Bachelor's Thesis

Energy and Environmental Engineering

2023

Timon Papenfuß

Control and Monitoring System Design for a Thermal Compound Network

TURKU AMK
TURKU UNIVERSITY OF APPLIED SCIENCES

Bachelor’s Thesis | Abstract

Turku University of Applied Sciences

Energy and Environmental Engineering

72 pages

Timon Papenfuß

Control and Monitoring System Design for a Thermal Compound Network

This paper deals with the development and implementation of a control and monitoring system for a thermal compound network with long-term heat storage. The work represents a case study, an exemplary technical design and implementation in a facility in the municipality of Kaustinen, Finland. The study is being commissioned by the Finnish start-up Heliostorage Oy. The paper deals with the question of how to design an efficient, concise, and reliable control and monitoring system for a composite thermal system. While solar thermal collectors are used as a heat source, the borehole thermal energy storage acts as a storage unit. With this setup, a heat pump is supplied with a higher feed-in temperature, which increases the coefficient of performance. The microprocessor unit ATmega 2560 is used for embedded on-site control of the system and implements different communication protocols. The monitoring system consists of a FastAPI backend interface, a PostgreSQL database, and a Grafana user interface. Implementing the C++-based microprocessor results in a reliable and fast control system. The developed remote user interface can monitor and optimise the resulting heat network performance. In this way, the system can help to reduce the dependence on external energy supply by matching shifts in thermal energy demand with availability. This renewable thermal energy supply is a promising solution to reduce CO_2 emissions and is contributing to the transformation of heating systems.

Keywords:

Thermal Energy, BTES, Renewable Energy, Energy Efficiency, C++, Solar Energy, Control System, Embedded System

Content

Figures

Tables

Equations

Appendices

List of abbreviations and symbols

ADC	Analog Digital Converter
API	Application Programmable Interface
AVR	Advanced Virtual RISC
BHE	Borehole Heat Exchanger
BTES	Borehole Thermal Energy System
CAD	Computer-Aided Design
DB(MS)	Database (Management System)
FC	Function Code
HMI	Human Machine Interface
HP	Heat Pump
HTF	Heat Transfer Fluid
HTTP(S)	Hypertext Transfer Protocol (Secure)
IDE	Integrated Development Environment
IoT	Internet of Things
I^2C	Inter-Integrated Circuit
JSON	JavaScript Object Notation
PLC	Programmable Logic Controller
P&ID	Pipes & Instrumentation Diagram
REST	Representational State Transfer
RISC	Reduced Instruction Set Computing

RTC	Real Time Clock
RTD	Resistance Temperature Detector
RTU	Remote Terminal Unit
SHS	Sensible Heat Storage
SPI	Serial Peripherical Interface
SQL	Structured Query Language
TCP/IP	Transport Control Protocol/Internet Protocol
TMY	Typical Meteorological Year
TSO	Transmission System Operator
UI	User Interface
VPS	Virtual Private Server

1 Introduction

With the start of the Ukrainian war and the resulting energy crisis in Europe, the rising energy costs and the related economic dependency have created a desire for more energetic independence among companies and households. Especially with increasing the share of renewable energies and their fluctuating availability, storage systems are becoming an important part of the reliability of the current energy supply. Small-scale storage systems for balancing supply and demand on an hourly or daily basis are already being used more frequently, but long-term storage, especially for thermal energy, continues to be challenging. These systems are not yet widespread. To resolve this issue, thermal energy storage systems, such as borehole thermal energy storage (hereafter BTES), can be installed to store the thermal energy over a longer period and help to reduce the external heat energy demand (Alva et al., 2018; Miocic et al., 2022.)

To efficiently store and retrieve energy in such a system, a control system is needed that centrally manages and communicates with the installed components. There is a need to improve overall system performance and manage the charge and discharge process. Sensible setpoints and assumptions should be made in order to enhance the control (Ridder et al., 2011.)

For real-time analysis of the system performance, data must be stored and processed. For this purpose, a structure must be set up to manage this data transmission and processing, so communication with the controller can be automated.

To bring forward this field of study, in this bachelor's thesis the following research questions are covered:

- What are the main components of an efficient thermal network and what is their purpose?
- How can a control system combine these thermal devices?
- **How should a scalable and efficient system for controlling and monitoring a thermal network be designed in practice?**

In this bachelor thesis, the implementation of a control and monitoring system for a thermal compound network is discussed and evaluated based on a case study. In the first part, the general layout of the underlying hardware is discussed. This includes the thermal system components included and the electrical connections used for the microcontroller of the unit. Based on this, the development of the control system and the backend connection to the server and the monitoring unit is described. The implemented system is then discussed and a further outlook for future installations and scalability is given.

2 Methodology

The development of the new control and monitoring design begins with an initial investigation of the current scientific and practical state of the art. This consists of two key components: firstly, an examination of the theoretical background. This includes the fundamental principles of thermal systems and, more specifically, the operation of the BTES, as well as the theoretical fundamentals of interfaces and programming. This phase involves a comprehensive review of the available literature on the digital university library and the available online literature with the keywords *BTES, control system, thermal compound network,* and *data processing.* Secondly, the current hardware configuration of the existing thermal compounding system is evaluated. This includes an analysis of the pipes & instrumentation diagram (P&ID) and the related computer-aided design (CAD) drawings made by Heliostorage, as well as an in-depth inspection and assessment of the system on-site.

Based on the knowledge and understanding gained from the first phase, the key components required to complete the case study are identified. These include three key areas: software/controller development, associated backend development, and finally the practical implementation and integration of the new system. During this phase, the information gained from the literature review is integrated to create a novel system design. This work thus closes the recognised gap between the theoretical elaboration of controller designs and the practical implementation of systems in remote operation.

The final stage of the study involves a plausibility check of the new system and an analysis of the implementation results. This involves comparison and evaluation of the system performance data collected, along with a discussion of the limitations and further extension possibilities that have emerged during the course of the study.

3 Thermal System Background

3.1 System Layout

The case study system includes a BTES, a heat pump, a heat exchanger, a geothermal system, a solar thermal system, and buffer tanks. The system provides heating for an institutional building using stored thermal energy from the ground, solar energy, and a heat pump. The main system components, on which the commissioner is focusing, the BTES, the solar thermal collectors, and the heat pump will be discussed in the following chapters. Figure 1 shows the simplified thermal layout of the case study system in Kaustinen (Finland) with its interconnections. The main system components, which are described in the following are highlighted. Additional insight can be retrieved from Appendix 2, in which the case study system components can be seen in pictures.

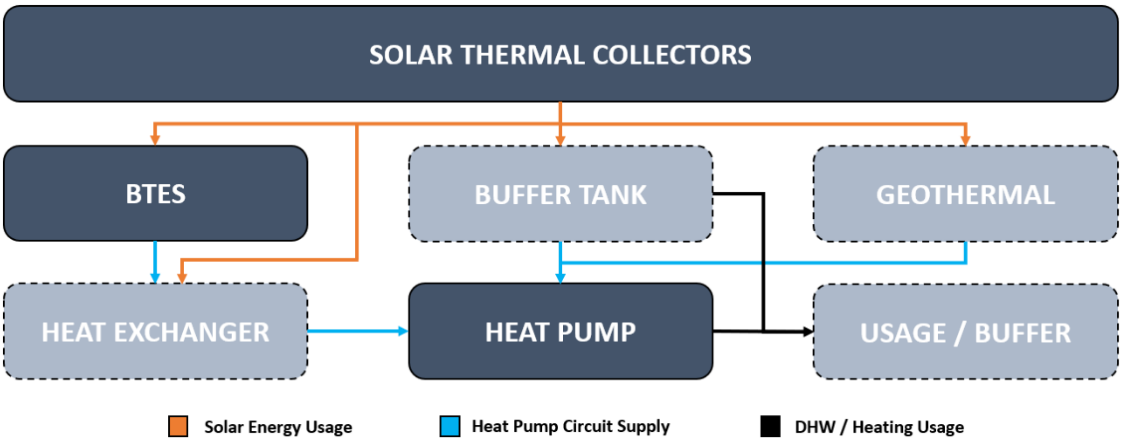


Figure 1. Simplified case study system layout and its heat flows.

3.2 Borehole Thermal Energy Storage

The key component of the assessed thermal compound system is the BTES. This system component describes a promising technology for long-term energy storage (Pavlov and Olesen, 2012; Rad and Fung, 2016). The BTES concept consists of several borehole heat exchangers (hereafter BHE) that can be used to transfer heat from the heat transfer fluid (hereafter HTF) to the ground as a storage medium and vice versa.

The BTES can help to shift excess energy supply to periods of higher demand, especially when combined with renewable energy supply systems or low-temperature waste heat, i.e. from periods of thermal surplus. For example, renewable energy from solar thermal collectors can be used in the summer to heat the ground around the boreholes. To accomplish this, the HTF heated by the solar collectors is passed through one or more BHE, which then release the heat into the ground. In winter, or more precisely during the heating period, this heat can then be used to heat the HTF again.

The BHE typically consists of a borehole with a depth between 30 m and 100 m, depending on the application. A U-, double-U- or concentric pipe is used inside the borehole. Inside these pipes the HTF can flow. To improve heat transfer to the outside of the borehole, grout is used to fill up the remaining space. On top of the BTES, there is a layer of heat insulation and a layer of soil to minimise heat loss through the surface of the ground. Its illustration can be seen in Figure 2.

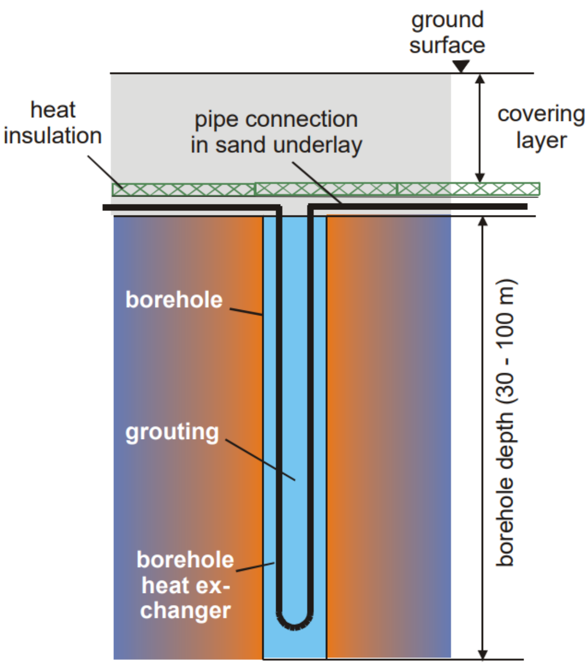


Figure 2. Borehole illustration (Mangold et al., 2003).

For the case study facility in Kaustinen, 37 boreholes are used, which are connected in three rings. In contrast to the common design, the BHEs only have a depth of around nine meters, which results in a storage volume of approximately $V_{ground} \approx 3960\ m^3$. From that a heat storage capacity of $Q = V_{ground} \cdot c_{p,ground} \cdot (T_{max} - T_{\min}) = 77\ MWh$ can be calculated, assuming a common heat capacity of $2\ MJ/(m^3 \cdot K)$ for the surrounding soil and a maximum temperature of $T_{max} =$

$40\ °C$ (Skarphagen et al., 2019). This is a usual temperature for low-temperature BTES (Rad and Fung, 2016).

With such BTES, an average efficiency of $\eta = 40\text{-}60\%$ can be achieved when fully charged. However, this is dependent on many factors such as soil quality, groundwater behaviour, borehole layout, or storage time (Rad and Fung, 2016.) In normal usage, an efficiency of 40% is hard to exceed (Skarphagen et al., 2019). In the case of this rather small storage volume in Kaustinen, an even higher loss factor can be assumed since the storage volume is smaller than in regular BTES.

The BTES works as sensible heat storage (hereafter SHS). Sensible means that the temperature rise of the storage medium is proportional to the energy input, as long as the second law of thermodynamics complies. There is therefore no phase change, which would indicate latent heat storage. These behaviours can be seen in Figure 3. The sensible storage properties result in the BTES having to be fed at a proportionally higher temperature as the state of charge of the BTES increases. This is particularly problematic in the late summer months when the BTES is already highly charged and the weather and decreasing radiation intensity no longer allow such high temperatures. To still maintain this working principle, the heat source, which in this case are solar thermal collectors, must supply the SHS with gradually higher temperature. It is necessary to control the BTES feeding temperature.

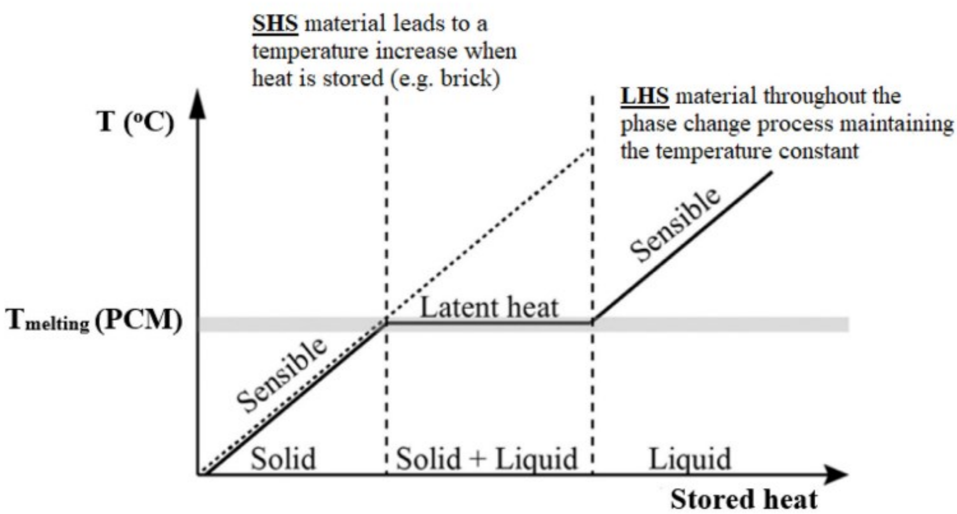


Figure 3. BTES as sensible heat storage (Sadeghi, 2022).

As the temperature of the storage tank rises, the losses from the storage tank itself increase. This is because the storage temperature rises disproportionately to the ambient temperature. This increases the temperature difference between the storage medium and the surrounding ground and this is the determining factor in the loss term in Equation 1. This equation describes a simplified assumption of a cylindrical storage medium that assumes heat conduction to the adjacent soil layers and upwards and downwards as losses.

Equation 1. Heat loss equation (Nilsson and Rohdin, 2019).

$$\dot{Q}_{loss} = \underbrace{2\pi k L \frac{T_2 - T_1}{\ln(r_2/r_1)}}_{\text{cylindrical}} + \underbrace{A \frac{T_2 - T_1}{\Delta x_1/\lambda_1} + A \frac{T_2 - T_1}{\Delta x_2/\lambda_2}}_{\text{plane}} \tag{1}$$

where

$\dot{Q}$	heat transfer rate $[W]$
k	ground conductivity factor $\left[\frac{W}{m \cdot K}\right]$
L	borehole length $[m]$
$r_1; r_2$	inner and outer radius $[m]$
$T_1; T_2$	temperature at inner and outer radius $[K]$

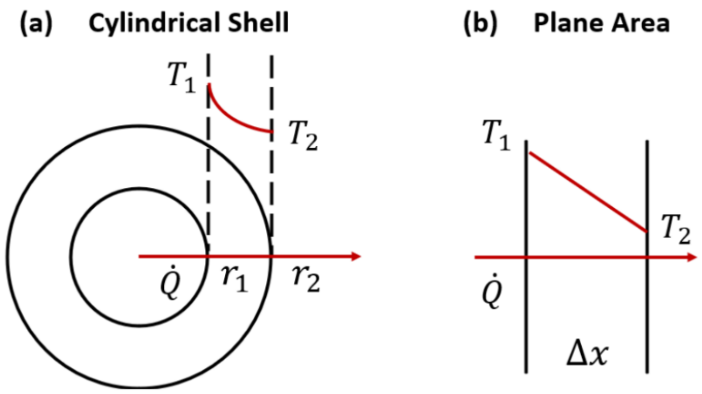


Figure 4. Heat conduction temperature gradients.

From that equation, it can be determined that the cylindrical shape of the storage helps to minimise losses in comparison to other shapes because the outer wall area gets reduced. With a constant ground conductivity factor k, other shapes would increase the energy losses of the BTES. While a spheric layout would provide the optimal storage volume-to-surface ratio r, it is not practically feasible, making a cylinder with a 1:1 depth-to-diameter ratio an optimal choice (Skarphagen et al., 2019). Whereas in practical implementation a hexagonal shape is used (Villa, 2020). The provided equation also shows the logarithmic temperature gradient from the outer wall of the tank to the surrounding ground temperature. This temperature gradient is shown in Figure 4.

In terms of the optimal storage method, it can therefore be said that the outer wall temperature of the storage tank should be kept as low as possible to minimise losses. In order to achieve an acceptable storage temperature that can also be used in the heat pump or for direct use, this warmer temperature can be maintained in the centre of the BTES, causing the temperature to gradually decrease towards the outside. This is where the design and connection of the BHE come into the picture. In this case, the already mentioned ring design was chosen. While the heat is charged from the inner rings of the BTES, it is dissipated from the outside to maintain a high internal temperature.

3.3 Solar Thermal System

Solar thermal collectors are an effective way of collecting the sun's energy and converting it into usable heat. In this case study, a solar thermal collector system on an east and west-facing roof is used. It has an inclination angle of 35° and a solar collector area of $A_{coll} = 260\ m^2$ on each side. The first step in understanding how this system works is to estimate the amount of solar irradiation available.

The PVGIS typical meteorological year (hereafter TMY) data can be used for the estimation of solar radiation within one year (Joint Research Centre, 2016). This data provides information on the typical weather conditions in Kaustinen (63° 32′ N, 23° 42′ E), including the amount of solar irradiation that can be expected throughout a typical year on a horizontal plane. Analysing this data provides a better understanding of how much energy can be collected by the solar thermal collector system. From the TMY the global horizontal irradiation ($E_{G,hor}$) and the diffuse horizontal irradiation ($E_{diff,hor}$) can be extracted. From the correlation shown in Equation 2 also the direct horizontal irradiation ($E_{dir,hor}$) can be calculated which is needed in the further course of calculations.

Equation 2. Global horizontal irradiation composition (Quaschning, 1996).

$$E_{G,hor} = E_{diff,hor} + E_{dir,hor} \tag{2}$$

The solar irradiation on the east and west-facing roof surfaces can be calculated using an equation that converts the solar irradiation on a horizontal plane to the irradiation on the tilted plane. This equation takes into account the sun height and the azimuth (Quaschning, 1996). The resulting irradiation values can further be used to provide an overview of the power generated by the solar thermal system. The used Equation 3 and explanatory Figure 5can be seen in the following.

Equation 3. Global solar irradiation on tilted plane (Quaschning, 1996).

$$E_{G,tilt} = E_{dir,hor} \cdot \frac{\cos(\theta_{tilt})}{\sin(\gamma_s)} + E_{diff,hor} \cdot \frac{1}{2} \cdot (1 + \cos(\theta_{tilt})) + E_{G,hor} \cdot A \cdot (1 - \cos(\gamma_E)) \tag{3}$$

where

E	irradiation $[W/m^2]$
θ_{tilt}	inclination angle [°]
γ_s	sun height [°]
γ_E	collector tilt angle [°]
$G, dir, diff$	index for global, direct, diffuse irradiation
$hor, tilt$	index for horizontal & tilted plane

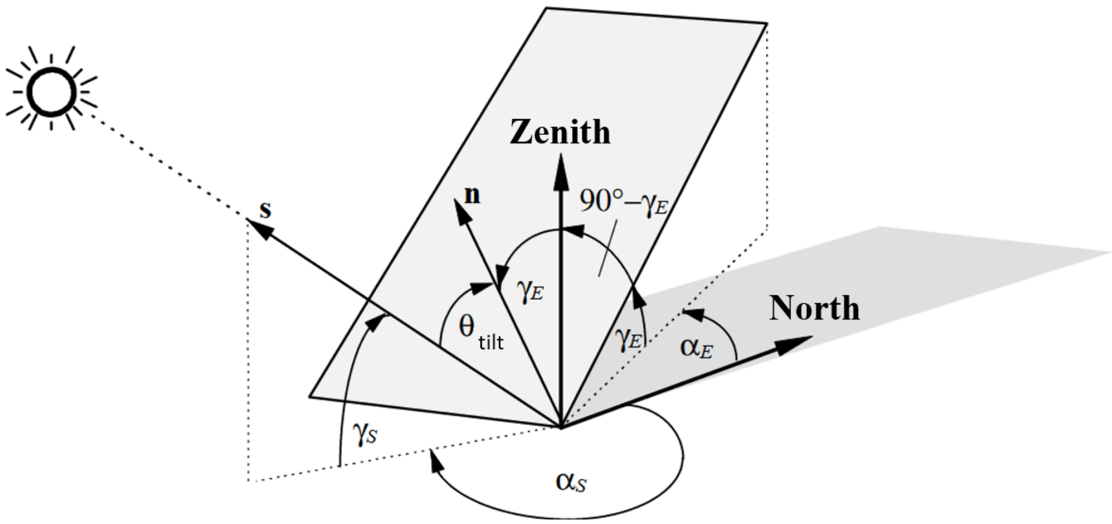


Figure 5. Angular correlation for irradiation on tilted plane (Quaschning, 1996).

As a result of the analysis, the course of the solar irradiation over the described TMY is illustrated in Figure 6. Based on these values, during a whole year, a total amount of 551 MWh of solar radiation is reaching both roofs combined. In the next step, the system efficiency should be taken into account. However, the available solar energy is much greater than the available storage capacity of 77 MWh, as can be seen from this dimension. This is particularly evident in the significantly reduced solar output in winter, while a surplus is expected in summer due to reduced heating demand in combination with high irradiation intensities.

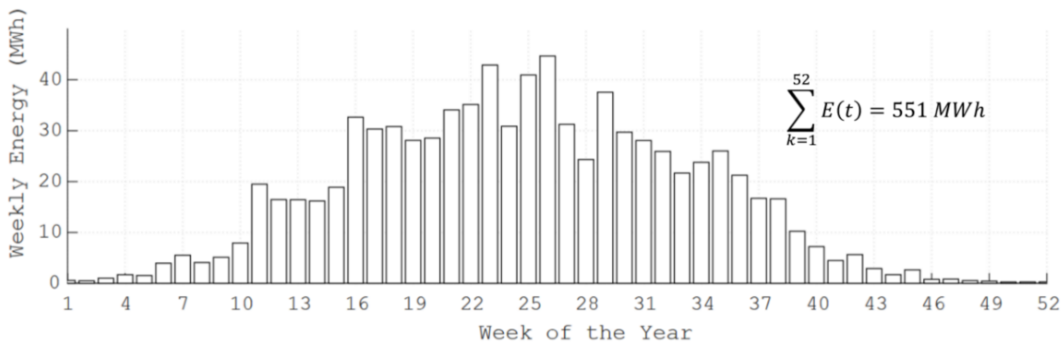


Figure 6. Weekly energy available for solar collectors from TMY.

As mentioned before, one consideration when designing a solar thermal collector system is the efficiency of the collectors themselves. In this case, a patented design of Heliostorage Oy is used that includes medium-efficiency collectors without vacuum tubes. This means that with a high temperature difference between the solar collector and the outdoor temperature, there would be comparatively high losses in northern countries with cold climates (Buzás et al., 1997). However, due to the cost-effective layout of the roof and the large surface area that is covered by the design, a significant amount of power can be obtained (Kalogirou, 2004.)

The deviation of air temperature and average collector temperature dT is a decisive factor in the loss equation of solar thermal collectors. Conduction and convection losses are proportional to the second power of dT according to the equations used by Kalogirou in 2004 (Kalogirou, 2004.) In addition, according to Stefan-Boltzmann law, radiation losses are proportional to the 4th power of its absolute temperature (Wellons, 2007). Thus it is necessary to keep these parameters as low as possible. In a simplified solar thermal heat supply equation, Equation 4, a simplified efficiency factor is used. So conduction and convection losses, which are represented by c_1 can be assumed to be proportional to dT, while the radiation losses which are represented in c_2 are assumed to be proportional to the second power of dT. This equation can be used for a temperature difference of up to $120\ K$ (Eismann, 2017.)

Equation 4. Solar collector heat flow with semiempirical efficiency equation for $\Delta T < 120\ K$ (Eismann, 2017; Kalogirou, 2004).

$$\dot{Q}_S = E_G \cdot A_{coll} \cdot \left(\eta_0 - \frac{c_1 \cdot \Delta T}{E_G} - \frac{c_2 \cdot \Delta T^2}{E_G}\right) \tag{4}$$

where

η_0	optical efficiency $[-]$
$c_1; c_2$	efficiency factors $\left[\frac{W}{m^2 \cdot K}; \frac{W}{m^2 \cdot K^2}\right]$
$\Delta T = \left(\frac{T_{out} - T_{in}}{2} - T_{outdoor}\right)$	temperature difference $[K]$
E_G	global irradiation on tilted plane $\left[\frac{W}{m^2}\right]$

The pump control design must therefore be adapted to keep the average roof temperature as low as possible while still allowing good heat transfer to the BTES and feeding the heat pump with adequate input temperature. To keep thermal losses as low as possible, it is necessary to control the output temperature of the solar thermal system. This aspect is discussed in Chapter 3.3.1. In all cases, there are higher losses with higher temperature differences to the outside. This can also be seen in an exemplary efficiency curve of solar thermal collectors in Figure 7. With increasing temperature difference between the solar collector cover and absorber, the correlated absorber efficiency drops significantly. Therefore it is important to keep the temperature difference at a minimum value while still allowing a heat transfer in the BHE and not getting in conflict with the second law of thermodynamics. With that strategy, the efficiency of the collector absorber can be kept in the upper part of the efficiency range.

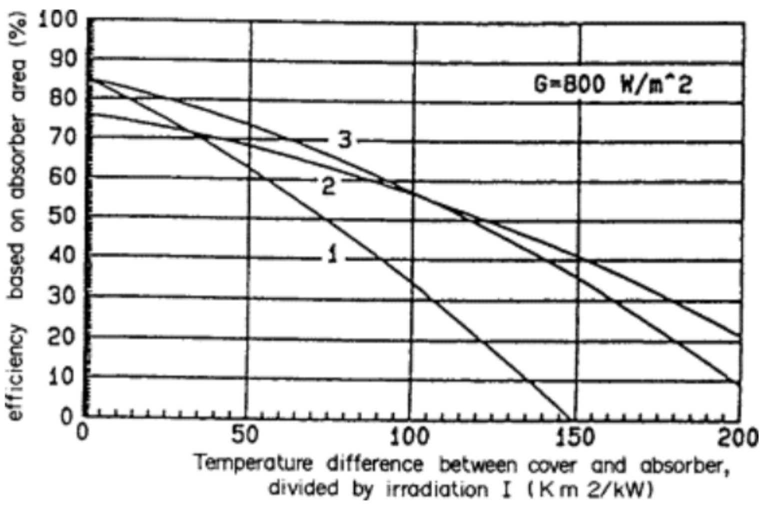


Figure 7. Solar thermal collector efficiency (Granqvist, 1991).

The indicated efficiency can also be optimised by interconnecting the different roofs to achieve the most suitable output temperature. For this purpose, the east and west roofs of the case study system can be connected differently. While a serial connection results in a higher output temperature, a parallel connection of the two roof surfaces results in a higher power with a lower maximum temperature. However, this interconnection is mostly necessary for south irradiation, as otherwise the east and west surfaces are not irradiated at the same time. Nevertheless, the less irradiated roof side can still be used to preheat the HTF for the hotter side. So it's sensible to use the east roof in the morning when the sun shines from the east and the west roof when the sun is setting in the west. Of course, certain threshold values must be adhered to, which are necessary for determining the usage of the solar thermal roof.

3.3.1 Pump Control System

In order to have an understanding of how the system should be controlled, a closer look at control theory is necessary. One specific control loop that can be analyzed is the regulation of the output temperature of the solar thermal system in charging mode. There are various situations in which it is necessary to regulate the temperature leaving the solar field, such as ensuring that the solar return temperature is higher than the temperature of the BTES or storage tank explained in the previous chapter. The mass flow rate of the fluid has the highest influence on this parameter. The mass flow rate is directly dependent on the pump speed. The parameter which can be used for managing is the pump speed in per cent, which has to be sent as the setpoint to the pump and therefore can be used to optimize the system (Hrisca and Melis, 2018.)

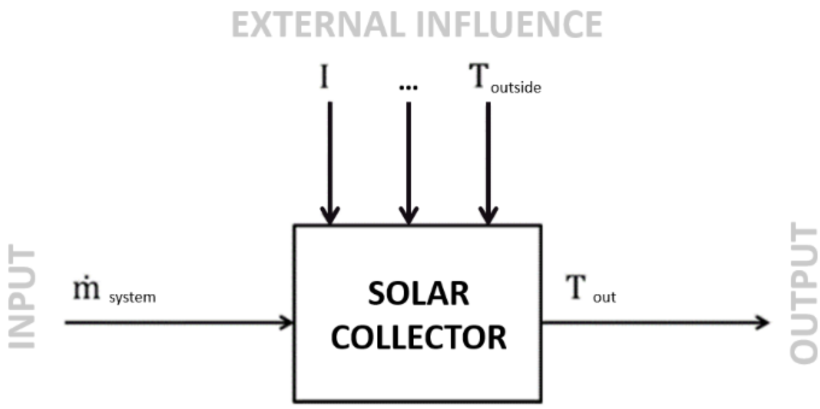


Figure 8. Simplified control parameters (based on Kannaiyan et al., 2020).

Designing the control system for this pump control is challenging due to the long dead time in the system. Dead time describes the delay between the change in input parameters and the adaptation in output parameters which is caused by factors such as thermal mass and fluid flow rate. The long pipes through which the HTF flows cause a time delay in these parameters.

However, to design an effective control system for this network, it is necessary to consider the various components within the control loop, which include the control variable, the setpoint, the controller, the disturbance variables, and the comparator. The feedback loop with its components is depicted in Figure 9. The control variable is the variable that is being controlled. The controller is the component that calculates the control action based on the difference between the setpoint and the measured value. The disturbance variable is any external variable that affects the performance of the system, such as changes in the irradiance, the outside temperature, the optical efficiency or the input temperature. The comparator compares the setpoint and the measured value of the control variable and sends the error signal to the controller.

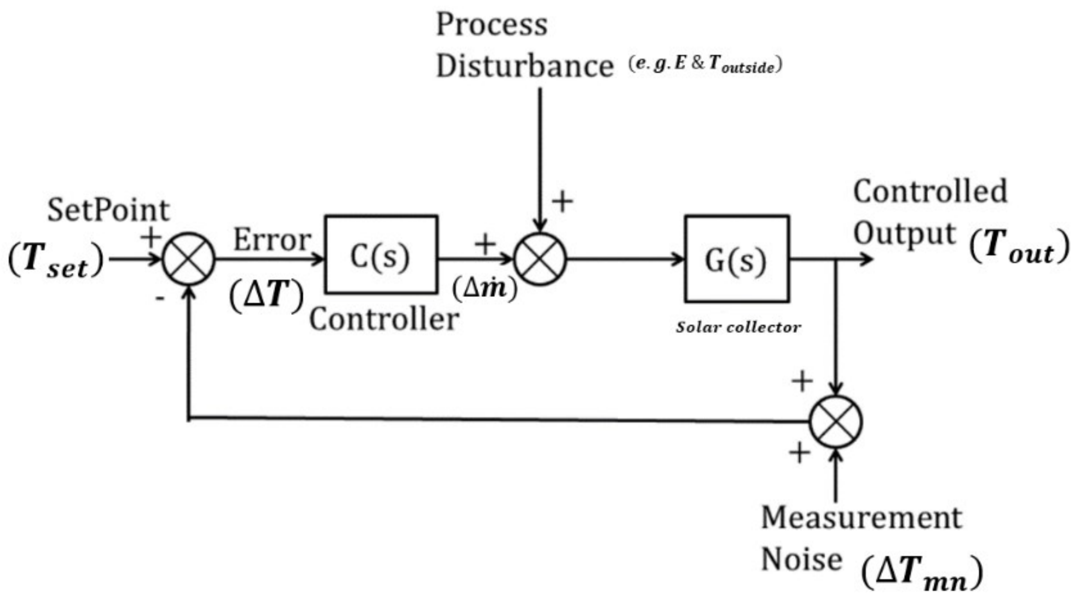


Figure 9. Control circuit diagram (Kannaiyan et al., 2020).

In the system under study, the control variable is the flow rate of the fluid that circulates through the system. The controller is a proportional-integral (PI) controller, which calculates the control action based on the error signal to the setpoint. The control mechanism is based on a paper from Kannaiyan et al.. It is a PI controller based on a simple power balance of the solar field. The PI controller is commonly used in control systems with dead time because it can account for the delayed response of the system. One example of the disturbance variable is the ambient temperature. The change in this external condition can affect the efficiency of the system as discussed in Chapter 3.3 about the solar thermal collectors. The comparator is located inside the control board where the sensor that measures the temperature of the HTF compares it to the setpoint temperature. The setpoint temperature in charging mode thereby is dependent on the BTES temperature to maintain a certain thermal temperature difference between the solar thermal output and the BTES temperature.

The control circuit of a solar thermal system can be designed using a feedback control loop. The performance of the control system can be improved by tuning the parameters of the PI controller based on the characteristics of the system.

3.4 Heat Pump

The heat pump in this thermal compound network is used to raise the temperature level from the low-temperature BTES or the geothermal supply of up to $T_{max,BTES} = 40°C$ to a higher level that corresponds to the use of domestic hot water (hereafter DHW), which is commonly around 60 °C. This is achieved by the thermodynamic vapor-compression cycle inside the heat pump. The isentropic compression in the compressor inside the idealised heat pump cycle increases the temperature level. For this, mechanical work has to be used, which in this case is generated by an electrical motor. The resulting temperature increase can be seen in the T-S-Diagram from states 1 to 2 in Figure 10. The additional work which has to be used is related to the temperature difference which has to be overcome (Sarbu and Sebarchievici, 2014.)

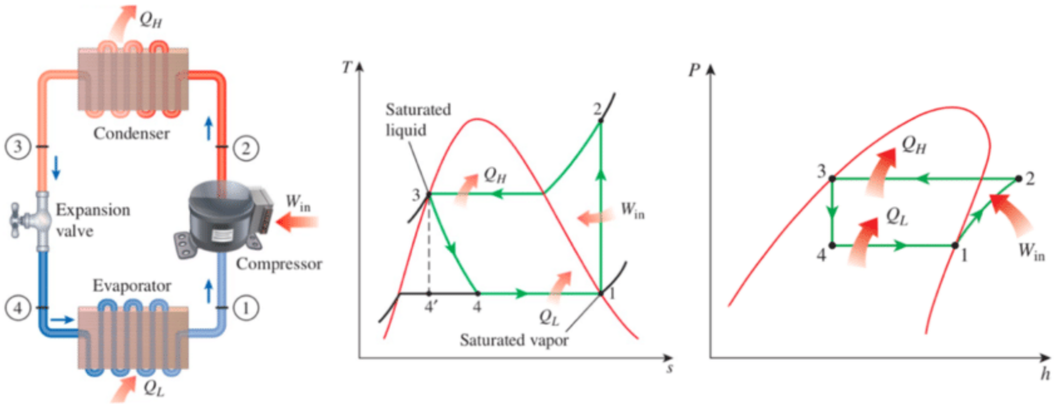


Figure 10. Ideal thermodynamic vapor-compression cycle (Edward Xu, 2018).

The most important aspect to reduce external dependencies of the facility is to reduce the electrical energy demand of the heat pump. The main part is formed by the mechanical work which has to be generated by the compressor. To reduce this, the heat pump brine-in side has to be fed with high temperature while avoiding a heat pump limp mode, which can be caused by a too high temperature input.

With this approach, the coefficient of performance (hereafter COP) of the heat pump can be increased, which is related to the characteristic curve of the COP in relation to the acquired temperature difference. The mathematical definition can be seen in Equation 5 and the disproportional change of the COP in comparison to the heat pump input and output temperature is shown in Figure 11.

Equation 5. Heat pump COP (Sarbu and Sebarchievici, 2014).

$$COP = \frac{E_U}{E_D} = \frac{\dot{Q}_{HP}}{P_{el}} \tag{5}$$

where

E_U	useful thermal energy $[kWh]$
E_D	drive energy $[kWh]$
$\dot{Q}_{HP}$	thermal heat pump output $[kW]$
P_{el}	electrical power $[kW]$

The ideal reversed Carnot cycle COP increases disproportionately with increasing evaporator inlet temperature. This concept makes it additionally important to provide the heat pump with a higher input temperature. For example, rising the input temperature from 0°C to 20°C with an output temperature of 60 °C increases the COP by $\Delta COP > 2{,}5$ from $COP_{(T_{cold}=0,\,T_{hot}=60)} = 5{,}6$ to $COP_{(T_{cold}=20,\,T_{hot}=60)} = 8{,}3$ (Zehnder, 2004). With the limits from the reversed Carnot cycle as the reversible border process, the COP of the heat pump can be calculated from Equation 6.

Equation 6: Reversed Carnot cycle efficiency.

$$COP_{rev,Carnot} = \frac{T_H}{T_H - T_L} \tag{6}$$

where

T_H	high output temperature
T_L	low input temperature

A visual representation of this thermodynamic relationship is shown in Figure 11. It shows the maximum possible COP as a function of the input temperature and the required output temperature for DHW or heating application.

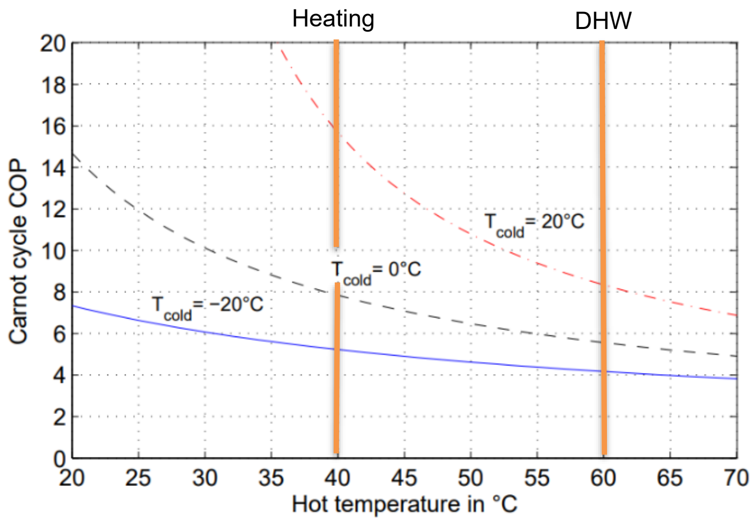


Figure 11. Heating COP for ideal HP cycle in relation to ΔT (Zehnder, 2004).

3.5 Additional System Components

3.5.1 Heat Exchanger

While the solar thermal circuit must use an antifreeze component to avoid freezing in Finland's cold winters, the heat pump input can't use the same fluid. Therefore a separate cycle has to be used for the roof and heat pump circulation. These two flow cycles are connected by a heat exchanger, which transfers the heat if there is a flow on both sides of the heat exchanger. Although it is an important component, the exact calculation is not further discussed. But it can be assumed that with the increasing flow on both sides, the heat transfer also increases. The heat transfer coefficients are based on Prandtl, Nusselt and Reynolds numbers (Böckh, 2014.)

3.5.2 Geothermal Boreholes

As an additional component inside the heat pump flow cycle, geothermal boreholes are installed. But because it is not the common layout used by the commissioner, it is not further discussed in this paper. Nevertheless, the control system has to take this into account. These boreholes mainly are designed as a backup system. The size of this heat source is not big enough for the usage of only these boreholes in the long term, so additional heat has to be supplied by the solar thermal and BTES flow cycle.

3.5.3 Buffer Tank

Two buffer tanks are installed for short-term deviations or fluctuations in the solar collector output. One is located upstream of the heat pump and the other is used to preheat the main DHW cylinder. The purpose of these tanks is to keep the solar energy and the heat pump more independent of each other. In this way, the buffer tank can be supplied with energy when the heat pump is not running, and the heat pump can be supplied with heat indirectly when the sun is not shining.

This configuration ensures that the heat pump is supplied with a high enough temperature to avoid high electricity demands with low brine-in temperatures.

3.6 System as a Network

To combine all these described system components, the control board is used to manage the flow between them and set valves according to the desired interconnection. For leading the flow to the most efficient components a priority list is used.

In the charging system status, the solar thermal collectors act as the heat source and the other system components, namely BTES, buffer tank, heat pump and geothermal boreholes act as heat sinks. Since the heat supplied to the buffer tank is directly used, it is the most efficient heat usage. This implies compliance with the required minimum temperature for usage. Besides the direct supply to the buffer tank, the supply to the heat pump, if active, has priority. This is because if the heat pump can be fed with a higher temperature, this also means an increase in COP and a reduction in the external power supply. This principle was described in Chapter 3.4 about the Heat Pump. Heat is then supplied to the BTES and to the geothermal boreholes. This implies a prioritization according to their efficiency terms. This finalises in the following structure for charging, discharging and solar discharging condition. If the heat supply or demand exceeds the first of these heat sources or heat sinks, the next component is connected in series or parallel to provide a higher power or temperature.

Heat sinks for charging status (Solar thermal collectors as heat source):

1. Buffer Tank / Heat Pump
2. BTES (Ring 1 → Ring 3)
3. Geothermal Boreholes

Heat sources for discharging status (Heat pump as heat sink; no irradiation available):

1. Buffer tank
2. Geothermal Boreholes
3. BTES (Ring 3 → Ring 1)

Heat sources for solar discharging status (Heat pump as heat sink; irradiation available):

1. Buffer tank
2. Geothermal Boreholes
3. Solar Collectors
4. BTES (Ring 3 → Ring 1)

4 Electrical and Communication Background

The control and monitoring system is highly dependent on the electrical hardware used and the communication interfaces available. So, for developing a system controller, which interacts with the devices and manages the facility, the existing hardware has to be examined.

4.1 Microcontroller Unit

As the heart of the controller system, the microcontroller processor ATmega 2560 is used. This processor is implemented on a self-designed printed circuit board (hereafter PCB) from a previous thesis by Bäck (Bäck, 2020). The microcontroller has the following specifications, which can be seen in Table 1. Staying inside these specifications is crucial for the further development of the system.

Table 1. ATmega 2560 based microcontroller specifications.

SPECIFICATION	VALUE	EXPLANATION
RAM	256 kB	Regular program memory, slow, big, read-only
FLASH	8 kB	Short-term data memory, smaller, read/write
EEPROM	4 kB	Independent memory, small, read/write
SERIAL PORTS	4	Serial connections to other devices
I/O PINS	86	Analog/digital pins for external communication

These specifications impose certain constraints on the controller layout. In particular, the flash memory of 8 kB is very limited. However, this control board allows a very efficient, fast and reliable solution. This is also due to the Harvard memory architecture. This architecture describes the distinction between program and data memory. A dual data bus system is used to access memory allowing the processor to access both program and data memory simultaneously. This can also be seen in Appendix 1, which shows the complete microprocessor layout.

The microprocessor used is based on the AVR structure, which is an acronym for advanced virtual RISC (Reduced Instruction Set Computing). The RISC architecture offers several advantages, including simplicity and efficiency by using a small set of simple and limited instructions. This allows the processor to run at high speed and without complex instructions, which in turn reduces power consumption. This makes the implemented hardware suitable for real-time data processing applications and IoT devices that provide low power consumption and fast processing (Nayyar and Puri, 2016.)

The ATmega 2560 also allows the use of the hybrid C/C++ and Arduino environment. The environment offers a wide range of open-source libraries, various implementation examples and extensive literature on its use, which makes programming the system particularly fast-forward. The C/C++ language, on which the processor is based, is a low-level programming language. This means that programming is closely linked to the hardware and based on the resources available. This makes the programs very fast and reliable, but on the other hand, there are more aspects to consider when programming, namely the memory location, the variable types, the available interfaces and other hardware dependencies.

4.2 System Interaction

The system must be designed to respond to changing environmental conditions, such as irradiance intensity or outside temperature. This requires the ability to access these conditions and adjust internal parameters accordingly. For example, the most important parameter is the system state, which controls the general flow in the circuit. The system state is depending on the roof temperature and the outside temperature. This means that certain conditions must be defined to change the flow and the state of the main system valves to direct the energy flow from the roof in the right direction. For example, to charge or discharge the BTES.

To adapt to changing conditions from the outside, intelligent decision-making based on sensors must be used. This allows real-time adjustments to be made. The sensor values need to be analysed and interpreted to determine the appropriate system status and other parameters. The system must be able to make decisions on its own based on the sensor data it receives.

To ensure the safety of the system, measures have been implemented in case the decisions made by the system are based on incorrect data or in case of a sensor fault. This ensures that the system responds appropriately to changing conditions and continues to operate effectively in the long term.

4.3 Control Interfaces

To achieve the mentioned behaviour described in Chapter 4.2, two different aspects are required. On the one hand, measuring instruments are needed. These help to generate data that represent the internal and external conditions. And on the other hand, there is a need for interfaces to regulate the behaviour of the devices based on the sensor data. To get an initial overview of the connected system instruments and controllable devices, they are presented in Table 2.

Table 2. PLC-connected instruments and devices.

INSTRUMENT	COUNT
Temperature Sensor	14
Flow Sensor	1
Pressure Sensor	1
Heat demand Sensor (Heat Pump [ON/OFF])	1

DEVICE	COUNT
Valve	16
Pump	2
PID Controller	1
Display (HMI)	1
Server	1
Signal LED	2

To communicate with these instruments and devices using only one processor unit, the processor acts as a master control unit. This means that all the decisions are made in the microprocessor unit and then sent to the peripherals. This architecture is called controller/peripheral architecture, also formerly known as master/slave layout. This is common for automation and control systems in Industry 3.0, which represents the use of a central control unit, to manage processes and facilities. This hierarchy can be seen in Figure 12, since there is used the ATmega 2560 as the main controller which sends and receives different signals to and from all the peripherals and devices.

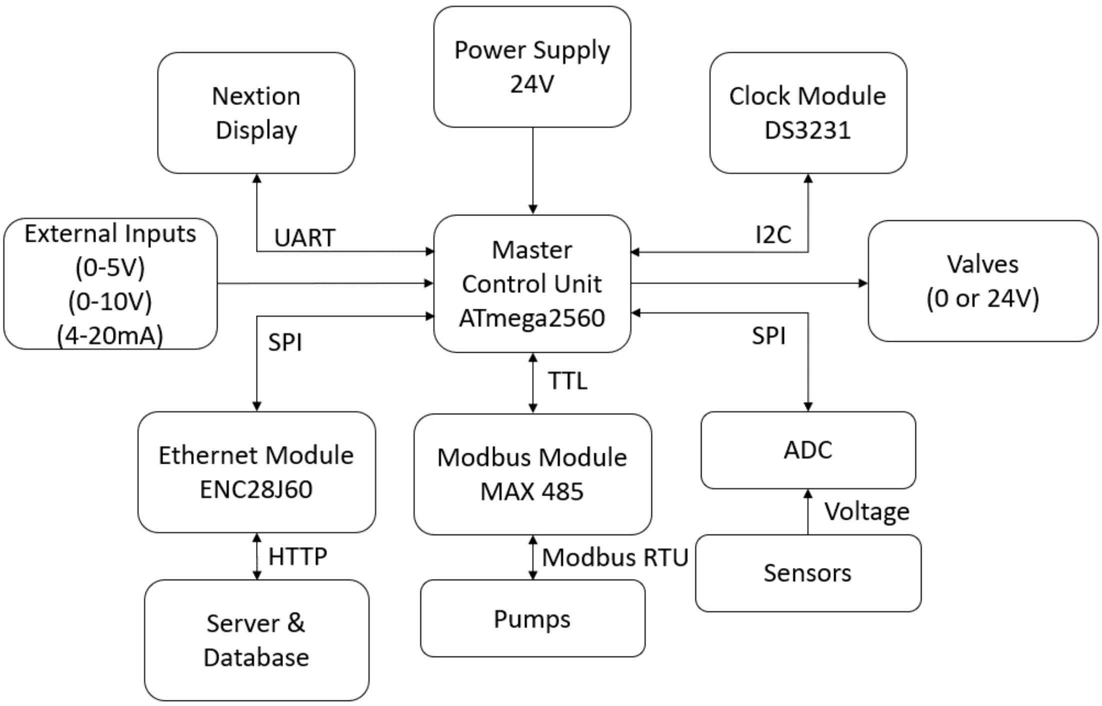


Figure 12. Master control unit interfaces (Bäck, 2020).

Different signals are used for reading the sensors or communicating with the devices. This means, that each instrument has a certain way it has to be read and each device has a certain way it is communicating with the master control unit. While the used sensors are based on analog input signals, the device communication is spread into various communication protocols. In this case, the device communication is divided into the serial peripheral interface (SPI), the inter-integrated circuit (I^2C), RS-485/Modbus RTU protocols, Universal asynchronous receiver-transmitter (UART), and hypertext transfer protocol (HTTP) interfaces.

These internal and external interfaces are to be implemented in the controller code and have to process, request and send the correct signals and data to the instruments and devices. These communication methods with the corresponding device type can be seen in Table 3. They are to be explained in detail in the following sections.

Table 3. PLC communication methods.

DEVICE	COMMUNICATION	I/O
Temperature Sensor PT1000	Analog $(0 - 5\ V)$	I
Flow Sensor	Analog $(1 - 4\ V)$	I
Pressure Sensor	Analog $(4 - 20\ mA)$	I
Heat Demand Sensor	Digital $(0\ /\ 2{,}5\ V)$	I
Valves	Digital $(0\ /\ 24\ V)$	O
Pumps	Modbus RTU (internal TTL)	I/O
Server/Database	HTTP (internal SPI)	I/O
Internal Clock	Inter-Integrated Circuit (I^2C)	I/O
Display (HMI)	UART	I/O
Debugging Interface	UART	I/O

4.3.1 Analog Sensors

The microcontroller processor is based on a voltage of $U_{processor} = 5V$. This is also important for the used sensors, which are using the board's voltage supply. The connected sensors work by changing the voltage on the analog input pin to a measurement-correlated value. For converting this analog voltage value to a digital measurement, a 10-bit analog-digital converter (hereafter ADC) is used, which according to the bit size can generate a resolution of $r = \frac{5V}{2^{10}-1} = 4.8mV$. This means the ADC will output a value between 0 and 1023, where each step is corresponding to a voltage change of $\Delta U = 4.8mV$ in a range of $5V$.

As an example the implemented PT1000 temperature sensor is examined. The PT1000 sensor with platinum wire and resistance of $R = 1000\ \Omega$ using a 2-wire system is a resistance temperature detector (RTD). RTDs have a known resistance at a certain temperature and a proportional change per temperature change. An example voltage on the processor pin $U_{meas} = 3.4\,Volt$ is correlated to an ADC output of $ADC = \frac{3.4V}{48mV} = 696$ digital output, which can then be mapped according to the datasheet resistance table to determine the temperature. But still, there is necessary a certain calibration which is described in Chapter 5.3 about Hardware Implementation.

The used pressure and flow sensor work in a similar way. The current or voltage on the appropriate sensor lead is interpreted by an operational amplifier circuit into a $U_{meas} = 0\ to\ 5\ V$ voltage input. This range describes the measurement range. This voltage is then translated by the ADC into a digital value and then mapped and processed according to the datasheets.

4.3.2 Serial Peripherical Interface (SPI)

The serial peripherical interface (hereafter SPI) describes a serial connection of the processor to other internal and external peripherals. Internal peripherals can be the ADCs, DACs, or Ethernet ports. External peripherals can be displays, memory storage or sensors. SPI describes a full-duplex solution, i.e. communication in which the partners can send and receive data simultaneously without having to wait for the other partner to complete the request or data package. For communication, four wires are needed. A continuous serial clock (SCLK) signal, two contrary data lines (CIPO/COPI) and a chip select (CS). While the serial clock is used for synchronizing the controller and peripheral, the chip select is used for selecting the right peripheral. This interconnection is shown in Figure 13 (Wootton, 2016b.)

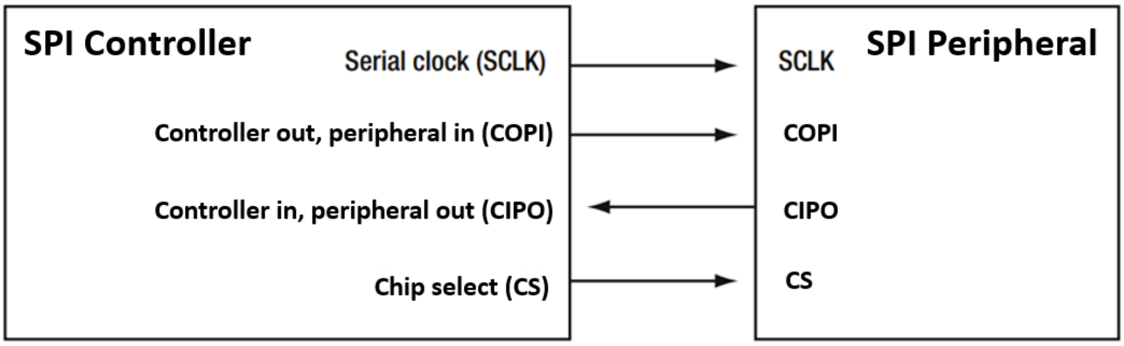


Figure 13. SPI communication structure (Wootton, 2016b).

In the used controller board this SPI is used to communicate with the ADCs, the ethernet module, meaning for controller board internal and controller-near external communication.

4.3.3 Inter-Integrated Circuit (I^2C)

Another serial connection that is used to communicate with the internal clock module DS3231 is the inter-integrated circuit (hereafter I^2C). Compared to the SPI, I^2C communication only describes a half-duplex communication protocol. This means that data can only be sent one after the other. This is because I^2C uses a shared bus architecture with a clock line (SCL) and a single bidirectional serial data line (SDA). This layout can be seen in Figure 14 (Hemmanur; Wootton, 2016a.)

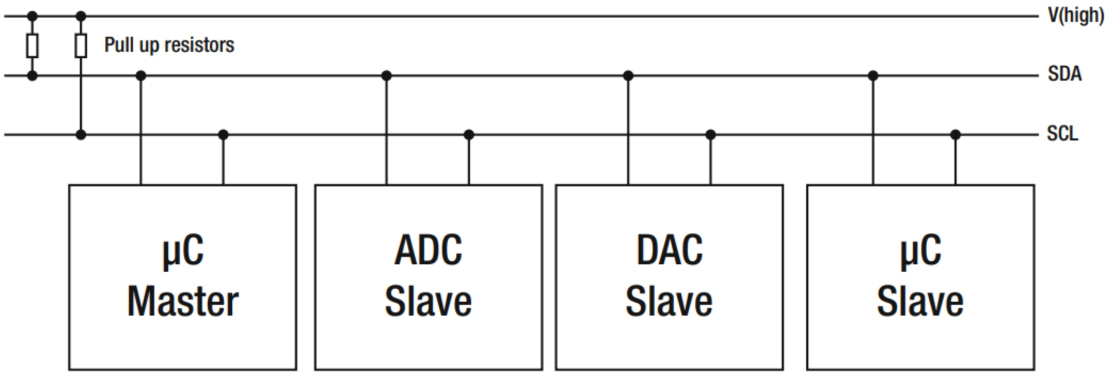


Figure 14. I^2C connection layout (Wootton, 2016a).

This protocol is used for processor communication with the internal clock module DS3231.

4.3.4 RS-485 & Modbus Remote Terminal Unit (RTU)

Another widely used industry standard communication layout is the RS-485 setup with Modbus RTU communication protocol. It is commonly used to control and monitor various devices over a serial connection. In this case, the pumps allow a Modbus interface connection. To control and monitor the used pumps, a reliable communication protocol is essential.

RS-485 provides a robust physical layer that enables communication over long distances. It is the industry standard for serial connection wiring. This also makes it ideal for larger industrial applications where the devices may be located remotely from the control system.

Modbus RTU provides a widely accepted communication protocol for exchanging data between devices in industrial automation and control systems. The Modbus massage protocol contains a particular message layout. It is built out of the application data unit (hereafter ADU) and the protocol data unit (hereafter PDU). While the PDU contains the actual data which is processed by the receiver, the ADU adds the header and footer to the message. The header and footer contain metadata for the transmission of the PDU. The address inside the header describes the place where the data should be transmitted and the error check in the footer is important for the consistency check of the received data. This message layout can be seen in Figure 15.

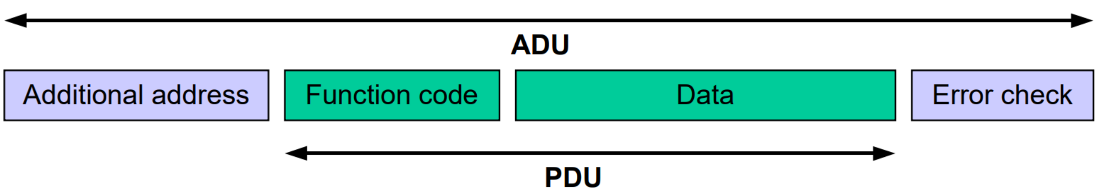


Figure 15. Modbus message layout (Modbus Organization, 2006).

There are defined different function codes that can be used in the Modbus RTU communication structure. These include reading and writing different data types to registers and coils. Registers represent multi-bit data storage and coils single discrete data storage locations. The main function codes (FCs) can be seen in Table 4.

Table 4. Modbus FCs (Modbus Organization, 2006).

FC	DESCRIPTION	FC	DESCRIPTION
01	READ COIL	**05**	WRITE SINGLE COIL
02	READ DISCRETE INPUT	**06**	WRITE SINGLE REGISTER
03	READ HOLDING REGISTERS	**15**	WRITE MULTIPLE COILS
04	READ INPUT REGISTERS	**16**	WRITE MULTIPLE REGISTERS

By using RS-485 to transmit Modbus messages, the installed pumps can be controlled and monitored. The Modbus protocol allows different types of data to be exchanged, such as setpoints, error codes and performance data. This enables to monitor the pump performance, detecting faults or malfunctions and making adjustments to the process if necessary (Axelson, 1995; Modbus Organization, 2006.)

4.3.5 Hypertext Transfer Protocol (HTTP)

To transform the used ATmega 2560 into an IoT device, an internet connection has to be established. For this, the Hypertext Transfer Protocol (hereafter HTTP) is used. This is a communication protocol based on the request/response paradigm, which is used on top of the TCP/IP connection. The used system describes a client/server architecture, i.e. it is used to establish a communication interface between the controller board and a server (Yannakopoulos, 2003.)

An HTTP message consists of a header and a body. The header contains the content length, the content type, the connection type, the HTTP version, the endpoint, and the applied method. The used HTTP version 1.1 defines different request methods, such as GET, POST, PUT, and DELETE. These methods allow different types of interaction between the ATmega 2560 and the server. For example, the GET method retrieves data from the server, while the POST method sends data to the server for processing. More information on server-side data processing is given in Chapter 5.2 about the Backend System Development. With this information, the following data body can be processed by the server. To send the data both parts must be consistent, otherwise, an error will occur. In this case, the sent body consists of data in the format of JavaScript Object Notation (hereafter JSON). The JSON body contains the information which is needed for monitoring the thermal system performance. For example power figures, temperatures or used setpoints.

With the applied interface the time-reliant data and sensor values can be sent to the server and be accessed via the web interface. This is a crucial point for conversion to Industry 4.0 and IoT, in which cloud interconnection is a key component (Arivalahan et al., 2021; Deekshath et al., 2018.)

5 Development & Implementation

5.1 Controller Development

For the code development several structures, algorithms and programs are used. The architecture used is designed to ensure applicability to other systems.

5.1.1 Integrated Development Environment (IDE)

As an integrated development environment (IDE) Visual Studio Code with the PlatformIO extension is used. This is a common setup for embedded system design. Both the IDE and the extension are free open-source software, making it a cost-effective yet proven technology. PlatformIO is then used for handling the compilation and uploading of the microcontroller code. I.e. it takes care of the so-called bootloader and programmer (Grokhotkov, 2023.) The bootloader is responsible for running and executing the program code. The programmer is used for transferring the compiled code from the Desktop PC to the PLC. In this case, the Arduino MegaCore bootloader and the *Arduino as ISP* programmer are used to upload the bootloader via SPI. The CP2102 USB to UART bridge is used to upload the C++ code. Visual Studio Code thereby is used as a common IDE for code development.

5.1.2 Object-Oriented Programming

For better expansion, adaptation and reusability of the developed code, an object-oriented programming approach is used. This paradigm is utilized in the whole process of the control system development.

Object-oriented programming is a programming paradigm, that tries to group data and behaviour into entities. It facilitates understanding because it represents human thinking. In real life, an object can be shown as an object with properties and a certain behaviour. For example, a LED has a value *brightness* and behaviour *turn on* and *turn off*. In programming, classes are used as templates for digital versions of these objects. These classes then can be used to derive specific objects from them. The classes can have attributes and methods, which represent the associated property and behaviour of an object (Bossard, 2021.)

This grouping of related attributes and methods is a decisive factor for reusability and further use in other projects. Since classes are stand-alone environments, they can easily be imported into further projects.

Another important part of object-oriented programming is the inheritance of class attributes and methods. It is possible to inherit these aspects from another class and easily add and extend new features to the new classes. This makes it particularly fast-forward since it is not necessary to rewrite certain features for a new class. In the case of this project, the classes were derived from a Base class. This for example contains a name and an ID of a certain object. This Base class is then used further for the DeviceBase, SensorBase, PumpBase, LedBase and ValveBase. These classes represent real-life objects with their basic attributes and methods. From these classes, the specific sensor, pump and valve classes are derived. This final stage then also contains the real communication interfaces to the devices, so that they can be directly represented in the code. These relationships between the classes can be seen in the inheritance diagram in Figure 16.

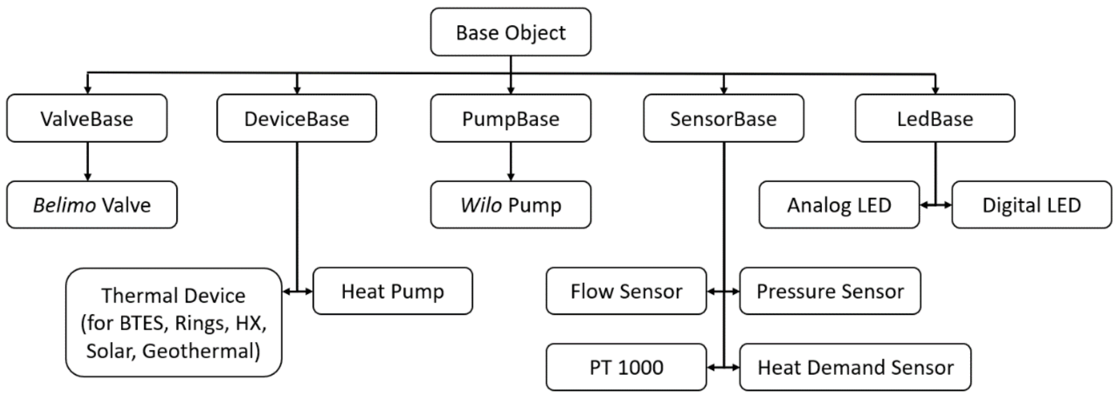


Figure 16. Inheritance diagram of used classes.

This system layout provides an easy way to implement a simulation mode in which the system can run without real connections to the system devices. For this, the second-level ”Base” objects can be used, which do not require physical connections to the specific devices. Instead, sensor, pump or other device values can be set within the program. This is especially useful for debugging and system behaviour analysis. There is more on this specific use in Chapter 5.3 about Hardware Implementation.

5.1.3 Code Development

As already discussed, for the code development the programming language C++ in the Arduino framework is used. To integrate the broad microcontroller libraries, which are developed by the Arduino community, C-header data is used to import the specific libraries to the main controller code. The integrated libraries with their use cases are shown in Table 5.

Table 5. Integrated Arduino libraries and related use cases.

ARDUINO LIBRARY	USE CASE
ModbusMaster.h	Modbus RTU Communication
Wire.h	I^2C Communication
SPI.h	SPI Communication
ArduinoHTTPClient.h	HTTP Communication
DS3231.h	RTC Integration
EthernetENC.h	Ethernet Module Integration

The used libraries facilitate the communication setup to the different devices. They keep track of the general structure of the messages or their integration. For example, the Modbus library automatically generates the ADU, so that only the PDU, the function and the data have to be declared by the user. Integrating these necessary external libraries is also the first step in the code.

In the next step, the classes are defined. They are based on the structure from Chapter 5.1.2. Since the exact declaration of these classes would go beyond the scope of discussion, only the general procedure for these classes is described. The generated classes consist of a constructor, destructor and class-specific methods and attributes.

The case-related objects are then created from these classes. These objects are similar to the PLC-connected devices. Since object-oriented programming is used, each physical entity of the system is represented by an object in the Arduino program. In the same step, also the global variables are set, such as the current time and date. These variables are set globally because it is necessary to use them in the following *setup()* and *loop()* code. These code functions are particularly important for the Arduino environment. The *setup()* function will be run once the system is started and the *loop()* function is then repeated as long as the microcontroller is turned on (Purdum, 2015.)

After initializing the global objects and parameters, the *setup()* function is executed. This function is initializing the communication protocols, which are used to interact with the different devices. So, for example, the serial connections are initialised, the I^2C communication to the clock is set up and the HTTP connection is established. The main initialisation steps described above are shown in Figure 17.

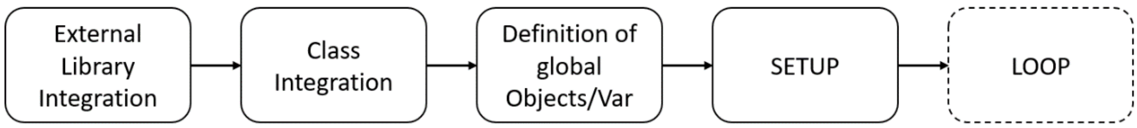


Figure 17. Initialization diagram.

As the last step after the initialisation, the *loop()* function is executed. This is the main control code and contains the decision and communication function of the controller. This code will be run continuously as long as the controller board is connected to the power supply. Since the ATmega 2560 does not allow sub-processes, there is a need for prioritization in the control code itself. Only one process can be executed at a time. The priority in terms of response time is direct human interaction. This means, that the HMI, the display and the Serial debugging connection are prioritized in the control loop to a response time of less than a second. The next bigger loop includes the sensor reading and adaptation. This is meant to be an adjusting control loop. For example, the pump speed can be adjusted to optimize the power output, while not changing the system state. This procedure is described in Chapter 3.3.1 about the Pump Control System. This loop is executed every minute. To add a certain stability to the control code, the main decision-making is not executed in the fastest way possible. This means, that the decision about the system state for instance is only being made every 5 minutes. This allows it to stay at a certain system state for a longer time without having to send signals to each device one after the other. This also helps to avoid the mentioned problem with the dead time of Chapter 3.3.1. This loop setup can be seen in Figure 18.

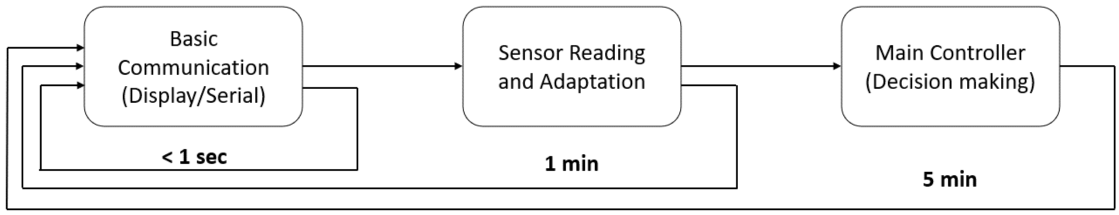


Figure 18. Loop execution diagram.

As mentioned above, the system state decision and other control tasks are performed in the main controller. The decision-making within this main controller is based on a truth table. This truth table is a table that represents a boolean expression table where the conditions for each device and state are defined. These conditions are based on the experience and developments of Heliostorage Oy. These conditions are then translated into the corresponding boolean decision blocks, also called if-statements. There are different control codes for different devices. These conditions are then used to decide between the system states of charging, discharging, solar discharge and off. The main interconnections for these states are described in Chapter 3.6 System as a Network.

One example control code, which is executed in the main controller is the buffer tank control. Since this code is quite independent, it can be controlled by a simple decision. As an input only the current tank temperature and the temperature of the upstream pipe flow are needed. Then the decision can be made by comparing these temperatures including a certain hysteresis temperature and the setpoint of the tank temperature. Finally, the tank valve is set according to the resulting status. This decision process can be seen in Figure 19.

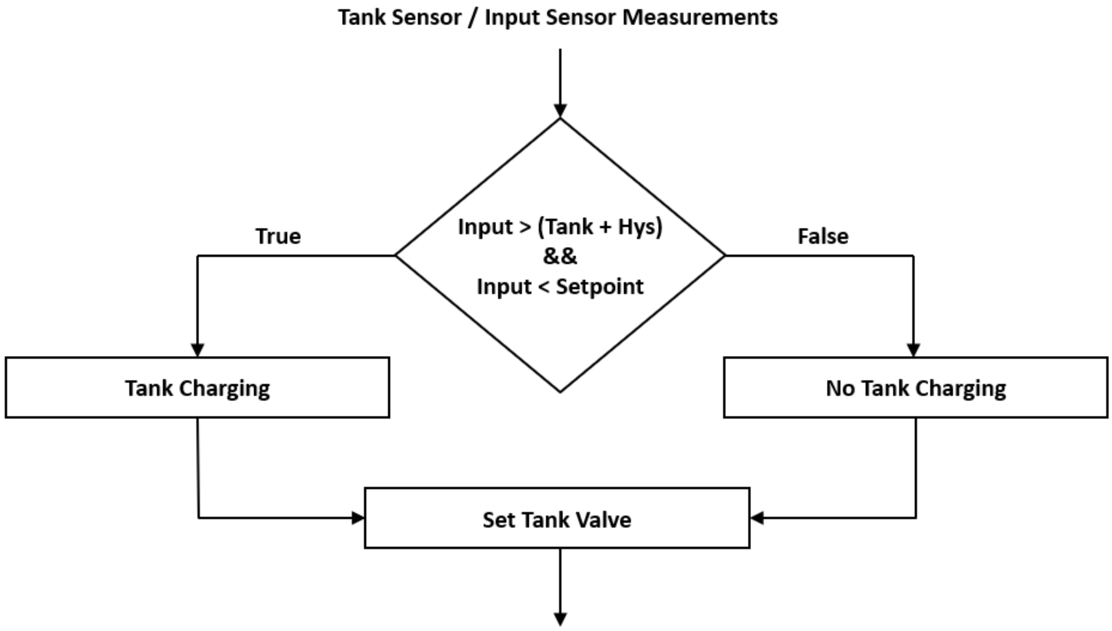


Figure 19. Schematic tank control statement.

Similar conditions are applied to all the system components. Their interconnection is based on each other and would go beyond the scope of discussion.

5.2 Backend System Development

The server architecture is a Linux VPS provided by the commissioner Heliostorage. This system is a remote server, especially used for testing purposes. VPS is an abbreviation for Virtual Private Server, which means that the physical server is shared by several parties but virtualization is used to separate the access and permissions. They are virtually separated and private. It allows full access to the root directory and all permissions to the dedicated VPS are granted. In this context, Linux describes an open-source operating system (OS), which is known for its reliability and customization possibilities. The server is reachable at its IP address via a secure shell (SSH) connection.

To integrate a data collection system and remote monitoring system on this server, the data needs to be transferred, stored, and visualized. Without a monitoring system, it would not be possible to easily analyse the behaviour and optimize the control in the further course. The main components to implement these functionalities are described in the following subchapters.

5.2.1 Docker Architecture

Docker is a popular containerisation platform and provides a flexible architecture for deploying applications on servers. The Docker engine enables the creation, management and execution of Docker containers on a host system. Docker containers in this context are lightweight, portable and isolated code execution environments (Chung et al., 2016.) Inside this environment, different data processing applications can be implemented, which take over the mentioned tasks of data transfer, storage and visualization.

For understanding the Docker architecture, the key components have to be explained. These include Docker images, which are created from Docker files and serve as templates for the containers. Docker containers encapsulate applications and dependencies. Docker volumes ensure persisting data across containers. Docker networks enable communication between containers and with the host system. And Docker Compose defines and runs multi-container applications, which provides an all-in-one solution for server maintenance (Docker Documentation, 2023.)

The benefits of the Docker architecture for server deployment include portability, enabling consistent deployment across environments, scalability, and isolation. It also provides security and stability for containerised applications.

In this system, the Docker engine is used to provide an environment for running applications, that perform the data processing, data storage and data visualization. A database (hereafter DB) system, an Application Programmable Interface (hereafter API) and a Graphical User Interface (hereafter GUI) application are used in combination to automate this data processing (Gkamas et al., 2022; Govindapillai et al., 2022.) This interaction between the different applications can be seen in Figure 20. A similar data processing architecture for solar thermal applications has been developed in a paper by Tsvetanov et al., with an MQTT Broker instead of the REST-API (Tsvetanov et al., 2022). An open-source full-stack application with the same server components (PostgreSQL and FastAPI) is developed by Ramírez (Ramírez, 2023). This shows the practical feasibility of combining these server applications.

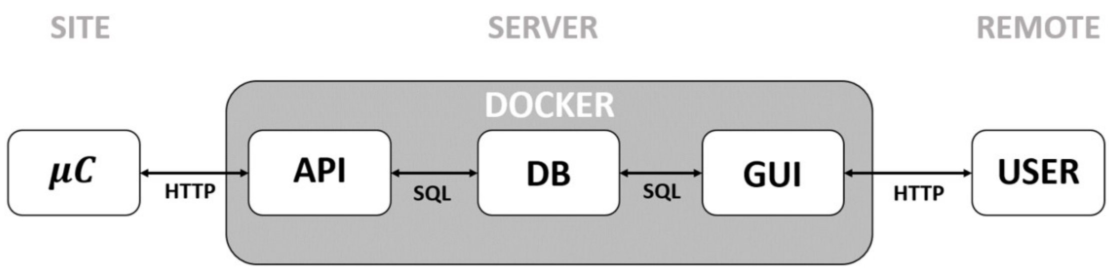


Figure 20. Server-side architecture and Interfaces.

5.2.2 Database System PostgreSQL

The first component inside the Docker environment is the PostgreSQL Docker container. This is used to store measurement and process data from the control system. PostgreSQL is an open-source industrial database that is often used in Docker containers due to its free availability and flexibility. As a popular and powerful database management system (hereafter DBMS), PostgreSQL provides robust features for storing, retrieving and managing data in a structured way. Docker allows PostgreSQL to be deployed in a containerised environment (Gkamas et al., 2022.)

In the context of PostgreSQL, the Structured Query Language (hereafter SQL) is used to establish a relational database. The term "relational" denotes the establishment of a connection between diverse values and properties. It can be conceptualized as a tabular structure wherein distinct values are stored in separate columns and rows based on a predefined schema (Kleuker, 2016.)

A key aspect of database design when using PostgreSQL in Docker is to adhere to common standards for structuring the database schema. This includes implementing proper normalisation techniques. The database is developed to comply with the third normalization form (3NF) and foreign keys are inserted to establish relationships between tables. Database normalisation is a process that ensures data integrity and reduces redundancy in the database structure, resulting in a more efficient and organised database (Kleuker, 2016.) Docker provides the flexibility to define and manage the PostgreSQL database schema within the container, providing a consistent and repeatable environment for database design and development. The used schema correlation is represented in Figure 21.

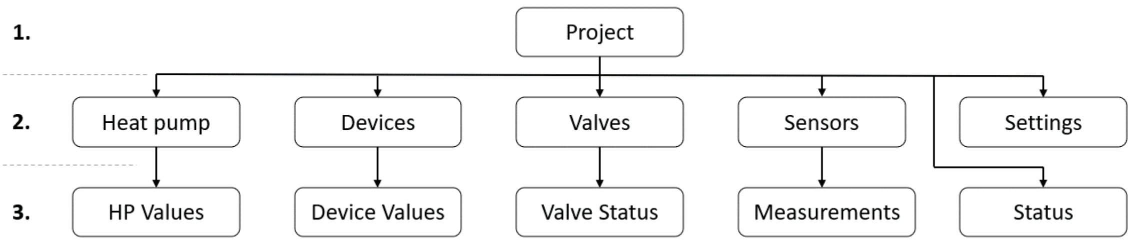


Figure 21. Database schema structure.

This structure contains three levels of database tables. The first table contains project-specific data such as contact data of the project responsible persons or other general information. The second layer contains the device information, where all the project-related sensors, valves, devices and heat pumps can be added. At this level, the general information about these devices is stored. Such as name, type, ID and unit. These first two layers are time-independent database tables. In contrast to that, the third layer of the database structure represents time-reliant values, such as measurement data, status information, and power or energy values.

Foreign keys are used to link the database tables together for efficient use and easier querying. This helps to request important information for a specific project, sensor or time. So for example each measurement is dependent on a sensor which in turn is dependent on a project.

The used structure is enabling the addition of new entities or projects without the need to restructure the entire database.

With this setup, the microcontroller can send the measurement data, status information and calculated values to the API, which sorts the time-dependent values into the third layer of database schemas.

5.2.3 Application Programmable Interface FastAPI

To connect the microcontroller system to a database, an interface is required to transfer the data from the local setup to the server-side storage location. This is necessary to store the performance and control data and further be able to analyse the system operation. Therefore an API is needed.

One way of doing this is to directly execute SQL queries from the Arduino. However, as the computing power and memory are very limited, it is preferable to transfer the data in a less computing-intense format and reduce the computing power on the local device. Therefore an additional serverside interface is used, to process a less computing-intense HTTP request and translate it into SQL queries (Arduino Getting Started, 2023; Norris, 2015.)

For this, a so-called RESTful API is used. REST in this context means Representational State Transfer, describing an architecture for communication (Kotz and Wenz, 2022). In this specific case, the Python-based interface FastAPI is used. This API is running on the VPS server and is listening on an endpoint related to the domain name. This API is used to process HTTP requests with JSON data, which already is explained in Chapter 4.3.5 on the Hypertext Transfer Protocol (HTTP) request layout. The incoming data can be parsed using Python's built-in JSON module, which converts the JSON data into a Python dictionary.

Once the data has been parsed, it can be validated using FastAPI's data validation capabilities to ensure that the data conforms to the expected schema (Lathkar, 2023.)

Once the incoming data has been validated, it can be stored in the PostgreSQL database using SQLAlchemy with the database driver *psycopg2*. *Psycopg2* provides several functions for performing database operations, such as connecting to the database, executing SQL queries, and committing changes to the database. By using these functions, FastAPI can store the incoming data in the PostgreSQL database with minimal code, while still allowing error control and request validation.

5.2.4 Graphical Monitoring System Grafana

A visualisation and analysis tool is needed for better monitoring of the collected performance data stored in the installed PostgreSQL database. For this purpose, the widely used visualisation and query tool Grafana is used. It is an open-source program that can be easily added to the Docker host system using the official Grafana Docker image.

To connect the Grafana visual interface to the database as a data source, a custom dashboard is created. This contains the most important data in a visual format so that it can be easily analysed and the decision-making further improved. This is particularly useful for troubleshooting control behaviour, as the external conditions and the resulting behaviour can be compared and analysed.

5.3 Hardware Implementation

For the physical implementation of the system, the code first has to be uploaded to the control board. To upload code to the ATmega 2560 with bootloader using the VS Code PlatformIO extension, the appropriate board and port settings must be selected in the PlatformIO configuration file. Once the configuration has been set up, the code can be written in the Arduino framework and compiled using the

PlatformIO build command with the setting described in Chapter 5.1.1 on the Integrated Development Environment (IDE). After compilation, the compiled code can be uploaded to the ATmega 2560 using the PlatformIO upload command, and then the code can be executed.

For troubleshooting the designed system, the already mentioned simulation mode can be used. Running the ATmega controller in simulation mode allows testing and troubleshooting the system by setting all incoming pin values and sensor inputs to specific values. This provides a controlled environment to analyse the behaviour of the system under different conditions and identify any potential problems that may be encountered. By simulating inputs and sensors, it is possible to test the functionality of the system without a need for physical components. The ability to run this kind of rapid prototyping allows to test different scenarios quickly and easily. In general, this debugging can be thought of as an iterative process that will be continued over time. Since not all real system parameters can be predicted and tested in advance, this process is repeated on-site. For this approach, onboard debugging is used, where the results can be seen and adapted in real-time (Kagane and Shaji, 2021.)

Once this simulation step has been completed, the switch to the physical system can be made. This step also involves pin connection planning, in which each device with its connection interfaces has to be organized in a way that all the devices can be connected to the available controller board pins. The available pins can be seen in Chapter 4.1 on the Microcontroller Unit, while the used interfaces and the necessary pin connections are described in Chapter 4.3 on the Control Interfaces. Combining these two requirements in the pinout diagram can be seen in Appendix 3.

The final hardware implementation on-site involves designing and building the physical components of a system to support the desired functionality. In the context of implementing the controller board, this involves connecting various cables such as Ethernet, Modbus, valve and sensor cables, as well as the power supply. To streamline the process and ensure efficient cable management, a cable harness is used. The use of a harness helps reduce the likelihood of errors such as misconnections or shorts and simplifies troubleshooting by clearly identifying cables and connectors. In addition, harnesses improve the overall appearance of the system by organising cables and reducing clutter.

The installation of the controller board also includes the mounting of the PT1000 temperature sensors required for the control structures and power calculations. These replace, among other sensors, the previously unreliable digital temperature sensors. In order to obtain an accurate measurement with these new sensors, they have to be calibrated. This is done by comparing the resistance value with the PT1000 resistance table. This allows a potentiometer to be set to the appropriate resistances representing specific temperatures. This exact resistance is compared with the measured value via the board's internal measurement circuit. Due to the linear behaviour of the PT1000 sensor, two measurements are required to determine the coefficients.

After an iterative process of debugging and troubleshooting, a controller board code was developed that responds to incoming sensor signals in a predetermined way. To get there, several issues had to be resolved relating to the implementation on-site. While the simulated conditions were in a closed environment, in the physical system multiple errors and external uncertainties can occur. These especially include wiring errors and communication problems.

For example, while the network conditions in the test network are very open, the local network on-site is equipped with access restrictions. This means that the board's IP address cannot be obtained via the widely used Dynamic Host Configuration Protocol (DHCP). Instead, a static IP address must be set, approved by the IT administrator on site, and given the necessary permissions. This was essential to allow HTTP requests from the controller board to the server.

The final control cabinet can be seen in Figure 22.

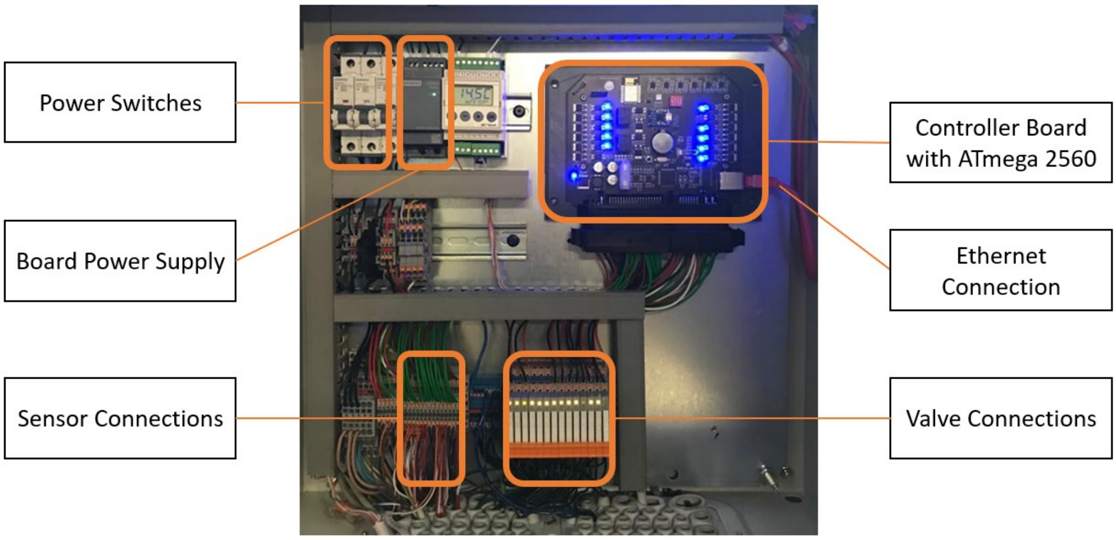


Figure 22. Developed control cabinet.

6 Results

A comprehensive control and monitoring system for a thermal network was developed and has been implemented in the school building in the municipality of Kaustinen for this study's purposes. To evaluate the system in detail, the results are examined and analysed below.

The thermal compound network has been assessed and a control and monitoring system has been developed. The ATmega 2560-based control system takes in sensor signals, processes them and takes action depending on the input. For this, several communication protocols are used, which are implemented in the embedded PCB. It can control the included pumps, valves and other devices depending on available irradiation or demand. In this way, it can control the heat flows and react to internal and external influences.

It can be concluded that the system fulfils its defined tasks. The system switches successfully and automatically between charge, discharge, solar discharge and offline modes. However, a closer look reveals a wide range of possible extensions that the commissioner Heliostorage can add in the future.

After solving the final complications with the on-site network and sensor calibration, the controller board successfully combines the assessed thermal components, resulting in an interactive network. This means the control system combines the heat pump, the BTES and the solar thermal collector roof into an efficient network. In this network, the power flows in the expected directions, meaning from and to the correct devices according to the mentioned truth table and prioritisation. This can be seen from the debugging output in combination with the corresponding states of the valves and pumps on-site. It can also be seen from the resulting data collected on the server. One exemplary data collection can be seen in Figure 23. As can be seen from the dashboard developed, the graphs show the power data and the correlated temperature and flow values for a day in mid-May.

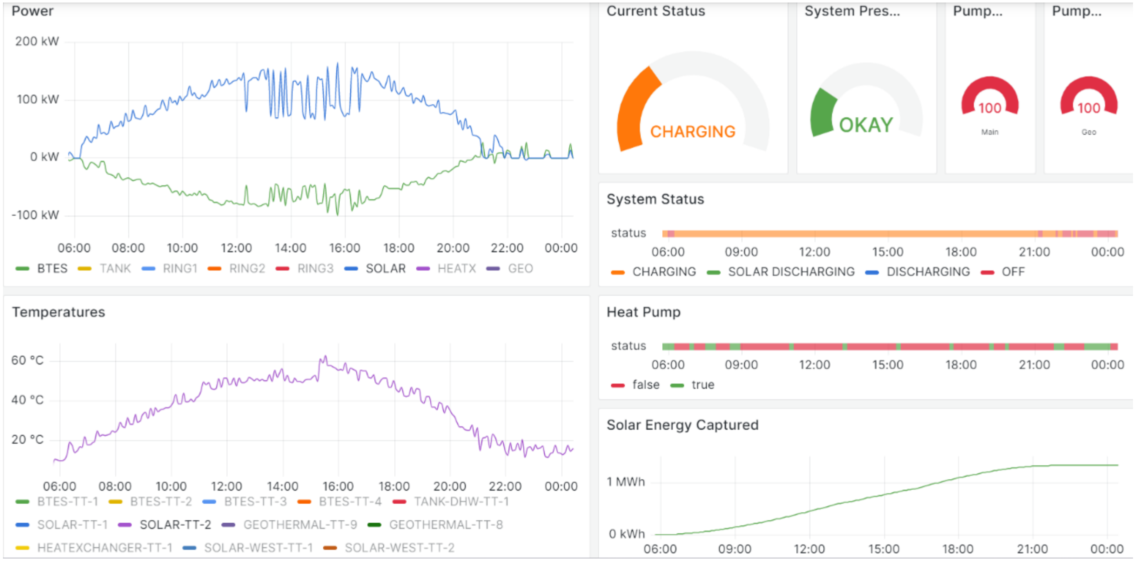


Figure 23. Developed Grafana dashboard with system variables mid-May.

While the maximum power generated by the solar thermal system is approximately $P_{solar} = 150\ kW$, the largest share of $P_{BTES} = -90\ kW$ is fed directly into the BTES system for the charging process. As can be seen in the diagram in the bottom left corner of Figure 23, this is done at a temperature of about $T_{solar} \approx 50\ °C$. A further $P_{Tank} = -10\ kW$ of the power generated by the solar thermal system is fed directly into the buffer tank, which fills quickly due to its small dimension. The remaining power is then used to heat the geothermal boreholes, which have a power of around $P_{GEO} = -50\ kW$. This helps to cool down the fluid returned to the solar collector to a temperature of around $T_{return} \approx 10\ °C$. In this way, approximately $E_{solar} = 1{,}3\ MWh$ of energy were converted with the solar thermal collectors on this specific day.

While the pump control is optimized for a low-temperature output, the pump speed increases as long as the optimal temperature difference is not reached. Since the described solar thermal output temperature of $T_{solar} \approx 50°C$ is high in comparison to the ~30°C that was used as a setpoint in this case, it can be seen, that the pump does not allow the high mass flow which is required. This results in high-temperature outputs on sunny days when the pump size does not allow any more increments of the pump speed. This leads to the conclusion that the pump is undersized for this kind of operation. But the described power-related control concept is proven, as it reaches maximum speed when the irradiation is high and gradually lowers the pump speed in the evening when the irradiation decreases.

When comparing the old with the new system power in Figure 24, it can be seen that the power on the first day of operation already outperforms every power generation during the beginning of the year, between January and April 2023, when the old system was installed. While the power in the old system barely exceeded $P_{max,old} = 40\ kW$ in the current year, the new system already captured power of $P_{max,new} = 150\ kW$ on the first day of real operation. This exorbitant increase in power generation of the roof can be mainly explained by the new control code and the change of sensors and wiring. The previously unreliable sensors and external devices were causing faults, which were also responsible for the long system downtimes shown in diagram (b) in Figure 24. The irradiation data may vary, but it is not significant when compared to several months with certainly some sunny days.

(a) (b)

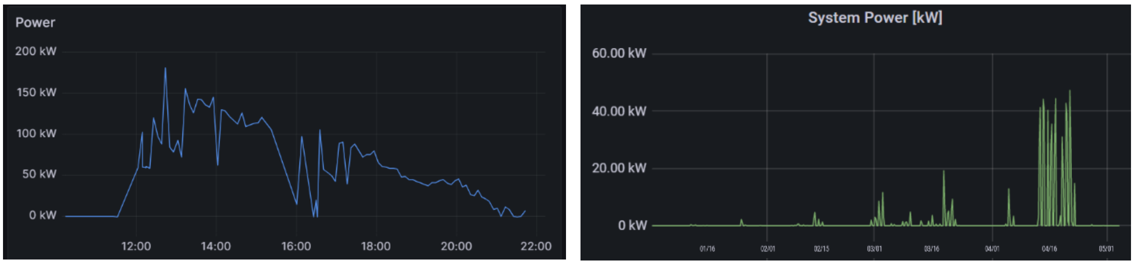


Figure 24. Comparison of power (a) on the first day of operation in May 2023 of the new system and (b) between January and May 2023 with the old system.

Comparing the heat pump consumption of the new and the old installation, the improvements are evident. Even though both systems were operated under almost similar weather conditions and over the duration of one week, the new installation outperformed its predecessor. While the previous system was practically inoperative due to sensor faults, the heat pump needed to produce $6440\ kWh$ of thermal energy in one week. This was generated by an electrical energy supply of $2108\ kWh$ to the heat pump. This gives a COP of about 3. Due to the improvements in the new system, the heat is supplied directly to the buffer tank and the heat demand was reduced by almost 70% to only about $2010\ kWh$. Meanwhile, the brine-in side of the heat pump was also supplied at a higher temperature, resulting in a buffer in electrical energy consumption. The consumption of electrical energy is reduced to $500\ kWh$ at a COP of 4. The total reduction of electrical energy usage adds up to $1608\ kWh$ comparing these two weeks. For a better overview see Table 6. Additionally to this reduction of the thermal energy demand and the heat pump input, the BTES got charged to supply the system in winter. These results cannot yet be extrapolated over the course of the year, as the different conditions throughout the year can lead to different results, but they do provide an outlook on the performative potential.

Table 6. Heat pump characteristics for one week of operation in comparison.

VALUE	OLD	NEW	DIFFERENCE
Thermal Energy Supply	$6440\ kWh$	$2010\ kWh$	$-\ 69\ \%$
Electrical Energy Demand	$2108\ kWh$	$500\ kWh$	$-\ 76\ \%$
COP	3.05	4.02	$+\ 31\ \%$

Since the previously analysed data is processed with the developed backend system, it can be seen that also this unit works in the expected way. Processing, storing and displaying the data sent by the control board works with a high degree of reliability for the defined tasks. The used Docker Engine provides a reliable solution for the server-side architecture, which can be seen in the non-existent off-time and unreachability. Overall the system provides a solution, that is avoiding data losses and server errors. However, there is still a lot of room for improvement, particularly in terms of data security and user-friendliness. This will be the subject of the further discussion in the outlook below.

7 Discussion

As analysed in the previous chapter, the installed system can efficiently control and monitor the described thermal system with its components. While the controller, as an embedded system, controls the system on-site and independently makes decisions about heat flows, the server acts as a data processing unit that displays the parameters resulting from the decisions in a meaningful and clear way. This provides a significant improvement in data analysis and controller operation. It contributes to a high increase in the overall system efficiency and ensures correct operation.

The installed system helps to reduce the facility’s external energy supply, thus minimising the external dependency of the school’s heat supply. This is particularly evident in the reduction in electrical energy required to operate the heat pump. In the analysed period of the year, the increased COP is mainly related to the higher heat pump input temperature, which is achieved by solar thermal collectors. The BTES itself only got charged in the observation period, so that the results obtained are related to the operation of the solar thermal collectors. These observations agree with a paper from Girard et al., in which an increased COP for ground source heat pumps in combination with solar thermal collectors was described. For higher latitudes as in Finland an increase of $\Delta COP \approx 0.2$ was observed in the course of a year. The higher increment in COP that was observed in this work could be related to the short observation period.

The storage of thermal energy produced by the solar thermal system is an important factor in lowering the all-year energy demand. The use of BTES allows the thermal energy produced in summer to be stored until the winter months. This helps to meet the increased heat demand in winter and supports the heat pump with a higher input temperature. These results were gained with the observations and conclusions of Rad and Fung or Pavlov and Olesen that underground thermal energy storage is a promising solution for shifting energy from solar thermal energy excess periods in summer to heat demand periods in winter. However, this behaviour needs to be monitored in the long term.

The increased use of locally produced energy and the associated reduction in the need for additional energy will lead to a significant improvement in the facility's CO_2 emissions. This is because the reduced electrical power consumption reduces the load on the local power grid. Every kWh of electric energy that can be saved and is not provided by the Finnish electricity mix reduces carbon emissions. According to Fingrid, the Finnish national transmission system operator (TSO) the emission factor (EF) for electricity consumed in Finland in the first quarter of 2023 is about $EF = 55\ g(CO_2)/kWh$ (Fingrid, 2023). With reduced consumption of $\Delta E_{el} = 1608\ kWh$ of electrical energy in this first week of operation, the savings imply a reduction in CO_2 emissions of $\Delta EF_{total} = -1608\ \mathrm{kWh} \cdot\ 55\ \mathrm{g(CO_2)/kWh}\ =\ -88\ \mathrm{kg(CO_2)}$. While CO_2 is the main greenhouse gas (hereafter GHG) which is responsible for human-induced climate change, its reduction in the coming years is central to many countries' climate politics and action plans (IPCC, 2021.) This also applies to the heating sector, in which the implementation of this kind of thermal network would provide an improvement, especially compared to conventional gas or oil heating. However, also a complete carbon-neutral heating system would be possible, by implementing renewable electricity supply for the heat pump. It brings forward heat system transformation.

As this system is also a pilot project for the commissioner for the new ATmega 2560 controller board, major changes are expected as a result of its successful implementation. The proof of concept which has been provided in this paper, closes the gap between the practical implementation and the theoretical control strategy. The basic code development and fundamental feasibility demonstrated by this system will greatly simplify the production process for other systems that can be developed on the same structure. Due to the object-oriented programming that is used, this code can also be applied to future projects, even if they do not contain the same components. By demonstrating the feasibility of code development for the specially designed controller board, the number of external modules that need to be used is reduced, significantly reducing potential sources of error and the additional cost of external components that would otherwise be required. The controller code development can therefore be more easily scaled up to a bigger production process. Additionally, the use of the ATmega 2560 based controller board and the open-source server software offer a low-cost solution, that can easily be implemented in additional use cases. This allows to increase the number of the installed thermal network systems with BTES, solar thermal collectors and heat pumps that include the developed controller board.

8 Conclusion & Outlook

In summary, the system created provides an efficient solution for controlling and monitoring a thermal system. This includes a reliable controller through the use of the fast and precise ATmega 2560 processor and the associated board. The database and visualisation system connected via the API processes the incoming data and automatically generates a clear summary of the system performance.

By implementing this system, power flows can be observed and the resulting energy and efficiencies calculated automatically. However, for a more detailed analysis of this data, a long-term analysis must be carried out. This is due to the long-term functioning of the BTES in combination with the solar thermal system and the heat pump. With the first classification in the Results and Discussion part, only an insight into the functioning at a certain time point in the early summer months can be given.

As a potential improvement that has been noticed in the development phase, the security layer of the server could be increased by using the HTTPS protocol instead of HTTP. For this, TLS/SSL certificates would be needed. By using this additional security measure, the likelihood of data extraction from unauthenticated externals could be avoided. In the same step, a reverse proxy server should be included, that takes care of exactly this measure. One common possibility is the use of the Nginx image in the Docker application. This means, that the used Docker Engine can be easily extended with the mentioned container. Another way is the migration of the system into a full-stack application developed by Ramírez et al.. This system uses the same system components as already implemented, so PostgreSQL in combination with FastAPI in the Docker environment and would provide built-in HTTPS, test options and user-friendly dashboards based on the Vue framework. This would also improve the usability for clients (Ramírez, 2023.)

Additionally, a further point of research has been discovered. As the controller decisions are now made based on the system parameters which are developed from previous projects and experience, in the future predictive algorithms could be used to optimize charging and discharging periods based on weather or irradiation forecasts. For example, approaches by Saloux and Candanedo or Xu and Dubljevic could be followed.

9 References

Alva, G., Lin, Y. and Fang, G. (2018) ‘An overview of thermal energy storage systems’, *Energy*, vol. 144, pp. 341–378 [Online]. DOI: 10.1016/j.energy.2017.12.037.

Arduino Getting Started (2023) *Arduino - MySQL | Arduino Tutorial* [Online]. Available at https://arduinogetstarted.com/tutorials/arduino-mysql (Accessed 23 April 2023).

Arivalahan, R., Balaji, S., Kamalakannan, M. and Vinoth, T. (2021) ‘DEVELOPMENT OF ARDUINO BASED MICROCONTROLLER THROUGH INTERNET OF THINGS (IOT) FOR THE MEASUREMENT AND MONITORING OF …’, in *International Journal of Electrical Engineering and Technology*, pp. 50–61.

Axelson, J. (1995) *Networks for Monitoring and Control Using an RS-485 interface* [Online]. Available at http://delta-electronic.com/article/wp-content/uploads/2008/09/rs4851.pdf.

Bäck, E. (2020) ‘Planering av styrning för värmesystem’ [Online]. Available at https://www.theseus.fi/bitstream/handle/10024/341164/B%C3%A4ck_Emil.pdf?sequence=2 (Accessed 22 March 2023).

Böckh (2014) *Wärmeübertragung*, Berlin, Heidelberg, Springer Berlin Heidelberg.

Bossard, A. (2021) ‘Autonomous on-Chip Debugging for Sensors Based on AVR Microcontrollers’, in *Journal of Sensor Technology*, pp. 19–38.

Buzás, J., Farkas, I., Biró, A. and Németh, R. (1997) *Modelling and Simulation of a Solar Thermal System* [Online]. Available at https://www.researchgate.net/publication/280884699_Modelling_and_Simulation_of_a_Solar_Thermal_System?enrichId=rgreq-62a77f34e4fc876dcc32724cfe96244f-XXX&enrichSource=Y292ZXJQYWdlOzI4MDg4NDY5OTtBUzozNzIxMTgzMTY0Mzc1MDRAMTQ2NTczMTMyODU1Mw%3D%3D&el=1_x_2&_esc=publicationCoverPdf.

Chung, M. T., Quang-Hung, N., Nguyen, M.-T. and Thoai, N. (2016) 'Using Docker in high performance computing applications', *2016 IEEE Sixth International Conference on Communications and Electronics (ICCE 2016): Ha-Long City, Quang Ninh, Vietnam, 27-29 July 2016.* Ha-Long City, Quang Ninh Province, Vietnam, 7/27/2016 - 7/29/2016. Piscataway, NJ, IEEE, pp. 52–57.

Deekshath, R., Dharanya, P., Dimpil Kabadia, K. R., Deepak Dinakaran, G. and Shanthini, S. (2018) 'IoT based environmental monitoring system using arduino UNO and thingspeak', in *International Journal of Science Technology & Engineering*, pp. 68–75.

Docker Documentation (2023) *Docker overview* [Online]. Available at https://docs.docker.com/get-started/overview/ (Accessed 23 April 2023).

Edward Xu (2018) *Green Energy Island Integration and Operation Optimization Research 绿色能源岛集成与优化运行研究* [Online]. Available at https://www.researchgate.net/publication/326912104_Green_Energy_Island_Integration_and_Operation_Optimization_Research_lusenengyuandaojichengyuyouhuayunxingyanjiu.

Eismann, R. (2017) *Thermohydraulische Dimensionierung von Solaranlagen: Theorie und Praxis der kostenoptimierenden Anlagenplanung*, Wiesbaden, Germany, Springer Vieweg.

Fingrid (2023) *Real-time CO2 emissions estimate* [Online]. Available at https://www.fingrid.fi/en/electricity-market-information/real-time-co2-emissions-estimate/ (Accessed 10 May 2023).

Girard, A., Gago, E. J., Muneer, T. and Caceres, G. (2015) 'Higher ground source heat pump COP in a residential building through the use of solar thermal collectors', *Renewable Energy*, vol. 80, pp. 26–39.

Gkamas, T., Karaiskos, V. and Kontogiannis, S. (2022) 'Performance Evaluation of Distributed Database Strategies Using Docker as a Service for Industrial IoT Data: Application to Industry 4.0', *Information*, vol. 13, no. 4, p. 190.

Govindapillai, L. C., J, K. P., D, N. K. and H, M. B. (2022) 'Industrial PC based Automated Store Management System', *2022 International Conference on*

Edge Computing and Applications (ICECAA). Tamilnadu, India, 10/13/2022 - 10/15/2022, IEEE, pp. 118–122.

Granqvist, C. G. (1991) *Materials science for solar energy conversion systems*, Oxford, New York, Pergamon Press.

Grokhotkov, I. (2023) *ESP8266 arduino core documentation* [Online]. Available at https://media.readthedocs.org/pdf/arduino-esp8266/latest/arduino-esp8266.pdf.

Hemmanur, K. 'Inter-Integrated Circuit (I2C)' [Online]. Available at https://www.egr.msu.edu/classes/ece480/capstone/fall09/group03/AN_hemmanur.pdf (Accessed 9 April 2023).

Hrisca, L. and Melis, W. J. (2018) 'Pump Speed Optimisation for Solar Thermal System', *Solar Energy*, vol. 172, pp. 212–218 [Online]. DOI: 10.1016/j.solener.2018.03.021.

IPCC (2021) *Climate change widespread, rapid, and intensifying* [Online]. Available at https://www.ipcc.ch/2021/08/09/ar6-wg1-20210809-pr/ (Accessed 29 May 2023).

Joint Research Centre (2016) *JRC Photovoltaic Geographical Information System (PVGIS) - European Commission* [Online]. Available at https://re.jrc.ec.europa.eu/pvg_tools/en/ (Accessed 28 March 2023).

Kagane, A. and Shaji, J. (2021) *Implementation of sensor interfacing using Arduino* [Online]. Available at https://www.ijeast.com/papers/374-378,tesma605,ijeast.pdf.

Kalogirou, S. A. (2004) 'Solar thermal collectors and applications', *Progress in Energy and Combustion Science*, vol. 30, no. 3, pp. 231–295 [Online]. DOI: 10.1016/j.pecs.2004.02.001.

Kannaiyan, S., Bokde, N. D. and Geem, Z. W. (2020) 'Solar Collectors Modeling and Controller Design for Solar Thermal Power Plant', *IEEE Access*, vol. 8, pp. 81425–81446.

Kleuker (2016) *Grundkurs Datenbankentwicklung*, Wiesbaden, Springer Fachmedien Wiesbaden.

Kotz, J. and Wenz, C. (2022) *C# und .NET 6: Grundlagen, Profiwissen und Rezepte*, München, Hanser.

Lathkar, M. (2023) *HIGH PERFORMANCE WEB APPS WITH FASTAPI: The asynchronous web framework based on modern python*, [S.l.], Apress.

Mangold, D., Schmidt, T. and Müller-Steinhagen, H. (2003) *Seasonal thermal energy storage in Germany* [Online]. Available at https://scholar.archive.org/work/5w42bnjqxfgzxlexh3npu3yuia/access/wayback/http://www.solites.de/download/literatur/03-03.pdf.

MCUdude (2023) *Arduino hardware package for ATmega series* [Online]. Available at https://github.com/MCUdude/MegaCore#pinout (Accessed 14 May 2023).

Miocic, J. M., Alcalde, J., Heinemann, N., Marzan, I. and Hangx, S. (2022) ‘Toward Energy-Independence and Net-Zero: The Inevitability of Subsurface Storage in Europe’, *ACS energy letters*, vol. 7, no. 8, pp. 2486–2489.

Modbus Organization (2006) ‘MODBUS Application Protocol 1 1 b’ [Online]. Available at https://modbus.org/docs/Modbus_Application_Protocol_V1_1b.pdf (Accessed 12 March 2023).

Nayyar, A. and Puri, V. (2016) *A review of Arduino board's, Lilypad's & Arduino shields: Proceedings of the 10th INDIACom; 2016 3rd International Conference on Computing for Sustainable Global Development : (16th-18th March, 2016)* [Online], [Piscataway, NJ], IEEE. Available at http://ieeexplore.ieee.org/servlet/opac?punumber=7589474.

Nilsson, E. and Rohdin, P. (2019) ‘Performance evaluation of an industrial borehole thermal energy storage (BTES) project – Experiences from the first seven years of operation’, *Renewable Energy*, vol. 143, pp. 1022–1034.

Norris, D. (2015) *The internet of things: Do-it-yourself at home projects for arduino, raspberry pi, and beaglebone black* [Online]. Available at https://www.accessengineeringlibrary.com/binary/mheaeworks/463c3e36324b99c7/0d7b544fdc4d6299190965b457400b88261564f6b09536886170a671f0469f04/book-summary.pdf.

Pavlov, G. K. and Olesen, B. W. (2012) ‘Thermal energy storage - A review of concepts and systems for heating and cooling applications in buildings: Part 1- Seasonal storage in the ground’, *H V A C & R Research*, vol. 18, no. 3, pp. 515–538 [Online]. DOI: 10.1080/10789669.2012.667039.

Purdum, J. J. (ed) (2015) *Beginning c for arduino, second edition - learn c programming for the ardui*, Berkeley, CA, Springer-verlag Berlin And Hei.

Quaschning, V. (1996) *Simulation der Abschattungsverluste bei solarelektrischen Systemen* (Zugl.: Berlin, Techn. Univ., Diss., 1996) [Online], Berlin, Verlag Dr. Köster. Available at https://www.volker-quaschning.de/downloads/abschattungsverluste.pdf (Accessed 29 March 2023).

Rad, F. M. and Fung, A. S. (2016) ‘Solar community heating and cooling system with borehole thermal energy storage – Review of systems’, *Renewable and Sustainable Energy Reviews*, vol. 60, pp. 1550–1561 [Online]. DOI: 10.1016/j.rser.2016.03.025.

Răduca, E., Ungureanu-Anghel, D., Nistor, L., Haţiegan, C., Drăghici, S., Chioncel, C., Spunei, E. and Lolea, R. (2016) ‘Web server with ATMEGA 2560 microcontroller’, *IOP Conference Series Materials Science and Engineering*, vol. 106, no. 1, p. 12018 [Online]. DOI: 10.1088/1757-899X/106/1/012018.

Ramírez, S. (2023) *tiangolo/full-stack-fastapi-postgresql: Full stack, modern web application generator. Using FastAPI, PostgreSQL as database, Docker, automatic HTTPS and more* [Online]. Available at https://github.com/tiangolo/full-stack-fastapi-postgresql (Accessed 23 May 2023).

Ridder, F. de, Diehl, M., Mulder, G., Desmedt, J. and van Bael, J. (2011) ‘An optimal control algorithm for borehole thermal energy storage systems’, *Energy and Buildings*, vol. 43, no. 10, pp. 2918–2925 [Online]. DOI: 10.1016/j.enbuild.2011.07.015.

Sadeghi, G. (2022) ‘Energy storage on demand: Thermal energy storage development, materials, design, and integration challenges’, *Energy Storage Materials*, vol. 46, pp. 192–222 [Online]. DOI: 10.1016/j.ensm.2022.01.017.

Saloux, E. and Candanedo, J. A. (2021) ‘Model-based predictive control to minimize primary energy use in a solar district heating system with seasonal thermal energy storage', *Applied Energy*, vol. 291, p. 116840 [Online]. DOI: 10.1016/j.apenergy.2021.116840.

Sarbu, I. and Sebarchievici, C. (2014) 'General review of ground-source heat pump systems for heating and cooling of buildings', *Energy and Buildings*, vol. 70, pp. 441–454 [Online]. DOI: 10.1016/j.enbuild.2013.11.068.

Skarphagen, H., Banks, D., Frengstad, B. S. and Gether, H. (2019) 'Design Considerations for Borehole Thermal Energy Storage (BTES): A Review with Emphasis on Convective Heat Transfer', *Geofluids*, vol. 2019, pp. 1–26 [Online]. DOI: 10.1155/2019/4961781.

Tsvetanov, S., Papapostolu, T., Dimitrov, S. and Andonov, I. (2022) 'Smart Controller for Solar Thermal Systems', in Ahram (ed) *Human Interaction, Emerging Technologies and Future Systems V,* [S.l.], Springer International Publishing, pp. 618–624.

Villa, S. (2020) *A numerical methodology and analysis of borehole thermal energy storage performance* [Online]. Available at https://aaltodoc.aalto.fi/handle/123456789/47208.

Wellons, M. (2007) *The Stefan-Boltzmann Law* [Online]. Available at https://physics.wooster.edu/wp-content/uploads/2021/08/junior-is-thesis-web_2007_wellons.pdf.

Wootton, C. (2016a) 'Inter-Integrated Circuit (I2C)', in Wootton, C. (ed) *Samsung ARTIK Reference: The Definitive Developers Guide,* Springer Verlag, pp. 321–334.

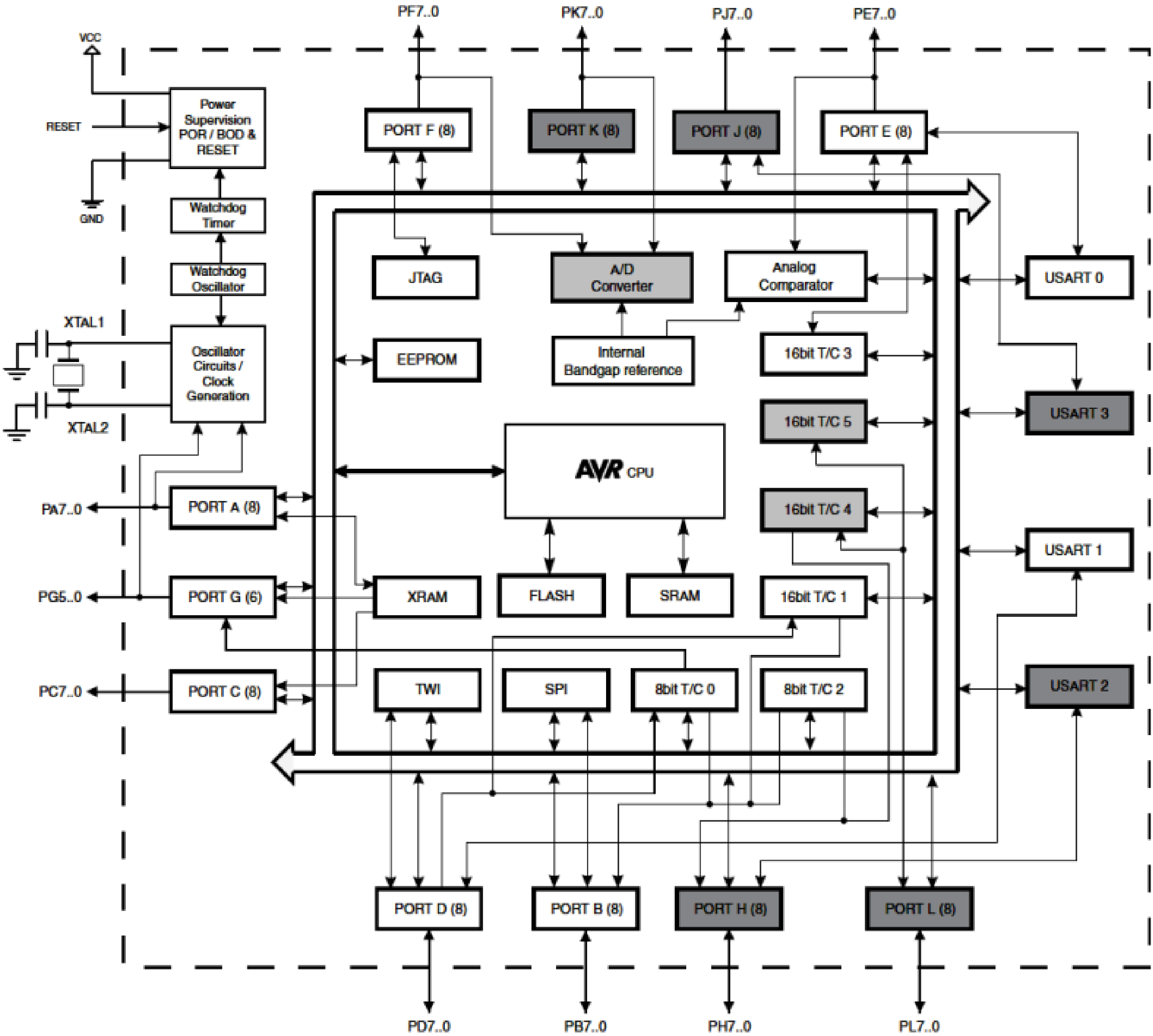
Wootton, C. (2016b) 'Serial Peripheral Interface (SPI)', in Wootton, C. (ed) *Samsung ARTIK Reference: The Definitive Developers Guide,* Springer Verlag, pp. 335–349.

Xu, Q. and Dubljevic, S. (2017) 'Model predictive control of solar thermal system with borehole seasonal storage', *Computers & Chemical Engineering*, vol. 101, pp. 59–72 [Online]. DOI: 10.1016/j.compchemeng.2017.02.023.

Yannakopoulos, J. (2003) *Hypertext transfer protocol: A short course* [Online]. Available at http://www.ece422.elimu.net/notes/html/http_short_course.pdf.

Zehnder, M. (2004) *Efficient air-water heat pumps for high-temperature lift residential heating, including oil migration aspects* [Online], Lausanne, EPFL. Available at https://www.researchgate.net/publication/37413723_Efficient_air-water_heat_pumps_for_high_temperature_lift_residential_heating_including_oil_migration_aspects.

AVR ATmega 2560 structure



Appendix 1. AVR processor structure (Răduca et al., 2016).

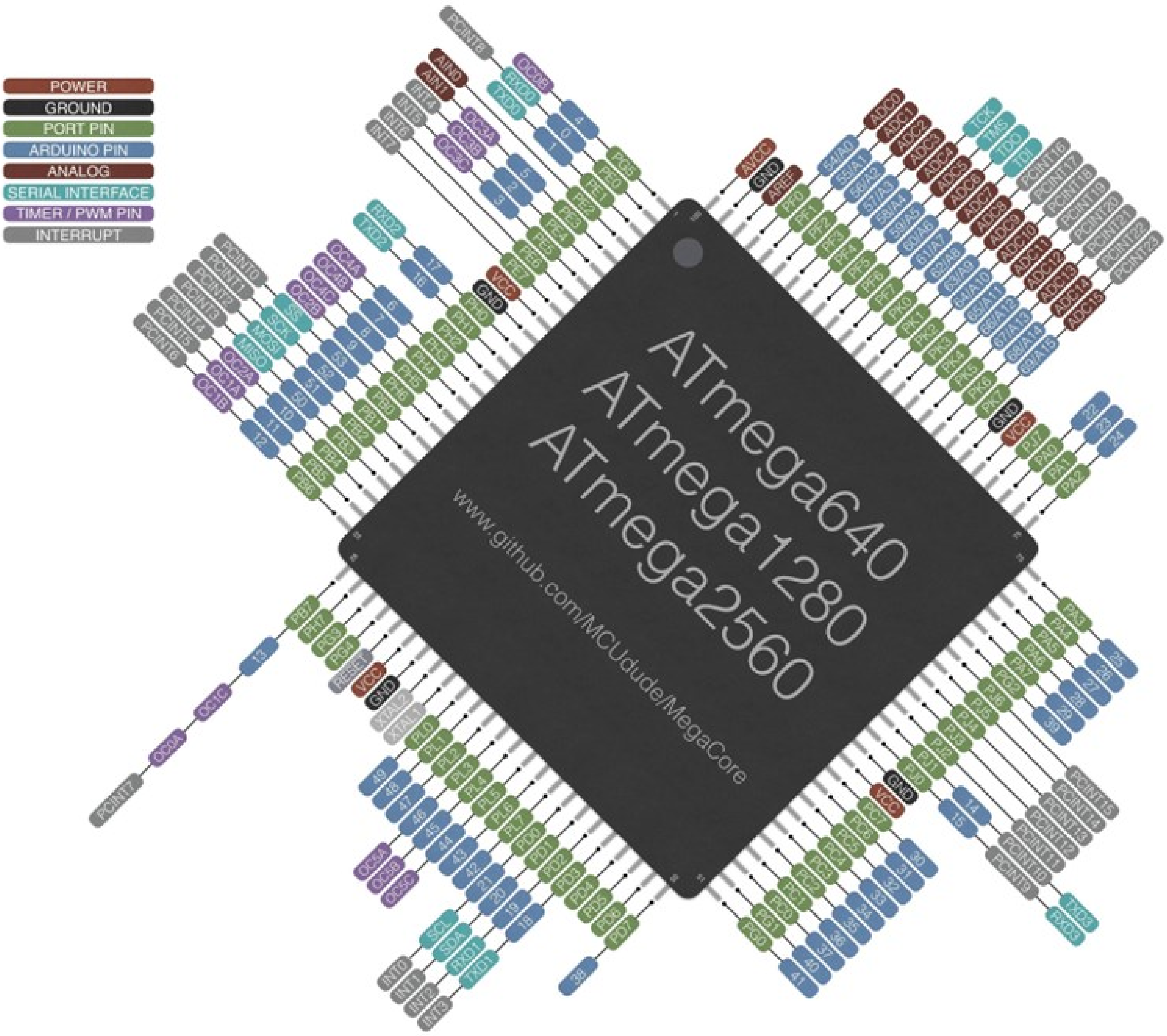
Case Study System Overview

Appendix 2.1. Solar thermal collector roof (Heliostorage Oy).

Appendix 2.2. BTES system under construction (Heliostorage Oy).

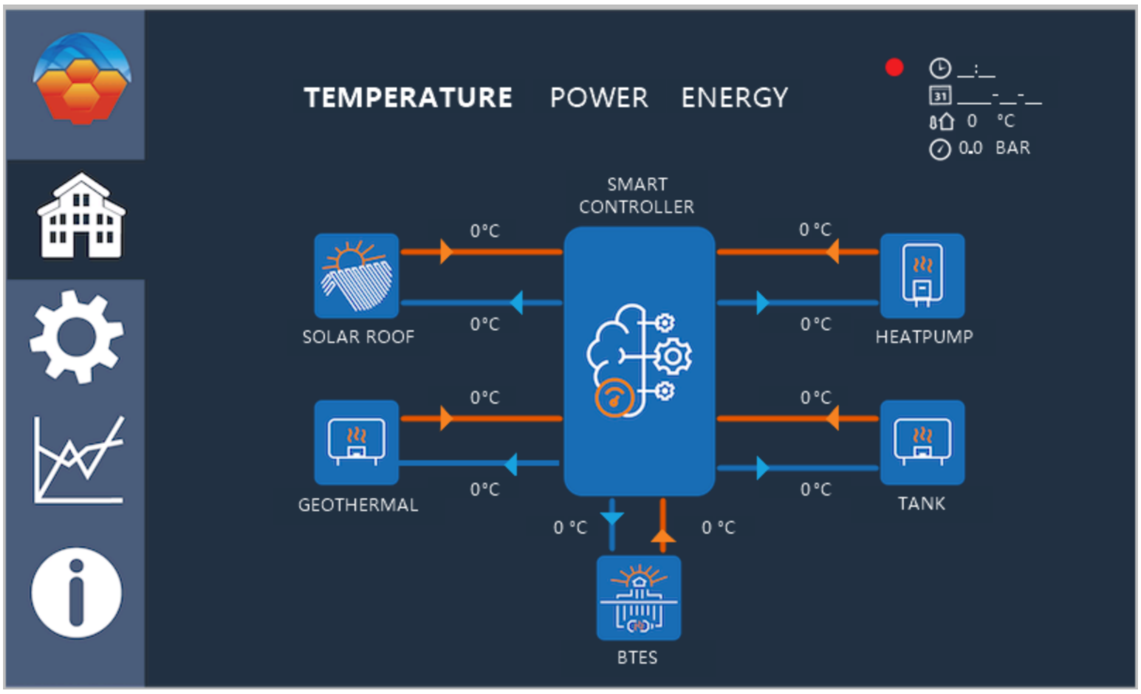
Appendix 2.3. Controller and piping system (Heliostorage Oy).

ATmega 2560 Pin Connection Diagram



Appendix 3. ATmega pinout MegaCore layout (MCUdude, 2023).

User Interface on HMI



Appendix 4. Display user interface for HMI on site (Heliostorage Oy).