Warsaw University of Technology

FACULTY OF
POWER AND AERONAUTICAL ENGINEERING

Institute of Heat Engineering

Bachelor's diploma thesis

in the field of study Power Engineering
and specialisation Power Engineering

Design and prototype construction of low pressure, low temperature steam boiler with superheating and web interface

Kuba Jałoszyński
student record book number 304457

thesis supervisor
PhD Eng. Karol Pietrak

Warsaw, 2023

Design and prototype construction of low pressure, low temperature steam boiler with superheating and web interface

Abstract

While performing scientific experiment, there can occur need for use of different kind of fluids, one of which is a steam. Despite type of a steam required, either wet, dry or superheated, typical supply sources such as co-generation power plant or similar facilities are rather not suitable for application in experiments, due to high variability of demanded supply. Big facilities typically operates and supply steam based on orders in advance, what is practically impossible to do in scientific facilities. Thus, such a steam generating custom unit was built in the laboratory of Thermodynamics Division, Warsaw University of Technology.

In this paper process of designing, constructing and testing is described, along with all design flaws which occurred, possible solutions to them, potential development path, etc. Throughout the project it was kept in mind that the unit is planned to be used in upcoming experiments, therefore safety was number one priority.

In first chapter project goals as well as objectives were briefly described along with demand for the project in the division. There is also available market offers review with available knowledge lookup on steam generating devices construction.

Second chapter is devoted to system design, with described decision making path, calculations, and all pre-construction work which had been done before the actual construction. Next, the construction process itself was briefly discussed. In both design and construction parts of work there were three main groups of tasks. These were classified as mechanical, electrical and programming parts.

Third chapter relates to whole software side of the generator. There are described hardware used, technologies applied and example user interface prepared for the demonstration of capabilities of the software.

Penultimate chapter - fourth, contains every operation and safety test before commissioning the device to nominal use. System characteristics were also measured and estimated at this point. Experimental data was compared with design values, and as it came up these differs by big margin, possible causes were pointed out.

The last chapter was left for conclusions after finishing the project, all noticed problems with recommendations for improvements and all entry-points for possible further development of system elements.

Keywords: Steam Generator, Experiment

Projekt i konstrukcja prototypu niskociśnieniowej, niskotemperaturowej wytwornicy pary ze sterowaniem bazującym na interfejsie sieciowym

Streszczenie

Przy wielu eksperymentach i badaniach naukowych zachodzi potrzeba użycia ogólnopojętych płynów o ustalonych parametrach. Jednym z takich płynów jest para wodna w jednym z możliwych rodzajów, pary mokrej, pary nasyconej, pary przegrzanej lub innych. Zazwyczaj do procesu wytwarzania pary wykorzystuje się duże gabarytowo kotły parowe opalane paliwami kopalnymi, a tak zwaną parę technologiczna, co oznacza parę o ściśle ustalonych i utrzymywanych parametrach, pozyskuje się z dużych bloków pobliskich elektrociepłowni lub podobnych instytucji technicznych na zamównienie.

W przypadku eksperymentu naukowego, gdzie zapotrzebowanie na parę jest znacznie mniejsze, a przede wszystkim bardzo nieregularne, nie jest możliwe skorzystanie z takiego źródła. Dodatkowo po rozeznaniu dostępnych ofert rynkowych, okazało się, że zakup takiego urządzenia byłby znaczącą inwestycją co z kolei mogło by ograniczyć możliwe wydatki na pozostałe części eksperymentów wykorzystujących taką wytwornice. Z tego powodu została podjęta decyzja o budowie samodzielnej takiego urządzenia w labolatorium wymiany ciepła i termodynamiki, w Zakładzie Termodynamiki Politechniki Warszawskiej.

W niniejszej pracy został przedstawiony proces projektowania i powstawania wytwornicy pary wraz z oprogramowaniem używanym do sterowania, opisanymi napotkanymi problemami, proponowanymi rozwiązaniami jak i sugerowanymi kolejnymi krokami rozbudowy. Ze względu na charakter urządzenia i jego planowane wykorzystywanie w labolatorium do nadchodzących badań, ekstremalnie ważnym czynnikiem było bezpieczeństwo użytkowania. Ze względu na nieregularność konstrukcji podjęto również stosowne przygotowania przed budową jak i wykonano testy bezpieczeństwa, które są opisane w rozdziale czwartym.

W rozdziale pierwszym zostały opisane cele projektu, przyczyny powstania projektu jak i streszczony został przegląd ofert rynkowyh. Rozdział zakończony jest przedstawieniem filozofii obliczeń i wzorów ogólnych zastosowanych później do wyznaczenia parametrów urządzenia.

Rozdział drugi zawiera opis procesu projektowania i konstrukcji urządzenia. Projekt urządzenia został rozdzielony na 3 główne kategorie: elementy mechaniczne (rama urządzenia, dobór materiałów, etc.), elementy elektroniczne i elektryczne (tor zasilania, połączenia elektryczne (schemat uproszczony), etc.) i część poświęconą oprogramowaniu (dokładniej opisane w rozdziale 3).

Kolejny rozdział został poświęcony opisowi stworzonego oprogramowania. Został opisany użyty mikro komputer pod względem parametrów, użyte technologie oraz stworzony interfejs użytkownika.

Czwarty rozdział zawiera zebrane dane, zaprezentowane w formie wykresów z omówieniem uzyskanych wyników. Zostały w nim również opisane wykonane testy bezpieczeństwa.

Ostatni rozdział został poświęcony na podsumowanie pracy, opis proponowanych poprawek i wyciągnięte wnioski.

Słowa kluczowe: Wytwornica pary, eksperyment naukowy

Contents

Praca powstała w ramach realizacji projektu pt.: Badania wymiany ciepła i wilgoci w wielowarstwowym ubraniu ochronnym poddanym obciążeniom cieplnym i parowym, nr. projektu: 2020/37/B/ST8/04012 finansowanego przez Narodowe Centrum Nauki.

The work was created as part of the project: Research on heat and moisture exchange in multi-layer protective clothing subjected to heat and steam loads, no. project: 2020/37/B/ST8/04012 financed by the National Science Centre.

Chapter 1

Introduction

This work describes the construction of a low pressure, low temperature steam generator, built in the laboratory of thermodynamics division. Project has high utility value as it is planned to use it for further research in the division. It was kept in mind that this project should be as simple as possible for cost reduction, thus attempt made was based on either customizing and recreating existing construction or putting together existing solutions whenever it was possible.

1.1 Goals of the project

The goal was to design and build custom steam generator as a cheaper alternative to more expensive commercial products available on the market, while achieving higher nominal parameters (steam temperature, steam pressure, steam mass flow rate). One of the objectives was to create a universal control system, which allows to create custom control software based on universal web interface for easier customization in the future.

1.2 Demand for the project

Utility of the project, as it was mentioned in previous paragraph is very high as it is planned to operate the steam generator in upcoming experiments. As it is shown in the next section, commercial units are not suitable for such a use case due to strictly limited nominal parameters without easy way to manipulate those. Units which satisfied those criteria were exceeding significantly estimated budget for the project. What is more most of commercial units are driven by direct controls - user has to be on side of the device which would create hazard if system malfunctioned. Based on the fact that the machine will be operated by third party the safety was one of the highest priorities during whole project. Key factor considered while designing the control system was to keep a user away from the device, by allowing fully remote control of the system, reducing any potential risk of system failure.

1.3 Market offers overview

In pre-design phase market offers were reviewed for commercial plug&play units. Main limitation was that the unit has to be driven by electricity, considering that major part of available units were driven mainly by natural gas or diesel fuel possible number of choices plummet. Taken into consideration had been 3 steam generators manufacturers:

- JUMAG[1]
- STIGEN[2]
- BABCOC WANSON[3]

Detailed offers on polish market of available models of each company are available in the appendix 1.

*I*t quickly came up that electrical units parameters would be hard to match with minimum required. Best and only ones fitting the criteria were JUMAG's models EDI20 and EDI40 along with STIGEN models EKP30 and EKP60. Main flaws of these units were too low mass flow rate and no option for superheating the steam. The price was also an issue, companies were not able to provide actual prices without very detailed survey which was not possible to be provided as no precise calculations of system parameters had been done. Informal price range estimation was in range of 30 000 to 50 000 [EUR[4]], which was significantly exceeding project's budget.

Finally, after research on parts availability, prices, fees, it was estimated that handmade construction would be in range of 10 000 to 30 000 [PLN[5]], thus further work was fully directed to the development of custom design.

1.4 Available knowledge review

Small factor steam boilers are commonly used for heating, power generation, and process applications in a variety of industries. In recent years, there has been growing interest in the development of small factor steam boilers that are more efficient, reliable, and environmentally friendly. This has led to significant advancements in steam boiler technology, including improvements in design, construction, and operation.

1.4.1 Working medium - water vapour

Steam is a working fluid commonly used in power generation and industrial processes. Steam main advantage is high energy density compared to many other working fluids. This means that a small amount of steam can carry a large amount of energy. Steam is non toxic and rather non reactive medium. Most important is steam (water vapour) availability and ease of handling with it, while it

[1] Distribution in Poland provided by *Eltar-M.B. Tatarczyk Sp. z o.o.*
[2] Manufacturer is also the distributor. *STIGEN Sp. z o.o.*
[3] Manufacturer is also the distributor. *Babcock Wanson Polska Sp. z o.o.*
[4] Euro
[5] Polish zloty

can be generated using a variety of fuels. Biggest disadvantage of steam is it's cohesiveness which increases maintenance cost and potentially lead to mechanical failures of a device. Overall, steam is a reliable and widely used working fluid that is well-suited for many industrial applications. Its unique properties make it an ideal choice for power generation and heat transfer processes.

1.4.2 Steam Generator

The main elements of a small factor steam boiler include a combustion chamber or furnace, a heat exchanger supplying heat to the fluid, typically with either high temperature flue gasses[6] or higher temperature pressurized water[5] as a heat source or as in case of this project an electrical heater directly heating up the water, safety systems, and a control system. Most common approach to steam generator construction is continuous flow system with no steam accumulation, but it requires much greater heat supply and this kind of the device typically operates at much higher parameters. In this project device operates intermittently, at first water is boiled and then saturated steam is superheated, secondly the steam is being released with no heat supply.

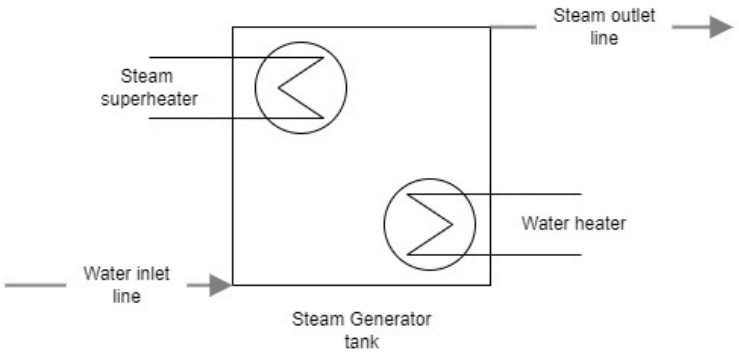


Figure 1. Simplified steam generator diagram

1.4.3 Steam boilers classification

Steam boilers are classified most often based on working medium - steam, parameters. Most widely accepted classification categorizes steam boilers into 4 main groups (Classification: American Society of Mechanical Engineers (ASME) Boiler and Pressure Vessel Code Section I[2], complementary with European Standard EN 12952 or International Association for the Properties of Water and Steam Technical Guidance Document):

Low-pressure boilers: These boilers operate at a steam pressure of up to 15 pounds per square inch (psi) or 1.03 megapascals (MPa) and a temperature of 250°F (121.11°C) or below.

High-pressure boilers: Boilers operating at a steam pressure above 15 psi or 1.03 MPa and a temperature of 250°F (121.11°C) or above.

Subcritical boilers: Boilers operating below the critical pressure of water (3,208 psi or 22.06 MPa) and a temperature of 540°C or below.

Supercritical boilers: These boilers operate above the critical pressure of water (3,208 psi or 22.06 MPa) and a temperature of 540°C or above.

Steam boilers also can be classified based on other various factors, such as the type of fuel used, the type of construction, the mode of operation, and the application. Most common classification as partly showed by M. D. Zidane[8] is by fuel used and type of construction, namely:

Fire Tube Boilers - these boilers have a cylindrical shell with a large furnace and are commonly used for low-pressure steam production.

Water Tube Boilers - here the water circulates through tubes that are surrounded by hot gases. They are commonly used for high-pressure steam production.

Electric Boilers - boilers which use electricity as their source of energy and are commonly used in small-scale applications, also used in this project.

Biomass Boilers - these boilers use biomass as their source of fuel, such as wood chips or agricultural waste.

Oil and Gas Boilers - boilers which use oil or gas as their source of fuel and are commonly used in industrial applications.

Based on above classifications device described in this paper is classified in border region of low and high pressure classification. Expected operational pressure is $10[\mathrm{bar}]$ or lower, thus it eventually fell into low pressure, low temperature group. Important note is that these classification varies a lot, thus using different division and having in mind that it has near border parameters it could be classified into other group.

1.4.4 System calculations - general formulas

Calculations were splited into two main steps: first is the water heating, evaporation and eventually steam superheating, with assumed no mass flow in or out the system and no work done. Second is steam release were no water evaporation and no work done were assumed.

Theoretical system parameters are eventually compared with experimental data. This was limited by number of installed sensors and theirs accuracy. Part of parameters, ex. ambient temperature before the tests were measured in discontinuous manner.

Steam generated

First part of device operation is steam generation. It is assumed that there is no work done by the system and system itself is closed. Taken into account are initial point with no steam in the system and final point - moment when heaters were turned off. To calculate generated steam mass having pressure measurements, equation based on first thermodynamics law[3] is derived:

$$\mathrm{dU} = \mathrm{dQ} - \mathrm{dW} + \Sigma_{\mathrm{j}=1}^{\mathrm{n}} \mathrm{i_j dm_j} \tag{1}$$

assuming:

- $\mathrm{dW} = 0[\mathrm{W}]$
- $\mathrm{dm} = 0[\mathrm{kg}]$ - constant system mass
- $\mathrm{m_{steam\_initial}} = 0[\mathrm{kg}]$ - cold start

with:

$$\mathrm{U_{water}} = \mathrm{m_{water} i_{water}} \tag{2}$$

$$\mathrm{U_{steam}} = \mathrm{m_{steam}}(\mathrm{i_{steam}} - \mathrm{p}\nu_{\mathrm{steam}})$$

further integrating:

$$\mathrm{U_2} - \mathrm{U_1} = \mathrm{Q_{heating}} - \mathrm{Q_{loss}}$$

$$\mathrm{U_1} = \mathrm{m_{water1} i_{water1}}$$

$$\mathrm{U_2} = (\mathrm{m_{water1}} - \mathrm{m_{steam2}})\mathrm{i_{water2}} + \mathrm{m_{steam2}}(\mathrm{i_{steam2}} - \mathrm{p_2}\nu_{\mathrm{steam2}})$$

$$\mathrm{Q_{heating}} = \mathrm{P_{heaters} t_{heating}}$$

$$\mathrm{Q_{loss}} = \mathrm{Q_{convection}} + \mathrm{Q_{radiation}}$$

after substitution and reorganization one obtains:

$$\mathrm{m_{steam\_generated}} = \mathrm{m_{steam2}} = \frac{\mathrm{Q_{heating}} - \mathrm{Q_{loss}} + \mathrm{m_{water1}}(\mathrm{i_{water1}} - \mathrm{i_{water2}})}{\mathrm{i_{steam2}} - \mathrm{i_{water2}} - \mathrm{p_2}\nu_{\mathrm{steam2}}} \tag{3}$$

Steam mass flow - steam release

Second part of device operation is steam release phase. In this phase there is no heating and still occur heat losses thru non insulated parts of the device and energy loss due to outflow of steam mass but no work is being done. Assumption was made that there is no more water evaporation as it was impossible to calculate evaporation rate with available data, this means that presented solution is more an approximation not an exact calculation. This leads, based on first law of thermodynamics described by J. Banaszek et al.[3] to:

$$\mathrm{U_3} - \mathrm{U_2} = -\mathrm{Q_{loss}} - (\mathrm{m_{steam3}} - \mathrm{m_{steam2}})\mathrm{i_{steam2}} \tag{4}$$

using general internal energy formulas:

$$\mathrm{U_2} = \mathrm{m_{water2} i_{water2}} + \mathrm{m_{steam2}}(\mathrm{i_{steam2}} - \mathrm{p_2}\nu_{\mathrm{steam2}}) \tag{5}$$

$$\mathrm{U_3} = \mathrm{m_{water3} i_{water3}} + \mathrm{m_{steam3}}(\mathrm{i_{steam3}} - \mathrm{p_3}\nu_{\mathrm{steam3}}) \tag{6}$$

$$\mathrm{Q_{loss}} = \mathrm{Q_{convection}} + \mathrm{Q_{radiation}}$$

and assuming $\mathrm{dm_{water}} = \mathrm{m_{water3}} - \mathrm{m_{water2}} = 0$:

$$\mathrm{m_{steam3}} = \frac{\mathrm{m_{steam2}}(2\mathrm{i_{steam2}} - \mathrm{p_2}\nu_{\mathrm{steam2}}) - \mathrm{m_{water}}(\mathrm{i_{water3}} - \mathrm{i_{steam2}}) - \mathrm{Q_{loss}}}{\mathrm{i_{steam3}} + \mathrm{i_{steam2}} - \mathrm{p_3}\nu_{\mathrm{steam3}}} \tag{7}$$

finally:

$$\mathrm{dm_{steam}} = \mathrm{m_{steam3}} - \mathrm{m_{steam2}} \tag{8}$$

Chapter 2

System design

Actual system design begun with system characteristic values calculation and estimation. Taken into account were power limit of 3 phase domestic grid connection in the division of thermodynamics laboratory, with maximal power not exceeding $\mathrm{P_{max}} = 40[\mathrm{kW}]$.

2.1 Required system characteristics

System parameter calculations are based on required steam parameters (design assumed parameters) with minimum to meet parameters considered as successful and maximum parameters, assumed as a safety limit, dependent on material strength (PN16 cast iron piping) are shown in table 1.

2.2 Initial system parameters calculation

Initial calculations were based on design parameters as well as basic review of available parts.

2.2.1 Required volume

Required volume of the device was calculated using:

$$\mathrm{V_{steam}} = \dot{\mathrm{m}}_{\mathrm{steam}} \nu_{\mathrm{steam\_10bar}} \mathrm{t_{operation}} \tag{9}$$

where (assumed generated steam $\mathrm{x} = 1$, values obtained using pyXSteam[1] Python module):

- $\dot{\mathrm{m}}_{\mathrm{steam}} = 12.5[\frac{\mathrm{kg}}{\mathrm{h}}] = 0.00347[\frac{\mathrm{kg}}{\mathrm{s}}]$
- $\nu_{\mathrm{steam\_10bar}} = 0.19[\frac{\mathrm{m^3}}{\mathrm{kg}}]$
- $\mathrm{t_{operation}} = 30[\mathrm{s}]$

which gives $\mathrm{V_{steam}} = 0.0198[\mathrm{m^3}]$, $\mathrm{m_{steam}} = 0.1[\mathrm{kg}]$. It was assumed that during steam release there should be water in liquid state in around 25[%] of steam volume, meaning $\mathrm{V_{water}} = 0.25\mathrm{V_{steam}} = 0.00495[\mathrm{m^3}]$, this assumption was made to ensure safety and avoid potential problems

[1] https://pyxsteam.readthedocs.io/en/latest/

Name	Value	Unit
Nominal parameters		
Pressure	10	[bar]
Temperature	200	[°C]
Mass flow rate	12.5	$[\frac{\mathrm{kg}}{\mathrm{h}}]$
Operation time (steam release total time)	30	[s]
Steam characteristics	Superheated steam	-
Minimum parameters		
Pressure	10	[bar]
Temperature	180	[°C]
Mass flow rate	10.0	$[\frac{\mathrm{kg}}{\mathrm{h}}]$
Steam characteristics	Saturated steam	-
Maximum parameters		
Pressure	16	[bar]
Temperature	220	[°C]

Table 1. Nominal, minimum and maximum generated steam parameters desired

with water heater overheating (if exposed from the water). This gives total volume equal to $\mathrm{V_{total}} = 0.02475[\mathrm{m^3}]$.

2.2.2 Required power

Power required was calculated using equation 3 with assumption of no heat losses, heat up time (arbitrarily assumed, reasonable value) of $\mathrm{t_{heater}} = 1800[\mathrm{s}]$ and $\mathrm{m_{steam}} = 2[\mathrm{kg}]$ (mass assumed well above required value to ensure no problems with heat supply - left 20x margin of error) rearranging equation one obtains:

$$\frac{\mathrm{Q_{heating}}}{\mathrm{t_{heater}}} = \mathrm{P_{heater}} = \frac{\mathrm{m_{steam2}}(\mathrm{i_{steam2}} - \mathrm{i_{water2}} - \mathrm{p_2}\nu_{\mathrm{steam2}}) + \mathrm{m_{water1}}(\mathrm{i_{water2}} - \mathrm{i_{water1}}}{\mathrm{t_{heater}}} \tag{10}$$

using assumptions and values from subsection Required volume and expanding it with initial pressure to be equal $\mathrm{p_{inital}} = 0.1[\mathrm{bar}]$ - sub-vacuum , assumed $\mathrm{m_{water}} = 10[\mathrm{kg}]$, fluids parameters for $\mathrm{p} = 10[\mathrm{bar}]$ (obtained using pyXSteam Python module):

- $\mathrm{m_w} = 10[\mathrm{kg}]$
- $\mathrm{p_{final}} = 10[\mathrm{bar}]$
- $\nu_{\mathrm{steam\_10bar}} = 0.19[\frac{\mathrm{m^3}}{\mathrm{kg}}]$

- $\mathrm{i}_{\mathrm{water\_0.1bar}} = 197.39[\frac{\mathrm{kJ}}{\mathrm{kg}}]$
- $\mathrm{i}_{\mathrm{water\_10bar}} = 762.68[\frac{\mathrm{kJ}}{\mathrm{kg}}]$
- $\mathrm{i}_{\mathrm{steam\_10bar}} = 2777.12[\frac{\mathrm{kJ}}{\mathrm{kg}}]$
- $\mathrm{Q}_{\mathrm{loss}} = 0[\mathrm{kJ}]$ - design assumption, no reasonable estimation is possible at this point

it is obtained $\mathrm{P}_{\mathrm{heater}} = 5.17[\mathrm{kW}]$.

2.2.3 Available heaters power and it's characteristic times

Heater selection was made based on heater power (the higher the better - increased system dynamics), heater power (the lower the better - power supply limits) and *characteristic times* 1) steam generation starts, where $\mathrm{m}_{\mathrm{steam}} >= 0[\mathrm{kg}]$ and 2) steam generation ends, where $\mathrm{m}_{\mathrm{steam}} = 2[\mathrm{kg}]$ - designed maximal mass of generated steam, lower water mass could lead to dropping water level below water heater level and eventually damage it from overheating). Characteristic times shows predicted device operational times (from cold start to ready to use). Lower bound of characteristic time is strictly theoretical value as in sub-vacuum pressure water supplied from atmospheric pressure with start to cavitate, thus this value is treated only as simple estimation rather than accurate calculation. Using equation 3 steam mass function of time can be derived dependent on available heaters power and with given example (parameters values equals to listed ones in sections Required power and Required volume) it is obtained: Saturation temperature $\mathrm{T}_{\mathrm{sat\_10[bar]}} = 179.89[\mathrm{C}]$, and $\mathrm{m}_{\mathrm{steam}} = 5.48 * 10^{-4}\mathrm{Pt} - 1.8923$ - describes only time required for water to start boiling as simple estimation, to obtain accurate calculations cavitation of the water should be taken into account:

1. $\mathrm{P}_{\mathrm{heater}} = 2[\mathrm{kW}]$

$$\mathrm{m}_{\mathrm{steam}} = 0.0011\mathrm{t} - 3.082 \tag{11}$$

2. $\mathrm{P}_{\mathrm{heater}} = 6[\mathrm{kW}]$

$$\mathrm{m}_{\mathrm{steam}} = 0.0033\mathrm{t} - 3.082 \tag{12}$$

3. $\mathrm{P}_{\mathrm{heater}} = 8.7[\mathrm{kW}]$

$$\mathrm{m}_{\mathrm{steam}} = 0.0048\mathrm{t} - 3.082 \tag{13}$$

With minimal time to obtain steam is calculated for each (heat up water to boiling point):

1. $\mathrm{t}_{\mathrm{min}} = 2801.8[\mathrm{s}]$
2. $\mathrm{t}_{\mathrm{min}} = 933.94[\mathrm{s}]$
3. $\mathrm{t}_{\mathrm{min}} = 642.08[\mathrm{s}]$

and minimal time to completely evaporate water ($\mathrm{m}_{\mathrm{steam}} = 2[\mathrm{kg}]$):

1. $\mathrm{t}_{\mathrm{min}} = 4620[\mathrm{s}]$
2. $\mathrm{t}_{\mathrm{min}} = 1540[\mathrm{s}]$

3. $t_{min} = 1181.86[s]$

As expected, the highest power gives the smallest heat up times, but considering relatively small differences between 2. and 3. while giving significant power savings, heater with $P_{heater} = 6[kW]$ was considered as best option with operational time in range $t_{6kW} \in [933.94; 1540][s]$.

2.3 Control system and user interface

Software is based on Django - Python web framework. To speed up code creation, user interface was made using HTML with minimal CSS formatting. The overall design must have been the simplest possible, while providing sufficient readability of control panel and system parameters. More information about how exactly software is created is available in chapter 4.

2.4 Electrical & electronics design

Main assumption during design process was using ready-to-use modules minimizing work required. There are two main electrical parts of the system: power box - with all high voltage lines, with control signal lines connecting it with a main computer unit, electronics box - storing all low voltage devices.

2.4.1 Power lines

System is connected to the 3 phase grid. Power control is meant to be implemented using solid state relays (SSR) to directly connect or disconnect elements to the grid. There are two low voltage supply lines, first is independent, always on, 5[V] supply line for Raspberry Pi computer. It is designed to operate separately from the rest of the system to maintain a control even in case of emergency shutdown (total power cutoff). The second is 12 [V] line supplying power to the rest of electronics. Simplified diagram is presented below, on figure 2.

2.4.2 Transistor switches module

All SSR elements requires minimal signal voltage of 4 [V], but due to maximal 3.3 [V] on Raspberry Pi output there occur need to mount additional amplifier circuit. BJT transistor were used as it's characteristics match required ones (low Base current to activate, high β coefficient). Final circuit schematic is presented in appendix 4.

2.4.3 Low voltage devices

Low voltage electronics were designed to be placed in separate container to avoid any potential interference from power lines as well as to protect these from shorting circuit with high voltage lines. Pt100 temperature sensors are connected via MAX31865 DAC converted to Main computational unit

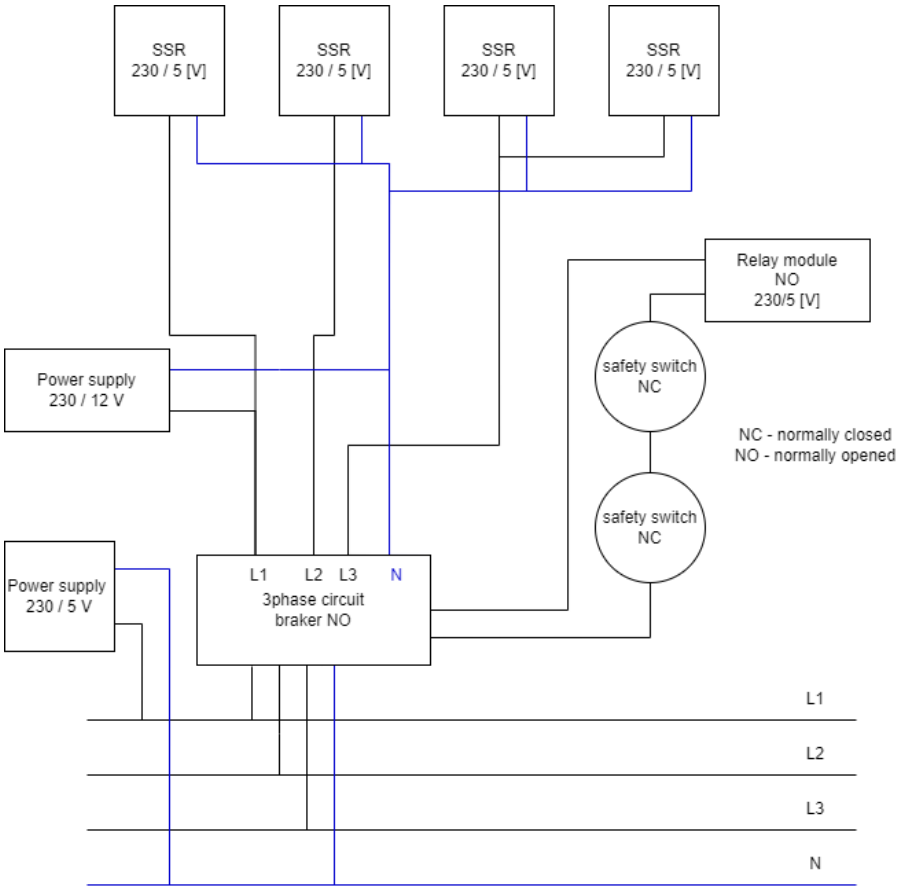


Figure 2. High voltage connections and supply lines

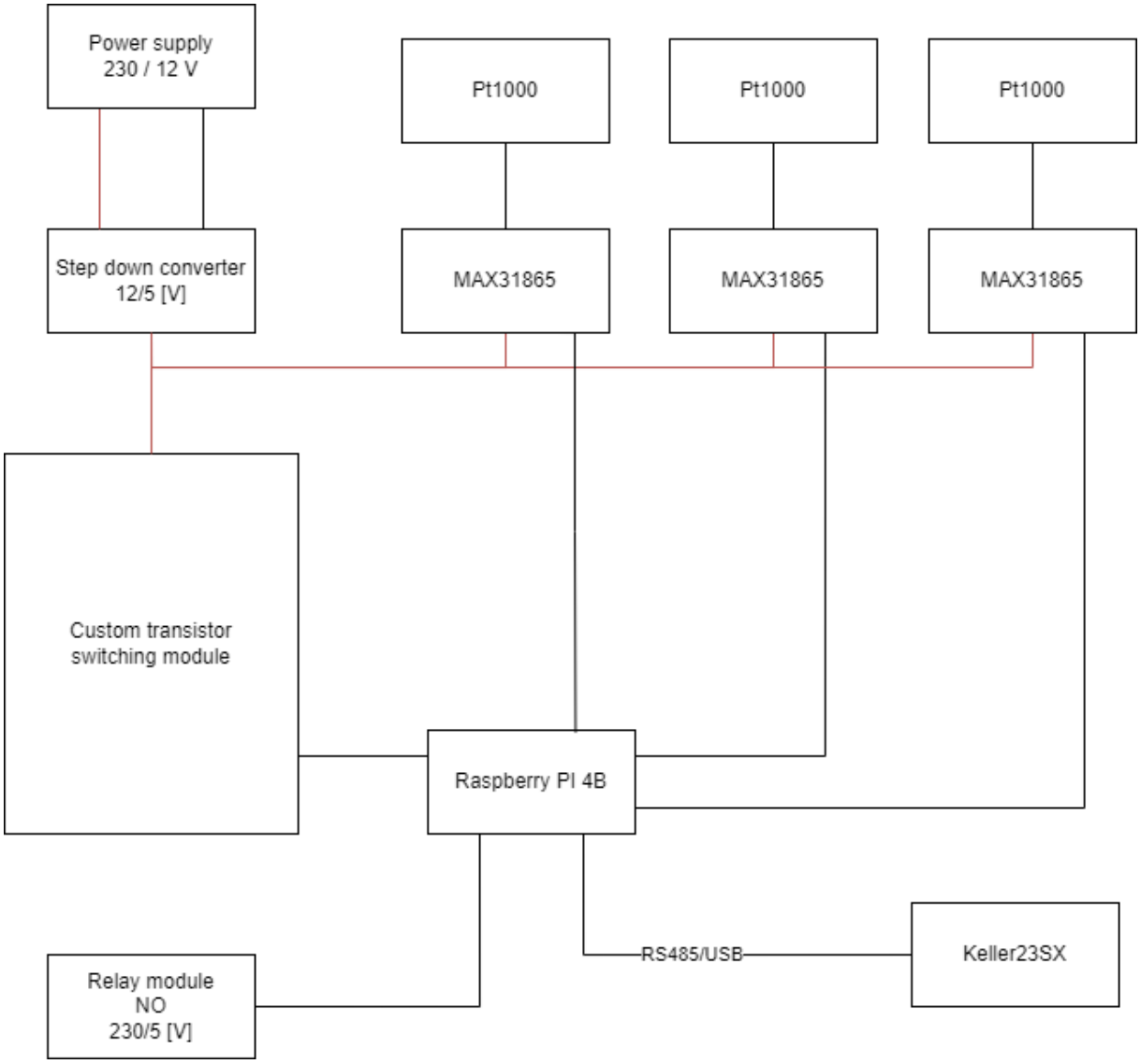


Figure 3. Low voltage connections

(MCU) with SPI interface. Keller23SX pressure sensor uses RS485 interface, transistor switching module is connected with 5 independent pins to the MCU, each controls different heater or valve and finally MCU controls relay circuit breaker with single pin. Low voltage connections are shown on figure 3.

2.4.4 Device model

To estimate overall device dimensions and space required, simple device body 3D model was created (presented on figure 5). Device schematic is available on figure 4. On schematic there are marked positions of mounter elements: heaters, sensors, etc. In a final build there are part of the device which were not insulated properly due to problems that occured during assembly. These surfaces

are marked on schematic with red boxes. Complex shapes such as pipe collars, were simplified and surface area was calculated for simpler shapes, ex. cylinder. This simplification was made due to fact that potential calculations with use of surface area (heat losses calculations) were used to rough value estimation rather than exact calculation. Ultimately all exposed areas should be insulated.

2.4.5 Device assembly

Based on initial calculations and parameters estimation supplied with market research for parts available, device assembly was started. Due to miscalculations, onsite project changes and other minor reasons selected insulation for the device did not fit, thus part of the surfaces were left non insulated. Assembled device is presented of figure 6. Non insulated surfaces are marked on figure 4.

Actual device volume (with installed heaters, etc.) was measured experimentally and came up to be equal $\mathrm{V}_{\mathrm{total\_assembled}} = 0.02329[\mathrm{m}^3]$. Assembled electric and electronics box is presented on figure 7.

2.4.6 Heat losses calculation

Finished the device assembly it was possible to estimate heat losses of the device. As was mentioned in previous subsection, the device lacks insulation on part of it's surfaces (marked on figure 4). Due to lack of direct continuous device surface temperature measurement assumption was made that:

$$\mathrm{T}_{\mathrm{surface}} = \mathrm{T}_{\mathrm{steam}}(1 - \epsilon(\Delta\mathrm{T})) \tag{14}$$

where $\epsilon(\Delta\mathrm{T})$ and $\Delta\mathrm{T} = \mathrm{T}_{\mathrm{steam}} - \mathrm{T}_{\mathrm{ambient}}$ is some (predicted that small) temperature drop. This drop was estimated with several discrete measurements averaging at$\mathrm{T}_{\mathrm{steam}} = 154[\mathrm{C}]$, $\mathrm{T}_{\mathrm{surface}} = 152[\mathrm{C}]$ and $\mathrm{T}_{\mathrm{ambient}} = 24[\mathrm{C}]$ measured on heated device in steady state, giving $\epsilon(\Delta\mathrm{T} = 154 - 24) = 1.32[\%]$. It is also expected that $\epsilon(24 - 24) = 0[\%]$. Using these two point linear approximation of temperature difference was derived as:

$$\epsilon(\Delta\mathrm{T}) = 1.01558 * 10^{-4}\Delta\mathrm{T}[\%] \tag{15}$$

Heat losses estimation was based on Cengel[4] approximation formulas for convection and radiation.

Heat radiation was estimated using:

$$\mathrm{Q}_{\mathrm{rad}} = \sigma\epsilon(\mathrm{T}^4_{\mathrm{surface}} - \mathrm{T}^4_{\mathrm{ambient}})\mathrm{A}_{\mathrm{tot}} \tag{16}$$

and heat convection:

$$\mathrm{Q}_{\mathrm{conv}} = \alpha\mathrm{A}_{\mathrm{tot}}(\mathrm{T}_{\mathrm{surface}} - \mathrm{T}_{\mathrm{ambient}}) \tag{17}$$

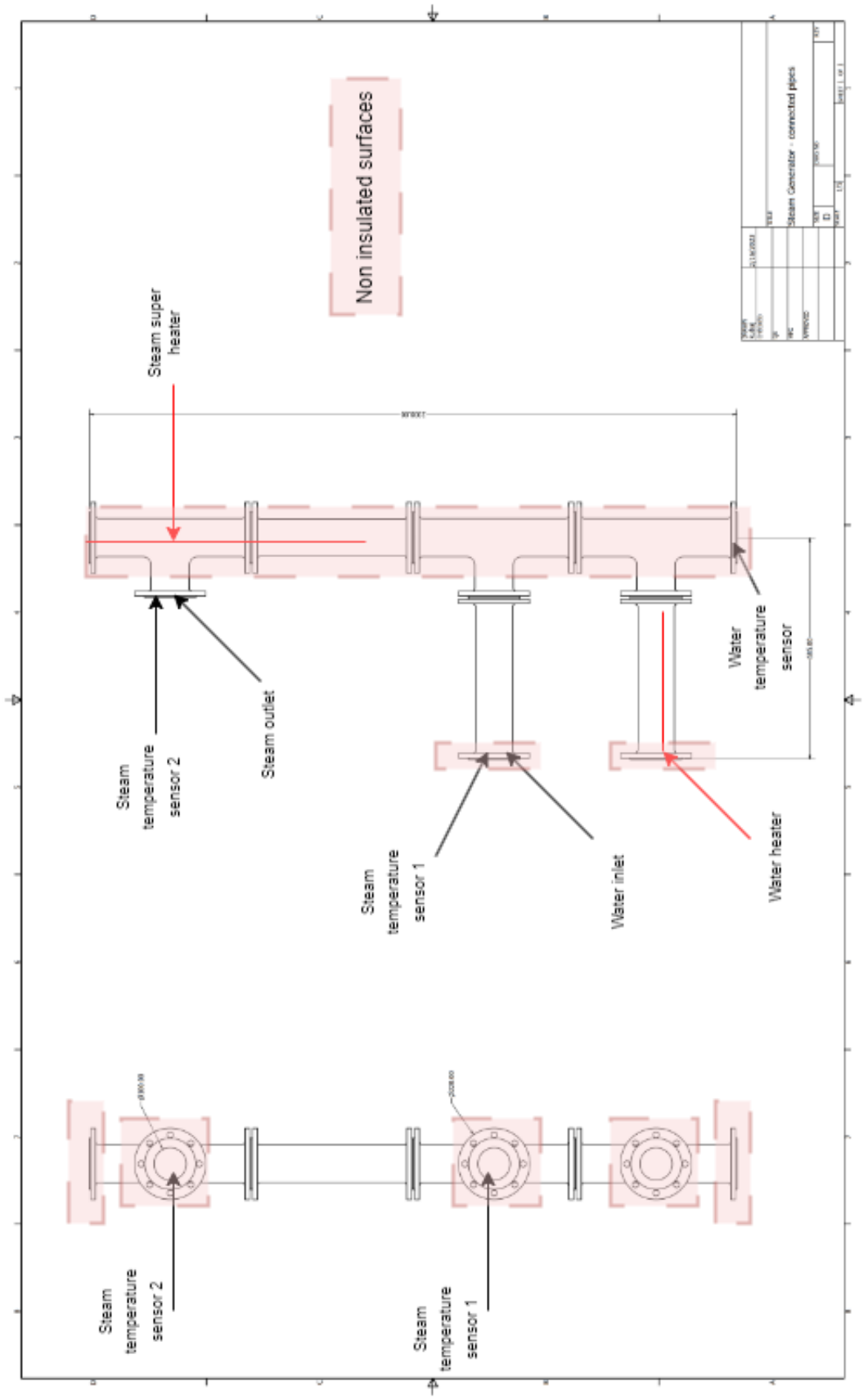


Figure 4. Device schematic with marked positions of mounted elements

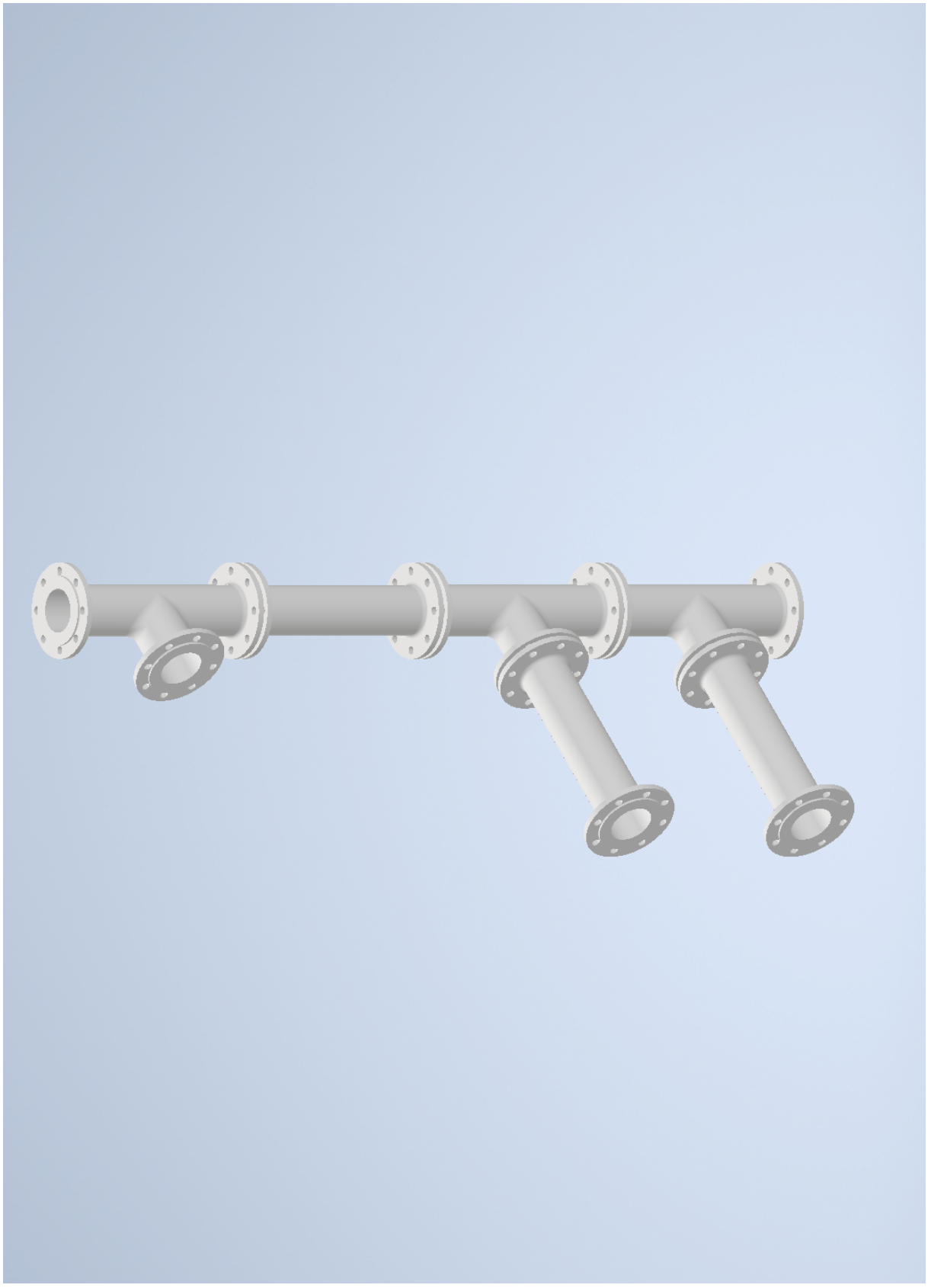


Figure 5. Device body 3D model

Figure 6. Assembled device

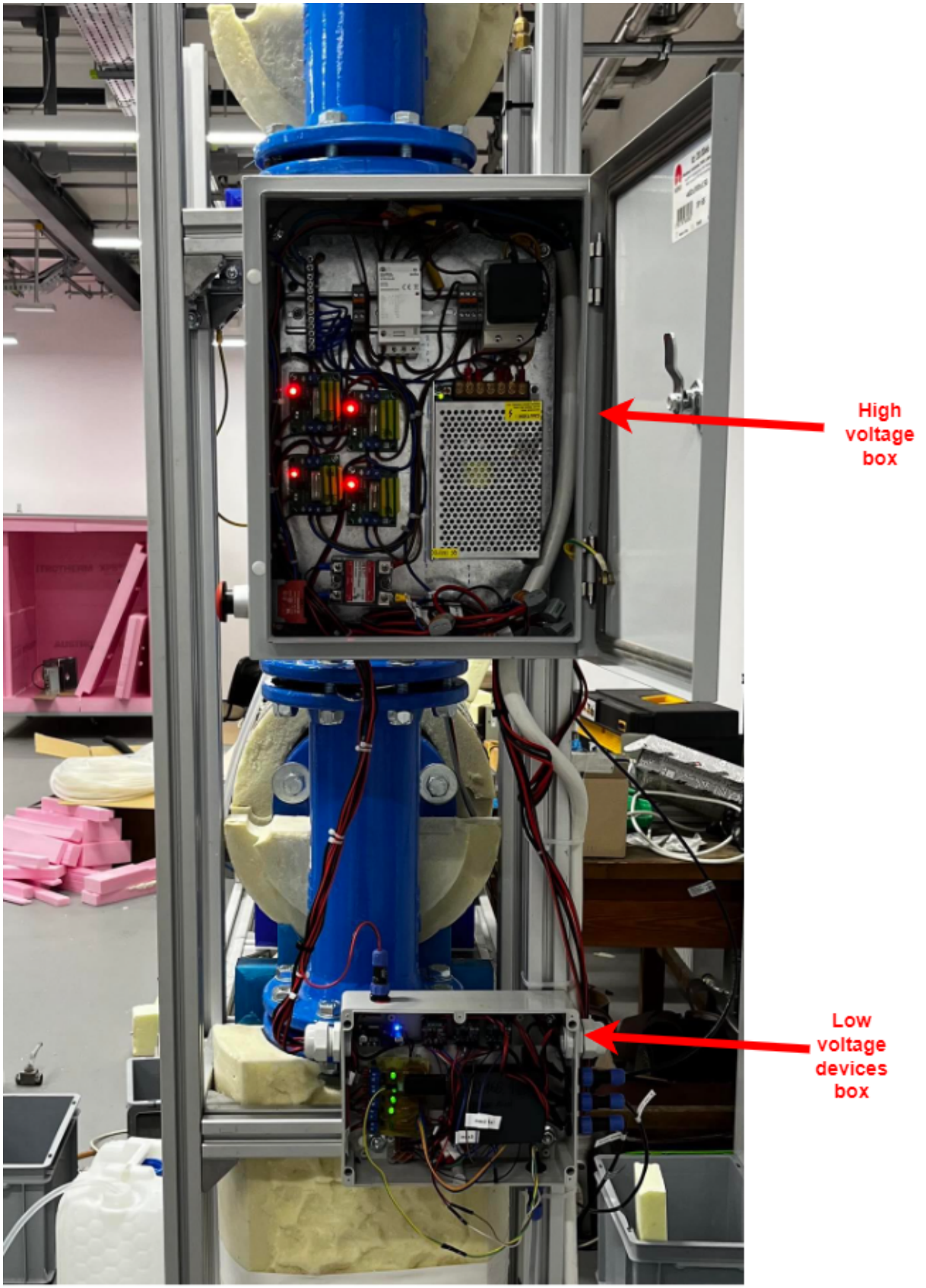


Figure 7. Assembled electric and electronics boxes

where $\mathrm{A_{tot}}$ is total area of non insulated parts of the device (non insulated surfaces are marked in appendix 7), which is approximated with:

$$\mathrm{A_{tot}} = \mathrm{A_{vertical\_cylinder}} + 4\mathrm{A_{flat\_plate\_covers}} = 2\pi\mathrm{RL} + 4\pi\mathrm{R_{cover}^2} \tag{18}$$

where:

- $\mathrm{A_{vertical\_cylinder}}$ - Only non insulated area
- $\mathrm{R[m]}$ - Device body outer radius
- $\mathrm{L[m]}$ - Device non insulated length
- $\mathrm{R_{cover}[m]}$ - Cover radius

Convection coefficients where calculated using[4] which states:

$$\alpha = \frac{\mathrm{Nu_L L}}{\mathrm{k}} \tag{19}$$

where:

- $\mathrm{k}[\frac{\mathrm{W}}{\mathrm{mK}}]$ - thermal conductivity of cast iron
- $\mathrm{L[m]}$ - characteristic length

There are several non insulated surfaces with different Nusselt number (differs due to, ex. facing direction), according to ([9-21, 9-22, 9-24])[4] Nusselt numbers are calculated for vertical cylinder, and 3 facing directions of flat plate covers:

$$\mathrm{Nu_{L,cylinder}} = \mathrm{Nu_{L,vertical\_flat\_plate}} \tag{20}$$

if $\mathrm{D} >= \frac{35\mathrm{L}}{\mathrm{Gr_L^{\frac{1}{4}}}}$,

$$\mathrm{Nu_{L,vertical\_flat\_plate}} = (0.825 + \frac{0.387\mathrm{Ra_L^{\frac{1}{6}}}}{(1 + (\frac{0.492}{\mathrm{Pr}})^{\frac{9}{16}})^{\frac{8}{27}}})^2 \tag{21}$$

applicable in any range,

$$\mathrm{Nu_{L,horizontal\_flat\_plate,facing\_up}} = 0.54\mathrm{Ra_L^{\frac{1}{4}}} \tag{22}$$

applicable for $\mathrm{Ra} \in [1\mathrm{e}5; 1\mathrm{e}11]$,

$$\mathrm{Nu_{L,horizontal\_flat\_plate,facing\_down}} = 0.27\mathrm{Ra_L^{\frac{1}{4}}} \tag{23}$$

applicable for $\mathrm{Ra} \in [1\mathrm{e}4; 1\mathrm{e}7]$, where:

$$\mathrm{Ra} = \mathrm{Gr_L Pr} \tag{24}$$

and:

$$\mathrm{Pr} = \frac{\mu \mathrm{c_p}}{\mathrm{k}} \tag{25}$$

$$\mathrm{Gr_L} = \frac{\mathrm{g}\beta\Delta\mathrm{TL}^3}{\nu^2} \tag{26}$$

for:

- μ - dynamic viscosity
- $\mathrm{c_p}$ -isobaric specific heat
- k - thermal conductivity
- g - gravitational acceleration
- $\Delta \mathrm{T}$ - temperature difference between surface and ambient temperature
- L - characteristic length
- ν - kinematic viscosity

Chapter 3

Web controls

During the design process utility and safety were carefully balanced, thus decision was made that the nominal operation of the system should not require direct contact with a running device, thus one of precautions made is a remote control of the steam generator. User can freely operate the device within range of a local network to which the control unit is connected, while maintaining a safe distance in case of system malfunctions. Control system is also a proof of concept for using it in other similar devices in the laboratory, which also created a need for versatility and easy customization.

All source code with documentation is available publicly as a repository on Github.

3.1 System overall design

Control system is composed of two main part - server application and graphical user interface (GUI). These communicate with each other via universal REST API web interfaces. This allows to freely change GUI and implement it to any digital platform, in any programming language[1]. For the project simple, server-side rendered website was created as HTML combined with JavaScript (jQuerry) easily handles REST API communication. For the server side python framework - django application was chosen due to it's easy development and language capabilities. To improve system flexibility even more, whole application is encapsulated inside Docker software, which allows to almost instantaneous hardware replacement in case of failure or installation of new device.

3.2 Server side system

Server application main task is to handle control commands from incoming network traffic and directly execute them on the device. Very base of the system is the Docker software allowing to virtualize the program and making it independent on hardware and OS[2] used, it operates identically on microcomputers such as Raspberry Pi as well as on full scale desktop PC. Inside Docker containers there are build several services figure 8 handling all the traffic. Nginx (http server) provides initial

[1] Language must handle REST API communication or user has to add this feature manually
[2] Operating system, ex.: Windows, Linux, MacOS

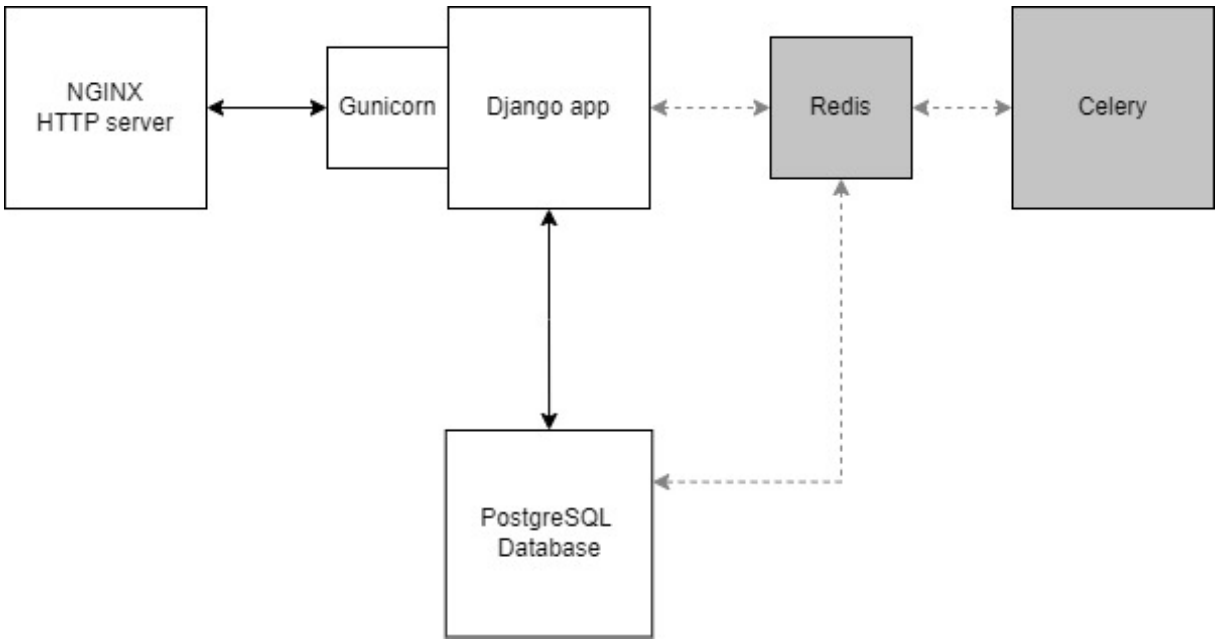


Figure 8. Server application servicies connections diagram

traffic routing to domestic services, it is the only entry point to the rest of the system. Django (working with Gunicorn) is the main application, it is responsible for all of direct commands execution on the device as well as the response send back to the user. Last key services present in the program are a PostgreSQL database, Redis message broker and Celery task scheduler. Celery is meant to move time demanding tasks, ex. set-point controller loop to the background to enable asynchronous system controls.

3.2.1 Hardware - Raspberry Pi

To avoid hardware bottlenecks yet maintaining ease of programming and hardware integration, the highest version available of Raspbbery Pi - Raspbbery Pi 4B model with 8 GB RAM memory was used. It offers enough computing power combined with a few GPIO pins and requires little power supplied even in peak performance. Base operating system for the computer is Raspberry Pi OS, debian based linux distribution. This gives access to all required programs and technologies for the project. Big advantage of Raspberry Pi compared to other possible computers is native support for serial interfaces, ex. SPI or RS485(UART). Actual Raspberry Pi parameters are presented in table 2.

Name	Value	Unit
Processor model	Cortex-A72	[-]
Processor speed	4x1.5	[GHz]
RAM size	8	[GB]

Table 2. Raspberry Pi 4B+ characteristics

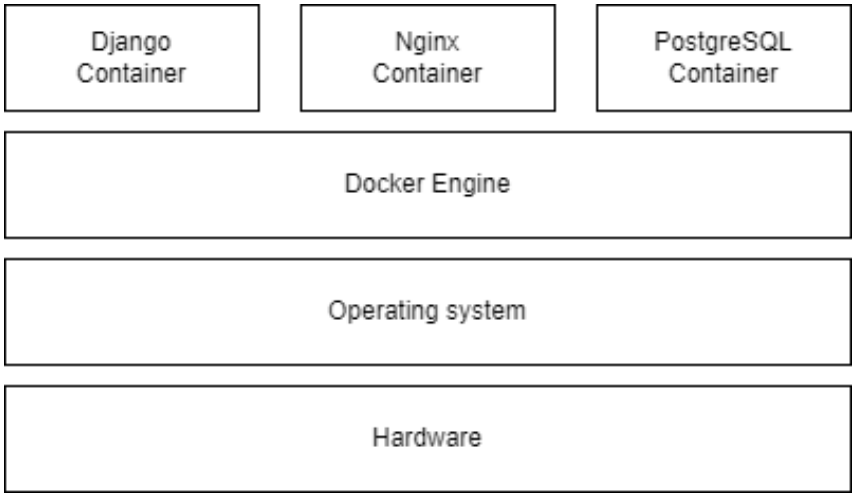


Figure 9. Docker virtualization diagram.

3.2.2 Docker

Whole software made and used in the project is based on *Docker* platform. It is used to create operating system level virtualization of build services, which allows to run those independently from used hardware or OS. The great advantage of containerizing software using Docker is easy version control and unwanted updates prevention, which could lead to version incompatibility and overall software failure. As this project deals directly with a hardware, virtualization gives another advantage of isolation of OS and it's kernel from the control software. This leads to increased security of the program and potential hardware damage made by malware. Docker virtualization layers are presented on figure 9.

3.2.3 Main application - Django framework

Main part of control program is Django[3] server application, which utilize REST API[1] for web communication. Django is Python web framework meant for easily scale-able applications, with built in basic security modules. Django default structure provided out of the box separates actual web services (web folder) and other application components (webcontrol folder). *SGcontroller* class instance is the main handle for all actions performed by the system, it's instance variables provides handlers of all sensors as well as active components (heaters). File *tasks.py* stores *Timeloop* class instance, which is used to run periodically two most important control loops of the program: *watchdog* and *pid_loop*. First is a safety control loop, watchdog checks if critical parameters set are not exceeded and if yes, watchdog loop turns off all active elements. Second is actual control unit, PID controller instance which is used to calculate control signal for heaters. Besides, there are 3 available class based Django views, first is base URL view, where user interface is served as well as user input commands are being processed and directed to SGcontroller instance. The next view available at

[3]Django documentation available at:https://docs.djangoproject.com/en/4.1/

update_data is meant to be called by user interface to get current parameters measurements and system state data in json file type. The last view is used to handle data download from database in form of text file. This kind (Class based views) approach gives huge ease of modifying the application while maintaining the code simple and easily readable as shown on Listing 1:

```
1
2  class Index(View):
3
4      def get(self, request, *args, **kwargs):
5          template = 'index.html'
6          return render(request, template)
7
8      def post(self, request, *args, **kwargs):
9          post_data = request.POST.dict()
10
11         print(post_data)
12
13         for ele in post_data.keys():
14             controller.control_params[ele] = int(post_data[ele])
15
16         if(controller.control_params['emergency_stop']):
17             controller.emergency_shutdown()
18
19         if(controller.control_params['soft_stop']):
20             controller.soft_shutdown()
21
22         if(controller.control_params['manual_mode']):
23             controller.control_loop()
24
25         if(controller.control_params['temp_setpoint']):
26             controller.pid.setpoint = controller.control_params['
    temp_setpoint']
27
28         if(controller.control_params['save']
29             and not controller.data_save_started):
30             controller.start_data_save()
31
32         if(not controller.control_params['save']
33             and controller.data_save_started):
34             controller.stop_data_save()
35
36         controller.control_loop()
37
38         template = 'index.html'
39         return render(request, template)
40
```

```
class UpdateData(View):
    def get(self, request, *args, **kwargs):
        res = controller.output

        return JsonResponse(res)

    def post(self, request):
        return HttpResponseBadRequest()

class DownloadFileView(View):
    def get(self, request):
        return HttpResponseBadRequest()

    def post(self, request):
        file_data = controller.get_data_from_db()

        return JsonResponse(file_data)
```

Listing 1. Available endpoints with handlers for basic REST API methods

3.2.4 Web connection

All incoming traffic to the control program, despite the source, either the web or local network, is handled by *NGINX* HTTP server. In this configuration NGINX serves as a reverse proxy service and it was chosen due to it's development potential and scaleability. It is possible to quite easily add additional services to the whole program.

3.2.5 Data acquisition

Software is delivered with basic data acquisition system, gathering data from all system sensors and storing them for further use. In prototype version there are two main ways of data save, backup file storing all parameters and system setting from system power on until hard shutdown (power cut off). This file works like systems back box, it is intended to recreate all events from data stored there, the file itself is saved as text file in hardwares mass memory. Primary data save is made with Django ORM[4] querry directly to a PostgreSQL database. Main data table stores measured values of all system parameters such as temperatures (water and steam), pressure, phases voltage, current and power as well as each active element current state. Database defined with Django ORM class provides easy way of modifying data being collected, adding new fields for additional information and most importantly it is more sequre than raw SQL querries. Database table model is provided in *models.py* file and it is shown on Listing 2:

```

class SteamGenerator(models.Model):
```

[4] https://docs.djangoproject.com/en/4.1/topics/db/queries/

```
    measurement_num = models.IntegerField(default=0)
    water_temp = models.FloatField(default=0)
    steam_temp_1 = models.FloatField(default=0)
    steam_temp_2 = models.FloatField(default=0)
    pressure = models.FloatField(null=True)
    heater_water1_power = models.FloatField(default=0)
    heater_water2_power = models.FloatField(default=0)
    heater_water3_power = models.FloatField(default=0)
    heater_steam_power = models.FloatField(default=0)
    valve = models.CharField(max_length=10, default='closed')
    voltage_ph1 = models.FloatField(default=0)
    current_ph1 = models.FloatField(default=0)
    power_ph1 = models.FloatField(default=0)
    voltage_ph2 = models.FloatField(default=0)
    current_ph2 = models.FloatField(default=0)
    power_ph2 = models.FloatField(default=0)
    voltage_ph3 = models.FloatField(default=0)
    current_ph3 = models.FloatField(default=0)
    power_ph3 = models.FloatField(default=0)
```

Listing 2. Database primary table model

3.2.6 Cybersecurity and general security system

Due to fact that software is web base and operates in network and being open for every connection cyber securing the app was considered. In a prototype version there are several defence stages, firstly the app is served in local network only protected with WPA2[7] and blocked access to the admin panel from outside of a network. Secondly, inside Django app there are installed a few middleware protection submodules providing basic security to the site for example CSRF blocking module.

```

MIDDLEWARE = [
    'django.middleware.security.SecurityMiddleware',
    'django.contrib.sessions.middleware.SessionMiddleware',
    'django.middleware.common.CommonMiddleware',
    'django.middleware.csrf.CsrfViewMiddleware',
    'django.contrib.auth.middleware.AuthenticationMiddleware',
    'django.contrib.messages.middleware.MessageMiddleware',
    'django.middleware.clickjacking.XFrameOptionsMiddleware',
]
```

Listing 3. Installed middleware security modules

On the other hand, to assure general safety a "watchdog" script was implemented. Initial version of a script runs periodic temperature and pressure changes. It is intended to develop this script to

assure more accurate and more automated safety check system including ex. pressure loss shutdown or system elements malfunction.

```
1
2 @t1.job(interval=datetime.timedelta(seconds=1))
3 def watchdog():
4     if(controller.output['water_temp'] >= MAX_TEMP or
5         controller.output['steam_temp_1'] >= MAX_TEMP or
6         controller.output['steam_temp_2'] >= MAX_TEMP or
7         controller.output['pressure'] >= MAX_PRESSURE):
8
9         controller.soft_shutdown()
```

Listing 4. Watchdog loop

3.3 User interface

User interface is meant to be easily adjustable and developed in the future, thus it uses REST API to connect to server side application. In a prototype version simple server served website with straightforward controls were written in Django HTML5[5] template and styled using CSS. There are also several Javascript services running in the background, providing asynchronous data update in a background using jQuerry library. The system was required to meet two conditions: provide simple control panel and demonstrate system capabilities.

3.3.1 User interface

The very basic control system is provided on single page website. There are 4 distinct sections of control panel: power controls, system measured parameters table, manual controls of all active elements and selected parameters graphs. User interface is presented on figure 10, sensors position is marked on figure 4.

Website is statically served[6] meaning that it is loaded ones after establishing connection.

3.3.2 Data fetch from and to the server

Website uses Javascript services to fetch data from the server and update displayed values on the website. The communication is bidirectional, meaning it also send user input to the server asynchronously, without need for any page reloads[7]. On the website there are also live graphs presenting aggregated sensors readout of systems parameters, these are made using CanvasJS Javascript library.

[5] https://docs.djangoproject.com/en/4.1/ref/templates/language/
[6] Website HTML code available in the appendix 2
[7] Script available in appendix 3

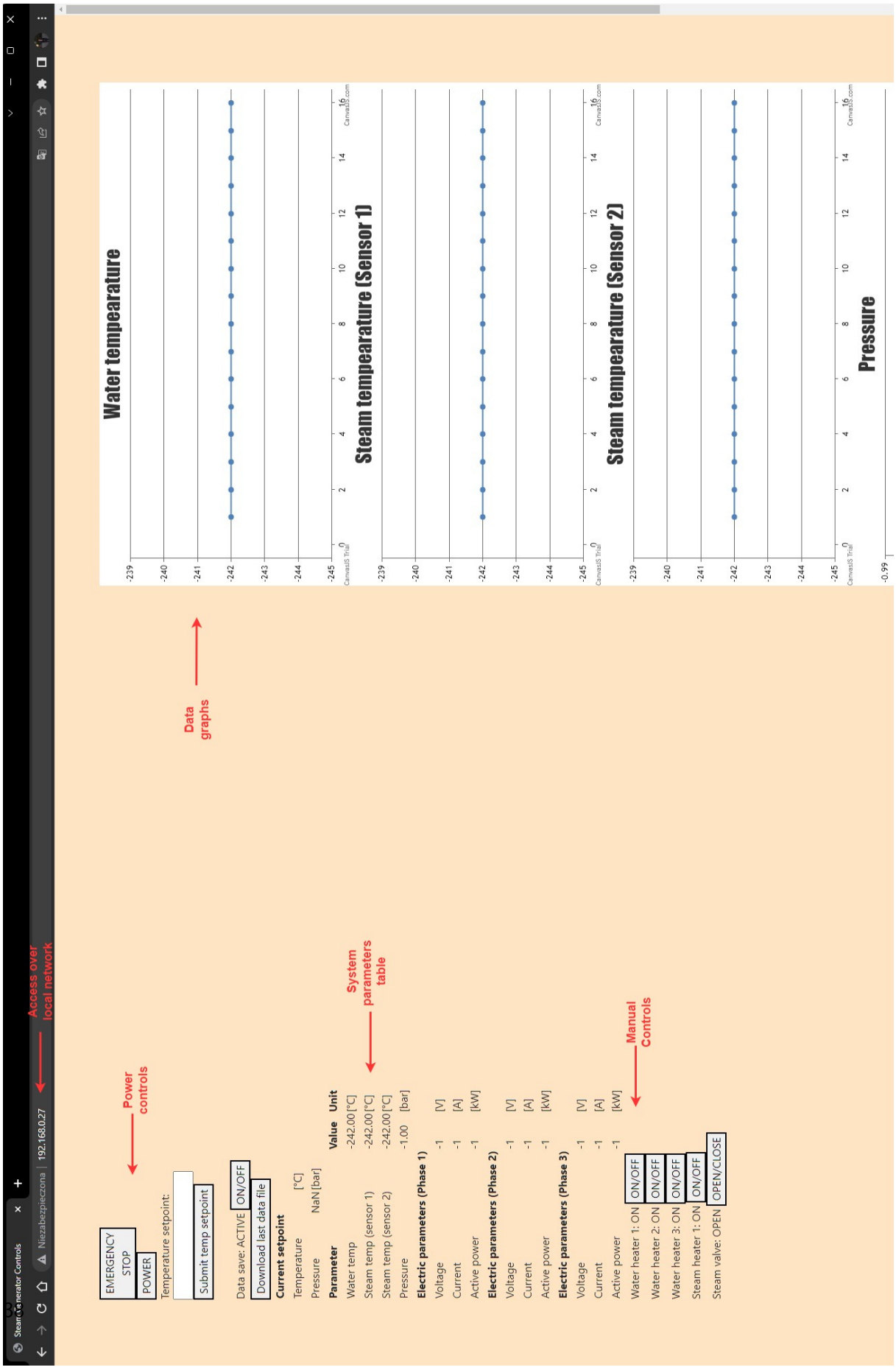


Figure 10. Prototype of a control panel as a single page website

Chapter 4

System tests

This chapter presents tests of the system performed during thesis preparation. There are more tests that should be performed before standard use of the device (mainly safety tests) and device improvements (described briefly in chapter 5).

4.1 Leak test

After each mechanical modification of the device a leak test should be performed. Leak test made lasted around $10[\mathrm{h}]$ (overnight test) and main check was, if the pressure drops anyhow with all valves closed. Overall pressure drop was around $0.15 \pm 0.1[\mathrm{bar}]$test was passed. Device was filled with compressed air for the test, thus small deviations were expected as surroundings temperature drops during the night compared with the day.

4.2 Safety tests

4.2.1 Power loss test

During power loss test, main power safety button was used to cut off system power. Each of three mounted buttons were tested (2 physical, 1 electrical ones) and each correctly cut off the power. Device was observed with manual measurement equipment and as predicted slowly started to cool down without any noticeable events. Rasberry Pi computer controls 1 electrical power button (power relay), in a event of power loss of computer it also correctly cuts off the power.

4.2.2 Over pressure and over temperature test

There is watchdog loop implemented in the software (code shown on listing 4) which constantly checks if system parameters (pressure and temperature) are kept in predefined range. During the test these were artificially increased over the predefined range. Software correctly triggered system shutdown in the moment of values exceeding thresholds.

4.3 Nominal operation tests

4.3.1 Nominal operation test no.1 (failed)

During first operation test two system failures occured. First, a data acquisition system malfunctioned, thus no data was gathered, only screenshots of data available in a control panel. Secondly, during steam release at much lower pressure than nominal, one of the rubber seals has broken causing sudden, uncontrolled steam burst. Figures 11,12 presents gasket after the event and figure 15 comparison of old to new one mounted. Position of gasket which failed is marked on figure 10.

Device inspection showed two main flaws which led to this event. Seal was tightened to hard which caused seal excessive stretch and decreased material strength. Besides, as it is shown on the graph (figure 14) steam mass flow was too high which led to excessive pressure drop and very possibly to cavitation inside the device, adding more stress to the material which eventually failed. As a side effect it came out that installed gaskets nominal operation temperature was $160[\mathrm{C}]$ and would eventually also lead to device failure.

After this event all gaskets were swapped for new appropriate[1] ones, leak test was performed again.

4.3.2 Nominal operation test no.2 (partially successful)

The second test gave more results as data was acquired correctly. Collected data is presented on figure 16. Actual positions of sensors are marked on schematic on figure 4.

[1] Market offer: https://sklep.polberis.pl/pl/p/Uszczelka-DN100-PN16-3mm-AF1000/10819

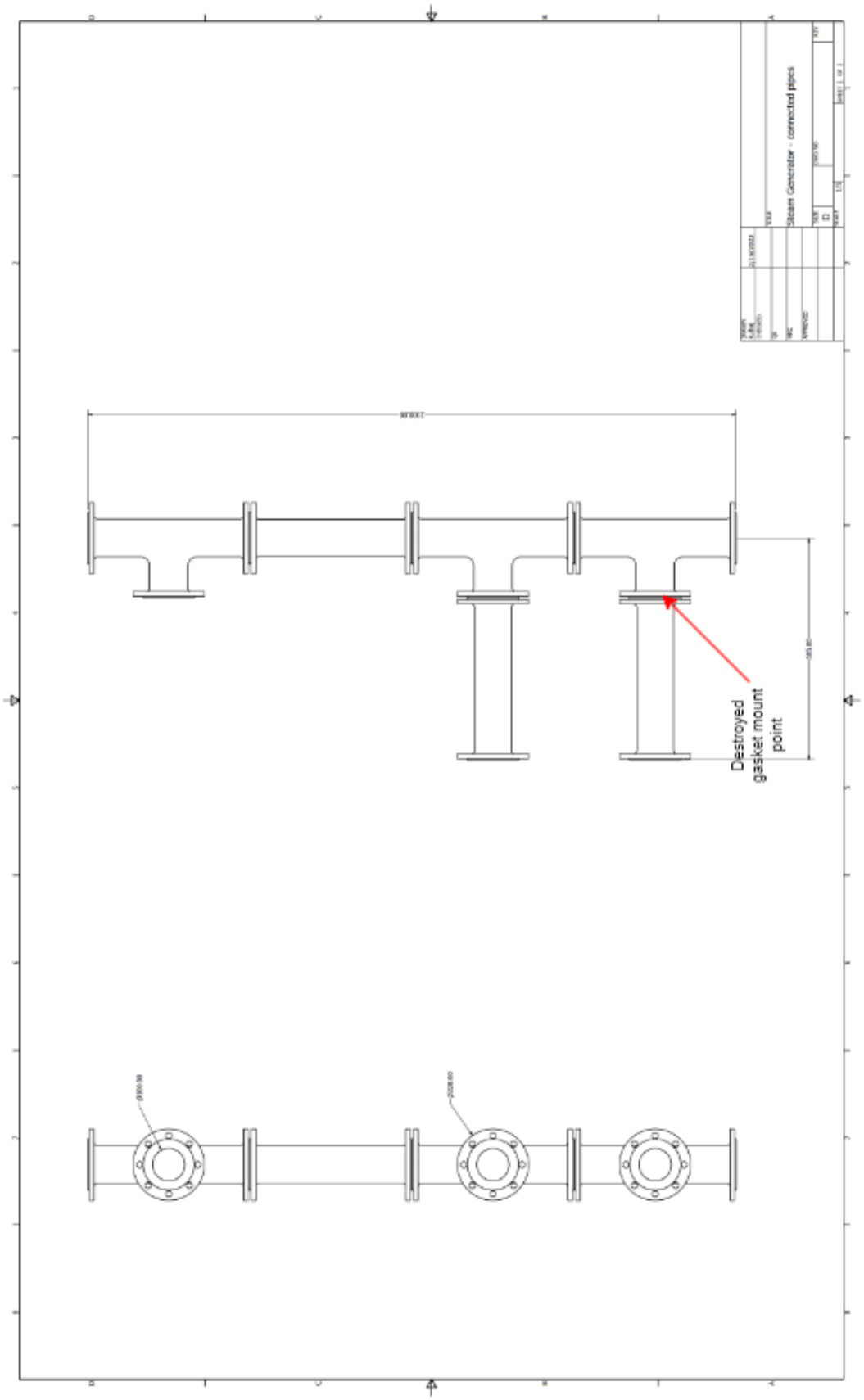


Figure 11. Position of destroyed gasket

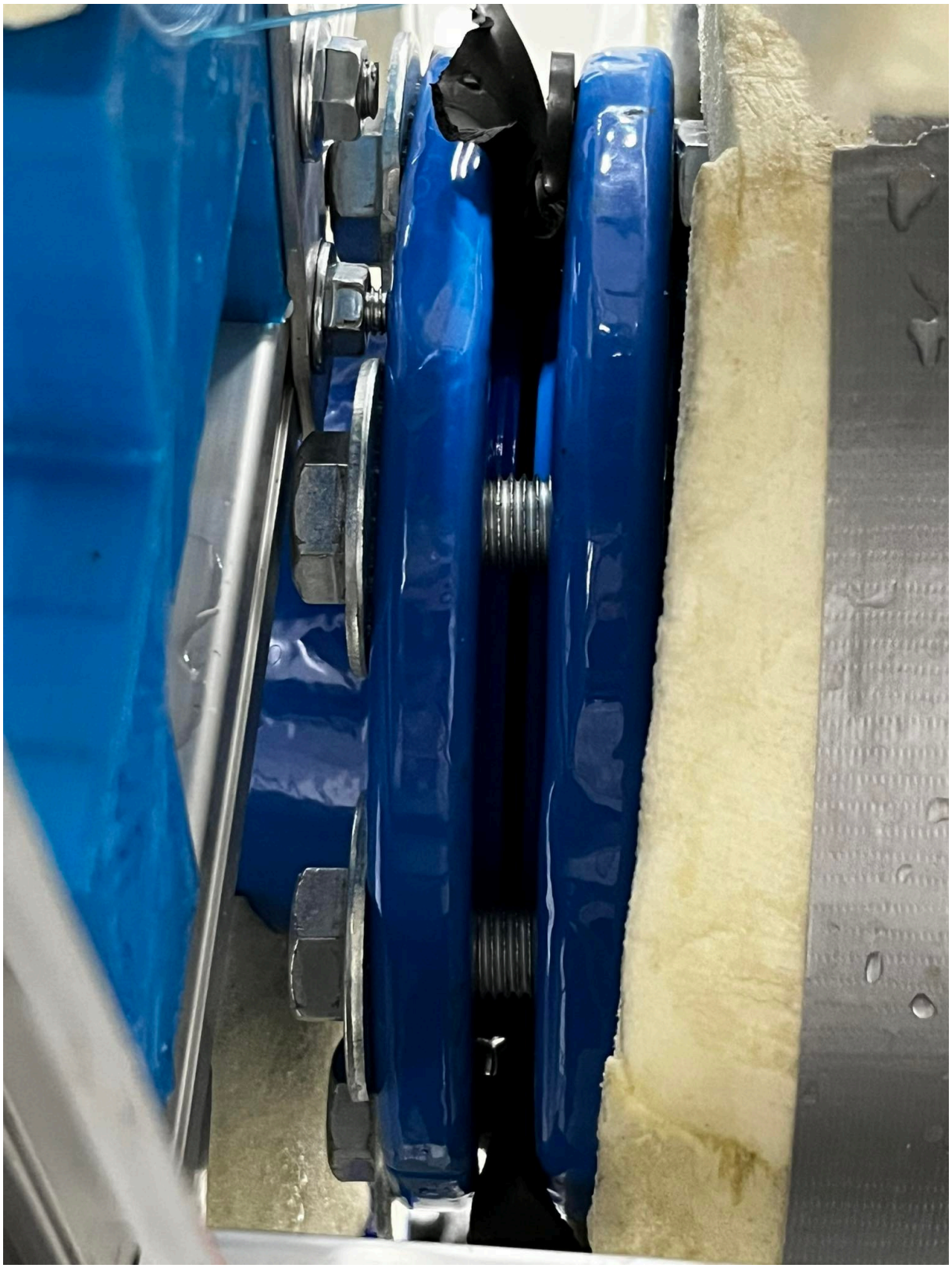


Figure 12. Installed broken rubber seal

Figure 13. Dismounted broken rubber seal

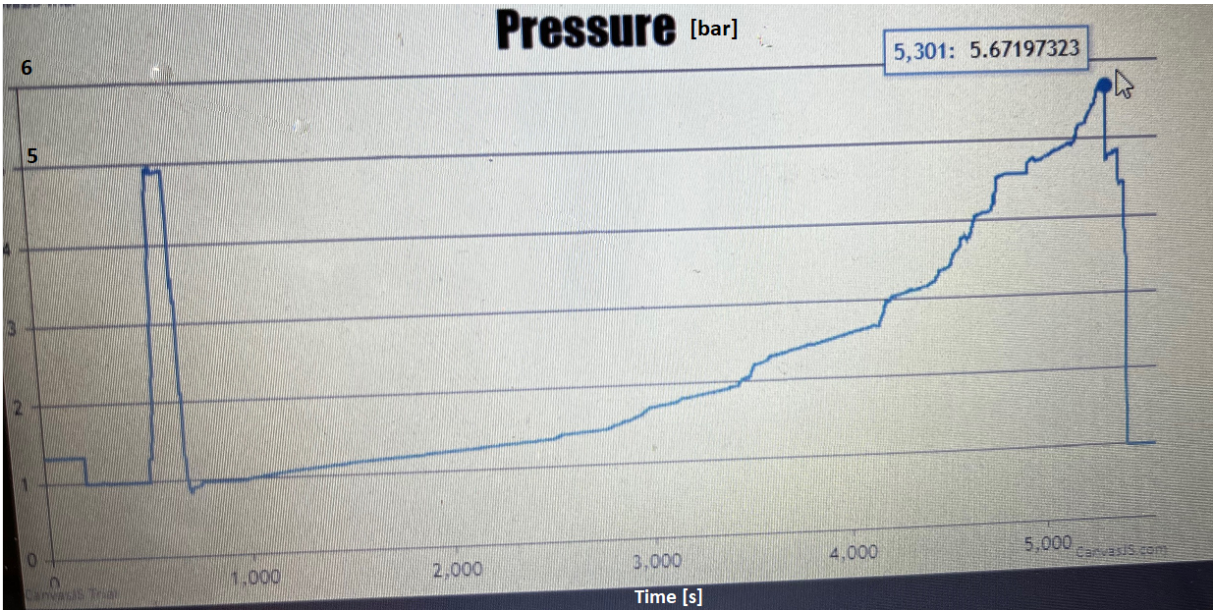


Figure 14. Pressure graph - failed test no.1

Figure 15. New seal by the old one

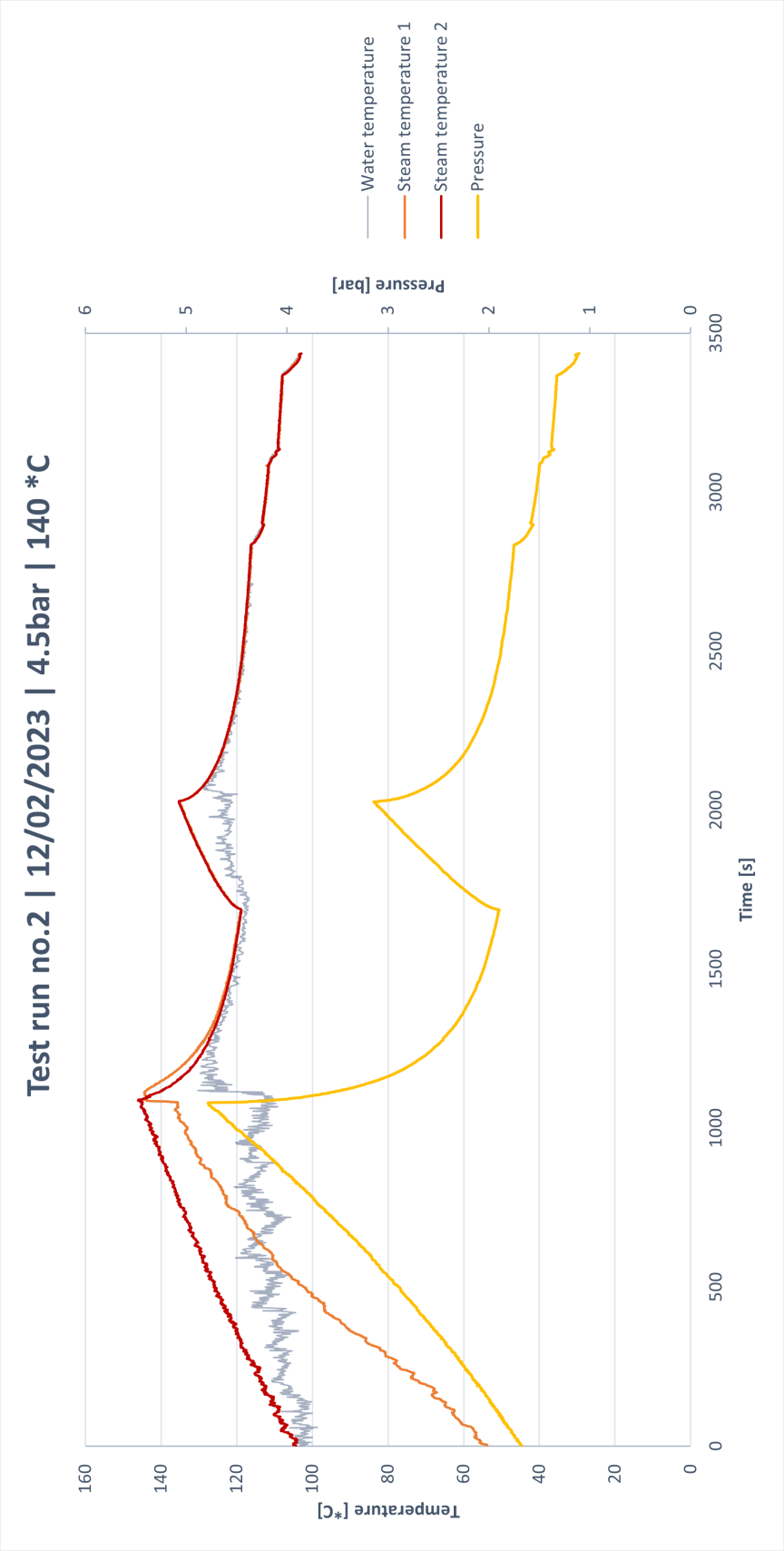


Figure 16. Test run no.2

During the test there occured small leakages when the pressure exceeded about $4.5[\mathrm{bar}]$, most probably caused by material thermal expansion. These did not prevent further testing (pressure drop was negligible), but further heat up was stopped as a safety precaution. Collected data gave insight on system behaviour and highlighted several major issues, but it allowed to confirm good enough data acquisition. As shown on the graph there is strong noise signal influencing water temperature sensor. Steam temperature sensors readouts were also disturbed slightly by noise signal. During the test there were two steam releases with gap between them. Mathematical models provided in equations 3,7 and 8 were impossible to apply due to air presence in the system, as the system was not stable enough.

4.3.3 Nominal operation test no.3

Third test, the last one, gave most accurate results compared with previous tests. Collected system parameters data is presented on figure 17. Graph presents system pressure, water, steam and saturation temperatures. There are clearly seen break points in a pressure line which corresponds to turning on next water heaters (in initial phase only one out of three heaters was on).

Before this test run air was remove from the system using vacuum pump, which made system behaviour much more predictable. Overall test was considered as successful and further system characteristics and calculations were based on test 3 results. During the test there were collected almost 100000 records of data with time interval between measurements of $\mathrm{T} = 1[\mathrm{s}]$. This gives accurate results to estimate device performance. The device lacked full thermal insulation thus there occured high heat losses thru a device walls. Test was conducted with ambient temperature $\mathrm{T_{ambient}} = 24 \pm 1[\mathrm{C}]$, initial water mass in the system $\mathrm{m_{water\_initial}} = 10.1437 \pm 0.0001[\mathrm{kg}]$, device emissivities were experimentally measured $\epsilon_{\mathrm{cover}} = 0.11$ and $\epsilon_{\mathrm{device\_body}} = 0.97$ with thermal imaging camera. Cast iron conductivity used for calculations $\mathrm{k} = 52[\frac{\mathrm{W}}{\mathrm{mK}}]$ and device dimensions $\mathrm{L_{cylinder}} = 1.62[\mathrm{m}]$, $\mathrm{D_{outter\_vertical\_cylinder}} = 0.118[\mathrm{m}]$, $\mathrm{R_{cover}} = 0.22[\mathrm{m}]$, $\mathrm{P_{single\_heater}} = 1850[\mathrm{W}]$.

4.4 System characteristics estimation

Whole data analysis was made using pandas[2] Python module to speed up process. PyXSteam[3] provided data point for water and steam at different pressures and pyfluids[4] was used to obtain accurate air parameters for calculations. Calculations where performed with use of Python script - available in appendix 6 and in project software repository on Github and results are presented on figures 18 and 19.

[2] https pandas.pydata.org docs getting_started index.html
[3] https://pyxsteam.readthedocs.io/en/latest/
[4] https://github.com/portyanikhin/PyFluids

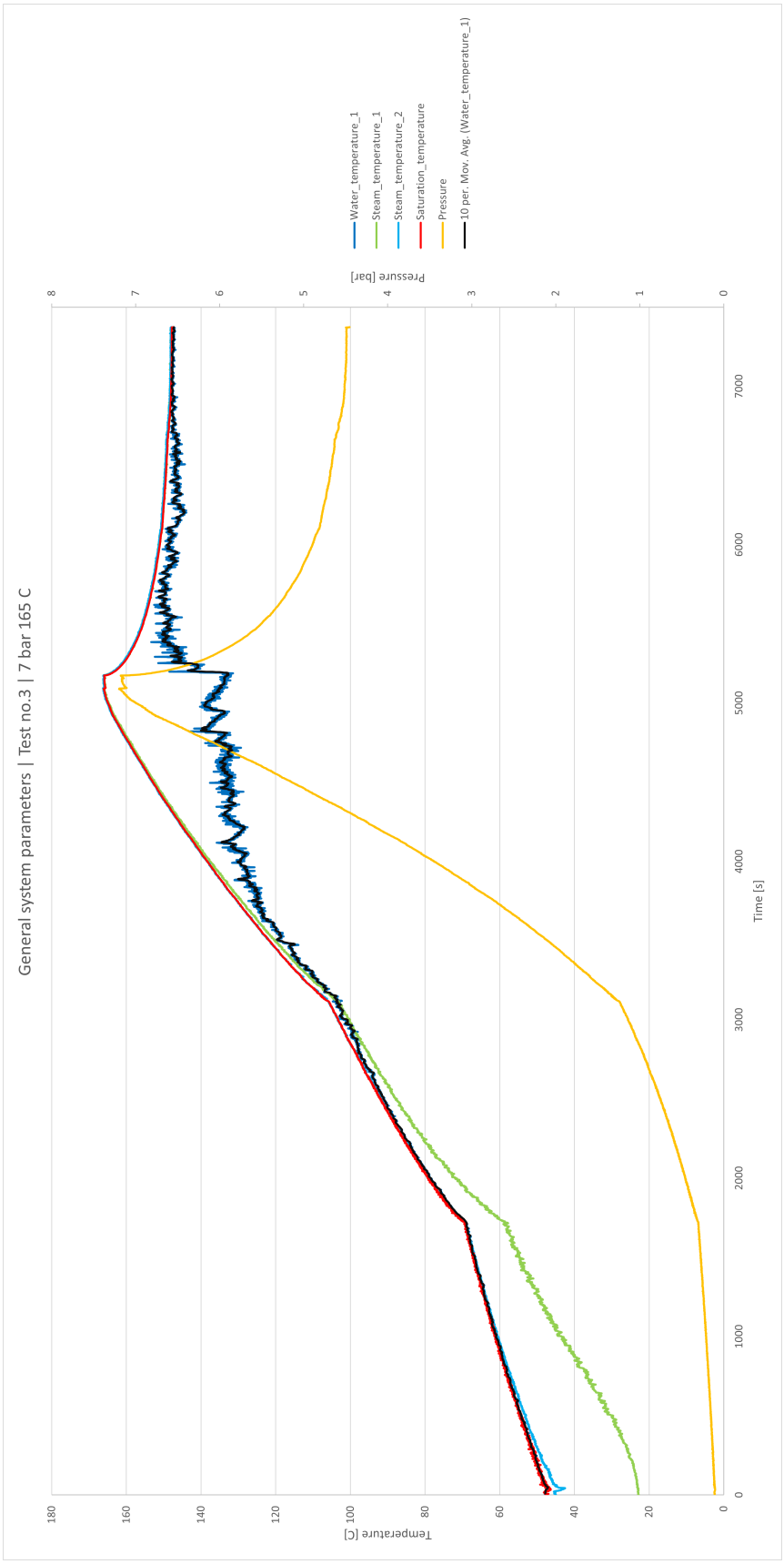


Figure 17. Test run no.3

4.4.1 System heat losses

Using equations 3,7,8 and 18-26 system heat losses were estimated for each time step. Figure 18 shows heat losses as a function of steam temperature throughout test no.3. It is expected that this holds for every other run (if no mechanical changes are made in the system). As it came up, main source of heat losses in the device is heat radiation. Measurements showed that cast iron used in the device body has high emissivity and heat radiation is much greater than heat convection.

4.4.2 Steam mass flow capacity

Keeping general philosophy presented in 1.5.3 firstly mass of generated steam was calculated. Due to steadiness of the system during heat up phase it was assumed that single iteration calculations would provide good approximation of steam mass value while reducing significantly calculation time. Using equation 3 with initial point at $\mathrm{t_{init}} = 0[\mathrm{s}]$,$\mathrm{m_{steam}} = 0[\mathrm{kg}]$ and final point at $\mathrm{t_{final}} = 4984[\mathrm{s}]$ (last data point with heaters on) mass of steam generated equals $\mathrm{m_{steam}} = 7.075[\mathrm{kg}]$.

This was used to further estimate steam mass flow at the outlet from the device. Due to sensors inaccuracies in data set occured point (pressure, mass of steam after next calculations iteration) that would give negative steam mass flow in steam release phase. These were assumed to be incorrect and caused by high noise. Steam mass flow calculations were iterated over every data point (time period between data points $\mathrm{dt} = 1[\mathrm{s}]$). This gave results presented on figure 19, were are presented total steam mass in the system at given moment, calculated steam mass flow using equation 8 and averaged mass flow value - made to increase readability. Initial sudden change of mass flow is a result of initial manual valve regulation. Average steam mass flow was $\dot{\mathrm{m}}_{\mathrm{steam}} = 0.000365[\frac{\mathrm{kg}}{\mathrm{s}}] = 1.361[\frac{\mathrm{kg}}{\mathrm{h}}]$. Mass flow is well under the design value (about 10 times), this is most probably caused by bad outlet valve setup, there is lacking electrically regulated throttling valve which allow to achieve higher mass flow, as system appears to be stable at given value and that there is possible increasing mass flow.

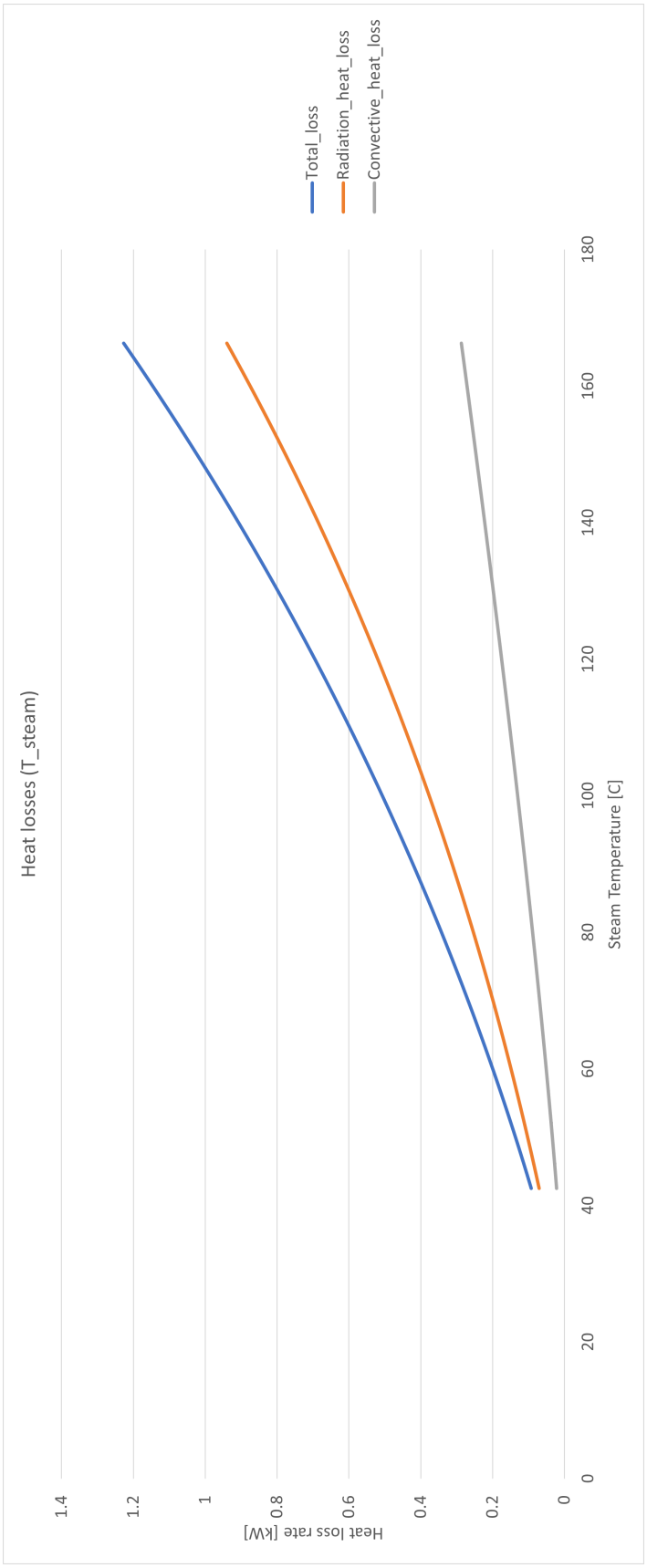


Figure 18. Device heat losses

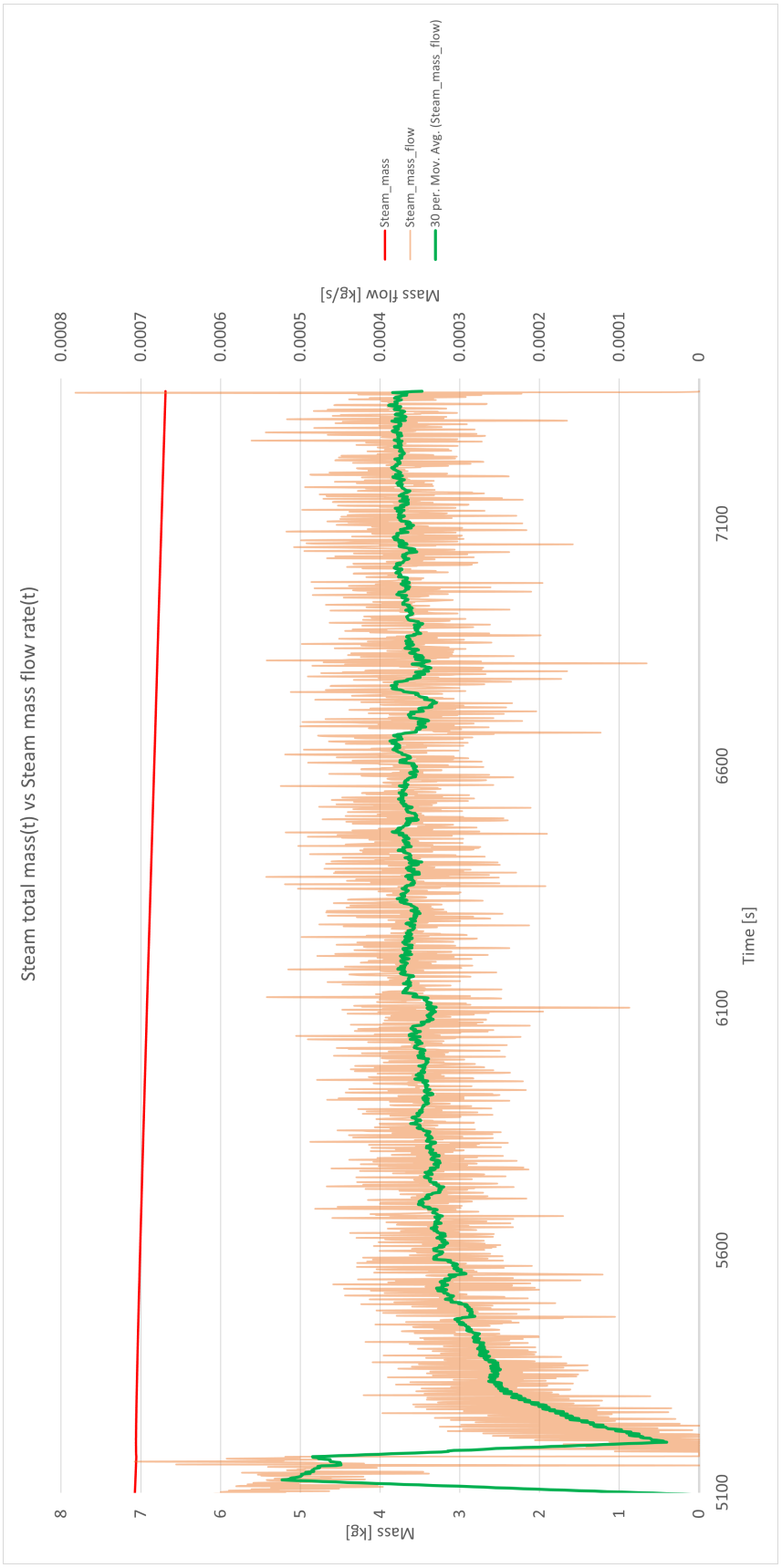


Figure 19. Steam mass flow

Chapter 5

Conclusions

This chapter is fully devoted to project summary, problems and improvements discussion. It should be noted that not all the things are listed and even after the modifications it is recommended to look constantly for other modifications improving device efficiency, dynamics, safety, etc.

5.1 Existing problems

During the assembly and tests many possible improvement options of the device came up as well as design flaws. This section contains a short description of most important existing problems, more about recommended changes are discussed in the next *Possible improvements* section.

5.1.1 Hardware

Heat radiation

Figure 17 shows the effects of electric elements overheating due to high heat radiation from the device.

Release valve and manual valve

Mounted valves handles are too short and too close to the device with no insulation and due to heat conduction these heat up so much that are untouchable without risking burns.

5.1.2 Software

Database

Currently most urgent matter is database which still malfunctions and data save is not reliable enough.

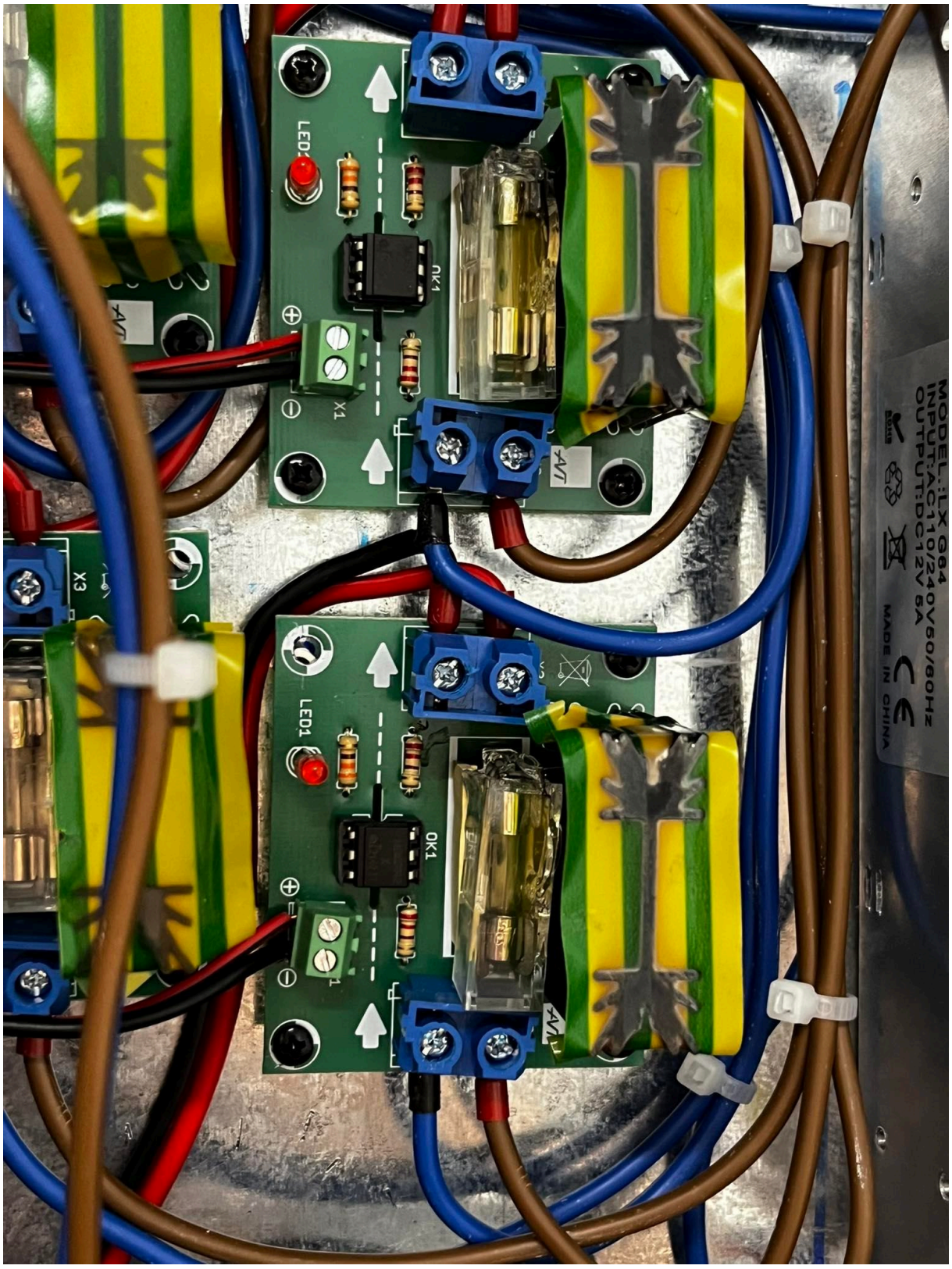


Figure 20. Installed SSR melted due to heat radiation

Automated software

After reboot software does not start automatically which could create danger in case of power loss. Also it is rather complicated (for person with no experience with programming) to restart the application.

PID

It is very unpleasant to get to the certain pressure or temperature setpoint as all controls are manual. PID controller still requires configuration as well as transfer function improvements.

5.2 Possible improvements

As it was shown in previous paragraph in the system still occur space for improvement. Below are presented all recommended improvements of the device, hardware and software upgrades. Note that these are not sorted by priority, thus it should be carefully considered during any upgrades.

5.2.1 Hardware

Hardware changes should be avoided if possible to lower overall upgrade effort. Given are the most important changes which can increase significantly system safety and usability compared to work required.

Electric and electronic boxes

In a electric (power) box SSRs should be swapped to ones that can handle 400 [V] supply voltage. This would allow to replace existing heaters with more powerful ones. Any new devices installed in electric box should be DIN mounted to cleanup inbox space. Low voltage (12/5 [V]) supply line should be simplified, 230/12 [V] power supply should be changed to 230/5 [V] version, with lower power - estimated 20 [W] should be enough to power all electronics. Raspberry pi should remain on different independent power line to ensure control over the system if other power lines are switched off. Electronics box should be changed to bigger one to cleanup inbox space. Any lines, despite if signal or power, coming in or out to electronics box should be connected via some kind of easily detachable connectors, as it is done in existing connectors with Pt100 sensors or Keller23SX sensor.

Insulation and frame

In prototype version due to bad frame design it is impossible to mount insulation on every piece of the device. It is necessary to remodel frame in such a way that the insulation would fit in. Without that excessive heat losses occur and device estimated characteristics accuracy is reduced.

Figure 21. Existing insulation

Frame support

Another possible and recommended upgrade is to change currently mounted 3D printed device supports. Best case scenario would be to make those out of thermal insulator yet material which offers strength similar to aluminum frame.

Heaters

Currently mounted water heater has lowered nominal voltage compared with designed one. This is a consequence of mounted SSRs with lowered nominal voltage. After electronics replacement, heater should be replaced as well, which would increase device thermal dynamics.

Steam super heaters can be replaced with shorter version which could reduce overall device height (approximately 2.5 [m] now)

Heat radiation shields and cooling

It is necessary to mount heat shields by electric and electronics boxes or to put thick layer of insulation to protect electric components from overheating and eventually failure. Combined with heat shields it would be a good idea to install electric boxes cooling. Other solution is complete device insulation which would also eliminate overheating problems.

5.2.2 Software

Only technical improvements are proposed, any visual changes are left as these are not a priority in current version.

GUI

Software is provided with prototype version of a GUI. It is recommended to either develop existing version on recreate it from scratch. In the first place live data graphs should be provided with function of dynamic scrolling thru data, range selection, graph freeze function. Readability of graphs is a next step, one should change overall appearance to clarify the data preview. Another big step forward would be option to zoom selected areas of a graphs and display multiple variables on a single graph.

Next connection controls or indicator should be added. These should indicate connection of main control unit with all elements and sensors. In following versions error codes should be added, providing user with at least basic instructions for reconnecting elements.

It is a good idea to move indicators of ex. temperature or heater on/off status onto device schematic with marked real position of a element. This should increase readability of a control system. Controls of three phase heater should have option to be combined, these are now three separate buttons.

In case of website development one should switch whole front end application to user side rendered solution, ex. ReactJS or similar to increase robustness.

Figure 22. Mounted 3D printed supports

Docker containers

One should separate server side application. Django and controller apps should be separated should be splited to smaller independent services to ensure robustness of safety systems. Django app should only serve responses to GUI requests and connect via some kind of a interface with controller application. Target solution is to rewrite the software to fully implement micro services architecture. This means that software parts should be able to run without any other elements.

Controller app

Controller app should be developed with focus on safety and reliability systems. At first, watchdog script should be extended to provide wider range system checks: connection checks, system responsiveness, auto repair, etc. Next, data pipe to and from database should be finished, in provided version database is unused.

PID controller should be added to allow automated control over achieving set point temperature without need for constant supervision.

Parameters calculation script

Based on calculations made in 5th chapter it is possible to add a script to automatically calculate system parameters. There exists Python modules such as *pyXSteam*[1] providing accurate water and steam properties in nominal range. Combined with threading the script can work in a background, avoiding program blocking if computational time would be to high.

Auto leak test

Using pressure sensor leak test should be automated. It should be simple script accounting for environment parameters changes (temperature changes) and monitor any unexpected leaks from the device (pressure loss).

5.2.3 Improvements summary

Below are gathered all proposed improvements to the prototype device:

1. Hardware
    (a) SSRs should be swapped to 400 [V] version
    (b) power supply lines should be simplified
    (c) lower power, power supply- estimated 20 [W]
    (d) bigger electronics box
    (e) electronic box connectors for all the lines

[1] https://pypi.org/project/pyXSteam/

2. Software
    (a) Graphs:
        i. dynamic scrolling thru data
        ii. range selection
        iii. graph freeze
        iv. overall appearance to clarify the data
        v. zoom selected areas of a graphs
        vi. display multiple variables on a single graph
    (b) Controls
        i. connection controls or indicator
        ii. error codes
        iii. move indicators onto device schematic
        iv. controls of heaters with option to be combined
        v. switch front end application to ReactJS
    (c) Server app
        i. Django and controller apps should be separated
        ii. fully implement micro services architecture
        iii. watchdog script: connection checks, system responsiveness, auto repair, etc.
        iv. data pipe (database) to be finished
        v. PID implementation

5.3 Project summary

This work presents process of "Design and prototype construction of low pressure, low temperature steam boiler with superheating and web interface". According to initially reviewed market offers self made device is still preferred, despite the problems and challenges during construction it offers higher parameters flexibility and what is most important potential to freely make modifications as needed.

To obtain more accurate parameters and insights on device operation it is necessary to improve data acquisition in terms of measurements quality as well as quantity. Currently most of calculations serve as fast approximations used for engineering proccess rather than accurate calculations.

Presented project and device prototype are considered partially successful. In terms of meeting design parameters it has failed. Due to many small errors, design flaws, etc. after consolidation it gave unsatisfactory results. However obtained data and experience gave a huge insights on the project and how to develop it in the future. The project will be continued in the future and developed further based on discussed improvement propositions as well as further redesign. This project gave enormous interdisciplinary knowledge gain in electronics, power machines, thermodynamics,

mechanics, programming as well as soldering, parts manufacturing, etc. in both theoretical and practical terms.

Bibliography

[1] A A Prayogi, M. N., Indrabayu, and Rijal, M., „Design and implementation of rest api for academic information system," IOP Conference Series Materials Science and Engineering, 2019.

[2] American Society of Mechanical Engineers, „Boiler and pressure vessel code. section i: Rules for construction of power boilers," American Society of Mechanical Engineers, Tech. Rep., 2019.

[3] Banaszek, J., Bzowski, J., Domański, R., and Sado, J., *Termodynamika Przykłady i Zadania*. Oficyna Wydawnicza Politechniki Warszawskiej, 1998, ISBN: 83—7207—015—6.

[4] Cengel, Y. A., *Heat Transfer. A practical approach. Second Edition*. McGraw - Hill, ISBN: 007-1230424.

[5] Kim, M., Lee, Y., and Lee, J., „Thermal design and operational aspects of steam generators in advanced water-cooled nuclear power plants," *Nuclear Engineering and Design*, vol. 239, pp. 1098–1107, 2009.

[6] Li, J., Zhang, P., Zhang, X., and Li, D., „A review on boilers energy use, energy savings and emissions reductions," *Applied Energy*, vol. 259, pp. 1045–1056, 2019.

[7] Sakib, A. N., Ahmed, S., Rahman, S., Mahmud, I., and Belali, M. H., „Wpa 2 (wi-fi protected access 2) security enhancement: Analysis & improvement," *Global Journal of Computer Science and Technology*, vol. 12, 2012.

[8] Zidane, M. D., „Classification of boilers," *International Journal of Emerging Technology and Advanced Engineering*, vol. 2, 2012.

List of Abbreviations and Symbols

ASME American Society of Mechanical Engineers 13

MCU Main computational unit 20, 22

dm Change of mass 15

dQ Change of heat of the system 14

dU Change of internal energy of the system 14

dW Change of work done by the system 14, 15

$\mathrm{P_{heaters}}$ Power of heaters 15

$\mathrm{P_{max}}$ Maximal available power draw from the grid 17

$\mathrm{Q_{convection}}$ Heat transfered with convection 15

$\mathrm{Q_{heating}}$ Total heat delivered during heating phase 15

$\mathrm{Q_{radiation}}$ Heat transfered with radiation 15

$\mathrm{U_1}$ Internal energy right after water insertion into the system 15

$\mathrm{U_2}$ Internal energy at the end of the heating 15

$\mathrm{U_3}$ Internal energy in given moment of steam release 15

$\mathrm{U_{steam}}$ Internal energy of steam 15

$\mathrm{dm_{steam}}$ Change of mass of steam 16

$\mathrm{dm_{water}}$ Change of mass of water 16

i_{steam2} Specific enthalpy of steam at the end of heating proccess 15, 16

i_{steam3} Specific enthalpy of steam in given moment of steam release 15, 16

i_{steam} Enthalpy of steam 15

i_{water1} Specific enthalpy of water at the beggining of heating proccess 15

i_{water2} Specific enthalpy of water at the end of heating proccess 15

i_{water3} Specific enthalpy of water in given moment of steam release 15, 16

i_{water} Enthalpy of water 15

m_{steam2} Total mass of water inserted into the device 15, 16

m_{steam3} Mass of steam in given moment of steam release 15, 16

$m_{steam\_generated}$ Generated mass of steam 15

$m_{steam\_initial}$ Initial mass of steam 15

m_{steam} Mass of steam 15

m_{water1} Total mass of water inserted into the device 15

m_{water2} Total mass of water at the beggining of steam release 15, 16

m_{water3} Total mass of water in given moment of steam release 15, 16

m_{water} Mass of water 15, 16

ν_{steam2} Specific volume of steam at the end of heating proccess 15, 16

ν_{steam3} Specific volume of steam in given moment of steam release 15, 16

ν_{steam} Specific volume of steam 15

p_2 Pressure inside the device at the end of heating proccess 15, 16

p_3 Pressure inside the device in given moment of steam release 15, 16

p Pressure 15

$t_{heating}$ Time of operation of heaters 15

Q_{loss} Total heat losses during given operation phase 15, 16

U_{water} Internal energy of water 15

List of Figures

List of Tables

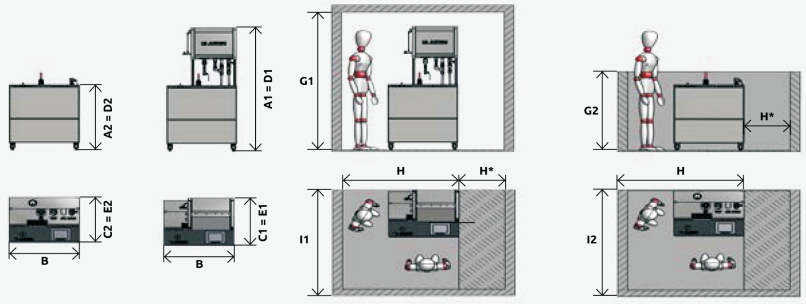
List of Appendices

Appendix 1
Available market offers

Dane techniczne

Rodzaj jednostki	EDI20	EDI40	EDI60	EDI80
Dopuszczalne ciśnienie (otwarcie zaworu bezpieczeństwa) PED 2014/68/EU kategoria II przy (PS*V < 200)	5,3 bara		3,4 bara	
Dopuszczalne ciśnienie (otwarcie zaworu bezpieczeństwa) PED 2014/68/EU kategoria III przy (PS*V < 1000)	12,5 bara		12,5 bara	
Wydajność pary do (przy temp. Wody zasilającej 15°C)	26,5 kg/h 0,44 kg/Min	53 kg/h 0,88 kg/Min	80 kg/h 1,32 kg/min	106 kg/h 1,77 kg/Min
Moc wytwornicy	20 kW	40 kW	60 kW	80 kW
Ciśnienie robocze	0,3-11 barów			
Czas rozgrzania	15 minut	7,5 minut	8 minut	6 minut
Pojemność zbiornika ciśn.	37,7 litra		58,8 litra	
Minimalna zawartość wody	14,3 litra		31,5 litra	
Przyłącze elektryczne	400 V / 50Hz			
Moc przyłączeniowa	22,2 kW	42,2 kW	62,2 kW	82,2 kW
Zabezpieczenie prądowe	min. 35 A - maks. 63 A	63 A	min. 100 A – maks. 125 A	125 A
Ciężar własny	190 kg	190 kg	250 kg	250 kg

Legenda do wymiarów (ilustracja przykładowa EDI20/40)



Wymiary

Ze zbiornikiem wody/kondensatu	EDI20/40	EDI60/80	Bez zbiornika wody/ kondensatu	EDI20/40	EDI60/80
Wysokość całkowita A1	1798 mm		Wysokość całkowita A2	1040 mm	1105 mm
Szerokość całkowita B	765 mm	1176 mm	Szerokość całkowita B	765 mm	1176 mm
Długość całkowita C1	703 mm		Długość całkowita C2	673 mm	
Minimalna wysokość transportowa D1	1798 mm		Minimalna wysokość transportowa D2	1040 mm	1105 mm
Minimalna długość transportowa E1	703 mm		Minimalna długość transportowa E2	673 mm	
Minimalna wysokość miejsca instalacji G1	2000 mm		Minimalna wysokość miejsca instalacji G2	1200 mm	
Minimalna szerokość miejsca instalacji H	1265 mm	1676 mm	Minimalna szerokość miejsca instalacji H	1265 mm	1676 mm
Opcjonalne miejsce dla konserwacji H*	500 mm		Opcjonalne miejsce dla konserwacji H*	500 mm	
Minimalna długość miejsca instalacji I1	1463 mm		Minimalna długość miejsca instalacji I2	1433 mm	

(/index.php?task=productprintpro&pid=36694&_wpnonce=9a61022315)

Seria VAP

Poproś o wycenę (https://www.babcock-wanson.com/pl/zapytanie-ofertowe/?quote=

Pobierz Broszura (https://r6n7c4f7.rocketcdn.me/wp-content/uploads/sites/56/2020

Gama VAP-LN firmy Babcock Wanson obejmuje wysokosprawne szybkie wytwornice pary, które zaprojektowano w celu spełnienia najbardziej rygorystycznych międzynarodowych przepisów dotyczących emisji NOx i CO.

Charakterystyka serii VAP

- Emisje zgodne z rozporządzeniami europejskimi
- Wytrzymała konstrukcja bez dużego zbiornika ciśnieniowego
- Dostępność pary już po trzech minutach od zimnego startu
- Niska objętość wody pod ciśnieniem, co zapewnia bezpieczeństwo nawet przy bardzo wysokim ciśnieniu
- Łatwa instalacja lokalna obok odbiornika pary, co obniża koszt instalacji rurowej
- Niezwykle kompaktowe wymiary
- W pełni zabudowana konstrukcja, w tym palnik firmy Babcock Wanson, pompa wody zasilającej i panel sterowania z wszystkimi elementami. Wszystkie podzespoły zostały okablowane i przetestowane w naszej fabryce.

Kategoria:

KOTŁY PAROWE (HTTPS://WWW.BABCOCK-WANSON.COM/PL/KATEGORIA-PRODUKTU/KOTLY-PAROWE/)

WYTWORNICE PARY (HTTPS://WWW.BABCOCK-WANSON.COM/PL/KATEGORIA-PRODUKTU/WYTWORNICE-PARY/)

VAP Premium – Kotły Parowe

VAP Premium	model	250 RR	400 RR	600 RR	900 RR	1200 RR	1500 RR
Produkcja pary (1)	Kg/h	250	400	600	900	1200	1500
Moc wyjściowa (1)	kW	175	280	419	628	838	1047
Szerokość A (2)	mm	1150	1400	1400	1600	1750	2000
Długość B (2)	mm	1150	1400	1400	1600	2000	2200
Wysokość C (2)	mm	1800	2100	2100	2300	2400	2700
Waga – wężownica napełniona	Kg	350	600	700	950	1350	1700

(1) Podana produkcja pary dotyczy temperatury wody zasilającej 80°C.

(2) Maksymalne ciśnienie pary wynosi 12 bar (wyższe ciśnienia na żądanie).

VAP Standard – Kotły Parowe

Types	150	250	400	600	900	1200	1500	2000	2500	3500	4250	5000	7000
Produkcja pary (kg/h) (1)	150	250	400	600	900	1200	1500	2000	2500	3500	4250	5000	7000
Moc wyjściowa (kW) (1)	106	176	281	423	635	846	1055	1410	1760	2468	2996	3520	

Dane techniczne[1]

Parametr / model	Jedn.	EKP 30	EKP 60	EKP 90	EKP 120
Wydajność pary nasyconej	kg/h	31	63	94	126
Ciśnienie robocze standard	bar	2,0 – 9,0	2,0 – 9,0	2,0 – 9,0	2,0 – 9,0
Moc cieplna	kW	21	42	63	84
Przybliżone wymiary jednostki					
- długość	mm	840	840	840	840
- szerokość	mm	850	850	950	950
- wysokość	mm	1800	1800	1850	1850
Masa	kg	230	250	270	300

[1] wersja farmaceutyczna jest produkowana na indywidualne zamówienie i powyższe dane mogą ulec zmianie

Realizujemy urządzenia w wykonaniu nietypowym na indywidualne zamówienia.

Zastrzega się możliwość wprowadzania zmian.
Urządzenie objęte ochroną patentową.

Figure 23. STIGEN available devices

Appendix 2

Django HTML website template

```
1
2  <!DOCTYPE html>
3  <html lang="en">
4    <head>
5      <meta charset="utf-8" />
6      <meta id="viewport" content="width=device-width, initial-scale=1" />
7      <title>Steam Generator Controls</title>
8    </head>
9    <body>
10     {% load bootstrap5 %} {% load static %} {% bootstrap_css %}
11     {% bootstrap_javascript %} {% bootstrap_messages %}
12
13     <link rel="stylesheet" href="{% static '/css/general.css' %}" />
14
15     <script src="https://ajax.googleapis.com/ajax/libs/jquery/3.6.1/jquery.
       min.js"></script>
16     <script src="https://canvasjs.com/assets/script/canvasjs.min.js"></
       script>
17     <script src="{% static '/js/data_fetch.js' %}"></script>
18     <script src="{% static '/js/stop_key.js' %}"></script>
19     <script src="{% static '/js/setpoints.js' %}"></script>
20     <script src="{% static '/js/post_rq.js' %}"></script>
21     <script src="{% static '/js/charts.js' %}"></script>
22
23     {% csrf_token %}
24
25   <div class="left_column_div">
26     <div class="stop_button">
27       <button id="emergency_stop" onclick="keySendPOST(this.id)">
28         EMERGENCY STOP
29       </button>
30     </div>
31     <div class="power_button">
32       <button id="power" onclick="keySendPOST(this.id)">
33         POWER
34       </button>
35     </div>
36     <div>
37       <label for="temp_setpoint">Temperature setpoint:</label><br />
38       <input type="number" id="temp_setpoint" name="temp_setpoint" /><br />
```

```
      <button onclick="sendSetpoints()">Submit temp setpoint</button><br />
    <br />
    </div>
    <div class="files_body">
      <div class="save_active">
        <span class="save_curr_state"
          >Data save: <span id="save">Inactive</span></span
        >
        <button id="save" onclick="keySendPOST(this.id)">ON/OFF</button>
      </div>
    </div>
    <div class="download_button">
      <button id="download" type="button" onclick="">
        Download last data file
      </button>
    </div>
    <div>
      <table>
        <th>Current setpoint</th>
        <tr>
          <td>Temperature</td>
          <td><span id="curr_temp_set">NaN</span></td>
          <td>[&#8451;]</td>
        </tr>
        <tr>
          <td>Pressure</td>
          <td><span id="curr_press_set">NaN</span></td>
          <td>[bar]</td>
        </tr>
      </table>
    </div>
    <div>
      <table class="params_table">
        <tr>
          <th>Parameter</th>
          <th>Value</th>
          <th>Unit</th>
        </tr>
        <tr>
          <td>Water temp</td>
          <td><span id="water_temp">NaN</span></td>
          <td>[&#8451;]</td>
        </tr>
        <tr>
          <td>Steam temp (sensor 1)</td>
          <td><span id="steam_temp_1">NaN</span></td>
```

```
            <td>[&#8451;]</td>
          </tr>
          <tr>
            <td>Steam temp (sensor 2)</td>
            <td><span id="steam_temp_2">NaN</span></td>
            <td>[&#8451;]</td>
          </tr>
          <tr>
            <td>Pressure</td>
            <td><span id="pressure">NaN</span></td>
            <td>[bar]</td>
          </tr>
          <tr>
            <th>Electric parameters (Phase 1)</th>
          </tr>
          <tr>
            <td>Voltage</td>
            <td><span id="voltage_ph1">NaN</span></td>
            <td>[V]</td>
          </tr>
          <tr>
            <td>Current</td>
            <td><span id="current_ph1">NaN</span></td>
            <td>[A]</td>
          </tr>
          <tr>
            <td>Active power</td>
            <td><span id="active_power_ph1">NaN</span></td>
            <td>[kW]</td>
          </tr>
          <tr>
            <th>Electric parameters (Phase 2)</th>
          </tr>
          <tr>
            <td>Voltage</td>
            <td><span id="voltage_ph2">NaN</span></td>
            <td>[V]</td>
          </tr>
          <tr>
            <td>Current</td>
            <td><span id="current_ph2">NaN</span></td>
            <td>[A]</td>
          </tr>
          <tr>
            <td>Active power</td>
            <td><span id="active_power_ph2">NaN</span></td>
```

```
          <td>[kW]</td>
        </tr>
        <tr>
          <th>Electric parameters (Phase 3)</th>
        </tr>
        <tr>
          <td>Voltage</td>
          <td><span id="voltage_ph3">NaN</span></td>
          <td>[V]</td>
        </tr>
        <tr>
          <td>Current</td>
          <td><span id="current_ph3">NaN</span></td>
          <td>[A]</td>
        </tr>
        <tr>
          <td>Active power</td>
          <td><span id="active_power_ph3">NaN</span></td>
          <td>[kW]</td>
        </tr>
      </table>
    </div>
    <div class="control_buttons">
      <div class="heater_w1_button">
        <span>Water heater 1: <span id="heater_1">NaN</span></span>
        <button id="heater_w1" onclick="keySendPOST(this.id)">ON/OFF</
button>
      </div>
      <div class="heater_w2_button">
        <span>Water heater 2: <span id="heater_2">NaN</span></span>
        <button id="heater_w2" onclick="keySendPOST(this.id)">ON/OFF</
button>
      </div>
      <div class="heater_w3_button">
        <span>Water heater 3: <span id="heater_3">NaN</span></span>
        <button id="heater_w3" onclick="keySendPOST(this.id)">ON/OFF</
button>
      </div>
      <div class="heater_st_button">
        <span>Steam heater 1: <span id="heater_st1">NaN</span></span>
        <button id="heater_st" onclick="keySendPOST(this.id)">ON/OFF</
button>
      </div>
      <div class="valve_button">
        <span>Steam valve: <span id="valve">NaN</span></span>
```

```
171         <button id="valve" onclick="keySendPOST(this.id)">OPEN/CLOSE</
    button>
172       </div>
173     </div>
174   </div>
175   <div class="right_column_div">
176     <div id="chartContainer1" style="height: 370px; width: 740px"></div>
177     <div id="chartContainer2" style="height: 370px; width: 740px"></div>
178     <div id="chartContainer3" style="height: 370px; width: 740px"></div>
179     <div id="chartContainer4" style="height: 370px; width: 740px"></div>
180     <div id="chartContainer_pid" style="height: 370px; width: 740px"></div>
181   </div>
182   </body>
183 </html>
```

Appendix 3

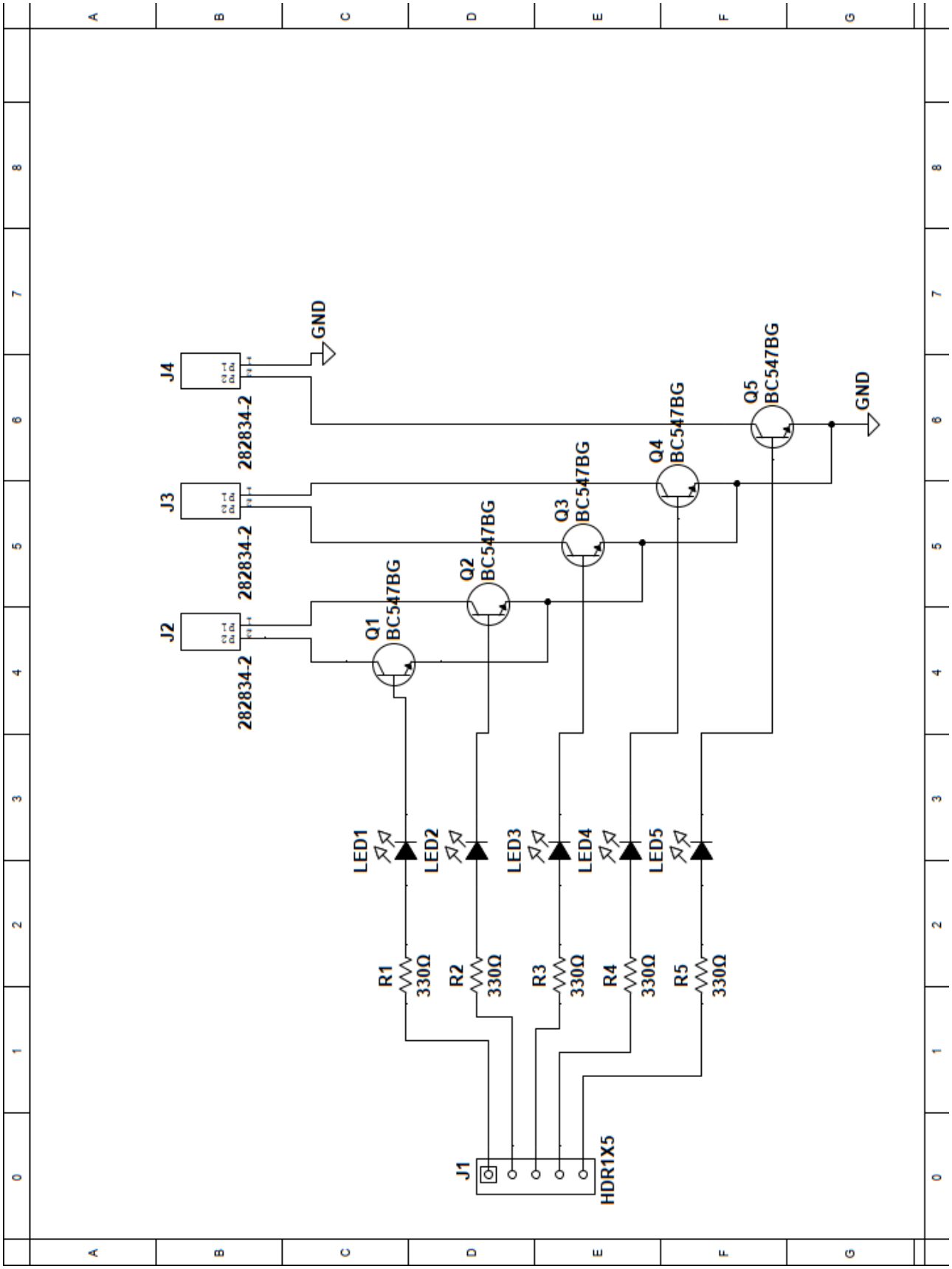
Javascript service: Data update

```
1
2  var json_data = {};
3
4  var refresh_interval = 200 // Miliseconds
5
6  function reloadData() {
7    $.getJSON("/update_data/", function (data) {
8      json_data = data;
9    });
10
11   console.log(json_data);
12
13   document.getElementById("curr_temp_set").textContent = json_data["
       curr_temp_set"];
14
15   document.getElementById("water_temp").textContent = parseFloat(json_data[
       "water_temp"]).toFixed(2);
16   document.getElementById("steam_temp_1").textContent = parseFloat(
       json_data["steam_temp_1"]).toFixed(2);
17   document.getElementById("steam_temp_2").textContent = parseFloat(
       json_data["steam_temp_2"]).toFixed(2);
18   document.getElementById("pressure").textContent = parseFloat(json_data["
       pressure"]).toFixed(2);
19   // Phase 1
20   document.getElementById("voltage_ph1").textContent = json_data["
       voltage_ph1"];
21   document.getElementById("current_ph1").textContent = json_data["
       current_ph1"];
22   document.getElementById("active_power_ph1").textContent = json_data["
       active_power_ph1"];
23   // Phase 2
24   document.getElementById("voltage_ph2").textContent = json_data["
       voltage_ph2"];
25   document.getElementById("current_ph2").textContent = json_data["
       current_ph2"];
26   document.getElementById("active_power_ph2").textContent = json_data["
       active_power_ph2"];
27   //Phase 3
28   document.getElementById("voltage_ph3").textContent = json_data["
       voltage_ph3"];
```

```
29    document.getElementById("current_ph3").textContent = json_data["
       current_ph3"];
30    document.getElementById("active_power_ph3").textContent = json_data["
       active_power_ph3"];
31
32    var temp = "OFF";
33    if (json_data["heater_w1"]) {
34      temp = "ON";
35    }
36    document.getElementById("heater_1").textContent = temp;
37
38    temp = "OFF";
39    if (json_data["heater_w2"]) {
40      temp = "ON";
41    }
42    document.getElementById("heater_2").textContent = temp;
43
44    temp = "OFF";
45    if (json_data["heater_w3"]) {
46      temp = "ON";
47    }
48    document.getElementById("heater_3").textContent = temp;
49
50    temp = "OFF";
51    if (json_data["heater_st"]) {
52      temp = "ON";
53    }
54    document.getElementById("heater_st1").textContent = temp;
55
56    temp = "CLOSED";
57    if (json_data["valve"]) {
58      temp = "OPEN";
59    }
60    document.getElementById("valve").textContent = temp;
61
62    temp = "INACTIVE"
63    if(json_data["save"]){
64      temp = "ACTIVE"
65    }
66    document.getElementById("save").textContent = temp;
67  }
68
69  setInterval(reloadData, refresh_interval);
```

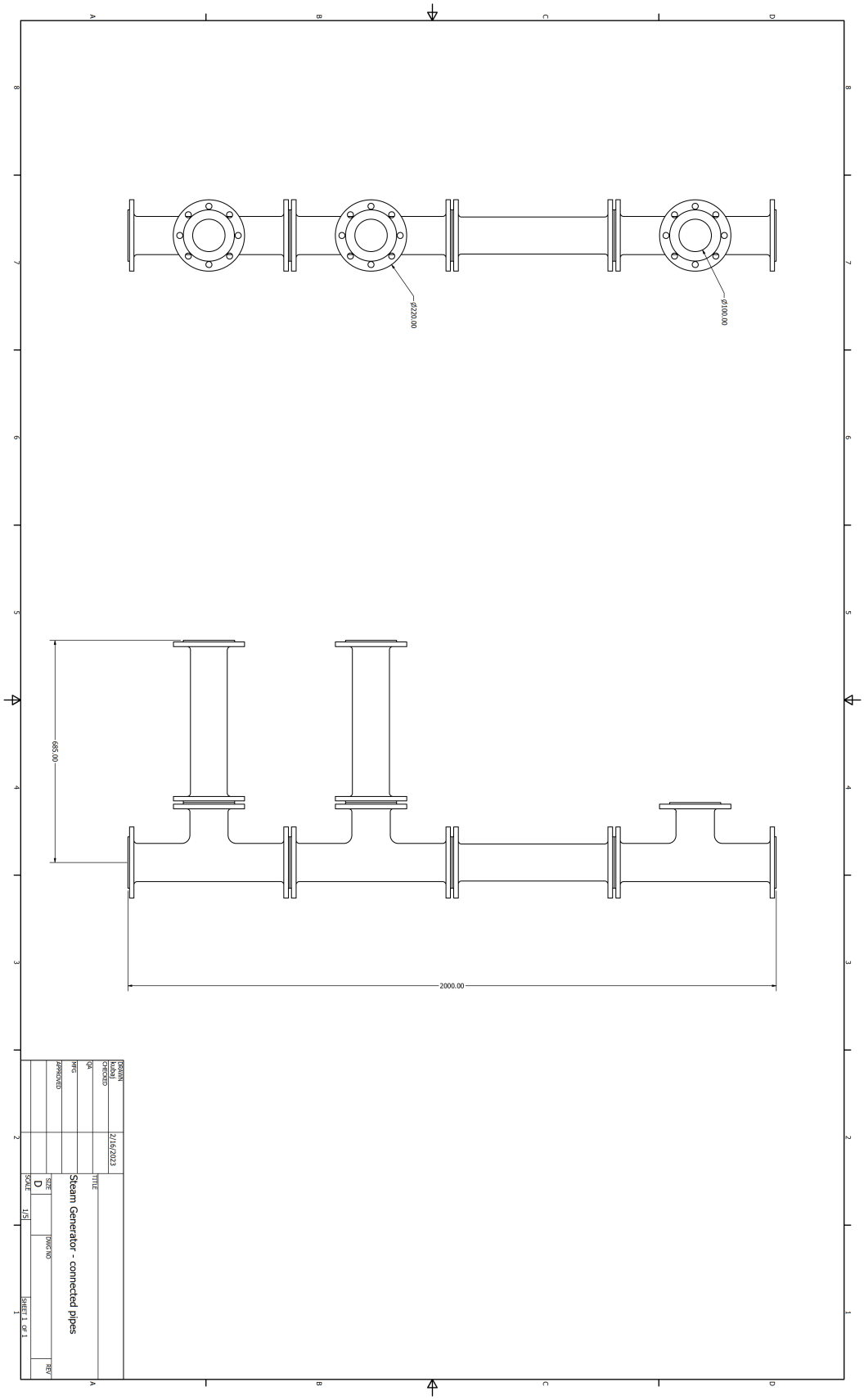
Appendix 4

Transistor switches module simplified schematic


J4
282834-2
GND
J3
282834-2
J2
282834-2
Q1
BC547BG
Q2
BC547BG
Q3
BC547BG
Q4
BC547BG
Q5
BC547BG
GND
LED1
LED2
LED3
LED4
LED5
R1
330Ω
R2
330Ω
R3
330Ω
R4
330Ω
R5
330Ω
J1
HDR1X5

Appendix 5

Simplified technical drawing of device prototype


Ø220.00
Ø100.00
685.00
2000.00
DRAWN
Kuhan
CHECKED
QA
MFG
APPROVED
2/16/2023
Steam Generator - connected pipes
SIZE
D
DWG NO
REV
SCALE
1:5
SHEET 1 OF 1

Appendix 6

Python script for system parameters calculation

```
1
2  import pandas, numpy, pyfluids, time
3  from pyXSteam.XSteam import XSteam
4
5  def process_data(file='data.txt', separator=',', header=None) -> pandas.
       DataFrame:
6
7      data = pandas.read_csv("data.txt", sep=",", header=None)
8
9      for col in data:
10         data[col] = data[col].map(lambda x: str(x).replace('(', ''))
11         data[col] = data[col].map(lambda x: str(x).replace(')', ''))
12         data[col] = data[col].map(lambda x: str(x).replace('nan', ''))
13         data[col].replace('', numpy.nan, inplace=True)
14
15     data.dropna(axis=1, inplace=True)
16
17     # drop columns with names
18     drop_index = list(range(0, 34,3))
19     drop_index.append(25)
20     drop_index.append(28)
21     drop_index.append(31)
22
23     data = data.drop(drop_index, axis=1)
24
25     data.columns = ['Water_temperature_1',
26                     'Steam_temperature_1',
27                     'Steam_temperature_2',
28                     'Heater_water_ph1',
29                     'Heater_water_ph2',
30                     'Heater_water_ph3',
31                     'Heater_steam',
32                     'Valve',
33                     'Pressure']
34
35     data = data.convert_dtypes().astype(float)
36
37     data = data[data.Water_temperature_1 >= 0] # rm rows with false
       temperature readouts
```

```
38
39      return data
40
41  def gr(g, beta, delta_T, L, ni) -> float:
42      return (g * beta * delta_T * pow(L, 3))/(pow(ni, 2))
43
44  def pr(mu, cp, k) -> float:
45      return (mu * cp)/ k
46
47  def ra(pr, gr) -> float:
48      return gr * pr
49
50  def nu_v_plate(ra, pr) -> float:
51      return pow( (0.825 + (0.387 * pow(ra, 1/6))/pow((1 + pow( 492 / pr ,
        9/16)), 8/27)), 2)
52
53  def nu_h_plate_down(ra) -> float:
54      return 0.27 * pow(ra, 1/4)
55
56  def nu_h_plate_up(ra) -> float:
57      return 0.54 * pow(ra, 1/4)
58
59  def A_cylinder(r, h) -> float:
60      return 2 * 3.14 * r * h
61
62  def A_plate(r) -> float:
63      return 3.14 * pow(r, 2)
64
65  def T_out(T1, k, Tamb, alfa, L) -> float:
66      return (T1 * k + Tamb*alfa*L)/(k + alfa * L)
67
68  def Q_conv(alfa, A, delta_t) -> float:
69      return alfa * A * delta_t
70
71  def Q_rad(epsilon, T0, Tamb, A, sigma = 5.6703e-8) -> float:
72      return epsilon * sigma * (pow(T0 + 273.15, 4) - pow(Tamb + 273.15, 4))
        * A
73
74  def m_steam(data: pandas.DataFrame, mw) -> float:
75      data1 = data[data['Heater_water_ph1'] == 1]
76      data1 = data1[data1.index <= 5000]
77
78      Q_in = data1.sum()['Total_power'] / 1000
79      Q_loss = data1.sum()['Total_loss'] / 1000
80      i_w1 = data.iloc[0]['Water_enthalpy']
81      i_w2 = data1.iloc[-1]['Water_enthalpy']
```

```
    i_s2 = data1.iloc[-1]['Steam_enthalpy']
    p2 = data1.iloc[-1]['Pressure']
    ni2 = data1.iloc[-1]['Steam_specific_volume']

    print('Final heating time: %d [s]' % int(data1.index[-1]))

    return (Q_in - Q_loss + mw*(i_w1 - i_w2))/ (i_s2 - i_w2 - p2*1e2*ni2)

def dm_steam_valve(mw_init, ms_init, ip1, ip2, iw1, iw2, p1, p2, ni1, ni2, Q_loss) -> float:
    return (ms_init*(2*ip1 - p1*1e2 * ni1) - mw_init*(iw2-iw1) - Q_loss)/(ip2 + ip1 - p2*1e2 * ni2)

def T_surf(T_steam, T_amb) -> float:
    return T_steam * (1  - 1.01558e-4 * (T_steam - T_amb))

def Nu_condition(D, GR, L) -> bool:
    return D >= (35*L) / pow(GR,1/4)

init_time = time.time()

amb_temp = float(24) # [C]
dev_surf_temp = float(152) # [C]
dev_len = float(1.62) # [m]
dev_out_diameter = float(0.118) # [m]
dev_in_diameter = float(0.1) # [m]
cover_diameter = float(0.22) # [m]
dev_vol = float(0.024) # [m^3]
heater_power = float(1850) # [W]
total_mass = float(10.1437) # [kg]
k = float(52) # [W/m/K] - cast iron conductivity
cover_emisivity = float(0.11)
dev_emisivity = float(0.97)

################
# Heat losses #
################

air = pyfluids.HumidAir().with_state(
    pyfluids.InputHumidAir.pressure(101325),
    pyfluids.InputHumidAir.relative_humidity(0),
    pyfluids.InputHumidAir.temperature(amb_temp),
).as_dict()

if not dev_out_diameter >= (35 * dev_len)/(pow(gr(9.81, 3.4e-3, dev_surf_temp - amb_temp, dev_len, air['dynamic_viscosity']), 1/4)):
```

```
    raise Exception

####################
# Parameters calc #
####################

data = process_data(file='data.txt')

steamTable = XSteam(XSteam.UNIT_SYSTEM_MKS)

specific_volume = dev_vol / total_mass # [m^3 / kg]
x = True
y = False

for index, row in data.iterrows():
    total_power = (row['Heater_water_ph1'] + row['Heater_water_ph2'] + row[
    'Heater_water_ph3']) * heater_power

    data.loc[index, 'Saturation_temperature'] = steamTable.tsat_p(row['
    Pressure'])
    data.loc[index, 'Steam_enthalpy'] = steamTable.hV_p(row['Pressure'])
    data.loc[index, 'Steam_specific_volume'] = steamTable.vV_p(row['
    Pressure'])
    data.loc[index, 'Water_enthalpy'] = steamTable.hL_p(row['Pressure'])
    data.loc[index, 'Total_power'] = total_power

    T_surface = T_surf(row['Steam_temperature_2'], amb_temp)

    if Nu_condition(dev_out_diameter, gr(9.81, 3.4e-3, T_surface, dev_len,
    air['dynamic_viscosity'] / air['density']), dev_len):
        alfa_cylinder = nu_v_plate(ra(pr(air['dynamic_viscosity'], air['
    specific_heat'], air['conductivity']),
                                  gr(9.81, 3.4e-3, T_surface, dev_len, air['
    dynamic_viscosity'] / air['density'])),
                              pr(air['dynamic_viscosity'], air['specific_heat
    '], air['conductivity'])) * air['conductivity'] / dev_len

        data.loc[index, 'Alfa_cylinder'] = alfa_cylinder

    alfa_cover_v = nu_v_plate(ra(pr(air['dynamic_viscosity'], air['
    specific_heat'], air['conductivity']),
                                  gr(9.81, 3.4e-3, T_surface, cover_diameter,
     air['dynamic_viscosity'] / air['density'])),
                              pr(air['dynamic_viscosity'], air['specific_heat
    '], air['conductivity'])) * air['conductivity'] / cover_diameter

```

```
161     RA = ra(pr(air['dynamic_viscosity'], air['specific_heat'], air['
    conductivity']),
162                                         gr(9.81, 3.4e-3, T_surface,
    cover_diameter / 4, air['dynamic_viscosity'] / air['density']))
163     if RA in range(1e5, 1e11):
164         alfa_cover_h_down = nu_h_plate_down(RA)
165 
166     RA = ra(pr(air['dynamic_viscosity'], air['specific_heat'], air['
    conductivity']),
167                                       gr(9.81, 3.4e-3, T_surface,
    cover_diameter / 4, air['dynamic_viscosity'] / air['density']))
168 
169     if RA in range(1e4, 1e7):
170         alfa_cover_h_up = nu_h_plate_up(RA)
171 
172     data.loc[index, 'Alfa_cylinder'] = alfa_cylinder
173     data.loc[index, 'Alfa_cover_v'] = alfa_cover_v
174     data.loc[index, 'Alfa_cover_h_up'] = alfa_cover_h_up
175     data.loc[index, 'Alfa_cover_h_down'] = alfa_cover_h_down
176     data.loc[index, 'Surface_temp'] = T_surface
177 
178     conv_heat_losses = (Q_conv(alfa_cylinder, A_cylinder(dev_out_diameter
    /2, dev_len), T_surface - amb_temp) \
179                         + 2 * Q_conv(alfa_cover_v, A_plate(cover_diameter
    /2), T_surface - amb_temp) \
180                         + Q_conv(alfa_cover_h_down, A_plate(cover_diameter
    /2), T_surface - amb_temp) \
181                         + Q_conv(alfa_cover_h_up, A_plate(cover_diameter/2)
    , T_surface - amb_temp)) / 1e3
182 
183     rad_heat_losses = (Q_rad(dev_emisivity, T_surface, amb_temp, A_cylinder
    (dev_out_diameter/2, dev_len)) \
184                         + 4 * Q_rad(cover_emisivity, T_surface, amb_temp,
    A_plate(cover_diameter / 2))) / 1e3
185 
186     heat_losses =  conv_heat_losses + rad_heat_losses
187 
188     data.loc[index, 'Convective_heat_loss'] = conv_heat_losses
189     data.loc[index, 'Radiation_heat_loss'] = rad_heat_losses
190     data.loc[index, 'Total_loss'] = heat_losses
191 
192     if row['Valve'] == 1:
193         y = True
194 
195     if y:
196         if x:
```

```
            steam_mass = m_steam(data, total_mass)
            water_mass = total_mass - steam_mass
            x = False

        p1 = data.loc[int(index) - 1, 'Pressure']
        p2 = row['Pressure']
        ip1 = data.loc[int(index) - 1, 'Steam_enthalpy']
        ip2 = steamTable.hV_p(row['Pressure'])
        iw1 = data.loc[int(index) - 1, 'Water_enthalpy']
        iw2 = steamTable.hL_p(row['Pressure'])
        ni1 = data.loc[int(index) - 1, 'Steam_specific_volume']
        ni2 = steamTable.vV_p(row['Pressure'])

        data.loc[index, 'Steam_mass'] = steam_mass
        steam_mass = dm_steam_valve(water_mass, steam_mass, ip1, ip2, iw1,
    iw2, p1, p2, ni1, ni2, heat_losses)
        steam_mass_flow = data.loc[int(index) - 1, 'Steam_mass'] -
    steam_mass

        if steam_mass_flow <= 0:
            continue

        data.loc[index, 'Steam_mass_flow'] = steam_mass_flow

data.to_csv('out.csv')

data1 = data[data['Steam_mass_flow'] > 0]

print(data)
print("Average steam mass flow: %.6f [kg/s] = %.3f [kg/h]" % (data1['
    Steam_mass_flow'].mean(), data1['Steam_mass_flow'].mean() * 3600))
print("Produced steam mass: %.3f [kg]" % m_steam(data, total_mass))
print("Execution time: %.2f [s]" % (time.time() - init_time))


```