uncomplicated.

Considering the fact that the system is running locally on a computer, the web application is not required to be deployed for universal access. As such, the web application is accessed through localhost which is the a host address of the local computer. As such, the front-end is accessed through *http://localhost:3000* which is the web interface, and the back-end is accessed through *http://localhost:3001* which is the server. Other alternatives is not considered further in this report.

4.2 Server-side ExpressJS

The back end is developed using ExpressJS, which is responsible for the server-side of the application. Express is a web application framework written in JS and is the most common framework used for developing fast Node.js applications. Node.js is an open-source server environment with different frameworks and tools to simplify the development of building back end services like APIs and web applications.

ExpressJS is a framework built on top of Node.js and is one of the most popular HyperText Transfer Protocol (HTTP) server libraries and is commonly used to develop Representational State Transfer (REST) API. This provides a communication tool for web services where data can be represented as JavaScript Object Notation (JSON). A REST API works similar to a regular API, the only difference is that REST API interacts and correspond via an HTTP protocol. [32]

4.2.1 HTTP request

HTTP is the communication interface between the server and client-side, that allows requests and responds to data. The server is essentially responsible for sending and receiving data through these requests. Moreover, this communication is also used for communication between the server and the MPC algorithm, further explained in section 5.1.

Different resources can be requested by the client, and the server returns a response to these requests. In other words, the HTTP method is a request/response protocol. There are various HTTP methods that can allow information to flow between the client and the server, where the most common are GET and POST. A **GET** request returns data in order to access it from a specified resource on the server, demonstrated in code 4.1. Requesting data through GET from APIs is standard for both JS and Python. A **POST** request will send data to the server to update a

specific resource, demonstrated in code 4.2. The user can input the specified data and the server responds to this request. Sending requests over HTTP is most commonly accomplished by using the POST method. The data needs to be parsed into a format that is easily accessible on the server-side through a middle-ware called the body-parser package before it can be easily accessed. In the request object, the body property allows you to access the data parsed from the raw HTTP request.

```
1 app.get('/route', (request, respond)
      => {
2     console.log(function())
3     respond.json(function())
4 })
```

Code 4.1: HTTP methods for GET.

```
1 app.post('/route', (request, respond)
      => {
2     console.log(request.body)
3     let req = request.body
4 })
```

Code 4.2: HTTP methods for POST.

Routes

In order to retrieve data from the server, different routes are created to access the specified resource. The web application can receive a request to the specified route or endpoint with HTTP methods. The particular resource is accessed by an endpoint that performs an action on the Uniform Resource Locator (URL) path to the endpoint. The resources contain JSON data and a route is a way to locate a resource.

Creating different routes provides a better structure of the data on the server and the data is easy to handle when user requests are prompted. Eventually, more routes can be created to accommodate more functionalities in the web application or other IoT devices. The different routes created between the server and the client-side are presented in table 4.1, while the routes from the server back to the MPC scheme are listed in section 5.2.

Table 4.1: Routes for accessing data from the MPC scheme on the server.

Data	**Accessed through**
MPC data	$http://localhost:3001/MPC$
API data	$http://localhost:3001/API$

4.2.2 File watcher Chokidar

The server environment uses a file watching system called Chokidar to trigger a new GET request when new data is added or a data file is changed in the folder containing the data from the MPC scheme, demonstrated in code 4.3. Chokidar is a minimal and efficient file watching library for Node.js servers. This library is responsible for watching folders and files and reacting to updates or changes. Chokidar can monitor the folder for file additions or updates inside JSON files and use the path created to receive the data.

```
const chokidar = require('chokidar')
const watcher_MPCcomp = chokidar.watch('./FolderName/MPC-data', {})
watcher_MPC-data.on('add', (path) => {
  app.get('/MPC', (req, res) => {
    const newMPCcomp = require('../' + path)
    return res.json(MPC-data-file)
  })
})
```

Code 4.3: Chokidar with GET request.

The folder 'MPC-data' contains all relevant data retrieved from the MPC scheme such as weights, temperature, heat pump settings, consumption, etc. Depending on how the data is updated, the newest data is locally stored in this folder and the updates are further requested with a fetch in the front end. When the MPC scheme stores a new solution, the content within that file is updated. [33]

4.3 Client-side React

The front-end is developed with React which is responsible for building the user interface (UI) that allows the user to interact with the MPC scheme. React is a popular and simple component-based JS library and is widely used for building UIs. React is both efficient and flexible, which is advantageous in terms of handling dynamic data and a potential up-scaling of the application. [34]

In order to create the web interface, HTML (HyperText Markup Language) and CSS (Cascading Style Sheets) are used, which are generally the standard markup and styling languages for creating web applications. While HTML creates the main structure of the web page, CSS is used to format and design the layout. There are several CSS frameworks that can be integrated in order to make the styling considerably easier. In this project, Tailwind CSS is used, which is a modern and highly customizable framework.

4.3.1 Connecting ExpressJS and React

In order to develop a full-stack application, the Express back-end is required to be connected to the React front-end. Two important methods are used to achieve this; Fetch and HTTP requests. The Fetch API provides a JS interface for responses and requests between the server and client-side. Since fetch allows to make GET and POST requests on the client-side, it is able to communicate and retrieve data directly from the server ('http://localhost:3001/') demonstrated in code 4.5.

```
1
2  useEffect(() => {
3  fetch('http://localhost:3001/')
4    .then((res) => {
5      return res.json()
6    })
7    .then((json) => {
8      setData(json)
9    })
10   .catch((error) => {
11     console.log(error)
12   })
13 }, []
14 )
```

Code 4.4: Fetch the server data in React.

```
1
2  fetch('http://localhost:3001/', {
3    method: 'POST',
4    body: JSON.stringify(data),
5    headers: { 'Content-Type': '
         application/json' },
6  })
7    .then((res) => {
8      return res.json()
9    })
10   .then((json) => console.log(json))
11   .catch((error) => {
12     console.log(error)
13 })
```

Code 4.5: Post request from React to server.

The Fetch method can access resources on the server where the response is requested as a stream of data, typically stored as JSON. Moreover, the Fetch API becomes part of the HTTP pipeline by using GET and POST methods to request and send resources on the server. [35]

Moreover, React has introduced a new way of handling states referred to as React hooks. The two React hooks used extensively in the application are; useState and useEffect. With useState, the React state is added to a function component, where the state can be updated based on a setState call inside a function.

With useEffect the state will perform a re-render or update in relation to the stated dependencies. If the dependency is an empty array, the useEffect will only update during the first render, while having a state dependency will cause a re-render every time the state is updated. The useEffect is commonly used to re-render depending on a state with for instance a Fetch method, described in code 4.4. By having a fetch inside the useEffect, the state can be updated with the data from the resource, where the resource is typically the server. [36]

4.3.2 Charting library Recharts

In this thesis, a JS charting library called Recharts is used as the data visualizing tool, due to its intelligent, effective and neat graphical presentation. This library is built on React components and can present a variety of different charts as SVG elements. The web application utilizes three specific charts: LineCharts, AreaCharts, and ComposedfCharts. [37]

The web application is responsible for visualizing the MPC data intuitively. With large data sets, graphical displays are considered the optimal choice also taking into consideration that the data is updated in real-time. Accordingly, implementing a live chart allows for monitoring the real-time system, where the predicted trajectories can be displayed. This will also allow the data to be structured and customized, such as adding areas in the graph for highlighting a specific period or coordinates in the graph.

Limitations to charting library

There are some limitations to the Recharts library in terms of adding special features and customization. The charts are coded inside a Recharts component, which decides the type of charts that is rendered. In addition to these containers, other elements can be added, such as reference lines and reference areas. The reference elements can give some level of customization to the graphs, however, this can only reference one particular x-value or/and y-value or an area between coordinates. Thus, there is a limited amount of customization tools that are offered. For the particular system in this thesis, the library lacked some features needed to fulfill the customization in the graphical displays. Firstly, the data structure has to be in a specific format in order for the charts to read the values. This was a restriction since it required the server to handle the data in a non-generalized way. Secondly, highlighting parts of the graph is considered very helpful for the user. However, this customization has restrictions since it only allows to specify certain coordinates. Lastly, there are no additional hover effects that can highlight and explain certain periods in the graph.

Developing customized solutions that are not based on an existing library is time-consuming and difficult. Creating your own customized features inside the library is complex since the library is built on top of D3.js and React [38]. A possibility is to explore other charting library such as Charts.JS. Another alternative is to build customized SVG charts in React from scratch.

Chapter 5

Software implementation and data processing

In this chapter, an overview of the software implementation for the web application is presented, along with current adjustments to the MPC algorithm. In section 5.1, the data flow is explained, in which the data extracted from the control algorithm is processed. The following section, 5.2, presents the implementation of user interaction in the MPC scheme. Furthermore, section 5.3, presents the server framework for the REST API and section 5.4 presents the main components of the client-side development. Finally, section 5.5 demonstrates a quick guide for installing and using the web application on a local computer.

5.1 Data pipeline

The data pipeline represents the data flow from the collection point to the web interface, which includes the control algorithm, the server and the client-side framework. The smart house algorithm, which is dynamic system, is developed using Python. The data retrieved from measurements and house control algorithm must be processed by both the server and client-side in order to present the relevant data to the user. Since the MPC data is real-time, the pipeline requires a continuous flow and updating of data. The data pipeline is illustrated in figure 5.1.

The house data is collected by the Raspberry Pi and fed into the MHE and MPC scheme. The output data from the MPC scheme is acquired and stored in dictionaries and serialized in order to compress the data memory with Pickle, which is a Python built-in module. Due to the fact

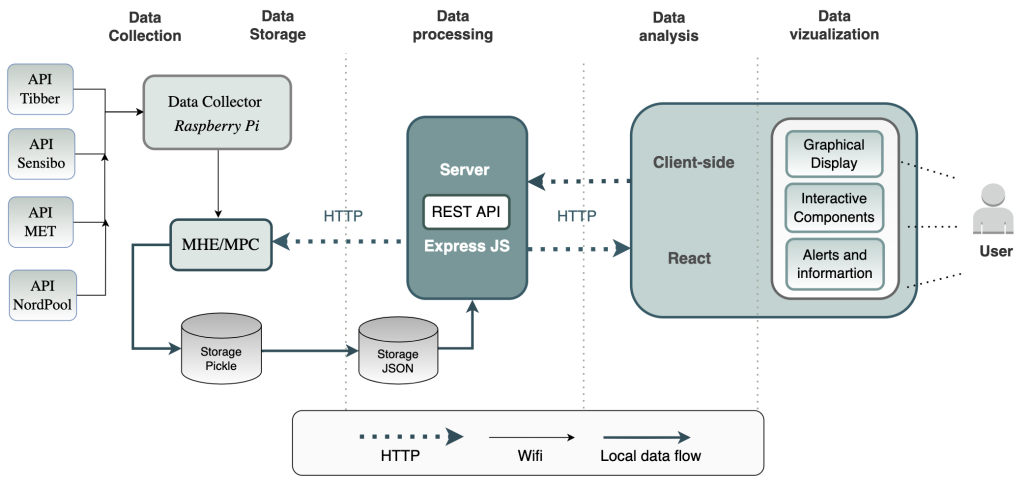


Figure 5.1: A basic illustration of the data flow between the data collection point and control algorithm to the server and client-side. The arrows represent the direction of the data and are explained at the bottom.

that the web application is developed using JS, the dictionaries are converted into JSON objects in order to read and utilize the data. Another approach can be to directly store the dictionaries as JSON objects in order to have a standardized and language-independent storage format that is compatible with both Python and JS.

The MPC receives the user request from the web interface, through the server as JSON objects where these are loaded into the Python script as dictionaries. Accordingly, the MPC algorithm is only receiving information from the REST API server, while the server and client-side receive and respond to HTTP requests between them. The data structure on the server is presented in appendix B. This allows the server to further process and transform the data into objects such that the client-side can perform data analysis to visualize the live data in the graphical and interactive components.

Currently, data is stored locally every 5 minutes according to the sampling time, under a folder named MPC-data, so there is no database included in the pipeline. A database can be convenient for persistent storage, however, in this thesis, this is not further investigated. The server is developed as a REST API in order to have easy access from both the MPC algorithm and the client-side. All communication between the different elements in the pipeline is enabled by HTTP methods as mentioned in section 4.2.

5.2 MPC comparison

Among the most important aspects of MPC, is the ability to predict future outputs. The predictions provide valuable information about how the heating is optimized. Moreover, the user can observe the anticipated heating and electricity cost through the web application and decide to change the optimization on occasions where the user is not satisfied with the performance, further detailed in section 6.2. However, the user might be oblivious to the effects of changing the optimization. Therefore, a comparison can be presented to inform the user of how the expected temperature and cost trajectories change according to the request made by the user.

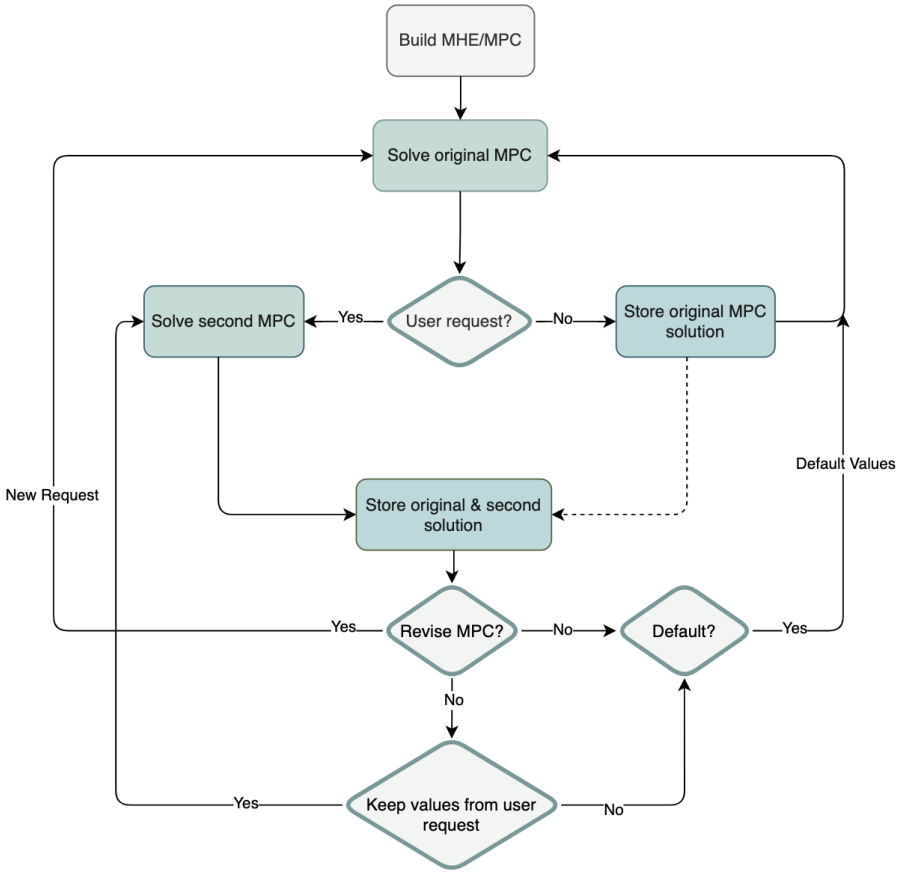


Figure 5.2: A simplified flowchart illustrating how the user requests are implemented in the MPC scheme to present a comparison.

In figure 5.2, a flowchart of how the comparison can be implemented in the house control algorithm is illustrated. This demonstrates how the MPC scheme reacts and responds to user interaction. The MPC and MHE are only required to be built once, which is advantageous in regard to computational time.

With the purpose of demonstrating a comparison of optimized outputs, the MPC scheme is run twice sequentially with different parameters. A sequential implementation is considered reasonable since the first solution can be used as the initial guess for the second MPC scheme. The original MPC scheme is made prior to this thesis and is considered 'optimal' with tuned values. The comparison is based on calculating a second MPC scheme based on user-specified values and presents the difference between the two solutions. Throughout the report, the comparison use the terms **original MPC scheme** and **second MPC scheme** to differ between the values in the web application.

At first, the user is presented with the original MPC scheme which allows the user to evaluate the performance of the optimal control. If the user decides to request a change in the optimization (e.g change weights or reference temperature), the second MPC scheme is executed with the values specified by the user. Further, the second solution is stored in order to present a comparison of the temperature and cost trajectories to the user. The comparison is initially presented as a potential output from the MPC, however, this is not implemented. After the comparison is presented, the user can decide to implement actions. Thus, three options are presented in order to implement the desired action to the MPC scheme after being presented with the comparison. A further description of the UI components responsible for handling the comparison in the web application is provided in section 6.1.4.

Table 5.1: Routes for accessing user requests in the MPC scheme.

User request	**Accessed through**	**Contains**
Request	$http://localhost:3001/Request$	Comparison (Boolean)
	$http://localhost:3001/ReviseMPC$	Final Submit (Boolean)
Values	$http://localhost:3001/TempSetting$	Reference temperature
	$http://localhost:3001/MPCWeights$	SpotGain and Cost24h
	$http://localhost:3001/ExtWeights$	BasePrice

The user requests have different routes and are loaded into the MPC scheme through HTTP methods. The response will create a python dictionary from the JSON format and extract the necessary information. Routes for posting data from web application through the server and finally to MPC scheme are provided in table 5.1.

5.3 REST API server

The server consists of different libraries and functions that allow data to be accessed through the web interface. Since the data is stored locally under the Express folder, it can be directly extracted from the files and further processed through functions that transform the data into a readable JSON object for the graphical interface. Eventually, the processed data is made available on the REST API routes by HTTP requests. A flowchart demonstrating the structure of the Express server is visualized in figure 5.3.

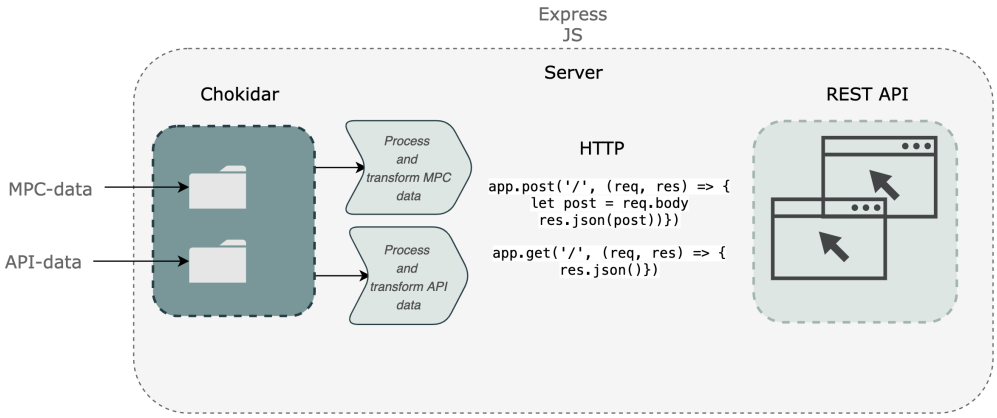


Figure 5.3: Flowchart describing the server developed in ExpressJS as a REST API.

5.3.1 Data handling

In general, there are two alternatives to establishing a 2-way communication between the MPC and the web application.

- The server receives information locally and directly from the Python script and the MPC receives information from the user through the REST API using HTTP requests.
- The server receives information about the MPC from the Raspberry Pi and the MPC receives information from the Raspberry Pi through the REST API using HTTP requests.

The first alternative requires the whole system to run on the local computer including the smart control algorithm and the web application. The second alternative focus on utilizing the Raspberry Pi to communicate the information between the MPC and the user, which can allow for remote handling of data. The current solution utilize the first approach for simplicity of operating a local data storage system.

Another element to consider is how often the server receives new information and updates from the MPC scheme.

- Update the data and predictions only when a user request is submitted.
- Update the data and predictions at every sampling instant corresponding to 5 minutes.

The first approach requires fewer updates due to the fact that the predictions will stay the same unless the user requests new data. The second approach is more accurate than the first in the sense that the MPC prediction do actually change at every sampling instant due to the MPC revising the plan. The user will, nevertheless, experience more frequent updates in data, and the predictions will change regardless of submitting new requests, which can cause confusion for the user. The current solution utilize the second approach, due to ease the data structure on the server-side of the application.

5.4 Web interface structure

All the UI elements belong to a parent container/component which makes up the main structure of the web interface. The simplified version of the application design is illustrated in 5.4. This represents the main components and provides an introduction to the functionalities of the web application. The main components are **Navigation bar**, **Sidebar**, **Filecontainer** and **File**.

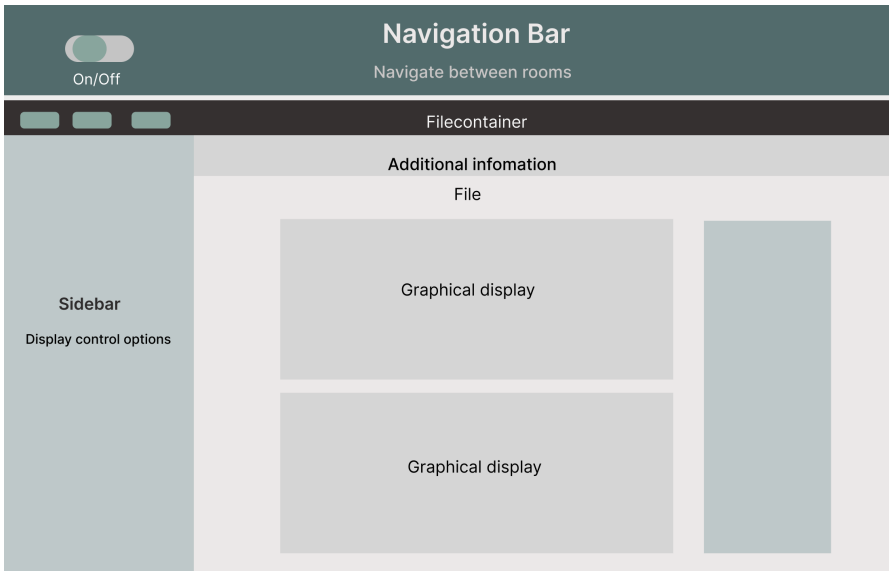


Figure 5.4: Overview of main components in the web interface.

The navigation bar is responsible for arranging the data displayed in the filecontainer, such as selecting the room environment. The sidebar is responsible for presenting different ways the user can interact with the optimization (control options) in the MPC scheme. Finally, the filecontainer and file is responsible for presenting the graphical displays and other visualizing tools.

The data displayed in each component is fetched from the REST API, processed, and rendered to the interface with interactive functionalities. The data extracted from the server is processed in a downstream fashion through the different components, illustrated in figure 5.5.

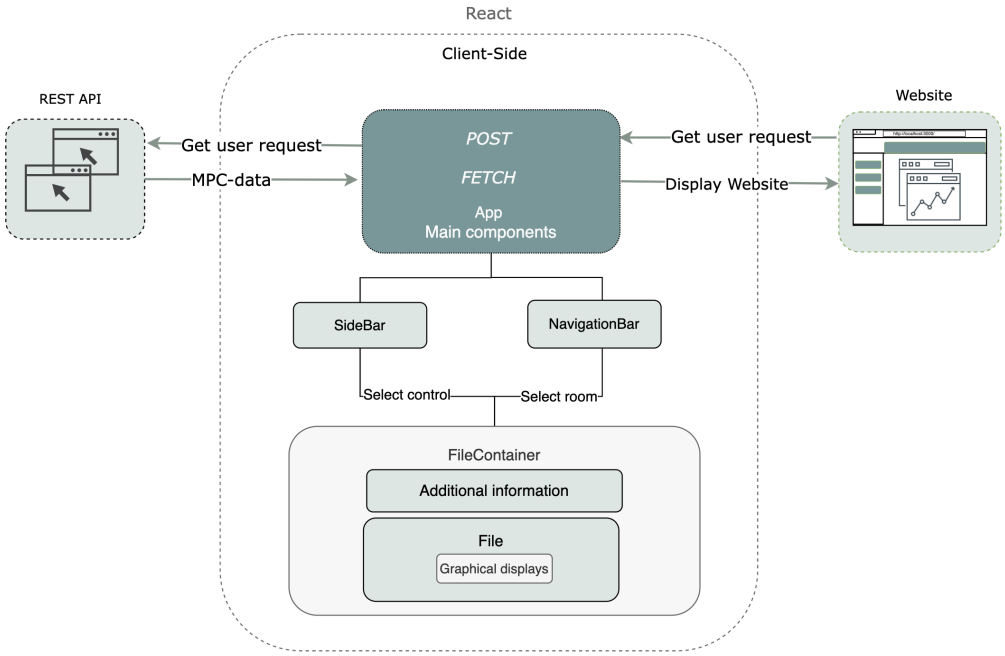


Figure 5.5: A basic illustration of the client-side framework presenting the main components and communication between REST API and web interface.

5.5 Quick installation guide

In this section, a quick installation guide is provided with the intent of demonstrating the application to the reader and/or to a person developing the application further. The application is developed specifically for the case study in Trondheim and you will need the entire software on your computer to run a complete simulation. There are no requirements to run the entire software for the smart house system (the software for running the house control e.g MPC scheme) if you only want to check out the web application. However, there will be no dynamic data or content displayed in the web application that is related to the smart house. A quick demonstration of the web application is provided in this **link**.

5.5.1 Installing and running the software

You will need to have the software running on your local computer in order to test it. The project folder is uploaded to Github and in order to clone the repository from Github, an account is required. The repository is named **POWIOT-WebApp** and contains all relevant information about the web application developed with JS. The front-end folder is named **app-React** and the back-end folder is named **app-Express**. A more detailed explanation of software installation is provided below.

ExpressJS and React

In order to run the web application, Node.js has to be installed. Node.js is a run-time environment that includes everything you need to execute a program written in JavaScript. In a web browser, navigate to https://nodejs.org/en/download/ and click the Installer button to download the latest version.

For JS applications, in general, a package.json file is included under the application folder. This file is responsible for containing all the dependencies needed to execute and run the code. All developer dependencies are not required for a regular user (only required for further development). By default, **npm install** will install all modules listed as dependencies in package.json. The list below provides the necessary steps to run the server Express JS and the frontend React in the browser on your computer. The procedure has to be executed once for the server and once for the client-side which are located under different folders.

- Open the terminal in the preferred code editor (e.g Visual Studio Code).
- Navigate to the application folder (app-React/app-Express) which should at least contain a /src folder and a package.json file.

- Write **npm install** in the command line interface, this will install all dependencies.
- To run the application write **npm run dev** in the command line interface.
- Now the front-end should be available on http://localhost:3000/ and the server (REST API) is available on http://localhost:3001

5.5.2 Python (Optional)

In addition to the web application folder, the house control algorithm is provided under the folder named **POWIOT copy**, which is developed using Python. If the reader wants to test the web application with real-time data from the smart house, they need to navigate to a file inside the House-Control folder, named **HouseControl.py**, and install all dependencies.

Connecting MPC to server

All software implementations described in this thesis need to run on a local computer. In order to access the MPC data, the JSON files are stored directly on the Express server. Hence, the *path* is required to be replaced with where the application folder is located on your local computer: "/Users/.../.../app-Express/MPC-data/" to access the MPC data (lines 3486 and 3505) and "/Users/.../.../app-Express/API/" to access the API data (line 2917).

Chapter 6

Results

This chapter presents a demonstration of the UI components in the web application and provides explanations relative to the control aspect these components are responsible for communicating to the user. In figure 6.1 an overview of the human-MPC interface before user interaction is introduced is illustrated. The results focus on the perspective of the user, presenting the challenges related to communicating the relevant information from the MPC scheme. In section 6.1, the dynamic arrangement of data is demonstrated. Further, in section 6.2, the graphical displays for the predictions are provided. In section 6.3, the detection mechanism for displaying certain MPC behavior is explained. Finally, the human-MPC interaction is presented in section 6.4.

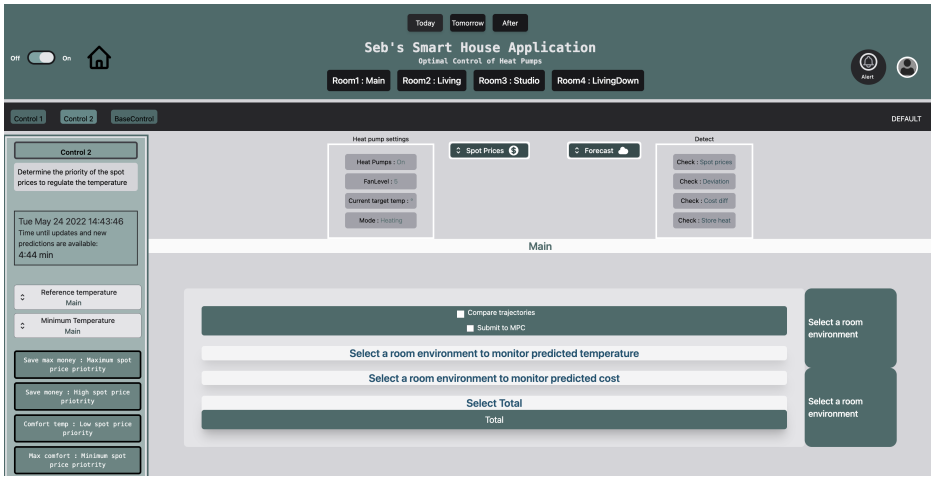


Figure 6.1: Overview of the interactive user interface components in the web application.

6.1 Dynamic arrangement of data

The output from the MPC contains large amounts of data and the components presented in this section are responsible for dynamically arranging this data. Considering the user, these components will allow the data to be organized intuitively when navigating the web application. The main challenge relates to how to present different content based on user interaction.

6.1.1 Navigation bar

Taking into account that the smart house consists of four rooms with an individual heat pump controlling the temperature, the application should provide an environment for each room. Accordingly, the data is arranged into different objects for each room, allowing the user to filter the data by selecting the room in the navigation bar, illustrated in figure 6.2. The corresponding object contains all relevant information for that particular room, such that the environment can be monitored and controlled by the user independently.

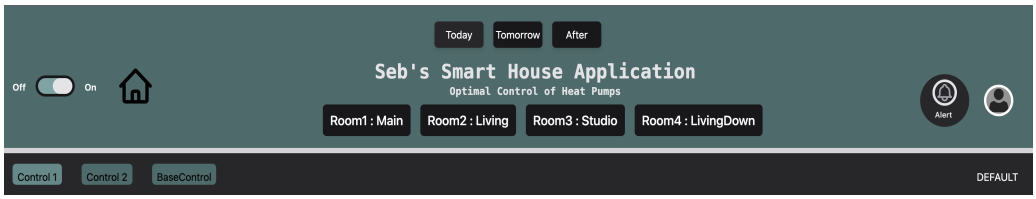


Figure 6.2: A navigation bar component that allows for selecting a room environment and day to view the prediction from MPC.

The navigation bar is a responsive header placed at the top of the page providing an overview of the system. Four buttons are implemented for room selection; Room1:Main, Room2:Living, Room3:Studio, and Room4:LivingDown. By making the navigation bar interactive, the user is able to have four different environments to manage the heating in the smart house. In general, the different environments will have different settings depending on the preference of the user. Typically, the living room is warmer than the basement, hence the thermal comfort criteria vary between the rooms.

Another element in the navigation bar allows the user to select between days, showing different predictions. Since the MPC provides predicted trajectories for the next 48 hours, the user is instantly presented with a lot of data. In order to arrange and structure the data, the user can request to separate the predictions by three buttons; Today, Tomorrow, and After. In this manner, the data is structured intuitively and provides relevant information depending on the preference of the user.

6.1.2 Show different graphs in the same display

The graphical displays can visualize predictions of different types of data such as reference temperature, room temperature, target temperature, electricity cost, and comparisons. In order to sort the data and avoid presenting all graphs in the same plot at all times, the user can choose to include or not include certain elements in the graph. Thus, the graphical displays have buttons in the left corner in every graph with different options for displaying data.

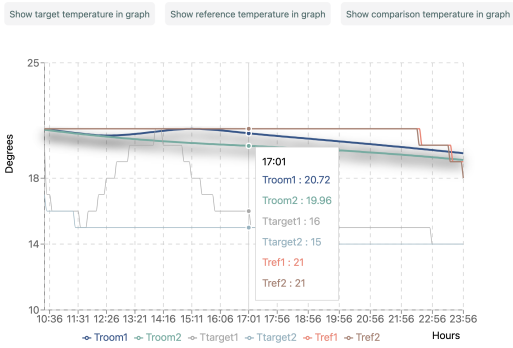


Figure 6.3: Three options in the left corner for displaying different temperature related graphs in the same component.

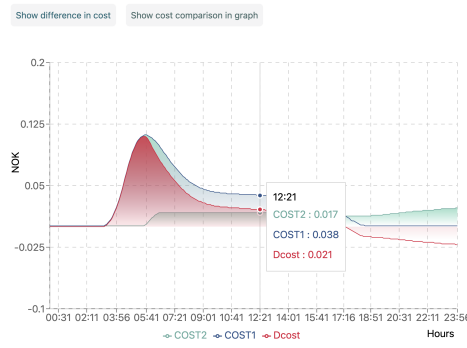


Figure 6.4: Two options in the left corner for displaying different cost related graphs in the same component.

In figure 6.3, all available data is displayed in the same graphical temperature component. Troom1 and Troom2 describe the predicted temperature between the original MPC and the second MPC, respectively. Ttarget1 and Ttarget2 describe the target temperature input to the heat pump, while Tref1 and Tref2 describe the reference temperature.

In figure 6.4, the predicted cost is displayed, where COST1 and COST2 describe the predicted cost between the original MPC scheme and the second MPC, respectively. The Dcost explicitly displays the difference in cost between COST1 and COST2.

6.1.3 Additional information

This component is responsible for making the additional information accessible to the user. This includes four main components: Heat pump settings, Spot prices, Weather forecasts, and Detection mechanism, presented in figure 6.5. The heat pump settings inform the user of the current mode, fan level, and target temperature. The other components are further detailed in section 6.2.2 and 6.3.

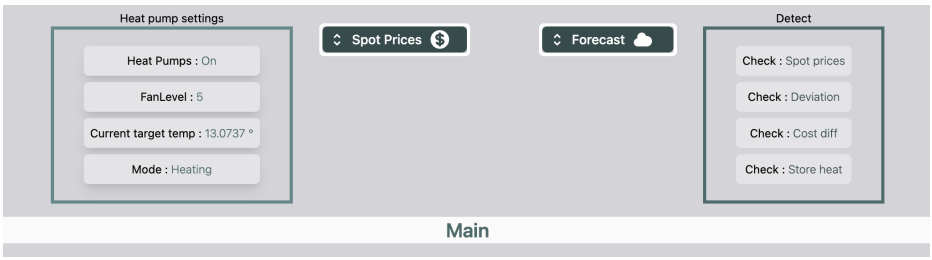


Figure 6.5: Component for displaying additional information.

6.1.4 Submit user requests

The web application is responsible for handling requests from users. Considering that the user wants to change the optimization, a request must be submitted through the web interface. The application includes several functionalities which provide the user with different components to interact with, where some are responsible for sending requests to the server. The submit buttons are essential for the data flow between the MPC scheme, server, and client-side. The server continuously updates data according to the user such that the MPC scheme can receive this information and act accordingly. The submit buttons correspond to the elements listed previously in table 5.1.

Revise the MPC

After the user has submitted the first user request for comparison, the user is able to make an educated decision on how to proceed forward with the control of the heat pumps. This decision will determine how the MPC will progress. The user is prompted with a pop-up for revising the MPC scheme, illustrated in figure 6.6. In this component, three options are presented; Return to default, Revise the MPC and Keep the values. The first option will disregard the changes made during the comparison, the second option allows the user to be presented with a new comparison and the third option will implement the changes made during the comparison.

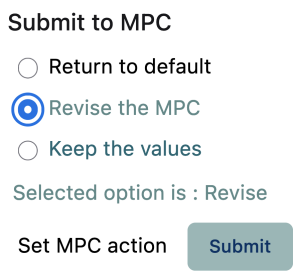


Figure 6.6: Component handling submit to MPC to implement action.

6.2 Graphical display

The graphical displays are one of the central components of the web application since they are responsible for visualizing the predictions from the MPC scheme. Two main graphical displays are developed to facilitate monitoring of the anticipated heating in each of the room environments. In regards to the user, there are two elements of importance; the comfort related to the temperature of the house and the monetary cost related to the heat pump consumption. Since these are occasionally conflicting elements in the optimization, the user has to decide what to prioritize; comfort, monetary cost, or a reasonable trade-off between them. Thus, visualizing the prediction of the temperature in the room and the related electricity cost is considered one of the essential elements in the web application.

The graphical components are presented in this section to demonstrate the visualization of MPC data. Throughout the results, tests are performed to demonstrate and validate the performance of the web application. All tests are performed on the *Main* room. Due to the fact that the MPC scheme is a real-time system the tests will vary from day to day, thus, the graphical displays presented will differ depending on the day the test was performed.

6.2.1 Graphical MPC comparison

The MPC scheme will receive updated information about the smart house and predict the trajectories by revising the plan at every time instant. As mentioned in section 5.2, the MPC will in addition receive user requests from the web application and possibly update the optimization problem relative to these requests. In order to provide information about how these requests ultimately affect the optimization and output of the MPC scheme, a comparison is visualized. This allows the user to evaluate if the performance has improved in terms of thermal comfort or cost.

Temperature

The comparison between the temperature trajectories informs the user of the anticipated temperature in the house. The temperature is decided by the user in terms of the ideal level of comfort and the room temperature is measured to adjust the difference. This difference can vary significantly under certain circumstances. Although, the MPC scheme tries to reach the desired temperature other factors are influencing the system. By visualizing the predicted room temperature in relation to the reference temperature the user can observe where deviations occur. In addition, the comparison can visualize whether the new trajectories perform better in terms of the preference of the user.

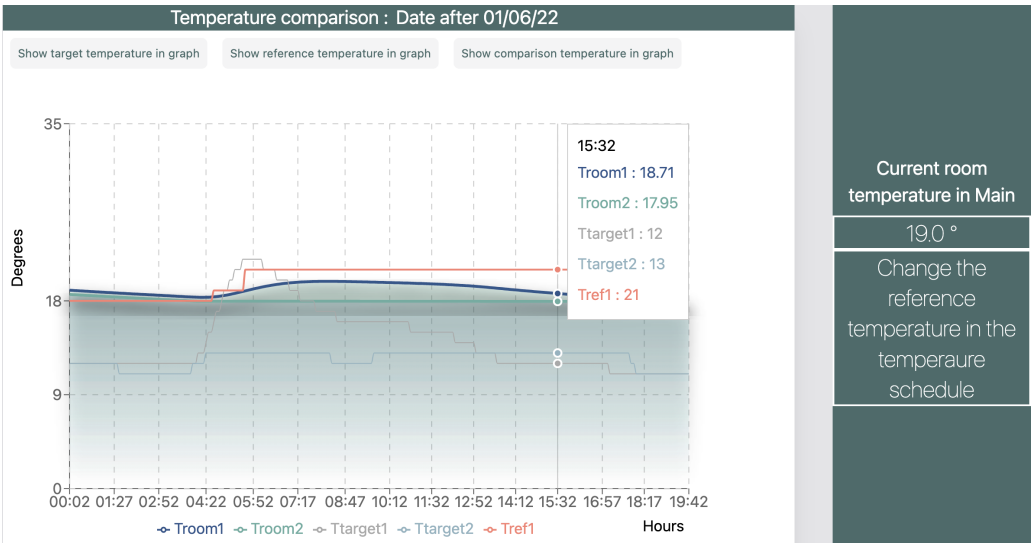


Figure 6.7: Graphical display of the predicted temperatures trajectories for comparison.

In figure 6.7 the graphical component for the temperature trajectories is presented. The comparison consists of Troom1 (blue) describing the predicted room temperature from the original MPC scheme and Troom2 (green) describing the predicted room temperature after submitting a user request. Additional information is displayed for the reference temperature and target temperature. On the right side, the current room temperature is displayed in order for the user to get direct information about the room temperature.

Electricity cost

The monetary cost is a result of the consumption related to the heat pumps and the spot prices. This is considered an essential element to visualize to the user. However, consumption and spot prices alone are not necessarily interesting parameters for the user. Thus, the graphical component presents the resulting electricity cost of multiplying these factors at every time instant. By visualizing a cost comparison, the user can become aware of the economic benefits of optimal temperature control. Furthermore, making the user aware of the cost related to for instance increasing the temperature.

In figure 6.8 the graphical component for the electricity cost trajectories is presented. The comparison consist of COST1 (blue) describing the predicted costs from the original MPC scheme and COST2 (green) describing the predicted costs after submitting a user request. In addition, Dcost (red) is plotted to highlight the difference in cost between the compared values.

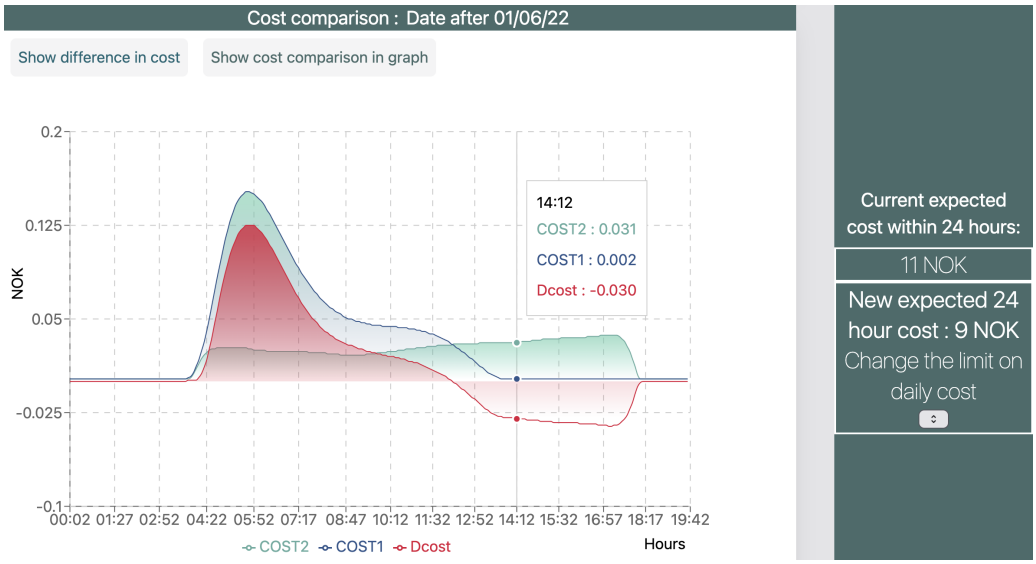


Figure 6.8: Graphical display of the predicted cost trajectories for comparison.

Furthermore, the current predicted 24 hour costs is displayed in a component besides the graph, which is the only MPC constraint the user can interact with. The predicted 24 hour cost, introduce a daily limit the user can specify to keep the electricity costs under. Another helpful element the expected 24 hour electricity cost can provide is how the comparison is affecting the overall cost. A new predicted 24 hour cost is presented when a new request is submitted for comparison.

The default value is 80 NOK, however, the MPC is rarely exceeding a daily cost of 50 NOK. If the user decides to submit a considerably low value for the constraint, the MPC may behave poorly to maintain this constraint/limit. This can result in the temperature being strictly lowered. Accordingly, the lowest value the user can submit is 20 NOK.

Currently, there are two options for presenting the cost; a 5 minute basis and an hourly basis. Displaying the cost every 5 minutes results in practically insignificant values. For the purpose of visualizing the cost in a more realistic manner, the hourly cost is calculated and presented in figure 6.9. Since the spot prices are hourly based, which means that the spot prices remain the same within one hour it more intuitive to present the hourly cost, however both approaches are included in the application.

Considering the MPC scheme is operating on a 5 minutes basis, the consumption is converted into an hourly basis by summing up the consumption. The number of samples per hour is $\frac{60}{5} = 12$, hence the hourly basis will consist of the sum of 12 samples. Normally, the graphs are displayed

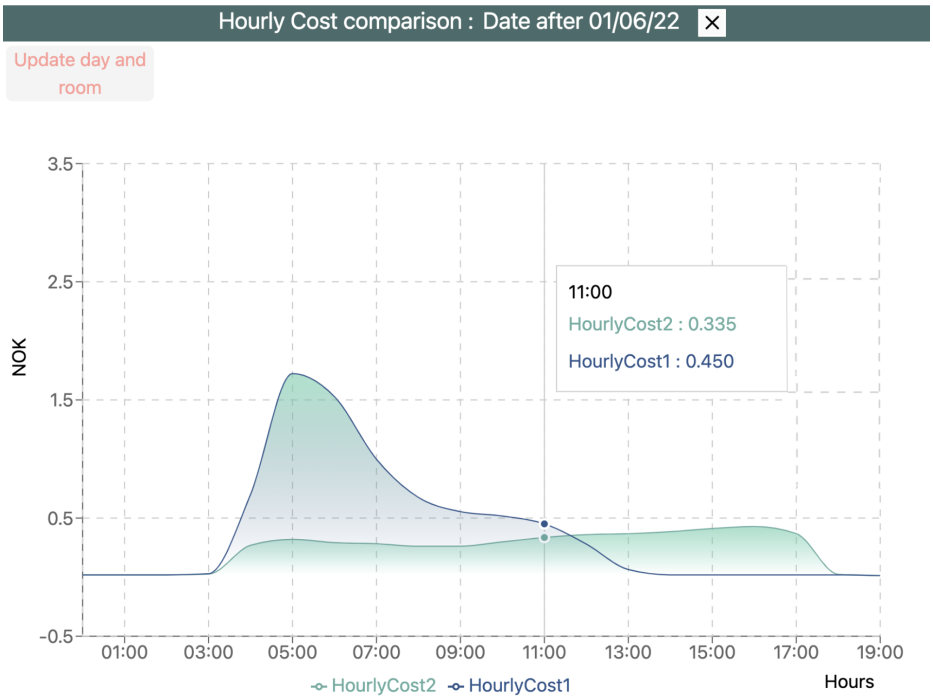


Figure 6.9: Graphical display of the predicted cost trajectories for comparison.

by selecting the day and room environment, however, for the hourly cost a different data structure is used. Therefore, the component is build differently than the other graphs and requires an additional function to be triggered in order to display the hourly values.

In addition to displaying the monetary cost for each room, the user might find it interesting to monitor the total electricity cost related to the four heat pumps, illustrated in figure 6.10. A comparison is provided for the total electricity costs as well. However, there are no options to view the total depending on the day. Instead the total cost is displayed for the entire prediction horizon. This will provide the user with an overview of the total costs related to the predicted heating in the house which can be beneficial in order to gain perspective of how the MPC scheme behaves. However, similarly to displaying the hourly cost the total electricity cost is triggered by an additional function.

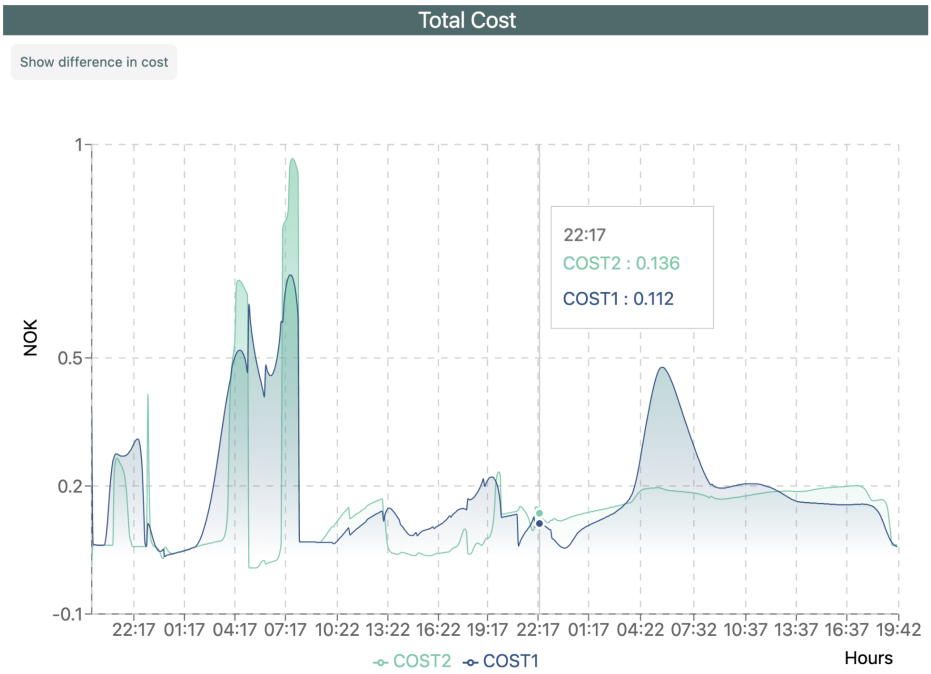


Figure 6.10: Graphical display for the total cost over the whole prediction horizon.

Change in predictions

The ability to predict future trajectories is one of the essential benefits provided by the MPC. However, the predictions will evidently change when the MPC revise the plan at every time instant. By presenting the predictions to the user, these will contain uncertainties and can noticeably change.

This concept should be explained to the user in order to avoid confusion as to why this occurs. As mentioned in section 5.3, the predictions are updated every 5 minutes corresponding to the sampling interval. Also taking into account that whenever a user request is submitted the MPC scheme will not immediately display the comparison. Increasing the sampling time will only result in higher computational time of building the MPC with more time steps and is avoided. To inform the user, a timer is implemented to explain that the MPC scheme is delayed and when the next update is expected, illustrated in figure 6.11.

Figure 6.11: A timer component informing the user when new predictions are available in the application.

Further information on why the interval is set to 5 minutes is not directly explained to the user. Nevertheless, this component will inform the user on when the comparison will be displayed and that new predictions are displayed regardless of submitting user requests. The component will be updated every 5 minutes and new trajectories can be expected. Another explanation is provided as a hover effect on the timer component, to inform the user about potential changes in predictions due to MPC revising at every time instant.

6.2.2 Day-ahead spot prices and weather forecasts

Aside from the main graphical components, additional information such as spot prices and weather forecasts are presented. The MPC's ability to take into account additional data such as day-ahead spot prices and weather forecasts in the optimization is considered helpful information to visualize to the user.

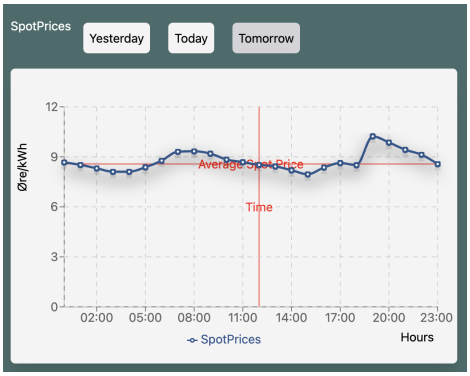


Figure 6.12: Graphical display of day-ahead spot prices.

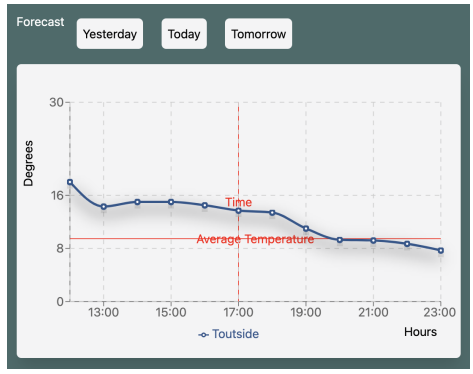


Figure 6.13: Graphical display of weather forecast.

The user can potentially view historical, current, and day-ahead spot prices in figures 6.12 and the weather forecasts of the outside temperature in figures 6.13. The data will depend on the available hourly spot prices from Nord Pool's API. The average is calculated and highlighted in red to indicate higher or lower values relative to the average. As such, the components provide an intuitive presentation of the data, in particular, understanding the spot prices is considered important for the user.

6.3 Detection mechanisms

This functionality is implemented in order for the web application to detect different aspects of the optimization and prediction that can be helpful for the user. Essentially, taking advantage of the predictive capabilities of the MPC scheme. By detecting significant differences in the comparison, the web application can highlight areas to inform the user of consequences related to adjusting weights and/or increasing temperature, further explained in section 6.4.

The web application can detect possibly unwanted behavior such as significantly lower temperatures and provide information as to how predicted temperatures and related costs are evolving. Based on what the application is able to detect from the predicted trajectories, the application is able to provide simple recommendations for the user. This functionality is implemented in order to inform the user on what type of actions they can implement to change the particular behavior, listed below. Accordingly, the user can potentially understand why this behavior occurs in the first place and further understand how to interact with the MPC scheme based on the recommendation. In addition, the components are responsible for transforming difficult language related to control and optimization into relatively simple concepts that are familiar to the user. This functionality is only implemented in the *Main* room for simplicity.

- Detect high spot prices
- Detect significant deviations from reference temperature
- Detect significant difference in electricity cost
- Detect heat storage

6.3.1 Detect high spot prices

In order to detect high spot prices, the application will need information on what is considered a high spot price. One potential approach could be to check for spot prices that are higher than the average. However, the likelihood of this occurring frequently is considered high. This will not necessarily provide the user with helpful information. Hence, a better approach would be to consider the specific times when the spot prices are significantly high during the day.

The current solution to detect high spot prices is to create a set containing values for when the difference between the current spot price and the average spot price is higher than 10 øre/kWh, described in equation 6.2. The array will contain the hours when the spot prices are significantly high. The calculation of the average spot price is provided in equation 6.1.

$$\mathrm{C_{spot,avg}} = \frac{1}{N}\sum_{k=1}^{N}\mathrm{C_{spot,k}} \tag{6.1}$$

Where the average spot price is denoted, $\mathrm{C_{spot,avg}}$. The N will vary depending on the available hourly spot prices on the Nord Pool site, using both the past spot prices and potentially the day-ahead spot prices. If the day-ahead spot prices are available N will be equal to 48. The high spot prices, denoted $\mathrm{C_{spot,high}}$.

$$C_{\mathrm{spot,high}} = \{C_{spot} \in \mathbb{R} \mid C_{spot} > C_{spot,avg} + 10,\ SP > 0\} \tag{6.2}$$

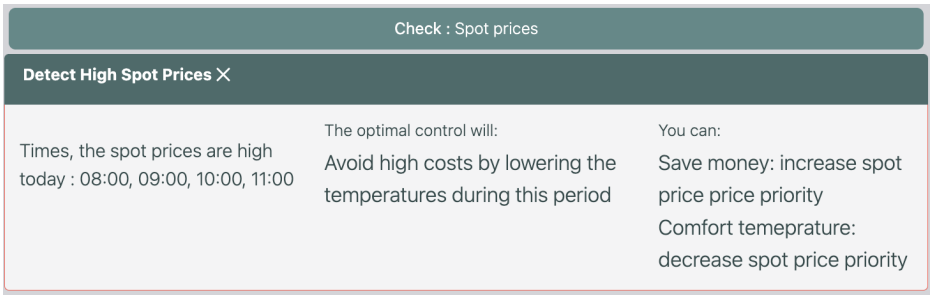


Figure 6.14: Detection component for explaining the concepts related to high spot prices.

The user is able to activate the detection by "checking" the spot prices as illustrated in figure 6.14. In this component, the user is presented with the hours the spot prices are considered high and information about how the MPC scheme will act accordingly. In addition, the user is presented with a recommendation on how to interact with the MPC scheme to potentially reduce the cost further or increase the thermal comfort.

In figure 6.15 an illustration of highlighting high spot prices in the graphical spot price component is illustrated. The highlight is activating by the detection mechanism. Albeit this alert component can detect high spot prices it is relative to the average. Occasionally, the spot prices are high and constant, which implies that the alert system will not detect any high spot prices relative to the average. Thus another component responsible for detecting deviation from the reference temperature is implemented and explained in the next section.

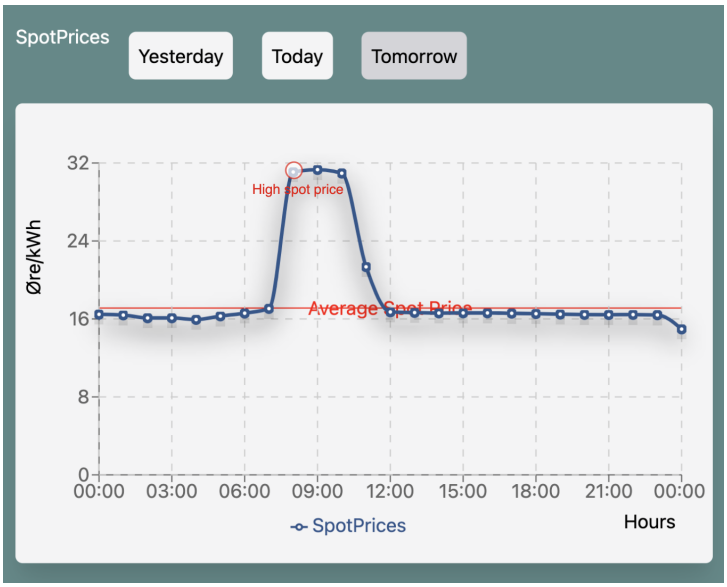


Figure 6.15: Graphical highlight of high spot prices.

6.3.2 Detect significant deviation from reference temperature

The user will experience deviations from the reference temperature as a result of the MPC scheme seeking a compromise between comfort and cost. There are several factors influencing the decision-making algorithm as mentioned in section 3.3.3. One of the most important concepts to communicate to the user is why the reference temperature is not achieved by the MPC. Accordingly, the web application is required to communicate how different elements are influencing the temperature trajectories and provides relevant information to the user, by using simple language to describe a relatively complex concept.

The current solution to notify the user about significant deviations in temperature is to create an set containing variables where the difference between the measured temperature and reference temperature is lower than -3 degrees, described in equation 6.5. The set containing this information is transformed into an array of objects, where the first element and last element are accessed. This is done in order to present the period where the deviation is occurring. The temperature difference are calculated as shown in equation 6.3 and 6.4.

$$T_{\text{diff1,k}} = T_{\text{room1}} - T_{\text{ref1}} \tag{6.3}$$

$$T_{\text{diff2,k}} = T_{\text{room2}} - T_{\text{ref2}} \tag{6.4}$$

Where k is the time instant over the sum of the prediction horizon N equal to 48 hours. $\mathrm{T}_{\text{diff,high}}$ is the set containing the temperature differences, $\mathrm{T}_{\text{diff1}}$ and $\mathrm{T}_{\text{diff2}}$ that are considered significant.

$$T_{\text{diff,high}} = \{T_{\text{diff1,k}}, T_{\text{diff2,k}} \in \mathbb{R} \mid T_{\text{diff1,k}} < -3 \cup T_{\text{diff2,k}} < -3\} \tag{6.5}$$

As a result of high spot prices, the MPC seeks to lower the temperature to avoid high costs. This information is considered important to provide to the user. As such the user can understand that the temperature in the house will be lower during these times. The detection component, presented in figure 6.16, provide information about how the MPC scheme will act when deviation occurs. In addition, the user is presented with a simple recommendation on how to interact with the MPC scheme to potentially avoid this behavior.

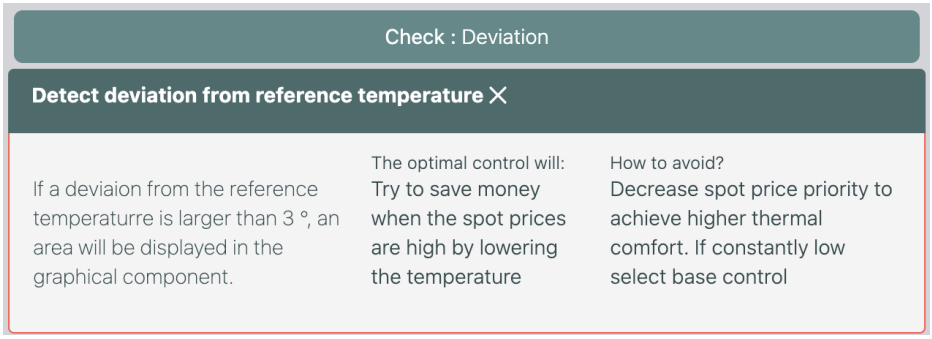


Figure 6.16: Detection component for explaining deviations from reference temperature.

If the alert component has detected a significant deviation, the graphical display will pinpoint the period where the significant deviation is found by displaying an area as demonstrated in figure 6.17. A gray area is displayed in the period where the Troom2 is expected to be deviating significantly from the reference temperature. This functionality can also detect deviations from reference temperature in Troom1 from the original MPC scheme as well, however, this was not the case during this test. The highlighted area can optionally be displayed or not by clicking on the detection button related to the deviation.

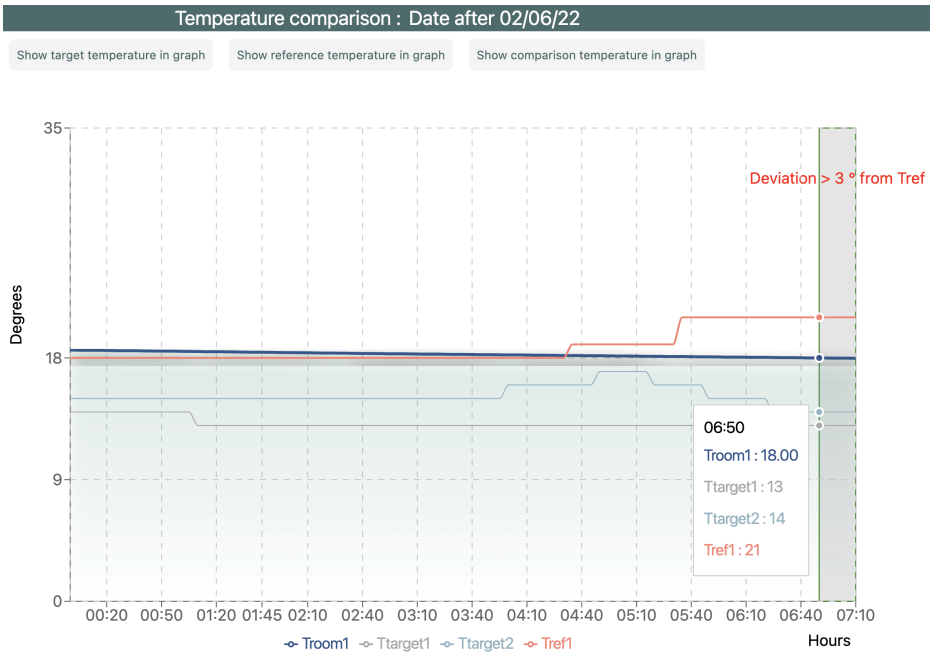


Figure 6.17: Graphical highlight of period with significant difference from the reference temperature.

6.3.3 Detect significant difference in electricity cost

The electricity costs related to the consumption can also provide valuable information in terms of cost savings and how the effects of changing the optimization influencing the costs. Similar to detecting significant deviations in temperature, detecting a significant difference in the cost i a set containing variables where the difference between the original MPC and the second MPC is greater than -0,05 NOK, shown in equation 6.7. The cost difference is calculated in equation 6.6.

$$C_{\mathrm{diff,k}} = P_{\mathrm{HP1,k}} * C_{\mathrm{spot,k}} - P_{\mathrm{HP2,k}} * C_{\mathrm{spot,k}} \tag{6.6}$$

Where k is the time instant over the sum of the prediction horizon N equal to 48 hours. $\mathrm{C_{diff,high}}$ is the set containing the temperature differences where $\mathrm{C_{diff}}$ is considered significant.

$$C_{\mathrm{diff,high}} = \{C_{\mathrm{diff,k}} \in \mathbb{R} \mid C_{\mathrm{diff,k}} < -0.05\} \tag{6.7}$$

The reason for choosing a negative value is to notify the user when the electricity cost is increasing as a result of changing the optimization. This will make the user aware of the negative effects of prioritizing thermal comfort. The negative value derive from either high spot prices or higher consumption during that period.

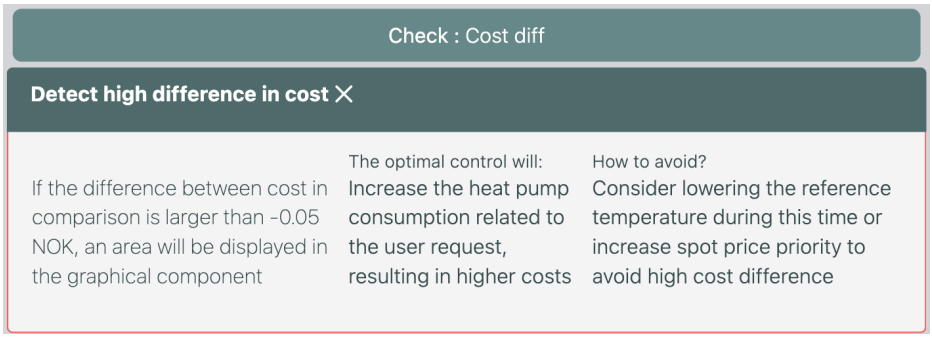


Figure 6.18: Detection component for explaining high difference in electricity costs.

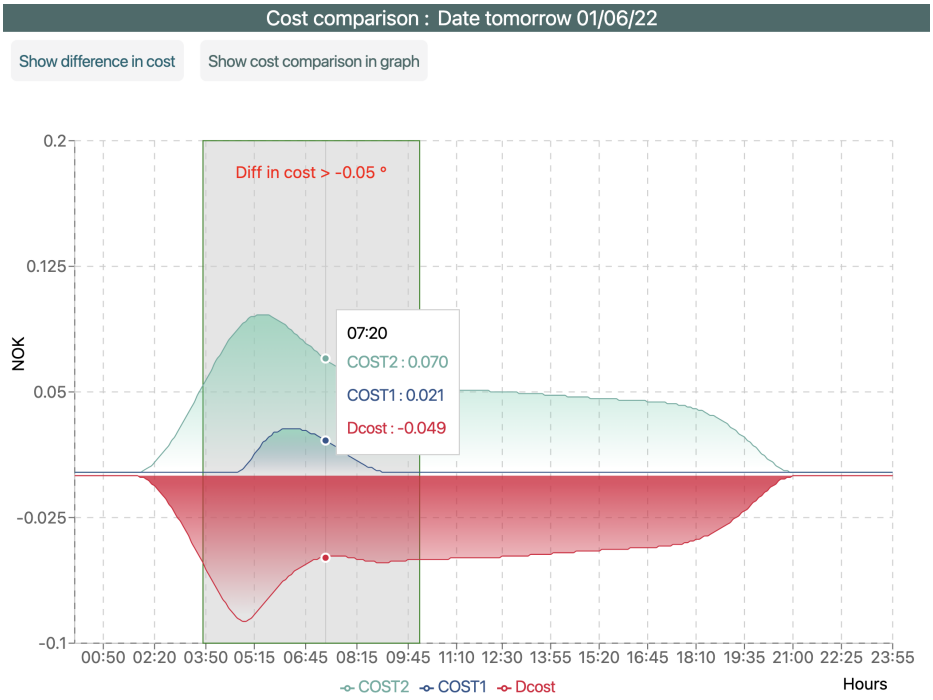


Figure 6.19: Graphical highlight of period with significant difference between costs in comparison.

The most effective way of visualizing this comparison is to present a graph containing both predictions. The cost difference may stem mostly from one specific hour. Accordingly, the application should be able to pinpoint where the greatest difference occurs, such that the user can revise the request to the MPC scheme and possibly avoid the high cost. The component in figure 6.18 will explain to the user that the new requested values have resulted in a higher cost due to the user wanting to achieve a higher thermal comfort increasing the consumption and related electricity costs.

In figure 6.19, an area for displaying where the difference between COST1 and COST2 is significant is presented. This area highlights the period with highest price difference between the requested temperature and the MPC scheme. In addition, the user can keep the new reference temperature in periods where the cost difference is not considered high.

6.3.4 Detect heat storage

The MPC scheme is able to store heat, by taking advantage of the thermal mass of the house and the price-signals. This concept can be visualized to the user by highlighting the period when the MPC scheme decides to store heat in the house before lowering the temperature in periods when the spot prices are high. This is an important aspect of how the MPC scheme is able to plan ahead and shift the load in the house to avoid heating when the spot prices are high.

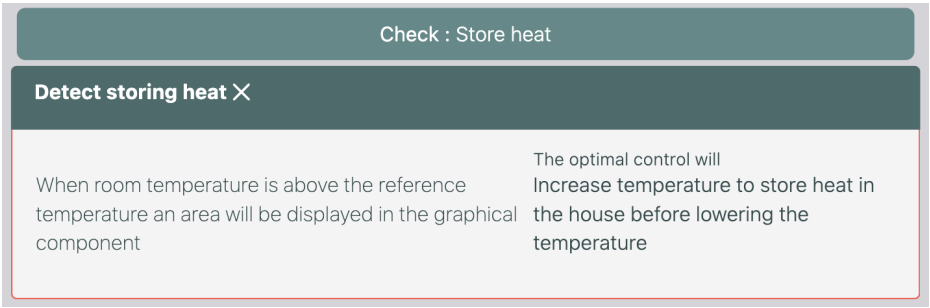


Figure 6.20: Detection component for explaining the MPC scheme storing heat.

The current solution to detect this behavior is to observe when the predicted room temperature is exceeding the reference temperature, explained in figure 6.20. When the user experience a higher temperature, it is considered beneficial in terms of electricity cost, however, some user might not prefer to have the house heated above the reference. However, there are no options for the user to avoid the MPC scheme preheating the household by storing heat because it is considered solely as an advantage of the MPC. In addition, this will require the user to interact with other weights in the MPC scheme, which is beyond the scope of this thesis.

This is considered a similar approach to load shifting, where the smart control is able to heat more when the spot prices are low in order to avoid heating when MPC can optimally choose to exploit this to even out the consumption. When the the spot prices increases the temperature in the house decreases. As presented in figure 6.21, the spot price before 8 AM are considerably low. In figure 6.22 the application has detected that the MPC is storing heat by highlighting this period in the graphical display. Troom1 is above the Tref, and after storing heat the temperature is reduced below the reference temperature. Storing heat when the spot prices are relatively low is one of the important aspect the MPC is able to perform in order to save money. In regards to the thermal inertia of the house, the room is able to maintain a relatively comfortable temperature although the MPC scheme is shutting the heating off afterwards to save money.

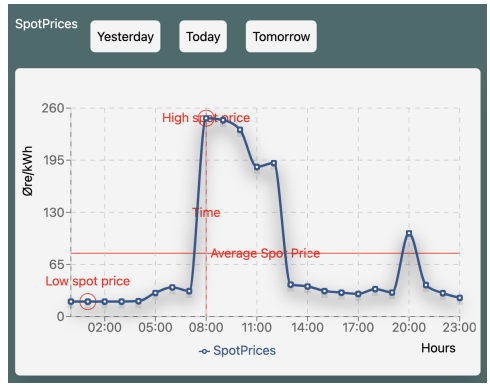


Figure 6.21: Graphical display of low spot prices.

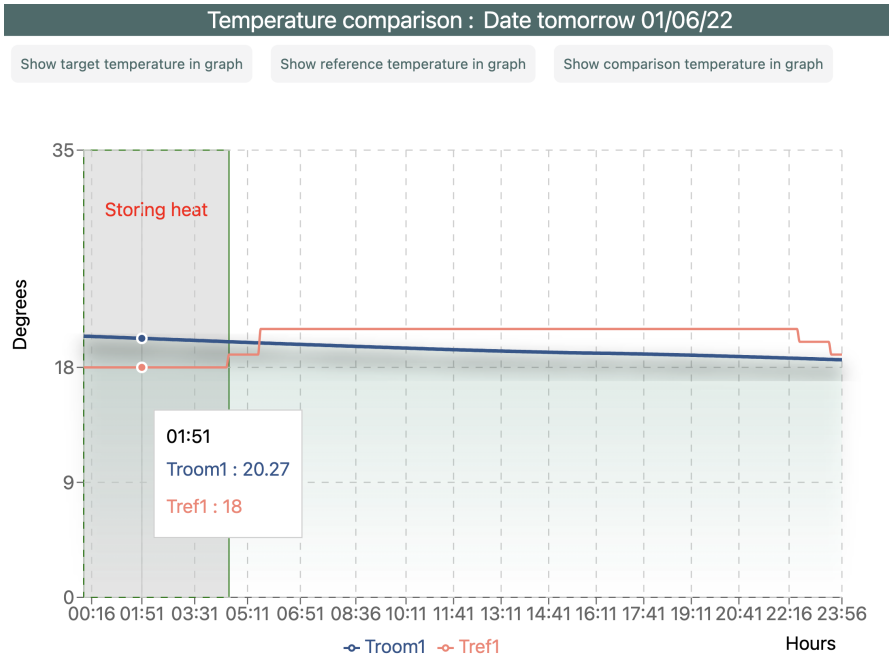


Figure 6.22: Graphical highlight of period when the MPC scheme stores heat in the house.

6.4 Human-MPC interaction

The web application is responsible for abstracting complex control theory into simple concepts. As mentioned in section 3.4, there are certain elements in the MPC scheme the user can change to influence the optimization problem through the web application. The term optimal is directly linked to the preference of the user, taking into account whether thermal comfort is more important than saving money and vice versa. This section presents the UI components responsible for demonstrating how the user can interact and change the optimization in order to achieve the desired output and to give the user the opportunity to decide the optimal performance.

6.4.1 Interactions with weights

The components presented in this section aim to explain how the user can interact and understand the SpotGain and BasePrice in the MPC scheme. Accordingly, these components are responsible for explaining how the changes in the optimization will affect the control of the heating system. From a user perspective, selecting weights will be similar to determining how much the user is willing to pay to achieve thermal comfort, giving the user the opportunity to select the optimization according to comfort and economic preferences.

In this part of the web application, the user is provided with three different options to interact with the MPC scheme, listed in table 6.1. These options will dictate how the spot market influences the temperature control. In addition, an experimental approach is used to validate the performance of using these control options, presented in section 6.4.4.

Table 6.1: Control options for interacting with the MPC scheme.

Option	Control aspect
Control 1	Optimal compromise between comfort and monetary cost.
Control 2	Decide priority of SpotGain weight
Control 3	Decide the calculation of the BasePrice

By selecting one control option, the user can interact with the MPC scheme. Control 1 describes the optimal compromise between thermal comfort and monetary cost, which is also referred to as the original MPC scheme. Control 2 demonstrates how the w_{spot} is adjusted to regulate the temperature and will directly interact with SpotGain weight in the cost function. Control 3 has an alternative way of regulating the temperature relative to the spot price market, where the calculated BasePrice determines the optimization. The three options will be detailed below.

Control 1 - Default

The first control option is referred to as the original MPC scheme (default) which has predefined weights (listed in table 3.4). In particular, the tuned w_{spot} weight is responsible for informing the MPC scheme how important cost savings are to achieve a certain trade-off and is by default set to 0.1. The initial weight is tuned specifically to achieve a reasonable compromise between thermal comfort and cost savings. In general, the user will experience small deviations from the reference temperature. However, larger deviations can occur when the spot prices are significantly high. This component is responsible for explaining this to the user and is considered important in order to describe the benefits of maintaining this compromise. Generally, the MPC scheme could optimize and manage the temperature of the house without any human interaction and automatically adjust. However, the application is responsible for communicating that if the user is unsatisfied with the original MPC scheme, then choose another control option. The MPC scheme will always run this as the default scenario for comparison, however, the control option is included in order to explain to the user what the MPC scheme is doing before the user decides to interact and change the optimization.

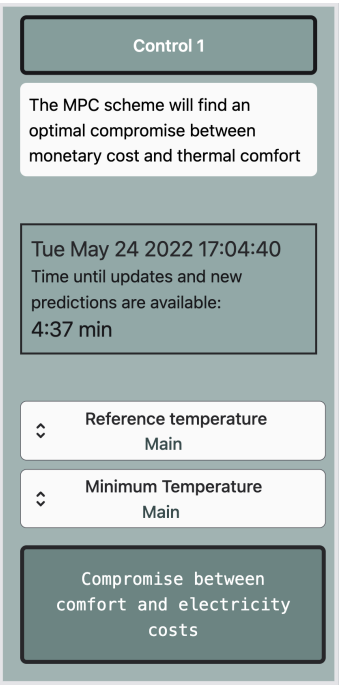


Figure 6.23: UI component for the user to select a reasonable trade-off between thermal comfort and cost savings.

Control 2 - SpotGain

One important implication of the optimal control of the heat pumps is the preference of the user. Some users will prefer thermal comfort over cost and vice versa. Ultimately, the user will arbitrate the optimal performance of the system. Accordingly, the user should have options to decide the influence of the w_{spot}, by providing the user with the proper tools. Without the proper knowledge of how weights are determined in the cost function, the user might not achieve the desired output. The challenge here is to convey the correct information about the effects of changing the values in the optimization and allow the user to interact.

Thus, in order to simplify the concept of weights, the user will have four options for setting the priority on the spot prices in the optimization, presented in figure 6.24. The approach is based on sorting the weight values from low priority to high priority in an effort to make the weight interaction more intuitive for the user. The options are tuned to achieve the desired results (see section 6.4.4), which entails that the user cannot directly interact with the weights, but rather determine how the MPC prioritizes depending on the weights to achieve a certain trade-off. The range of values the user can decide between is 0, 0.05, 0.5, and 1, where 0 is the minimum value and 1 is the maximum value for w_{spot}. Each component provides an explanation of the effects of choosing this priority e.g. "choosing high spot price priority will result in experiencing generally lower temperature by focusing on saving money", or "choosing low spot price priority will increase thermal comfort, experience higher costs". Selecting a weight value of 1 (i.e max priority), the optimization will significantly reduce the temperature in the room to reduce the monetary cost of heating. This is similar to setting the weight to the largest value since the spot prices are dominating the objective of the optimization. Thus, selecting values larger than 1 is not considered interesting for the user since it will provide the same results. Conversely, when the weight is 0 the MPC is simply tracking the reference temperature and disregarding the cost. Thermal comfort is achieved regardless of the spot prices since the user has neutralized the cost term by setting the weight to zero.

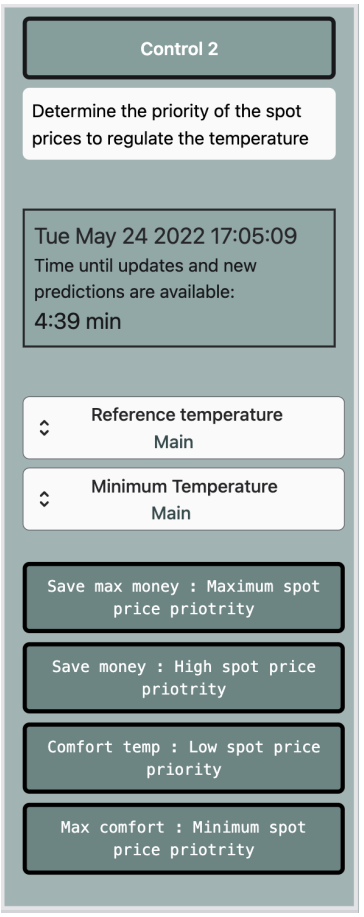


Figure 6.24: UI component for the user to interact with the spot price weight in the MPC scheme according to their preference.

Control 3 - BasePrice

The BasePrice is not considered a weight, however, it is included as a control options determining how the spot prices are influencing the optimization. The original MPC scheme can to some degree resemble load shifting, nevertheless, the cost function is not designed to specifically shift the load, but rather to avoid high spot prices to form a trade-off.

Optimally, the MPC attempts to avoid high spot prices by reducing consumption, but reducing the consumption at all times is a conflicting behavior because it directly affects the thermal comfort. In the efforts to diminish this behavior the user can choose to interact with the BasePrice. If the cost function is adjusted to only penalize when the spot prices are high relative to the minimum or average, the user can, in general, experience a higher thermal comfort. In addition, this will allow the MPC scheme to perform a better load shifting by taking into account the variations in the spot market better.

Interacting with the BasePrice is considered particularly helpful when the spot prices are high and constant. The user might not understand why the temperatures are constantly low and how to avoid this behavior.

By selecting the BasePrice control, the MPC scheme is to a greater extent able to neglect the effects of constant high cost and aim to heat the house closer to the reference temperature. Considering the calculation of the BasePrice (e.g equation 3.10), with low variation there will not be a large difference between the spot prices in general and the minimum. Thus, subtracting the BasePrice will result in the spot prices approaching zero in value in the cost function.

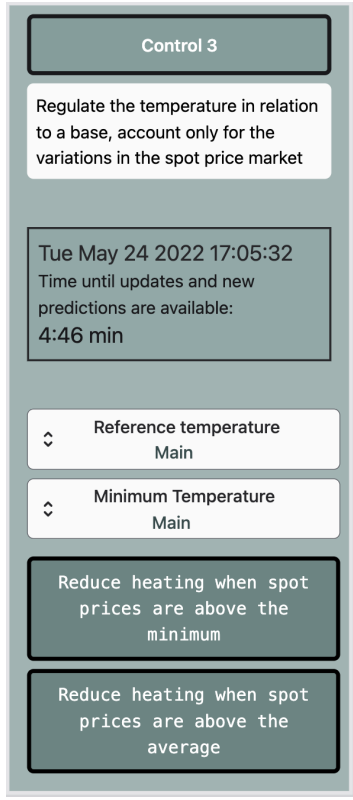


Figure 6.25: UI component for the user to decide the calculation of the BasePrice.

6.4.2 Reference temperature

The main purpose of installing a heating system, such as heat pumps, is to regulate the temperature. In order for the user to adjust the room temperature, the first parameter that needs to be considered is the reference temperature. The application is responsible for providing a user-friendly approach to setting reference temperatures for the individual rooms. The MPC scheme uses the reference temperature as input to the cost function in order for the MPC scheme to add cost when the room temperatures are below the reference. The temperature schedule is implemented in the sidebar component in an effort to make the accessible and allow the user to plan a daily heating schedule. The temperature component demonstrated in figure 6.26, is developed such that the user can input and submit the reference temperature to the MPC scheme.

Figure 6.26: Temperature schedule component for setting reference temperature.

The desired temperature is submitted as an hourly value, since allowing the user to set a new temperature every 5 minutes is considered unnecessary. This is based on the assumption that homeowners prefer relatively stable temperatures. In addition, an option to select the default temperature values is included as a default button. Considering the fact that the MPC scheme operates with a 5 minute sampling time, the hourly temperature values are interpolated in order for the MPC scheme to read the values. At every time step, the temperature setting closest to that time step is applied.

The temperature schedule can help the user become aware of the consumption related to heating by planning ahead. A potential challenge can be that the user consistently tries to increase the reference temperature since it is not achieved by the controller. However, increasing the temperature will cause the heat pump to generate more heat, and increase the power consumption and cost accordingly. This should be avoided and the user needs to be provided with such information. Thus, an information button is implemented to explain concepts related to the reference temperature.

6.4.3 Minimum temperature

The minimum temperature is decided by the user and can be implemented similarly to the reference temperature. The MPC scheme uses the minimum temperature as input in order for the cost function to strictly penalize when the room temperatures are below the minimum.

However, the minimum temperature is not required to be specified hourly, based on the assumption that users will have *one* minimum temperature. In cases where the spot prices are high and the w_{spot} is increased to avoid the cost, the room temperature tends to approach the minimum temperature. If the minimum temperature is considerably low, the user will experience low temperatures in the house overall. However, the user can decide to set the minimum temperature relatively close to the comfort temperature to avoid very low temperatures. This is related to the weight for avoiding temperatures below the minimum is considered high compared to other cost terms in the cost function, resulting in the MPC scheme rarely heating below this temperature.

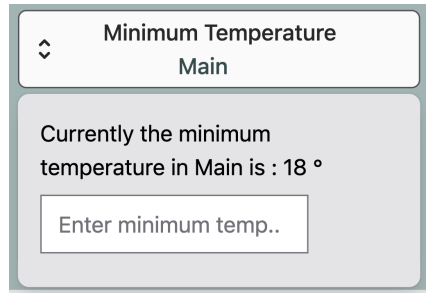


Figure 6.27: Temperature component for setting the minimum temperature.

6.4.4 Experimental results

In order to validate the performance of interacting with the MPC scheme, the temperature trajectories are plotted accordingly. All trajectories are compared to control option 1 and tested on the *Main* room, demonstrating the predictions 24 hour after initialization (Tomorrow).

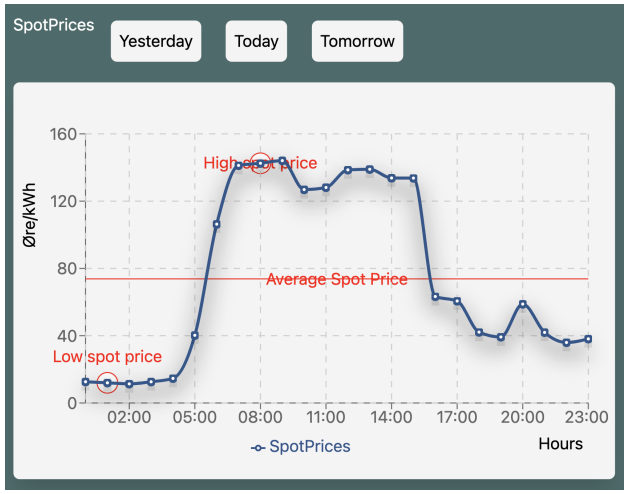


Figure 6.28: Spot prices on the day the experimental tests were performed.

In figure 6.28, the spot prices during the day the tests were performed are presented. This also highlights the period with low spot prices and high spot prices to give an indication of how the

MPC will predict the trajectories based on this information. In addition, taking into account that the spot prices are generally above the average, it will ultimately restrict the MPC to heat less when the spot prices are included in the optimization.

SpotGain

The first part of this analysis presents the influence of the spot market on the deviation of the temperature from the reference at various steady-state of the MPC.

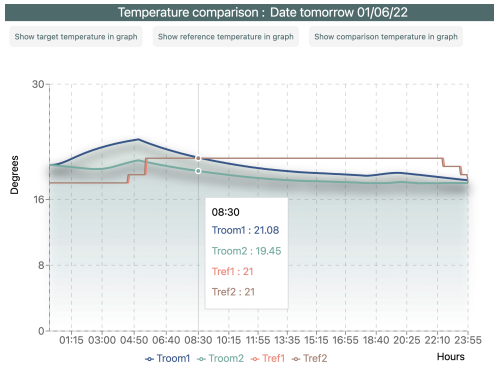


Figure 6.29: Illustration of the predicted temperatures with maximum spot price priority.

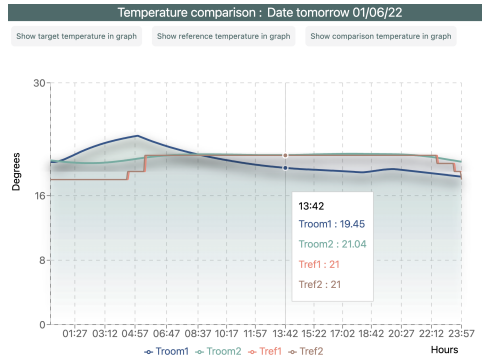


Figure 6.30: Illustration of the predicted temperatures with minimum spot price priority.

The maximum spot price priority ($w_{\text{spot}} = 1$) in figure 6.29 will result in the temperatures approaching the minimum temperature to avoid high costs in general and the user will experience a low thermal comfort. On the other hand, the minimum spot price priority ($w_{\text{spot}} = 0$) in figure 6.30 will ignore the spot prices completely and simply achieve thermal comfort by following the reference temperature.

Figure 6.31: The influence of maximum spot price priority on the 24 hour electricity cost.

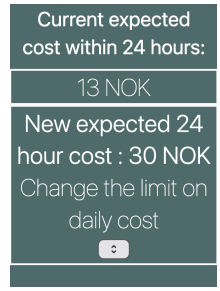


Figure 6.32: The influence of minimum spot price priority on the 24 hour electricity cost.

The resulting 24 hour cost of changing the spot priority to maximum and minimum are illustrated in figures 6.31 and 6.32, respectively. The 24 hours cost can give a good indication of the overall changes in electricity cost. By having the maximum priority the costs decreased by 1 NOK (from 13 NOK to 12 NOK), while for the minimum the cost increased significantly by 17 NOK (from 13 NOK to 30 NOK). This should encourage the user to avoid heating at thermal comfort at all times since the cost increases significantly.

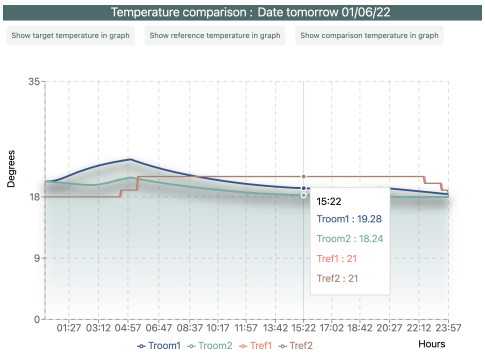


Figure 6.33: Illustration of the predicted temperatures with high spot price priority.

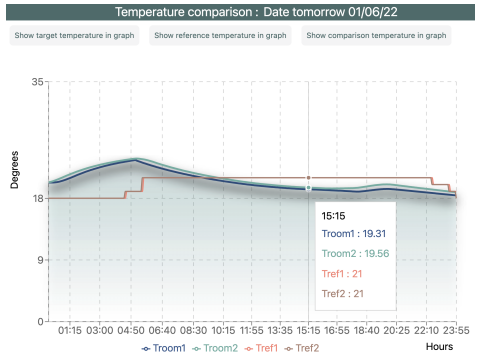


Figure 6.34: Illustration of the predicted temperatures with low spot price priority.

Furthermore, selecting high spot price priority ($w_{\text{spot}} = 0.5$), illustrated in figure 6.33, will result in generally lower temperatures and focus on saving money. Selecting low spot price priority ($w_{\text{spot}} = 0.05$), illustrated in figure 6.34, will result in generally higher thermal comfort. Due to the specific day the tests were performed, there can not be observed a significant difference between the maximum and high spot price priority, or the minimum and low spot price priority due to overall high spot prices.

BasePrice

The experimental results from interacting with the BasePrice are illustrated in figures 6.35 and 6.36. When the minimum spot price is subtracted it is economically advantageous for the user, since the MPC scheme is to a greater extent able to heat the house when the spot prices are low. Nevertheless, the 24 hour electricity cost will be slightly higher (increasing from 14 NOK to 18 NOK) compared to when the BasePrice is not accounted for. In terms of achieving thermal comfort when this is convenient the minimum spot price is a good alternative. This approach is considered the best option in regards to maintaining the compromise between comfort and cost.

On the other hand, when the average spot price is subtracted the user will to a great extent experience thermal comfort. The reference temperature is observed to exceed the reference temperature

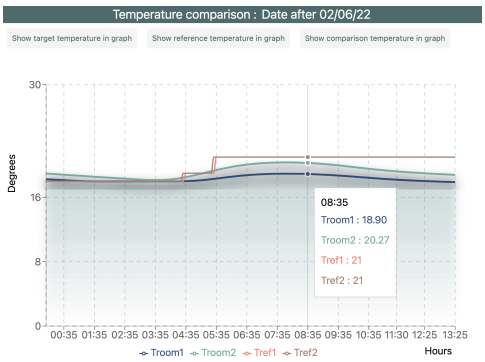


Figure 6.35: Illustration of BasePrice equal 1, i.e subtracting the minimum.

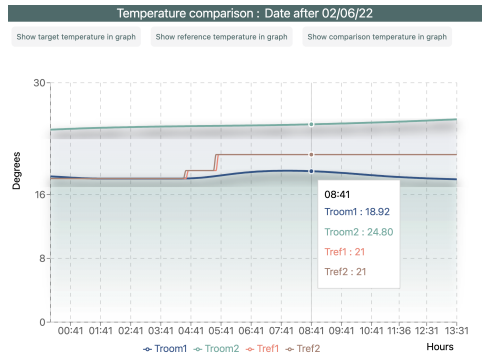


Figure 6.36: Illustration of BasePrice equal 2, i.e subtracting the average.

significantly. The MPC will only add cost whenever the spot prices are above the average. This is considered a good alternative, in theory, however, in practice the output is sub-optimal. The system is encouraged to heat more when the prices are below the average since the value is negative and interpreted as "earning money" in the cost function. The 24 hour cost increased significantly (from 14 NOK to 44 NOK), as well as the MPC scheme decides to heat above the reference temperature when the spot prices are below the average. By looking at the spot prices during the day the tests were performed (figure 6.28) the spot prices are lower than the average, before increasing significantly. However, the MPC is not able to decrease the temperature considerably albeit the spot prices are high during the day. This is partially due to the fact the MPC does not have a large penalty for temperatures above the reference and partially due to the MPC restricting rapid changes in temperatures. As such, choosing the BasePrice equal 2 is not considered optimal and therefore dismissed as a control option in the application.

Reference temperature

Finally, the experimental results for changing the reference temperature is illustrated to examine the effects of increasing and decreasing the reference temperature. The default reference temperature, Tref1, is 21 degrees. In figure 6.37, the effect of increasing the reference temperature is illustrated. Around 8 AM the reference temperature, Tref2, increased from 21 degrees to 25 degrees, approximately around the time when the spot prices increased, as shown in figure 6.28. The target temperature, Ttarget1, and Ttarget2 serve as input to the heat pumps and will decide how much power is drawn from the heat pumps. Considering the MPC scheme seeks to optimally choose the input, Ttarget 2 is increased before the reference temperature is increased in order to avoid consuming as much power when the spot prices are high. However, the MPC scheme does

not increase the room temperature significantly regardless of the reference temperature. This behavior can be connected primarily to the spot prices, nevertheless, the MPC scheme is to some extent failing to increase the temperatures. In order to reach the new reference temperature, the priority on the spot prices has to be decreased in addition to increasing the reference temperature.

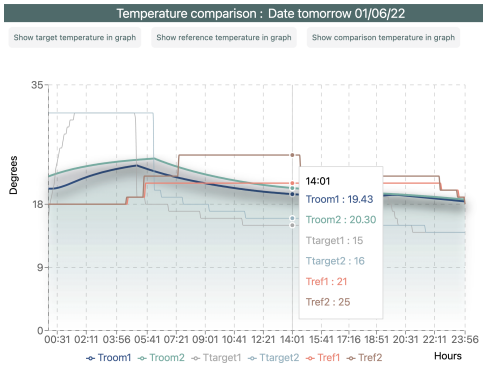


Figure 6.37: Illustration of increasing the reference temperature to 25 degrees.

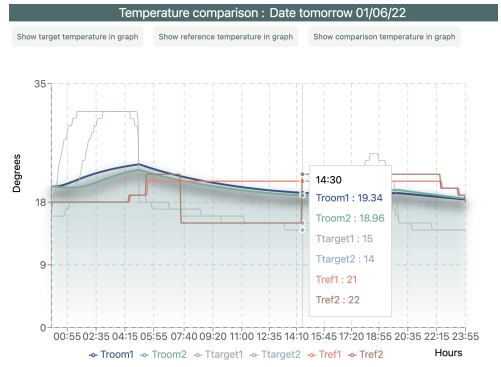


Figure 6.38: Illustration of decreasing the reference temperature to 15 degrees.

The 24 hour cost is only predicted to increase by 1 NOK when increasing the reference temperature considerably this day. Taking into account that the spot prices are relatively high this day, increasing the reference temperature does not increase the thermal comfort significantly. Conversely, in figure 6.38, the effect of decreasing the reference temperature in the same period from 21 degrees down to 15 degrees is illustrated. The input Ttarget2 is observed to stay below the Ttarget1 and is expected to consume less power during the high spot price period. Nevertheless, the 24 hour cost is not predicted to decrease, due to the MPC scheme not penalizing temperatures above the reference much. As a result the room temperatures are failing to decrease during this period.

Chapter 7

Discussion

This chapter presents a discussion of the software implementation and evaluates potential improvements for further development of this smart house system with a focus on the user perspective. Essentially, the discussion will address the challenges related to the human-MPC interface and how solutions to this can ease the adoption of complex control strategies in the average household.

7.1 Evaluation of software implementation

In this section, an evaluation of the software implementation is presented with regard to the graphical displays and interactive components of the web application. This discussion will mainly consider improvements to the current implementation, however, also considering alternative approaches to future development.

Overall, the software implementation presented in chapter 5 and 6 demonstrate that a web application developed using JS is a suitable HMI for this MPC scheme. Each UI element is responsible for communicating the relevant information to the user, which required a simple and intuitive design approach. Albeit several functionalities and interactive components are implemented, there is still room for improvements considering that there could be countless scenarios the user can encounter. Thus, there are limitations to this discussion to remain within the scope of the thesis.

7.1.1 Graphical comparison

The graphical comparison, presented in section 6.2, is considered a valuable implementation in terms of visualizing the effects of changing the optimization. The graphs include customized functionalities in order for the user to understand the anticipated heating in the house and the related costs. This will automatically assist the user with explanations without directly explaining (in text) the consequences of interacting with the MPC scheme. As such the user can make an informed decision to proceed with the request or disregard the changes. However, the comparison might be confusing to the user in terms of knowing what the current MPC scheme is optimizing for. In order to clarify this, the application should clearly display the current MPC actions. In this manner, the application could inform the user about whether the control algorithm is running the original MPC scheme to control the heat pumps or if the user requests are implemented as MPC actions.

Improve customization in graphical display

A potential implementation to improve the graphical display for electricity costs could be to highlight the priciest hour. The web application is in fact able to detect high-cost differences, however, detecting the priciest hour could give a good indication of when the highest spot prices occur and/or when the consumption is highest. Further, give a good indication of which period the user should tell the MPC to lower the temperatures considerably if the desire is to lower the costs. However, the reason for not implementing this currently, is due to the limitations of the charting library. Accordingly, highlighting without drawing a rectangle over the graph is not possible with the current library.

Albeit the charting library is restricted, it is possible to highlight periods by displaying areas, as demonstrated in section 6.3 (e.g significant difference between the room temperature and the reference temperature). This is the current approach to solving the challenges related to the charting library and is essentially provided as a potential solution although it is not optimal. As such, the web application is to some extent able to explain concepts by visualizing/highlighting and explaining.

In order to achieve more customization in terms of illustrations and explanations, the charting library might have to be changed to have more flexibility or possibly develop the SVG elements from scratch. However, as mentioned in section 4.3.2, this will require a lot of work since most libraries are built on top of other elements in order to simplify the work. Selecting a different library could potentially solve some of the related problems to displaying more information in the graphs and hover functions to highlight specific parts.

7.1.2 Selecting the appropriate trade-off

As presented in section 6.4.1, there are currently three control options implemented in the web application that allows the user to select how the MPC scheme takes into account the spot price market. Implementing intuitive tools for the user to interact with the MPC scheme has been one of the priorities in this thesis. To provide a simplified language, the UI elements are implemented as priorities rather than weights. Although it might be challenging for the user to understand exactly what the MPC is doing to control the heating, there is no single solution to explain this concept since the control will vary from day to day. Conveying the correct information for every single scenario is difficult and almost impossible. However, this might originate from the data handling not being generalized enough to accommodate more MPC-user scenarios.

Based on the experimental results in 6.4.4, the components are evaluated to provide sufficient information through these simplified explanations. These results also demonstrate the effects of adjusting the SpotGain and BasePrice. Selecting a low priority clearly shows that the thermal comfort level has been obtained as well as increased costs related to the consumption. Conversely, having a high spot price priority gives great ability to save money, however, this compromise the thermal comfort considerably. Ultimately, the user will be the person who arbitrates the optimal performance. Nevertheless, there should potentially be more information provided in terms of promoting energy-saving capabilities, that have not been accounted for in this thesis.

The reason for implementing three options is for the user to simply distinguish between how the spot prices are prioritized in the MPC scheme. Another reason for this particular approach is for the user to only decide to interact with one of the control options at once. There are currently no restrictions or guidance provided to the user, however, this should probably be implemented to include further explanations for the user to distinguish between the interactions.

Considering the user is satisfied with the original MPC scheme, control option 1 is selected. Selecting control option 2 will essentially let the user select how to prioritize the spot prices. By prioritizing spot prices, the user is informed that thermal comfort is compromised. Accordingly, the user can interact with the alternatives in control 2 to achieve the desired trade-off. By selecting control option 3, an alternative approach to account for the spot market is used, which allows for regulating the temperature according to a base price. However, if the user decides to select submitting more than one of the control options, the MPC might behave differently from what the user expects. This is just a potential discrepancy with the current solution, however, this might not have a major impact on the overall system. This is tested, however, all possible scenarios have not been investigated.

Another element to consider is the user confusing these control options with each other, which can be a possibility considering that all regulate the temperature based on the spot price market. The w_{spot} determines the importance of cost savings in the optimization, while the base price is not a weight, it is rather a tool for the user to regulate the heating relative to variations. Nevertheless, having three options will allow the user to have different possibilities of influencing the weights, which allows for more flexibility. It is assumed that the current implementation works well for the given purpose. However, if flexibility cause confusion, an alternative to these control options could be to have an even more simplified version, with three alternatives: low, neutral, and high. Although this approach was evaluated, the current solution is considered a better approach.

Interaction with other weights

The current implementations in the web application seek to balance the interactive components to provide sufficient information and flexibility as well as restrict the user interactions with the MPC scheme. More specifically, the application is restricted to only interacting with the w_{spot}, thus other weights are not adjusted by the user. The reason is simply that the trade-off between thermal comfort and cost savings is the main priority and requires only adjusting the priority of w_{spot}.

Nevertheless, one particular weight could be interesting to investigate, namely the $w_{\text{temp.above}}$. This weight is considered intuitive for the user to interact with since it determines how the MPC scheme heats above the reference temperature and stores heat. Depending on the preference of the user, heating above the reference temperature might not be ideal. Taking this into account, adjusting the $w_{\text{temp.above}}$ can select the priority with options such as "Allow heat storage" and "Avoid heat storage", demonstrated in figure 7.1.

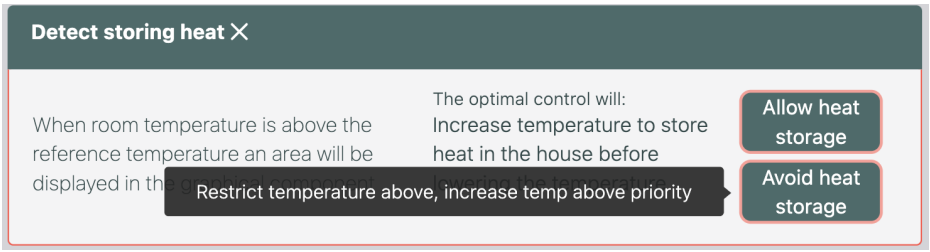


Figure 7.1: Potential implementation for user to restrict the MPC scheme to store heat in the house.

The first option will have a relatively low priority and the second option will have a high priority to strictly avoid the heating above. In addition, this can be advantageous when the user decides to decrease the reference temperature significantly (as shown in figure 6.38). Due to the temper-

atures above not being penalized very heavily, the MPC is unable to track a decrease in reference temperature and reflect that in the room temperature. This limits the ability of the user to lower the temperature during high spot prices. On the other hand, increasing the $w_{\mathrm{temp.above}}$, in general, can reduce the MPC flexibility to store heat during low-cost hours. Subsequently, increasing the electricity cost unnecessarily due to the inability of reducing consumption during high-cost hours, which is required to be communicated to the user. If this was to be implemented, tests have to be performed in order to evaluate the effects of changing this weight.

7.1.3 Improve detection mechanism

There are currently four different detection mechanisms that can inform the user about when the MPC predicts the temperature or cost to behave in a particular way (explained in section 6.3). The intention is to provide useful guidance by simply explaining what is happening in the current MPC scheme. This functionality is considered a reasonable approach to informing the user about the predicted output. However, the detection mechanism is based on identifying times where the MPC scheme is behaving in a certain way relative to a static number. Considering that the system is dynamic and changes all the time depending on the particular day, a static number to detect may not be sufficient. The detection mechanism should be relative to the data that is entering the web application in order to have a more dynamic functionality.

In addition, this functionality is able to provide some recommendations. If the web application detects high spot prices the user can choose between saving money by increasing the priority on the w_{spot} or higher thermal comfort by decreasing the priority on the w_{spot}. Although these recommendations are simple they can provide some valuable information to the user in terms of avoiding certain behaviors from the MPC, however, customized recommendations should be further investigated, explained in section 7.1.4.

One noteworthy aspect to mention is that the detection is currently only applied to one of the rooms, meaning the *Main* room is the only graphical display that can handle detecting capabilities due to difficulties with data structures. This should be implemented in all rooms when considering further development of the web application. Though there could be more intuitive and simpler approaches, the current implementation is assumed to function well for the given purpose.

Detect high and constant spot prices

One detection mechanism that should be considered implemented is high and constant spot prices. The reason for implementing this can be to inform the user that the temperatures will be constantly low during this period. In addition, propose a suitable interaction for eliminating this effect. The best solution to avoid constant low temperatures is for the user to interact with the BasePrice.

An alert or detection mechanism is not established for this scenario. Nevertheless, it is possible to detect high and constant spot prices by comparing the average historical spot price and the current average to calculate the standard deviation. If the average is high and the standard deviation is low the spot prices can be considered high with a low variation period/ constant. This could be implemented to improve the interaction with the MPC and possibly avoid such behavior if the user prefers thermal comfort during these periods.

7.1.4 Further implementations

There are several possibilities to implement new components and interactive elements in the web application. In particular, a further implementation should take into account that the predictive capabilities of the MPC scheme can be exploited further. However, with the current software, this will require more data processing. Some potential implementations are described below.

Turn off the smart house

An alternative to reduce the cost and energy consumption further with smart control is to have a functionality that allows the user to turn the heating system off or reduce the heating significantly. The web application could remotely tell the MPC to reduce the heating when the user expects to be away from home. By including a switch button in the web interface, the user can choose to "turn the heating off". This could be a relatively easy implementation, however, the MPC scheme is not currently designed to perform a shutdown, without stopping the code.

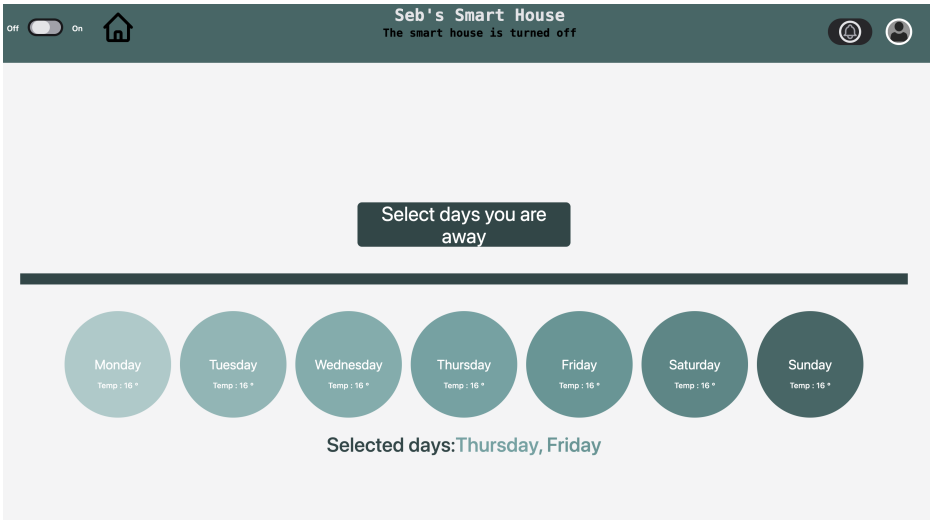


Figure 7.2: Potential implementation of "turn off" the smart house.

Including a turn-on/turn-off button can give the user an alternative way to lower the consumption by planning ahead. When the smart house is "turned off" the user can be prompted with a daily schedule, shown in figure 7.2. For instance providing a default value for the temperature settings to be 16 degrees or providing a button to tell the MPC to shut down the heat pumps completely. By specifying the days away, the web application can make sure the consumption is reduced during this period. Eventually, remote configurations of the heating system can also allow for the system to preheat the house before arriving back home again.

Display how the MPC revise

One of the challenges related to explaining the MPC scheme is the fact that the predictions change at every time instant. If the user expects a certain temperature in the room but experiences another temperature, this might be conflicting for the user and potentially lead to the user not trusting the system. Although the temperatures can be fairly accurate the uncertainties in the prediction apply, in particular, to the predicted electricity costs.

The uncertainties in the predictions can possibly stem from the length of the prediction horizon. However, the effects of reducing the prediction horizon to 24 hours are assumed to not have a major influence on the uncertainties. Explaining and displaying this concept should be further investigated. A potential solution can be to display past predictions of the MPC scheme to illustrate to the user how the MPC is revising its predictions all the time and clearly explain the uncertainties related to the predictions of the MPC.

Schedule for setting weights

Similar to the temperature schedule for setting reference temperatures, the control options for setting weights could have the same approach to scheduling. Considering a scenario where the user observes the temperature trajectories deviating significantly in the evening. During this period, the user wants to achieve thermal comfort and wants to avoid the MPC scheme lowering the temperature due to high costs. Allowing the weights to be scheduled can increase flexibility and exploit the predictive capabilities of the MPC further. This will also let the user save more money by potentially increasing the spot price priority during a short period of time during the day when the spot prices are significantly high. Accordingly, the schedule can let the user have a reasonable comfort temperature where this is feasible and let the user maximize cost savings in periods of high spot prices.

Another interesting element is having four different cost functions for the rooms in the smart house. Although the rooms have different temperature settings, there is only one cost function defined for all rooms. Hence, changing the weights will evidently affect all the rooms. The user

may desire to have a specific algorithm running in one room, where the preference is thermal comfort, while in other rooms the user might want to focus on saving money. This could indeed provide more flexibility, nevertheless, the user will have to interact with the MPC scheme more.

These approaches will require quite a few adjustments in the MPC scheme. The process of doing so is assumed not trivial. However, it could be an interesting aspect to further examine and can potentially provide more functionalities and improve the user experience.

Get customized recommendations

A possibility to enhance the smart control and take further advantage of the predictive capabilities of the MPC is to provide customized recommendations. These recommendations could be a functionality that provides explanations e.g. "here is the cost associated with your choices of temperature for tomorrow". When the user is scheduling the reference temperature, there could be implemented recommendations that inform the user of potential cost savings e.g "reducing the temperature to the minimum temperature between 23:00 and 08:00 you can save 10 NOK" or "choosing low spot price priority in this period will cost you 3 NOK". In response to recommendations, customers may choose to adjust their consumption patterns.

Accordingly, this will allow the user to exploit the predictive capabilities of the MPC scheme by providing recommendations. This can also contribute to the user acquiring a more conscious approach to heating the house and hopefully contribute to reducing consumption further. By quantifying the estimated cost savings between the original temperature trajectories and the altered trajectories, the consumer can be made aware of the consequences of interacting with the MPC prior to submitting requests.

Alternatively, to detect unusual behavior, the web application could have an alert system that is able to send automatic notifications to keep users informed on the newest information regarding the smart house. In addition, possibly be customized to what information the user wants to be notified with. This is currently not implemented and requires a considerable amount of time to create the appropriate data structure and functionalities.

Although there are some recommendations in the web application, these components are restricted to only displaying static information whenever the application has detected a certain behavior. A further improvement of these should be considered. The solution to include this is not considered trivial and will require additional code in the MPC scheme, data processing in the server, and client-side of the application. However, this is an interesting approach to consider for further development.

Other alternatives to calculate BasePrice

As mentioned in section 6.4.4, the calculation of the BasePrice based on the average of spot prices is not considered an optimal approach. However, there could be other alternatives to calculating a BasePrice. One possibility is to communicate to the MPC a relative spot market and divide the spot prices by the average. In this way, the MPC does not use the price but rather a percentage that explains e.g "the price now is 50% more expensive than the average". This could eliminate the effects of having negative values in the cost function when the spot prices are below the average because there will be no negative values. This might be a reasonable approach that has to be properly tested, however, this is not within the scope of this thesis.

7.2 Data processing

The functions responsible for handling the data structures on the server-side of the application are specific to the functionalities presented in the results of this thesis. However, these functions are not generalized, subsequently, complicating further development of the web application, such as adding more IoT devices to be monitored and controlled or adding functionalities. A number of these functions were developed to have the MPC data transformed into structures that can be understood by the charting library in React. In this way, the front-end development eliminates the need for multiple functions to further process data coming from the server. However, the server could be made more flexible by standardizing the structures further. As such, there are potentials for improving the data flow, lower risks of failure, and handling increased amounts of data.

7.2.1 Data flow between the server and MPC

An improvement that needs to be considered is the HTTP requests between the web application and the MPC. The current approach will have the comparison available at all times, due to updating the data every 5 minutes as mentioned in section 5.3.1. However, a better approach would be to only include the comparison when the user wants to compare how the trajectories will change if the user decides to change the parameters (e.g. weights). Due to difficulties with the server, it is not able to differentiate between the user requests.

On the other hand, the MPC scheme is able to differentiate the output depending on the requests, however, the data processing in the server is not general enough to handle different amounts of data. In other words, the data structure should be made more general to solve this issue. The current solution to this is to have a dynamic arrangement of data in the graph in order for the user to select if the comparison should be displayed or not.

Moreover, another important consideration is related to stopping and/or updating the server. In these cases, the requests from the web application disappear. Whenever the server is not running (either stopped/loss of Internet connection) the MPC scheme must include error handling for reading empty data or undefined requests to the server. This is included in the MPC algorithm since error handling is important in order to avoid failures in the house control algorithm. If the user request changes in the MPC scheme with empty values, the MPC is able to respond to these requests by using standard values, such that submitting empty values does not result in failure.

Run on Raspberry Pi

A potential approach to have more remote access is considering to run the MPC scheme directly on the Raspberry Pi. Currently, the MHE/MPC schemes can run on the Raspberry Pi, however, establishing communication from the Raspberry Pi to the web application has not been investigated in this report. Remote configurations, i.e the possibility to operate the system outside the house (local area network (LAN)), and have data stored in different places, have several advantages and should be further considered for optimal management of the smart house. These advantages are related to increased flexibility. A potential improvement for the web application development is to have it running on a separate computer and let the Raspberry Pi be responsible for retrieving the user requests from the web application and sending them to the MPC scheme. Possibly create an API from the Raspberry Pi in order for the server to get the latest outputs from the MPC and the Raspberry Pi will get the latest user request from the server/REST API, as illustrated in figure 7.3.

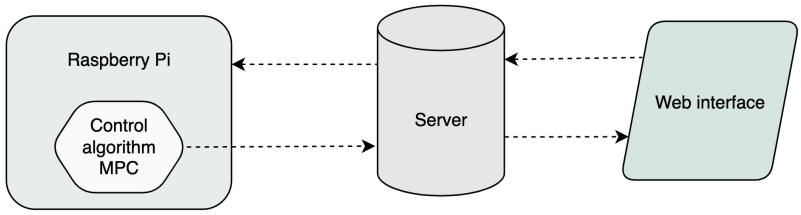


Figure 7.3: Illustration of a potential remote configuration by communicating with the Raspberry Pi.

7.3 Professional development and deployment

In general, web applications and services have the ability to provide flexibility to the user. Albeit this is the general case, one shortcoming of the current development is related to the local operation and storage system. To access the web interface, the current solution requires the user to have the software running on their local computer with local storage, as mentioned in section 5.3.1. This is not ideal for making the web application available for everyone or accessing it remotely. In order to make the web application run independently of the localhost, it should be deployed either on a dedicated domain, through a platform, or use cloud web services.

7.3.1 Security

Considering the fact that the control algorithm is executed locally, the data is stored locally and the web application runs on localhost there are no current security threats. Thus, security is not accounted for based on the assumption that there are no current risks. However, if the application was made commercial or public, the security aspect would be imperative. In general, for commercial IoT applications, security is regarded as one of the essential elements when developing smart home applications. For further development, the security aspect should be integrated in terms of protecting against potential leakage of sensitive and personal information.

7.3.2 Scalability and availability

One of the concerns related to the future development of the software and web application is the software scalability, referring to the measure and ability of the application to handle larger amounts of data and support other IoT devices.

The application is solely developed to support monitoring and controlling the heat pumps in the smart house. Nevertheless, the smart house includes various sensors and actuators which are not accounted for in this thesis.As mentioned in section 3.1.1, other IoT devices are currently installed in the smart house. Further development should consider including more devices to let the user monitor and control more smart appliances in the house. Subsequently, the web application will have to manage an increasing amount of data.

As mentioned in section 5.1, the data is stored locally on the server-side of the application. However, in order to access this data, the server has to be running with an Internet connection. An external database to store data has not been evaluated in this thesis due to limited time, however, the server resembles a nonrelational database. The advantage related to having a database is that the data resources are stored locally when they are offline, so they can still be used when there

is no server connection. Local databases are located on a dedicated computer or device, while a cloud database can be accessed through the Internet by utilizing a web server.

A potential solution to resolve this issue is to investigate the possibility of developing a cloud-based application with a dedicated database in order to scale according to the increasing load. In addition, this can take into account the security levels when developing the application and increase the availability. This could be further evaluated.

7.3.3 Smartphone app

With a web-based application, there is a lower availability compared to a phone-based application due to the requirement of having an Internet connection at all times. Additionally, considering the user wants to interact and monitor the system more frequently, a smartphone app is preferred over a web application since it also requires the person to have a computer. The reason for choosing to develop the user interface as a web application is due to the simplicity of integrating the application with the current smart house system.

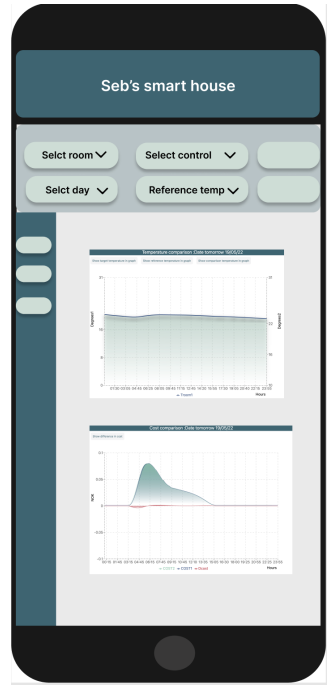


Figure 7.4: Web interface design for smartphone browser or a potential smartphone application.

There are possibilities to create a responsive design, that will allow the web interface to be compatible with the browser on smartphones. This will entail that every component is adjusted according to the size of the smartphone screen. Although the design of the interface is not the main priority in this thesis, design methods are considered one of the key elements for improving user-friendliness. By including a responsive design the user can be able to access the application through the phone for monitoring and controlling the smart house system. Since smartphones have smaller screens than computers some functionalities might be restricted on the smartphone, in particular, the option for multiple graphical displays and some options will be compressed inside components.

An alternative to extending the functionalities and accessibility of the web application is to create a native mobile app. This will require a whole different approach in terms of programming language and software technologies and they tend to be more advanced in terms of features and

functionality. Another concern related to developing mobile apps is that they normally take more time and the complexity is increased in terms of programming skills. This could potentially be a suggestion for future work related to user interface development.

7.4 User testing

This thesis focus on a user-centered approach to identify challenges related to understanding advanced control such as MPC schemes. However, the web application is developed by a person with familiarity related to control and optimization which could have had an impact on how the data is presented and explained. Thus, in this thesis, the user perspective is partially influenced by the author's knowledge of control theory and MPC. Nevertheless, the web interface is assumed to provide a sufficient amount of explanations and understanding of the MPC scheme to the average user.

An alternative to validate the software implementation and user experience further and indeed get a complete picture of how the average user can understand such a system, a user test should be conducted. Conducting user tests of the web application could also be a good alternative to determine other functionalities to add to the application. This will further contribute to getting a more correct understanding of the user perspectives and the ability to understand the system through interacting with the web application. In particular, this is interesting as the more advanced control methods are applied to optimize smart house appliances, and understanding the challenges can provide information on how to solve these questions by directly including the user.

7.5 Future of smart homes

An essential part of this thesis is to analyze how the user can be integrated into the system to facilitate the adoption of more complicated tools such as MPC. By developing a user interface, this thesis focus on a more practical approach to considering the user perspectives and potential adoption of such systems.

In comparison with other available software services for smart homes, this application is considered more comprehensive and user-friendly. However, future services and applications for home automation systems will rely on even more complex control strategies and require more data from numerous sensors. Thus, smart home applications should be able to handle an increased amount of data and functionalities.

An example of other applications with real-time graphical displays is Tibber. In addition to real-time graphical displays, this application offers more customization and advanced functionalities

that provide valuable information beyond just energy consumption. In addition, the user can specify high-level preferences to the MPC. Having a predictive algorithm can provide information ahead of time and highlight areas e.g when the spot prices are significantly high. Tibber claims that by following the consumption in real-time the user can reduce their energy consumption and acquire a behavioral change by becoming aware of the consumption patterns. However, this requires the user to shift their consumption.

With the MPC schemes' ability to optimize energy consumption automatically based on energy prices and weather forecasts, the user does not have to be burdened by shifting their consumption. The MPC can aid consumers in improving their energy consumption patterns automatically. In terms of efficient energy use, the household becomes an integrated part of the grid by installing optimized controller design in heating systems. By minimizing the consumers' electricity bills, the control strategy is directly contributing to enhancing the flexibility in the grid, which is an important implication of future smart homes and energy management systems.

Considering the MPC scheme in this thesis, future considerations for improving load shifting could be to examine the potential for energy/heating storage. If consumers find the MPC scheme helpful in reducing the overall electricity bill and the heating is used in a smarter way, it will have both operational and economic advantages.

This study is considered valuable for the integration of future smart home solutions, such that the current development of smart energy management systems and temperature controllers does not compromise the understanding of the user. This can also contribute to reducing the current gap between automation and adoption. The results obtained in this thesis can to some degree also support the theory of reducing concerns related to the adoption of more complex tools for controlling heating devices albeit there have not been conducted user testing methods in the scope of this thesis. Optimistically, including user interfaces in smart homes can contribute to curtailing the resistance from users and building trust. Accordingly, this thesis has addressed the main challenges related to developing a human-MPC interface and how solutions can ease the adoption of complex control strategies in the average household.

Chapter 8

Conclusion

The objective of this thesis was to evaluate the feasibility of implementing a web application designed for an MPC-based smart home control algorithm. This thesis has a focus on the more practical approach considering smart home implementation by supporting a user-friendly interface between the MPC and the homeowner.

The software implementation consists of a web interface developed using JavaScript, where the interactive components allow the user to monitor the predicted temperatures and related electricity costs. The predictive capabilities of the MPC scheme pose certain challenges in terms of uncertainties and intuitive communication, however, it offers valuable information to the user through graphical displays. The application take advantage of these predictions and offer customization, nevertheless, highly customized elements have been complicated to provide.

The web application can to a large extent explain, visualize and allow the user to influence the optimization of the MPC scheme by selecting a certain trade-off between comfort and cost savings in regard to their preference. The results from developing this web application have answered questions related to how the user should be integrated, potentially leading to easier user adoption for MPC-based heating control in residential buildings. Essentially, the human-MPC interface provides the user with the necessary functionalities to understand parts of the optimization, however, conducting user tests is recommended to get a more comprehensive measure of the web application's capability to effectively communicate the MPC data.

According to this thesis, future smart house solutions will be able to integrate more complex controller designs into heat pumps, such as MPC, to regulate the temperature without compromising the understanding of the user, by implementing customized interfaces. To conclude the thesis,

developing a human-MPC interface is viable for explaining and visualizing important concepts related to the control algorithm.

8.1 Further work

The results obtained in this thesis serve as the basis for further developing a complete application for the POWIOT project. Recommendations for further development of the software and related work presented in this thesis are listed in the following.

- Further considerations for the MPC scheme:
  - Develop a schedule for setting hourly weights.
  - Having different cost functions for each room.
  - Include an energy storage system to provide better demand response/load shifting.
- Improve and generalize data extraction and processing on the server-side to:
  - Potentially include more IoT devices already installed in the smart house.
  - Have a completely remote configuration or utilize the Raspberry Pi.
  - Potentially develop a smartphone app to increase the interoperability.
  - A cloud-based solution for better up-scaling and security.
- Develop more interactive and customized functionalities to:
  - Improve the graphical display of MPC data.
  - Provide more intuitive explanations such as customized recommendations that exploit the predictive capabilities of the MPC scheme to a greater extent.

References

[1] Ján Drgoňa et. al. *All you need to know about model predictive control for buildings*. 2020. URL: *https://www.sciencedirect.com/science/article/pii/S1367578820300584* (visited on 03/03/2022).

[2] Energi Norge. *Strømnettet i et fullelektrisk Norge*. Oct. 31, 2019. URL: *https://www.energinorge.no/contentassets/74f33e5598d64578bda89c1fa864e83a/rapport---stromnettet-i-et-fullelektrisk-norge.pdf* (visited on 02/2022).

[3] Statistisk Sentralbyrå. *Elektrisitet*. 2020. URL: *https://www.ssb.no/statbank/table/08311/* (visited on 02/2022).

[4] E24. *Direktiv om endringer i bygningsenergidirektivet*. Aug. 28, 2021. URL: *https://www.regjeringen.no/no/sub/eos-notatbasen/notatene/2016/des/revisjon-av-direktiv-om-bygningers-energiytelse/id2540198/* (visited on 04/2022).

[5] E24. *Ny strømprisrekord i Sør-Norge mandag*. Dec. 19, 2021. URL: *https://e24.no/privatoekonomi/i/Qy3jMx/ny-stroemprisrekord-i-soer-norge-mandag* (visited on 01/2022).

[6] Gérôme Bovet et.al. *Toward Web Enhanced Building Automation Systems*. Mar. 2014. URL: *https://link.springer.com/chapter/10.1007/978-3-319-05029-4_11* (visited on 04/2022).

[7] Energifaktanorge. *Energy use by sector*. 2020. URL: *https://energifaktanorge.no/en/norsk-energibruk/energibruken-i-ulike-sektorer/* (visited on 01/2022).

[8] Bomiao Liang et al. *Economic MPC-Based Smart Home Scheduling With Comprehensive Load Types, Real-Time Tariffs, and Intermittent DERs*. 2020. URL: *https://ieeexplore.ieee.org/document/9237994* (visited on 01/2022).

[9] BEN. *Internet of Heat*. 2020. URL: *https://www.benuk.net/Internet-of-Things-Heat-Pump-Controls.html* (visited on 02/2022).

[10] NVE. *Rapporter - vassmagasinstatistikk*. 2021. URL: *https://www.nve.no/nytt-fra-nve/rapporter-vassmagasinstatistikk/* (visited on 01/2022).

[11] NordPool. *Price formation*. 2021. URL: *https://www.nordpoolgroup.com/the-power-market/Day-ahead-market/Price-formation/* (visited on 01/2022).

[12] Tensio. *Nettleiepriser privat fra 1. januar 2022 - 31. mars 2022*. 2022. URL: *https://ts.tensio.no/kunde/nettleie-priser-og-avtaler/2022-nettleie-privat* (visited on 03/2022).

[13] Regjeringen. *Justerer innføringen av ny nettleiemodell*. May 6, 2022. URL: *https://www.regjeringen.no/no/aktuelt/justerer-innforingen-av-ny-nettleiemodell/id2911788/* (visited on 05/20/2022).

[14] Bjarne Foss and Tor Aksel N. Heirung. *Merging Optimization and Control*. Trondheim, Norway: NTNU, 2016.

[15] *Tibber*. URL: *https://tibber.com/no* (visited on 02/2022).

[16] Sensibo. *Sensibo*. 2021. URL: *https://sensibo.com/* (visited on 01/2022).

[17] Mathew Kevin. *Heat Pump Thermostat – Guide to Buy The Right Thermostat*. Jan. 13, 2022. URL: *https://thermostatguide.com/heat-pump-thermostat/* (visited on 02/22/2022).

[18] Tibber. *Sensibo Air - Smart termostat*. 2022. URL: *https://tibber.com/no/store/produkt/sensibo-air* (visited on 04/20/2022).

[19] Pervez Hameed Shaik et al. *A review on optimized control systems for building energy and comfort management of smart sustainable buildings*. Jan. 31, 2020. URL: *https://www.sciencedirect.com/science/article/pii/S1364032114001889* (visited on 01/2011).

[20] Zakia Afroz et al. *Modeling techniques used in building HVAC control systems: A review*. 2018. URL: *https://www.sciencedirect.com/science/article/pii/S1364032117314193* (visited on 02/22/2022).

[21] Abdul Afram et al. *Theory and applications of HVAC control systems – A review of model predictive control (MPC)*. 2014. URL: *https://www.sciencedirect.com/science/article/pii/S0360132313003363* (visited on 02/22/2022).

[22] Raffaele Carli et al. *IoT Based Architecture for Model Predictive Control of HVAC Systems in Smart Buildings*. Jan. 31, 2020. URL: *https://www.researchgate.net/publication/338962263_IoT_Based_Architecture_for_Model_Predictive_Control_of_HVAC_Systems_in_Smart_Buildings* (visited on 02/2022).

[23] Rasmus Halvgaard et al. *Economic Model Predictive Control for building climate control in a Smart Grid*. 2017. URL: *https://ieeexplore.ieee.org/document/6175631* (visited on 03/20/2022).

[24] Gianni Bianchini et al. *An integrated MPC approach for demand-response heating and energy storage operation in smart buildings*. 2018. URL: *https://ieeexplore.ieee.org/document/8264228* (visited on 03/25/2022).

[25] Hjørdis Amanda Schlüter et al. *Economic Model Predictive Control for Energy Systems in Smart Homes*. 2019. URL: *https://ieeexplore.ieee.org/document/8920663* (visited on 04/25/2022).

[26] Wenda Li et. al. *Motivations, barriers and risks of smart home adoption: From systematic literature review to conceptual framework*. Oct. 2021. URL: *https://www.sciencedirect.com/science/article/pii/S2214629621003042* (visited on 04/20/2022).

[27] Diba Malekpour Koupaei et. al. *An Assessment of Opinions and Perceptions of Smart Thermostats using Aspect-Based Sentiment Analysis of Online Reviews*. Dec. 2019. URL: *https://www.researchgate.net/publication/338010221_An_Assessment_of_Opinions_and_Perceptions_of_Smart_Thermostats_using_Aspect-Based_Sentiment_Analysis_of_Online_Reviews* (visited on 03/30/2022).

[28] Unified Info Tech. *10 Smart Home Apps That'll Make Your Life Easier In 2021*. Sept. 2019. URL: *https://www.unifiedinfotech.net/blog/top-smart-home-apps/* (visited on 03/2022).

[29] Unified Info Tech. *Investing in Smart Home App Development? Here's Your Guide*. Oct. 2019. URL: *https://www.unifiedinfotech.net/blog/smart-home-app-development-guide/* (visited on 03/2022).

[30] Mobindustry. *How to Develop an App for the Internet of Things (IoT)*. Mar. 2021. URL: *https://www.mobindustry.net/blog/how-to-develop-an-app-for-the-internet-of-things-iot/* (visited on 03/2022).

[31] Majid Ghassemi et.al. *Chapter 3 - Biosystems Heat and Mass Transfer*. 2017. URL: *https://www.sciencedirect.com/science/article/pii/B9780128037799000030* (visited on 04/2022).

[32] Robin Wieruch. *How to create a REST API with Express.js in Node.js*. Apr. 24, 2020. URL: *https://www.robinwieruch.de/node-express-server-rest-api/* (visited on 03/20/2022).

[33] OpenBase. *Chokidar*. 2021. URL: *https://openbase.com/js/chokidar/documentation* (visited on 03/15/2022).

[34] ReactJS. *A JavaScript library for building user interfaces*. 2022. URL: *https://reactjs.org/* (visited on 02/2022).

[35] MDN Web Docs. *Using Fetch*. 2021. URL: *https://developer.mozilla.org/en-US/docs/Web/API/Fetch_API/Using_Fetch* (visited on 03/2022).

[36] ReactJS. *Introducing Hooks*. 2022. URL: *https://reactjs.org/docs/hooks-intro.html* (visited on 02/2022).

[37] Jagreet Kaur. *JavaScript Chart Libraries*. Sept. 18, 2020. URL: *https://www.xenonstack.com/blog/data-visualization-with-javascript* (visited on 03/2022).

[38] D3js. *D3.js - Data-Driven Documents*. 2022. URL: *https://d3js.org/* (visited on 03/2022).

[39] Eric Törn. *IoT Software for Smart Houses*. 2021. URL: *https://ntnuopen.ntnu.no/ntnu-xmlui/handle/11250/2828765* (visited on 01/2022).

Appendix A

Smart house system overview

A detailed system overview of the smart house located in Trondheim, Norway, is illustrated in figure A.1. This illustration provides information about the abstracted layers of the system in addition to all the relevant components for this thesis.

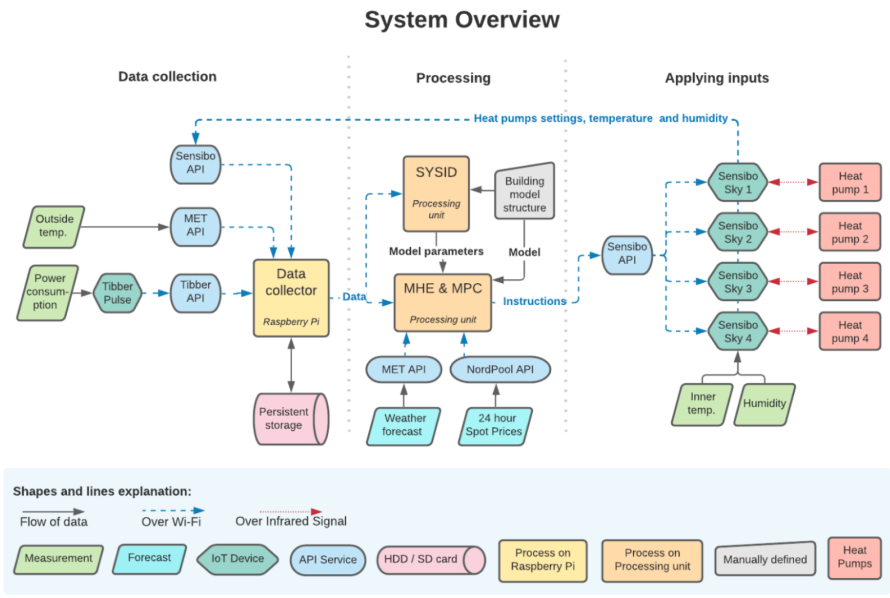


Figure A.1: A detailed flowchart of the smart house system and components. All arrows are described on the bottom of the figure [39].

Appendix B

JSON data structure on server-side

A snippet of the data structure on the server/REST API is provided in figure B.1. This data presents one sample, out of 576 samples, output from the MPC scheme on the *Main* room. This data is updated every 5 minutes and further sent to the client-side to perform data analysis to visualize the data.

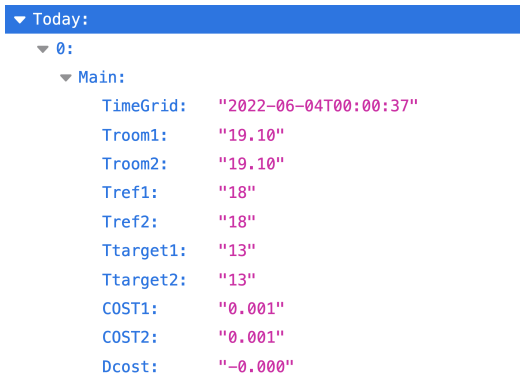
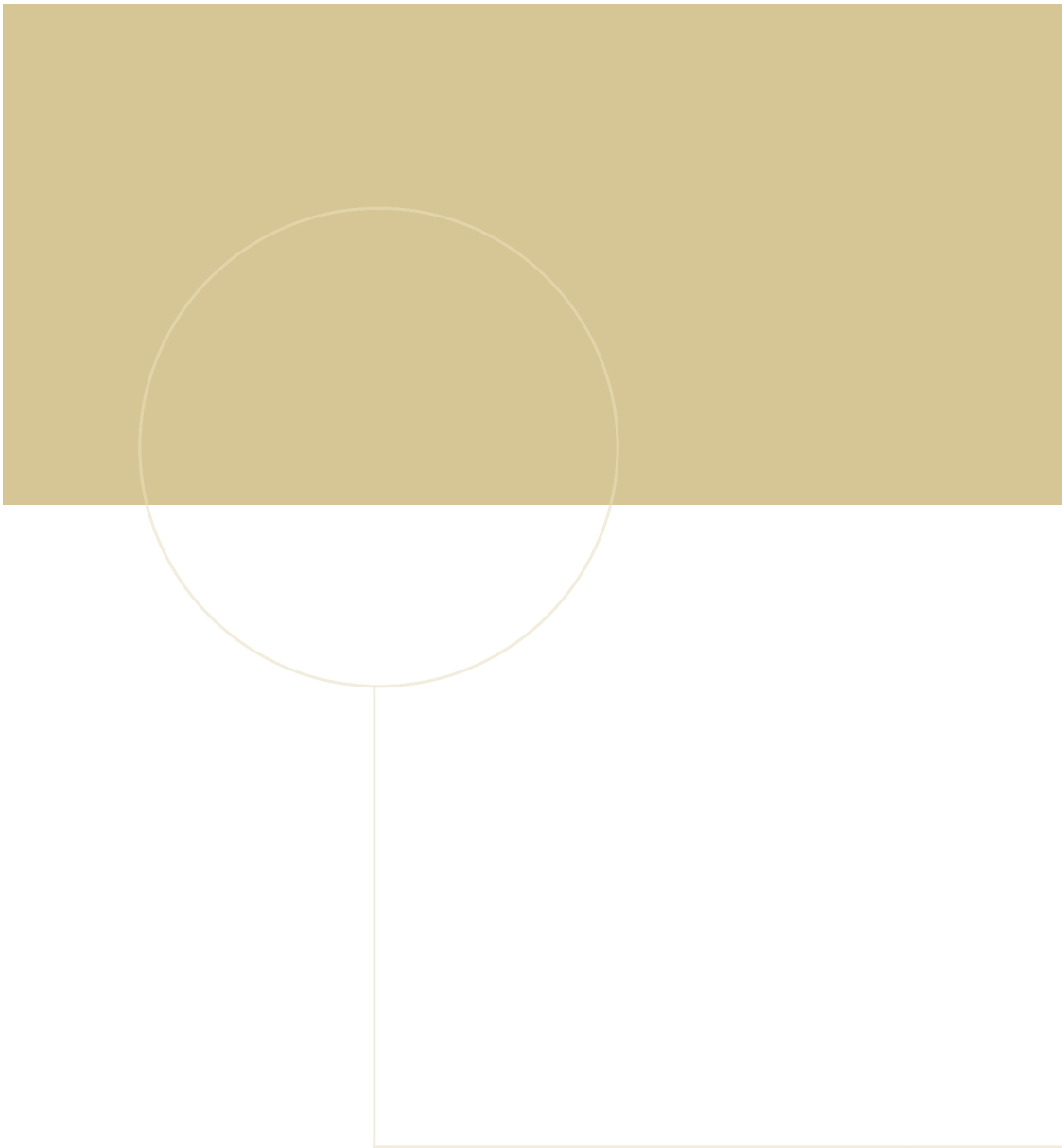


Figure B.1: JSON data structure on the server.



NTNU

Norwegian University of
Science and Technology