Optimal Energy Management for IoT Connected Microgrid Using MILP

Case of Meteorian Soderfjarden

Ahmed Mabrouk

Master’s Thesis

Master’s degree program in Automation Technology - Intelligent Systems

Vaasa 2024

MASTER THESIS

Author: Ahmed Mabrouk
Degree Programme and place of study: Automation Technology, Vaasa
Specialization: Intelligent Systems
Supervisor(s): Ray Pörn, Hans Linden

Title: Optimal Energy Management for IoT Connected Microgrid using MILP.

Date: 27.05.2024 Number of pages: 67 Appendices: 3

ABSTRACT

In this Master's thesis work, I had the honor to take part in implementing an energy optimization solution. The thesis work had as a goal to optimize the energy consumption for one of the greatest exhibitions in Vaasa region: Meteoria of Soderfjarden which is located in Sundom village. The implemented solution was developed to make the Meteoria Microgrid as autonomous in consumption as in energy generation from renewable sources. The Meteoria of Soderfjarden has already many options for generating electricity while taking advantage of wind and solar potential of Vaasa region.

The software solution was designed to solve a linear programming problem that had the objective of reducing fuel consumption. To achieve this, different software and frameworks were involved and taking advantage of the existing Novia IoT platform to collect the needed information for further computation during the optimization problem solving to find the possible optimal solutions.
Following to the solution development, a visualization dashboard of the results of this optimization was tested in real-time which was then published in the novia broker as well as the command values. The optimal values resulting from the MILP solving were then populated through the MQTT agent and the graphs accordingly.

The main optimization work was developed using PuLp framework and using Gurrobi as an agent solver. The MQTT agent was developed using Paho, Flask, and Plotly and was incorporated to visualize results in the form of a real-time dashboard. The resulting work is a model and a complete data storage system that can be depicted under the Novia MQTT platform to govern in real-time the energy inputs and outputs while fostering the use of energy storage and taking advantage of green resources to minimize fuel generator consumption.

Language: English
Key Words: Optimization, Renewable Energy, IoT, MQTT, MILP, visualization, Python

ACKNOWLEDGMENT

I want to express through this work my gratitude to anyone who has contributed to its fulfillment. Firstly, my thankful thoughts go to my supervisors, Ray Pörn and Hans Linden who have guided me and helped me to get my thesis complete with their precious guidance. Thank you for your advice and orientation during this research.

I want to send my deepest thanks to all Novia University of Applied Sciences staff, who work 24/7 to provide us have the best study conditions and resources. Also, I must thank my ex-supervisors and colleagues in the Technobothnia research center where I started the journey of research and development in the field of intelligent systems as I do appreciate this experience during which I had the chance to polish my skills and knowledge. Furthermore, I will not forget the help of my colleagues and friends who helped me with their suggestions and insightful ideas throughout this work. I would like to thank anyone who has provided me with the right materials and resources to accomplish my master's thesis.

Moreover, I would like to thank my parents in my home country who supported me during my study in Finland, and without them, this work could not see the light. You mean the world to me!

TABLE OF CONTENT

LIST OF FIGURES AND TABLES

LIST OF ACRONYMS

BESS	Battery energy storage system
CSE	Conventional sources of energy
CHP	Combined heat and power
DV	Decision variable
DRL	Deep reinforcement learning
DE	Differential Evolution
ESS	Energy storage system
EMS	Energy management system
GHG	Greenhouse gases
GCP	Google Cloud platform
GIS	Geographical information system
HRES	Hybrid renewable energy sources
HMG	Hybrid microgrid
HNG	Hybrid nanogrid
HCES	Hybrid clean energy supply
LP	Linear programming variable
LPSP	Loss of power supply probability
LCOE	Levelized cost of energy.
MOSC-MEMS	Multi-objective security constrained microgrid energy management system
MOSSO	Mult-objective salp swarm optimization
MODA	Multi-objective dragonfly algorithm
MOGOA	Multi-objective grasshopper optimization algorithm
MOALO	Multi-objective ant lion optimizer
MILP	Mixed Integer Linear Problem

NCSE	Non-Conventional sources of energy
PSO	Particle Swarm Optimization
RES	Renewable energy sources
ROA	Remora optimization algorithm
IEC	The International Electrotechnical Commission
IIoT	Industrial IoT
Rpi	Raspberry pi

1. Introduction

Hybrid microgrids (HMGs) are power plants located in isolated locations that cannot be accessible through the main grid and have a combination of more than two resources from which energy can be produced.

Although HMGs might differ in size and form, still the goal is usually the same: balancing the utility grid and integrating non-conventional sources of energy (NCSE). Renewable energy sources (RES) started to be more and more valuable in our age not only because of their role in compensating the ran out of conventional sources of energy (CSE) like coal and petrol, but also to mitigate climate harming caused by CSE as they are emitting a high amount of greenhouse gases (GHG) typically CO_2 and they are therefore considered as the most contributor of the climate change[1].

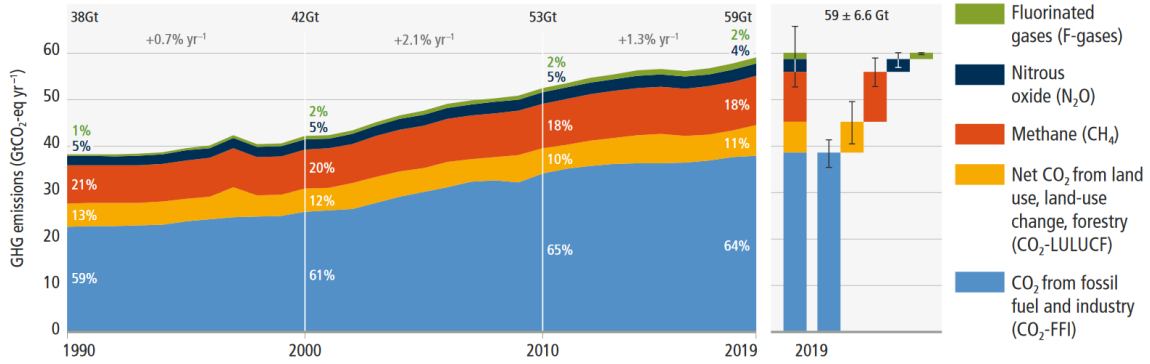

Figure 1: Global net anthropogenic GHG emissions 1990–2019, Source: IPCC (2022). https://www.ipcc.ch/ [2]

However, GHG emissions are only a consequence of satisfying the industrial revolution's needs in all sectors for energy and it was not possible to meet these needs without spinning diesel generators (DGs) by burning fuel and coal in thermal power plants in addition to nuclear power which is itself not considered as harmful in the short term as the process does emit considerably less GHG as a direct emission. On the other hand, considering the whole life cycle when producing energy but still considering its harmful impact the human health and is contributing to land environment pollution because of the reactors cooling water and disposal of the radioactive waste at the end of the generation cycle that require landfill surface [3].

For the above-mentioned reasons, considering off-grid micro/nano girds as alternatives has gained a lot of interest in the last few years for not only reducing the impact of rising temperature over the last decade but also contributing to reducing the loads by adding more electrification infrastructures and relaxing the demand considering the growing demand for electricity [2]

However, when it comes to off-grid microgrids and nano grids they are usually intended to be built around smaller areas that are not connected to the main grid for geo-economic reasons and they are not connected to the main grid by any means typically named hence by the name off-grid which means they are standalone utilities that depend on their electricity generators [4].

The ideal off-grid microgrid needs to be self-sufficient in regard to its available energy resources however since the NCSEs are approved to have an intermittent behavior and are highly dependent on the weather conditions as is the case for solar and wind energy (typically solar irradiance for solar and wind speed for wind) therefore the so-called “hybrid” microgrids is another solution that incorporate at least one CSEs to solve this issue. Thus, HMGs are expected to fill the production gap during bad weather conditions to fully achieve the energy self-dependency of these kinds of microgrids [5], [6]

In addition to its resilience and ability to guarantee the availability of electricity regardless of what the forecasting of NCSEs is in a specific daytime, the HMGs based on CSEs are most often associated with lower costs per kWh produced which makes them a great choice for the islanded locations where there is no interconnection to the main grid or difficulties to access power lines [7], [8].

One more crucial aspect of HMGs is their capacity to store the excess energy produced from the available NCSEs when the weather conditions are more convenient, the intermittency behavior of these sources allows them to produce energy that cannot be fully consumed at a given time since each microgrid is sized upon on its loads. HMGs are normally sized based on the worst-case production profile which is typically in the winter times because of the low solar irradiance when solar PV is one part of the HMG, therefore, a reliable energy storage system (ESS) is recommended in an HMG system to achieve power resiliency [9].

Energy can be stored in different forms namely thermal (heat or cold), kinetic (pumped hydro), chemical (battery storage), and hydrogen fuel cells [10], hence there should be a suitable storage system that meets the end consumers’ needs. For an island microgrid placed in cold locations, both heat and battery storage are more often adopted because of their long-lasting storage property that allows storing energy for a long duration and retrieving it directly in the form of heat or electricity when mostly demanded.

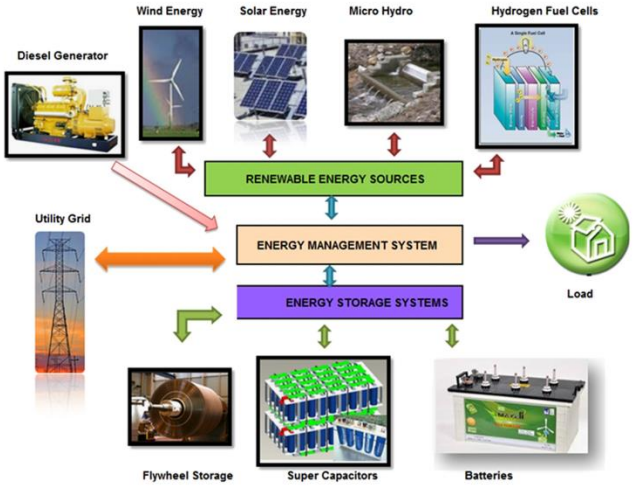

Figure 2: Typical HG layout [11]

However, when it comes to energy self-sufficiency in HMGs, multiple criteria must be met to keep producing and consuming energy as coherent and as balanced as possible. Therefore, in the early

phase of designing an HMG, engineers must consider providing the best technical details about what element should be incorporated, hence selecting the appropriate electrical switchgear for the plant concerning engineering design, financial budget, environment requirements, and its reliability goals [12]:

- Load profile : Power distribution along the consumption site.
- Generators : NCSE or CSE generator type that will meet the load demand.
- Production site : Location of HMG concerning the national grid and populated areas.
- Consumption site : Type of equipment that consumes energy (resistive, inductive,..).
- Weather conditions : These are related to where the HMG's site is operating.
- Geo-factors : Geographical conditions for risk assimilation [13].
- Storage : Type of ESS used for storing excess energy production.

In the same degree of importance these criteria must be respected, managing the power flow is also critically essential to operating HMGs optimally. The fact that these electrical infrastructures do not follow a deterministic fashion of producing energy but rather is more probabilistic which depends on the environmental conditions mostly when RES are the biggest parts of the HMGs bringing on the complexity of energy intermittence since those sources of energy are more often related to weather conditions while producing energy of the modern microgrids are governed by an Energy Management System (EMS) that controls constantly the power flow to optimize the whole production-consumption process to achieve resilience and reliable operations of a typical microgrid as well as contribute into balancing the main national grid [14]. Most of the on-grid microgrids are essentially served by the EMS that controls the national main grid where energy dispatching rules are inherited from the national dispatching center that controls power flow on a larger scale within the country, this makes the on-grid microgrids more secure on energy procurement even with fewer storage capabilities due to the limitation of the grid itself when serving as a dynamic storage system; Therefore, reducing the energy outages and enhancing balancing capabilities for the national grid is among the important aspects in modeling objectives of an optimization in a way that focus more on maximizing RES harvesting and incorporate it as main building block of HMGs [8].

However, this is not a simple task when it comes to HMGs, for the obvious reasons:

- No-grid connection: It must serve its storage systems and is highly vulnerability to blackouts.
-
- Hybrid RES: Different RES have different physical properties that make the HMGs a multimodal grid system [9].

1.1 Aim of research

Energy optimization is purely a mathematical topic that must be studied and formulated appropriately for further analysis and correct solving. Over the course of the last years, many researchers have exploited the leverage of high technologies based on AI to lower the bar and provide solutions based on metaheuristic techniques to provide software solutions rapidly and reduce time to market, however, the core problem behind all kind of optimization problems is still the same. In this thesis work, I will abord the deterministic approach.

The goal of this thesis work is to bring up a new feature in the IoT-based microgrids and enhance their performance regarding managing energy resources and loads. Previously, and based on the literature, the IoT part of micro-grid systems was mainly concerned with harvesting the data and storing it in some databases for further visualization and for simple automation processes, typically for guaranteeing the safe operations of each component of these off-grid systems. However, the desired result to be achieved through this research is to bring up a new feature to IoT systems and prove its operability in a real-life case (Meteorian Soderfjarden).

With this solution, the Meteorian Soderfjarden will have the ability to execute further computation and complexity that promote the use of green resources and therefore reduce fuel consumption costs without relying on human interaction. The core intention will be to embed an optimization-solving framework with the Meteoria IoT system and test its reliability in real-time data. Furthermore, extracting the useful data and visualizing it in the form of a dashboard will be a complementary part of this research to further interpret the optimizer results.

More precisely, through this thesis, I am going to explore the effectiveness of solving a mixed linear programming problem (MILP) which is a constrained optimization problem that will take into consideration an optimization objective function along with the constraints that bound the operation of the off-grid system, in our case is a ‘nano-grid’ or a small size micro-grid. In addition to the before stated objectives, a secondary area of this thesis will be to explore and test the interactions between the optimization inputs and outputs and how the IoT system that controls Meteorian Soderfjarden through Novia UAS servers using MQTT protocol will behave in general to guarantee a full deployment in the future.

Furthermore, this thesis work should at the end deliver the resources and the methodology on how the complete model will manage and solve the optimization problem of Meteorian Soderfjarden off-grid system as well as orchestrating the whole interface and data flow back and forth between Novia IoT platform and the internal/external data collected from the operational environment at Meteorian Soderfjarden site. Lastly, as with any thesis work, I will sort the topics of improvements to make the provided solution more solid and suitable for application on reel life, and what the challenges that we need to overcome for a reliable operation of the solution in such real case scenario for large scale micro-grids.

2. Literature review

To achieve energy optimization, some different techniques and approaches can be employed to do the tasks while they are different in principle, not all the optimization strategies can be used equally as the optimization itself depends on many criteria: requirements, the dimensionality of the problem, uncertainties and so on.

Optimizing an off-grid utility can be seen from various angles depending on what objective we want to achieve. Most of the time the ultimate goal of an off-grid power plant optimization is to reduce the dependency on the conventional resources (i.e Diesel fuel and oil). In Japan, a reinforcement learning technique (DRL) has been incorporated to solve an optimization problem for a double objective function of optimizing the operability of the renewable energy production system and minimizing the battery storage deterioration of an energy building [15].

In Ghulam Shah township in Pakistan, a site has a combination of resources that contains solar PV, Diesel generators, Combined heat and power (CHP) biomass generators, and a BESS for the surplus storage[16], employing a hybrid multi-criteria decision-making model (MCDM) which combines the analytical methods AHP, MOORA, TOPSIS and EDAS with the aid of the HOMER PRO software, to address an investigation study about on/off-grid hybrid clean energy supply (HCES)[16].

An optimization method has been explored to minimize the daily operation cost of a hybrid energy system, combining solar PV and pumped hydro storage, this was achieved by developing a modified version of the Crow search algorithm (CSA), and results showed that the optimal selection of the hybrid system components impacts positively in reducing the operation cost to the value of power purchased [17].

Concerning grid stability an effective multi-objective optimization model of an EMS when managing a microgrid operation composed of a wind turbine, ESS, and solar PV was investigated, the proposed multi-objective security-constrained microgrid energy management system (MOSC-MEMS) has to satisfy an objective function of minimizing power losses and operating costs and stabilize the main grid voltage in addition to comply to generation contingency constraints [18].

A multi-objective optimization problem for an off-grid microgrid located in Cameron was solved that combines solar PV, BSS, and hydropower, while incorporating a stochastic approach, mainly: Particle Swarm optimization (PSO) and Differential Evolution (DE). With a double objective function of minimizing the LCOE (Levelized Cost of Energy) achievable with a reasonable loss of power supply probability (LPSP)[19]. An optimization solution was presented for a standalone MG to supply 15 houses with electricity in a rural area in Djefla, Algeria. The author exploited the state-of-the-art metaheuristic algorithms in his study namely a variant of the PSO algorithm so called the multi-objective Salp Swarm algorithm (MOSSA). The MG of the study case constituted of solar PV, wind turbine, Batteries, and diesel generator. Taking into consideration the low LCOE and low LPSP as double objective functions. The assessment of obtained results was investigated to other metaheuristic techniques (i.e.: Multi-objective dragonfly algorithm (MODA), Multi-objective grasshopper optimization algorithm (MOGOA), Multi-objective ant lion optimizer (MOALO) [20].

From the system automatization perspective, with real-time control (RTC) of the MG, multiple techniques have been developed to control and supervise the power flow and status of MG entities

over time. A control system for a seawater desalination power plant powered by wind turbines and solar PV has been simulated [21]. The author demonstrated using the CISCO packet tracer tool the possibility to control the power plant through an internal server while granting the security interactions via an Access control list (ACL) which will filter the transmission packets thus, restricting communication from not allowable devices or persons [22].

A new information model for exchanging reel time data based on a novel cloud-based SCADA solution has been proposed that relies on the two IEC standards: IEC 61970 and IEC 61968, which can handle the interoperability of the MG, considering the information exchange between different entities of the MG. Reel time communication for an operational MG data exchange increases the efficiency of the future Grids, one of the recent propositions considers a cloud computing platform as a potential candidate for this purpose as it can decouple the software from the computing resources and therefore reduce energy usage through stack virtualization systems and simplifying automation, however, major challenges must be overcome by could computing systems, including and not limited to security, performance, and reliability [21].

An IoT-based system was suggested to control and monitor the state of typical MG, using mainly Wireless Sensor Network (WSN) equipment and metering devices he can ensure the high-speed data communication connected all together via Arduino boards hardware to control the MG entities which are in this use case: PV panel, wind system, battery, DC-DC converter and three phase inverters. Moreover, the author took advantage of the Remora optimization algorithm (ROA) to optimize the process of selecting hyperparameters fed to the Pulse Coupled Neural Network (PCNN) hence, enhancing the communication speed [23].

3. Outline of the proposed solution

For all the solutions explored, there is a lack of coming up with one solution that can tackle both the Optimization process and the automation process at once. In the proposed solution I will take advantage of the previous work of [24] on the same project and integrate a MILP Optimizer to control the efficiency of Meteor of Soderfjarden Nano grid (NG) as well as the real-time supervision using a cloud-hosted application.

The solution will be able to collect data, pre-process it trigger the optimization process based on the optimizer capabilities, and then send out the commands to the different components of the NA for better performance. The optimizer has an object function to minimize the DG fuel consumption, therefore maximizing the use of NCSE over CSE.

The Meteor of Soderfjarden has a hybrid model of a microgrid where the components are described in Table 1 [24].

Table 1: Meteorian Soderfjarden components.

Component	Device	Maximum power/energy	Other specifications
PV panels 1	Polycrystalline 275Wx12	3.3 kW	tilted 60°
Controller 1	TriStar TS-MPPT-60		Max 60A charge
PV panels 2	Polycrystalline 200Wx6	1.2 kW	tilted 30°
Controller 2	Victron MPPT 150/35		max 35A charge
Wind turbine	Tuule C200	3 kW	5 m diameter, 12 m height
Controller 3	Finnwind custom made	1.6 kW	
Charger Inverter	Victron Quattro 48/10,000	10 kW	Max 140A charge current
Control Unit	Victron CCGX		
Diesel generator	Cummins C11 D5	11 kW	230VAC 1-phase, diesel
Battery storage	Powerxon 2V AGM 1250Ah	60 kWh	About 40 kWh useable
Air source heat pump[a]	Mitsubishi FH25	2.2 kW [b]	Max 6.3 kW heat, 3.2 kW at −15 °C, used until Sep. 2022
Air source heat pump[a]	Panasonic HZ25XKE	2.0 kW [b]	Max 7.5 kW heat, 3.6 kW at −25 °C
Biomeiler[a]	50m^3 compost heater	0–2 kW	Used 2019–2021
Water tank[a]	500l hot water storage		
Water heater element		6 kW	Used when batteries are full
Convector[a]	Cooper & Hunter	4.6 kW	Connected to both the biomeiler and the water tank
Excess heaters	Space heaters	5.1 kW	Used when batteries are full

To forecast the required data for calculation and feeding the optimizer with the matching datatype inputs, a whole process of interfacing between Meteoria devices and the outside world is provided by Novia MQTT broker through which I have established all the development work operations in which all the data and commands will be collected and executed.

The used topics subscribed to for data harnessing are listed in Table 2:

Table 2: List of the Meteoria IoT broker topics.

MQTT topic	Measure
open/meteoria/airTemp	Outside air temperature
open/meteoria/airRH	Outside air relative humidity
open/meteoria/airPressure	Outside air pressure
open/meteoria/solarRadiation	Outside solar radiation
open/meteoria/solarRadiationAvg	Average solar radiation
open/meteoria/solarAltitude	Sun altitude
open/meteoria/solarAzimuth	Sun azimuth
open/meteoria/windSpeed	Outside wind speed
open/meteoria/windSpeedAvg	Average wind speed
open/meteoria/windDirection	Wind direction
open/meteoria/energy/batteryVoltage	BSS voltage
open/meteoria/energy/batteryCurrent	BSS current
open/meteoria/energy/batteryCapacity	BSS SoC
open/meteoria/energy/gensetPower	DG Power
open/meteoria/energy/windTurbinePower	Wind turbine power
open/meteoria/energy/solarChargePower	Solar power

The energy optimization will be carried out using the PuLP modeling script which is a well-known framework developed to model and solve MILP cases. Although the problem is not linear, some assumption work has been applied to solve the problem in a less time-consuming way and demonstrate it on the Meteoria site.

Following up on pulling and pushing data, a visualization part of this work will be accomplished by developing a real-time dashboard for all the parameters and decision variables of the optimizer as well as the data harnessed and pre-processed internally. In this part, I will take advantage of Flask as a visualization framework together with the Google Cloud Platform (GCP) that can serve as a host to our Realtime visualization application. The choice of GCP is based on the well-maintained platform and the variety of data analytics that can be associated with the application through the power of Google computing systems as well as the support in case of a problem.

4. Methodology

4.1 Proposed architecture

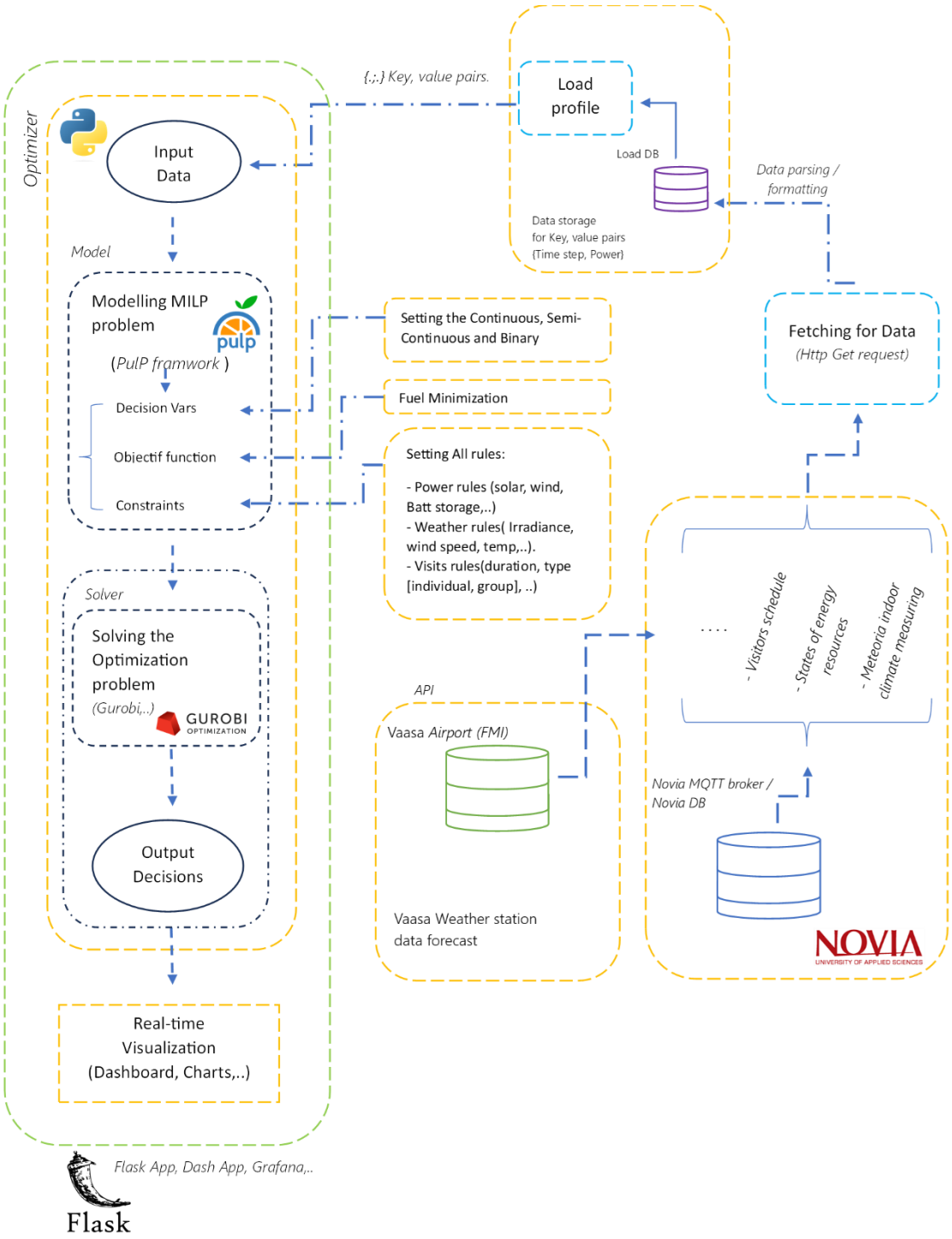

Figure 3: Architecture of the proposed solution

4.2 Constructing load profile

Based on the Meteoria site and the energetic appliances needed to keep the facility running as expected we should provide and guarantee the energy supply for those appliances and equipment at any given time. The goal of the Meteoria site is essentially agreed to be an exhibition for the public. This requires that visitors should have an excellent visit experience and that implies Comfortable environment conditions by providing heating or cooling to the site to get the suitable temperature inside the buildings (Main exhibition and the meteorite barn). With that being said, the heat pumps are the main consumer of energy in the Meteoria site and need to be supplied with energy as needed. However, this energy supply to heat pumps must be rationalized, in other words, it must be governed via other external requirements, and these are typically the weather conditions, scheduled visits, and the availability of energy supply from NCES and CES existing within the Meteoria MG.

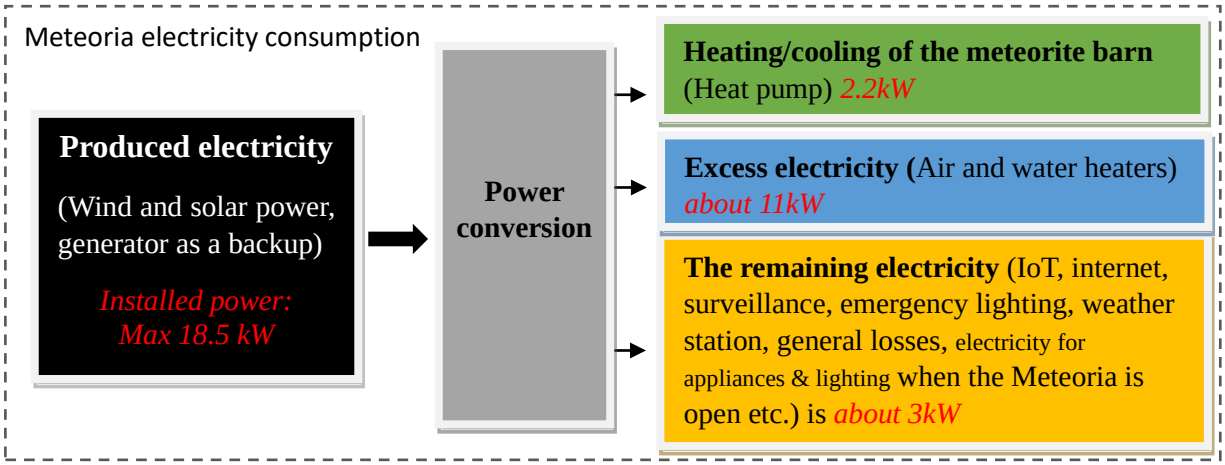

Figure 4: Power flow diagram at Meteoria

4.3 Constructing entry points

Entry points design all the data dictionaries needed for the optimizer as inputs to handle the computation and solve the energy optimization problem. These are typically:

The solar energy generated by PV modules tilted by 30 degrees. The solar energy generated by PV modules tilted by 60 degrees. Wind energy comes from the wind turbine. The energy demand from the Meteoria heat pumps.

To determine the solar energy available on site, we will need to have the measurement of available solar Irradiation which indeed through the Meteoria IoT sensing equipment we can access. Therefore, we can calculate the available solar energy on the solar modules tilted by 30 degrees using the formula:

$$P_{PV30} = G \,.\, S_{pv30} \,.\, \eta \qquad (1)$$

where

P_{PV30} : Power being generated power by solar modules.

S_{pv30} : Total surface of 30-degree tilted modules.

S_{pv30} : Total surface of 30-degree tilted modules.

G : On-site Solar Irradiance **(**measured by the on-site weather station and pulled out from the 'open/meteoria/solarRadiation' Meteoria MQTT topic)

η : The Solar PV thermal conversion rate which is usually in average 16%.

Then, the same calculation is applied to the P_{pv60} with respect to the total surface of available solar modules.

For wind energy, the Meteoria MG has a new wind turbine rated 5 kw and delivers up to 6.5 kw as maximum power at 25 m/s wind speed. The general known calculation formula for the output power can be drawn as follows:

$$P_{wind} = \frac{1}{2} \cdot {V^3}_{wind} \cdot C_p \cdot S_{turbine} \cdot \rho \qquad (2)$$

Where:

P_{wind} : The electric power generated by the wind turbine.

$V^3{}_{wind}$: The wind velocity in m/s (measured by the in-site weather station and

pulled out from the open/meteoria/windSpeed Meteoria MQTT topic)

C_p : The performance coefficient.

$S_{turbine}$: The surface covered by the turbine blades in m^2.

ρ : The Air density in kg/m^3.

Note that the C_p of Meteoria wind turbine is unknown, in theory, it is restricted by the Beatz limit, which is 59.3, this means that the maximal electric power we can generate out of the wind energy is 59.3 % of the available wind energy. In practice, the C_p ranges between 35% and 45%, therefore I will consider 0.35 as the worst scenario case.

Moving to P_to_hp, is the power that represents the heating demand to heat the Meteoria Meteorite Barn which is open to visitors. It is the main consumer in Meteoria MG. The main heating system is composed of a reversible heating/cooling pump which can both heat during wintertime and cool down during summer accordingly. It is designed to effectively during harsh cold winters up to -21 °C. To optimize the use of the heat pump, we need to calculate the demand for heating power that will be allocated to the heating system as a constraint when solving the problem. In theory, we achieve this by realizing the following formula:

$$P_{to_{hp}} = \frac{Q_H}{COP_{HP}} \quad (3)$$

where

$$COP_{HP} = \frac{1}{1 - \frac{T_L}{T_H}} \quad (4)$$

where

$P_{to_{hp}}$: The power needed for the heating system.

Q_H : The heating loss rate in KJ/h or W.

COP_{HP} : The Coefficient of Performance of the heating pump.

T_L : The outside temperature in °C (low temperature, measured by the in-site weather station and pulled out from the 'open/meteoria/airTemp Meteoria' MQTT topic).

T_H : The inside Meteorite barn temperature in °C (temperature to be maintained).

Due to difficult to determine the appropriate heat loss rate for the Meteorite barn since it depends on many factors such as the construction elements' U value (roof, windows, walls,.), the building materials used, and Thickness. The building is quite brand new or at least recently renovated, I found that 17% will be a reasonable estimation to go with to calculate the heating loss rate, meaning that the barn will lose around 17 kJ/h for every 100 kJ supplied heat.

Therefore, the heat loss rate could be estimated using the formula:

$$Q_H = U \cdot A \cdot \Delta_T \quad (5)$$

where

U : The total estimated U-value of the Meteorite barn.

A : The total area in m^2.

Δ_T : The difference in temperature (inside/outside),

Thus:

$$Q_H = 1.15 \cdot 35 \cdot (21 - (-20) Q_H = 1650 \cdot 25\, W$$

For example if the outside temperature T_L is -20 °C, the $P_{to_{hp}}$ will be estimated as :

$$P_{to_{hp}} = \frac{Q_H}{COP_{HP}} = \frac{1650.25}{\frac{1}{1 - \frac{-20}{21}}} = 3221,91\, W$$

5. Modeling the optimizer

5.1 Objective function

The goal of the entire solution is to save energy (e.i minimizing the consumption from CES) which in Meteoria is mainly the diesel engine therefore minimizing the steady-state fuel consumption of the diesel engine as well as the starting-up consumption when passing from off mode to on mode. In Addition, we aim to rationalize the usage of the available resources from non-conventional resources (solar and wind), which means we want to maximize the usage of energy coming from those resources.

This can be formulated in the objective function as follows:

$$\sum_{j=1}^{1}\sum_{k=1}^{n} FC_{k,j} \cdot \frac{\Delta t}{1000} + \sum_{j=1}^{1} K_j^{start} \sum_{k=2}^{n} Z_{k,j} \qquad (6)$$

where

FC : The fuel consumption of the diesel engine. (Linearized model).

Δt : The duration of consumption of the diesel engine.

K_j^{start} : The additional cost of starting the engine.

$Z_{k,j}$: Linearizing variable.

j : Number of genset used (one genset is used at Meteoria : j=1)

k : The number of time steps.

5.2 Parameters

Battery parameters

Q_{max} : Maximal energy charge stored in the battery.

Q_{init} : Initial energy charge stored in the battery.

Q_{fin} : Final charge stored in the battery.

eff_{to-bat} : Efficiency when charging the battery.

$eff_{from-bat}$: Efficiency when releasing from the battery.

Fuel consumption parameters

FC_{offset} : Fuel oil consumption offset.

Max_{fc} : Maximal fuel consumption.

Discretization parameters

Δt : Time step in discretization.

t_{max} : Max time in the simulation.

n : Number of time steps in the simulation.

t : Time vector in simulation (dimension(t)=n).

L : Vector of load profile (same length as time vector).

Site parameters

Irr_{sol} : Solar irradiance state (W/m^2).

V_{wind} : Wind speed (m/s).

T_L : Outside temperature (°C) (Atmosphere temperature).

T_H : Inside temperature (°C) (main exhibition temperature).

T_{barn} : Inside Meteoria barn temperature(°C).

Equipment parameters

Q_h : Estimated heat loss when temperature drops. 1 kJ/h = 0.0002777778 kW

R : Wind turbine radius (m).

C_p : Coefficient of performance.

ρ : (Rho) Air density (kg/m^3).

PV_{s_30} : Surface of the installed PV modules tilted by 30°.

PV_{s_60} : Surface of the installed PV modules tilted by 60°.

Power flow parameters

P_{PV-max} : Maximal power generated by Solar PV modules (for maximal irradiance (kW).

P_{PV-min} : Minimal power generated by Solar PV modules (for minimal irradiance (kW).

$P_{wind-max}$: Maximal power generated by the wind turbine (cut-out wind speed (kW).

$P_{wind-min}$: Minimal power generated by the wind turbine (cut-in wind speed (kW).

P_{hp-max} : Maximal power transferred to the main exhibition's heat pumps (kW).

P_{hp-min} : Minimal power transferred to the main exhibition's heat pumps (kW).

P_{sh-max} : Maximal power transferred to the space heater (kW).

P_{sh-min} : Minimal power transferred to the space heater (kW).

$P_{bio-max}$: Maximal power generated by biomeiler compost heater (kW).

$P_{bio-min}$: Minimal power generated by biomeiler compost heater (kW).

P_{wh-max} : Maximal power transferred to water heater (kW).

P_{wh-min} : Minimal power transferred to the water heater (kW).

$P_{stnb-max}$: Maximal standby power to keep essential equipment on standby mode (kW).

$P_{stnb-min}$: Minimal standby power to keep essential equipment on standby mode (kW).

P_{max} : Maximal power that can be generated by the genset (kW).

P_{min} : Minimal power that can be generated by the genset (kW).

5.3 Lp variables

Q^{bat} : Batteries state of charge. (kWh)

P : Power generated by the genset. (kW)

P^{load} : Power from genset to load. (kW)

P^{To-BSS} : Power from genset to batteries. (kW)

$P^{From-BSS}$: Power from the battery to the load. (kW)

P^{PV30} : Power from solar PV modules tilted by 30° (kW).

P^{PV60} : Power from solar PV modules tilted by 60° (kW).

P^{wind} : Power from wind turbine (kW).

P^{hp} : Power to heat pumps (kW).

P^{sh} : Power to space heater (excess heat (kW).

P^{bio} : Power from biomeiler compost heater (kW).

P^{wt} : Power to water tank (kW).

P^{vis} : Binary variable about booked group visits (kW).

P^{stnb} : Standby power (kW).

Y^{DG} : Binary state of genset.

Y^{To-BSS} : Binary charging variable for the BSS.

$Y^{From-BSS}$: Binary discharging variable for the BSS.

Y^{sh} : Binary state of a space heater.

Y^{hp} : Binary state of heat pumps.

Y^{bio} : Binary state of biomeiler.

Z : Additional cost for fuel oil consumption when starting Genset.

FOC : Genset fuel oil consumption. (g).

5.4 Lower and upper bounds.

BSS state of charge:

$$0.2.Q_{max} \leq Q_{BSS} \leq Q_{max}$$

Power generated by the genset:

$$P_{min} \leq P_k \leq 0.9.P_{max}$$

Power from genset to load:

$$P_{min} \leq P_{load} \leq 0.9.P_{max}$$

Power from genset to batteries:

$$P_{min} \leq P^{To-BSS} \leq 0.9.P_{max}$$

Power from BSS to load:

$$P_{min} \leq P_{From-BSS} \leq 0.9.P_{max}$$

Binary variable indicating the state of genset:

$$0 \leq Y_{DG} \leq 1$$

Binary charging variable for BSS:

$$0 \leq Y^{To-BSS} \leq 1$$

Power from BSS to the load:

$$0 \leq P^{From-BSS} \leq 1$$

Power from the battery to the load:

$$0 \leq Z \leq 1$$

Power from solar PV modules tilted by 30°:

$$P_{PV30-min} \leq P^{PV30} \leq P_{PV30-max}$$

Power from solar PV modules tilted by 60°:

$$P_{PV60-min} \leq P^{PV60} \leq P_{PV60-max}$$

Power from wind turbine:

$$P_{wind-min} \leq P^{wind} \leq P_{wind-max}$$

Power to heat pumps:

$$P_{hp-min} \leq P^{hp} \leq P_{hp-max}$$

Standby power:

$$0 \leq P_k^{Stnb} \leq 3\ kW$$

Power to water tank heater:

$$P_{wt-min} \leq P^{wt} \leq P_{wt-max}$$

Power to space heater:

$$P_{sh-min} \leq P^{sh} \leq P_{sh-max}$$

Power from biomeiler heater:

$$P_{bio-min} \leq P^{bio} \leq P_{bio-max}$$

Booked group visits:

$$0 \leq B^{bis} \leq 1$$

Genset fuel oil consumption:

$$0 \leq FOC \leq Max_{fc}$$

5.5 Optimization constraints

Power requirements:

$$L_k = \sum_{j=1}^{1}\sum_{k=0}^{n} P_k^{load} + \eta^{From_{BSS}}.P_k^{From_{BSS}} + P_k^{PV30} + P_k^{PV60} + P_k^{wind} \quad (7)$$

Power split in time step k from the DG:

$$P_k = \sum_{j=1}^{1}\sum_{k=0}^{n} P_k^{load} + P_k^{To\_BSS} + P_k^{Stnb} + P_k^{wh} \quad (8)$$

$$P_k^{load} = \sum_{j=1}^{1}\sum_{k=0}^{n} P_k^{To-hp} + P_k^{To-sh} \quad (9)$$

BSS charge balance:

Initial charge:

$$Q_0 = Q_{init}$$

Charge balance in steady state:

$$Q_k = Q_{k-1} + \eta^{To\_BSS}.\Delta t \sum_{k=0}^{k=10}\sum_{j=1}^{1} P_k^{to-BSS} - P_k^{From_{BSS}} \quad (10)$$

Final Charge balance:

$$Q_0 = Q_{Final}$$

Standby power:

$$P_k^{Stnb} = L_k - \sum_{j=1}^{1} P_k^{To-hp} + P_k^{To-sh} \quad (11)$$

Heat pump and biomeiler balance:

$$y_k^{heater} + y_k^{hp} \leq 1 \tag{12}$$

$$P_k^{hp} \leq Q_h * 1 - \frac{T_L}{T_H} * y_k^{hp} \tag{13}$$

$$P_k^{bio} \leq P_{bio-max} * y_k^{bio} \tag{14}$$

Excess heaters balance (water thank & space heater) :

$$Y_k^{sh} + P_k^{wh} \leq 1 \qquad \forall \; Q_k = Q_{max} \tag{15}$$

$$P_k^{wh} \leq P_{wh-max} * Y_{wh} \qquad \forall \; Q_k = Q_{max} \tag{16}$$

$$P_k^{sh} \leq P_{sh-max} * Y_{sh} \qquad \forall \; Q_k = Q_{max} \tag{17}$$

5.6 Implementation using PuLP

This chapter will describe the structure of the optimizer using PulP framework, to achieve the end goal of minimizing fuel consumption by the existing genset in Meteoria, which consequently will result in maximizing the use of RES when they are available on-site according to the forecasted data.

5.6.1 Introducing problem parameters

As the problem is an optimization problem, we should define all the parameters that are related and implicated in the optimization process. Those are mostly the constant values that define the references and or the power characteristics of each component. For instance, the genset has a rated power of 3Kva, therefore this information represents one parameter from the genset, of course, it could be more than one parameter for every component depending on the complexity of the problem we need to solve. In our case, there are many parameters we need to introduce. And they are all described in the code attached to the appendix.

5.6.2 Setting-up Decision Variables

When designing a layout of an optimization problem, it's very crucial to understand what physical properties are needed to accomplish the optimization process. This always involves the decision variable which is the variable that will contain the values of these physical properties that will be computed by the optimizer (precisely by the solver) and integrated into the flow of the optimization outputs. In our case of energy optimization in Meteoria of Soderfjarden site, the decision variables are described in section 5.3. and coded in the source code in the appendix.

There are mainly three types of decision variables considered in this problem:

- Continuous Integers (LpContinous),
- Binary (LpBinary)
- Integers (LpIntegers)

These variables together with the number of constraints will define the size of the problem to solve.

5.6.3 Upper and lower bounds

The upper and lower bounds are similarly one part of solving a MILP optimization problem. They represent the limits of each Linear programming variable (decision variables) that will not exceed in the physical components, these are very important as they help in the relaxation of the optimization problem and therefore find the optimal outputs easily without consuming much computation resources or turning a non-feasible problem to a feasible one.

5.6.4 Setting up the objective function

The objective function is the main consideration when the optimizer takes part in the solving process. There are typically two types of optimizations, either maximization or minimization. When well defined the solver in use will try to reach the target. In our case, it is a minimization optimization, which means the solver will minimize the use of fuel against the use of RES, mainly wind, solar, and the stored energy in batteries.

5.6.5 Setting-up problem constraints

Setting the problem constraints represents designing the interaction between all the decision variables under the optimization layout. Typically, all linear programming variables are defined previously in section 5.3. This process is the most time-consuming in the optimization process as it takes several iterations and switching considerations to which the solver relies. This will directly impact the relaxation process and even the optimizer outputs in a feasible problem. The designed constraints of our case are described in the mathematical form in section 5.5. Lp Constraints are implemented in the optimizer code available in the appendix.

The final adopted layer of the constraints in the client code could be different from what has been described in the 5.5 section as per the iteration's reasons and fine-tuning the optimizer.

6. Meteoria MQTT interface

Novia IoT platform has access to and controls all data information in Meteoria of Soderfjarden. This includes all devices connected to the same network. The main controllers are Raspberry Pi boards which are designed to control every section of the site. The site has 4 Rpi distributed to collect data or send upcoming commands from other devices. Below is the site configuration of how data communication is handled.

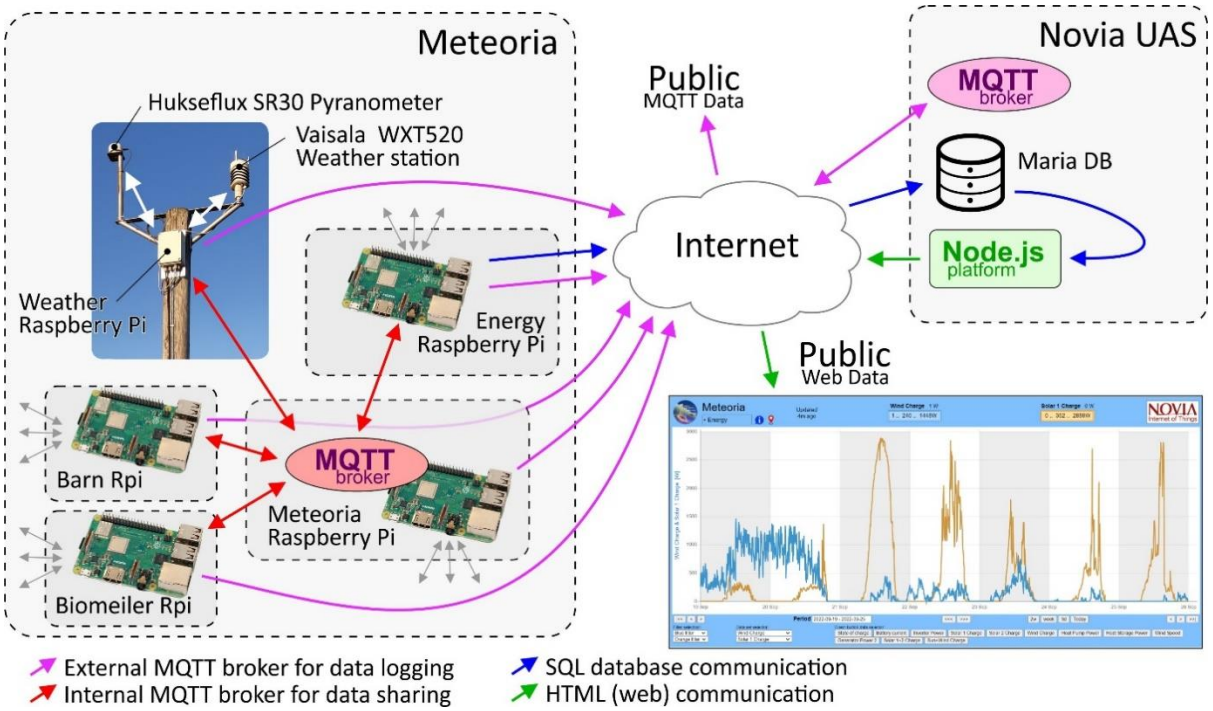

Figure 5: Data flow in Meteor of Sodenfiarden as in [24]

The Meteoria Rpi is the brain of Meteoria to and from which the measurements and control commands are issued. Energy Pi is attended for energy consumption measurement, Barn Rpi to communicate with different devices in the Barn and Biomeiler Rpi is assigned to communicate with Biomeiler (not operational for the time being).

The Meteoria site has one MQTT broker that is responsible for subscribing to and or publishing data to the internet, then Novia MQTT broker can access this data and publish it publicly via a designated web page for monitoring the different set measurements.

The charts from this web page give an overview of how the MG is behaving in general but still missing the process of optimizing the RES which we aim to achieve via this thesis work.

The optimizer uses value-pairs (e.g, python dictionary) data format and is different from what the MQTT broker is expecting, and the optimizer does not use the same type of data in his computation to solve the optimization problem. This is why a thorough work of data acquisition (DAQ) is needed to achieve the well synchronization back and forth.

6.1 MQTT client development

The solution needs typically two separate clients: One for subscribing to topics to fetch data and a second one that will be assigned for the publishing part to publish the Optimizer outputs that will be used by IoT devices in Meteoria. However, to minimize the resources and the dependencies I decided to develop one client that will do both operations at the same time.

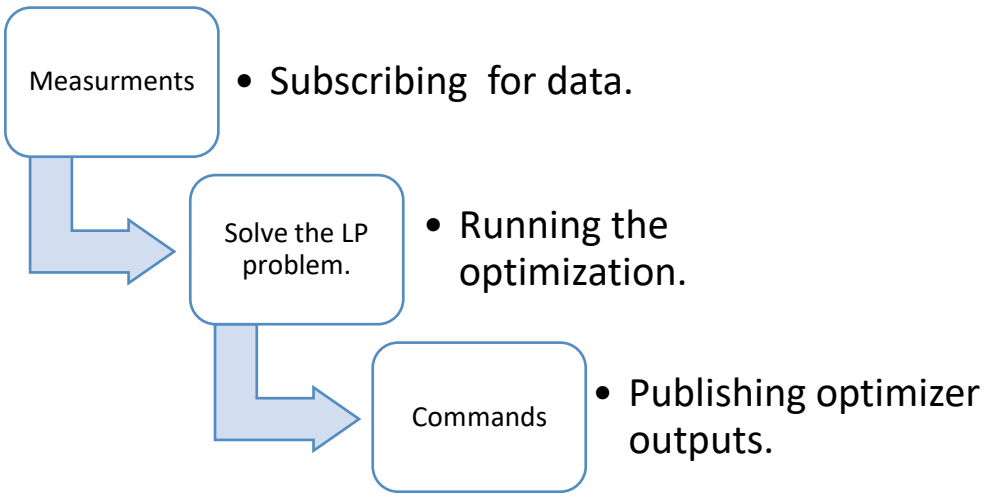

Figure 6: MQTT client structure

6.2 Data storage

The data which is been logged should be stored somewhere accessible to the optimizer to use this data for the computation process. Since the MQTT protocol is synched to subscribe/publish one packet at a time, I decided to put the data in a set of value-pairs dictionary with a size of the time steps needed for the optimizer to run the optimization, e.g: if the optimization is decided to run each 10 seconds, then the dictionary should be of a size of 10. Once the dictionary is fed with all value-pairs then the optimization can be run. The dictionary is then appended to a CSV file which is located within the same repository of the optimizer.

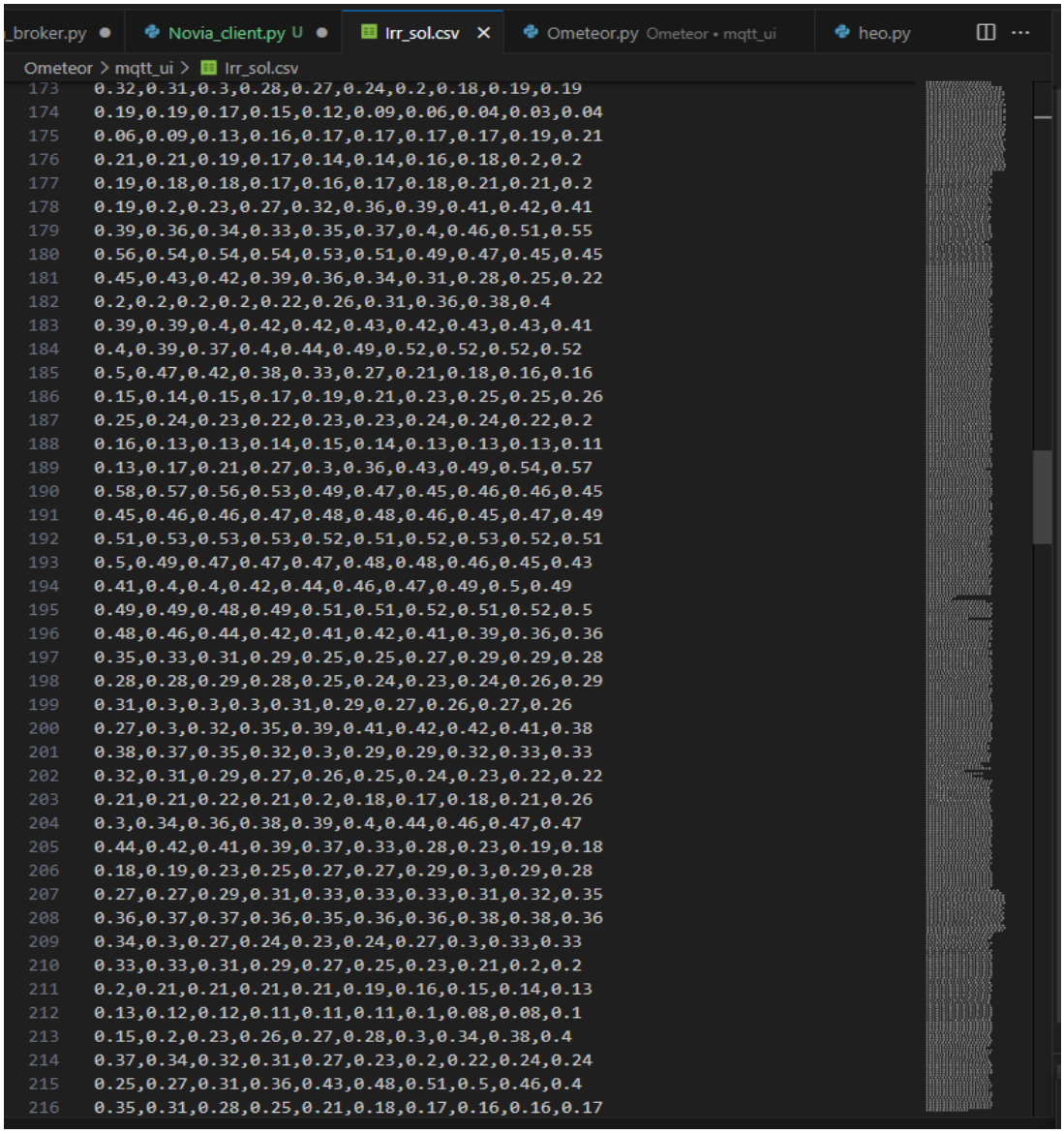

Figure 7: Snippet from data storage CSV file for solar irradiance.

6.3 Data acquisition development

The process of measuring electrical or physical phenomena, such as voltage, current, temperature, pressure, or sound, using a computer or a data logging device is known as data acquisition, or DAQ for short. The instrument used to make this measurement is known as a data acquisition system, and it usually consists of a signal converter (ADC/DAC), which transforms the analog signals from the sensors into digital values that a computer can process a set of sensors or transducers, and signal conditioning circuitry. Scientific research, design & development, manufacturing, and process control are just a few of the many uses for data-gathering systems.

Besides the fact that the DAQ systems usually are composed of different parts: Hardware and software, in this work I focus essentially on the software part which itself can have different software routines. This component is responsible for gathering digital data from the IoT devices of ADC-like systems. A control agent that can manage the synchronization, triggering, recording, data storage, and even pre-processing are some of the principles that are frequently involved. Usually, a computer or server is connected to and interfaced with by the data recording devices to provide the data for further visualization, storage, and post-processing. DAQ systems, however, can also function independently and standalone without requiring a constant PC connection. The DAQ development in this work will focus mainly on developing the codes that interface the raw data from the different Meteoria IoT platform topics which forecast the needed measurements like air temperature, solar irradiance and wind velocity.

These codes will help me interpret the data forecast and convert it to the right data types for the optimizer.

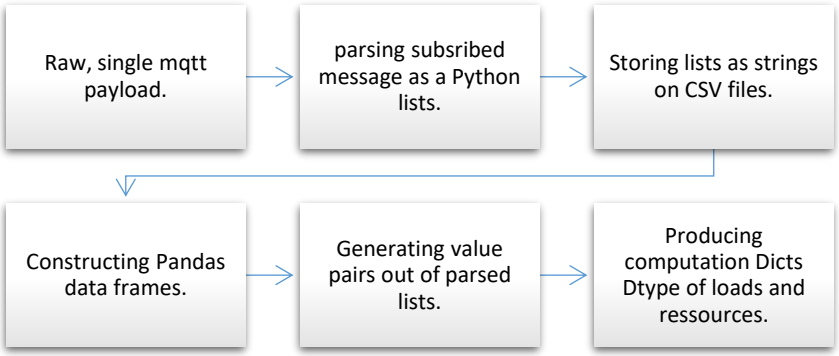

Figure 8: DAQ system for interfacing Meteoria measurements.

6.4 Running the full client

Running the full client will feed the Optimizer with the required inputs to compute the needed LPvariables, when accessing the values stored in these variables inside the 'Optim' optimization variable set a global variable to be accessed outside the Pulp Problem to generate control commands for the Genset to be turned-on or turned-off for the next 10s ahead of time.

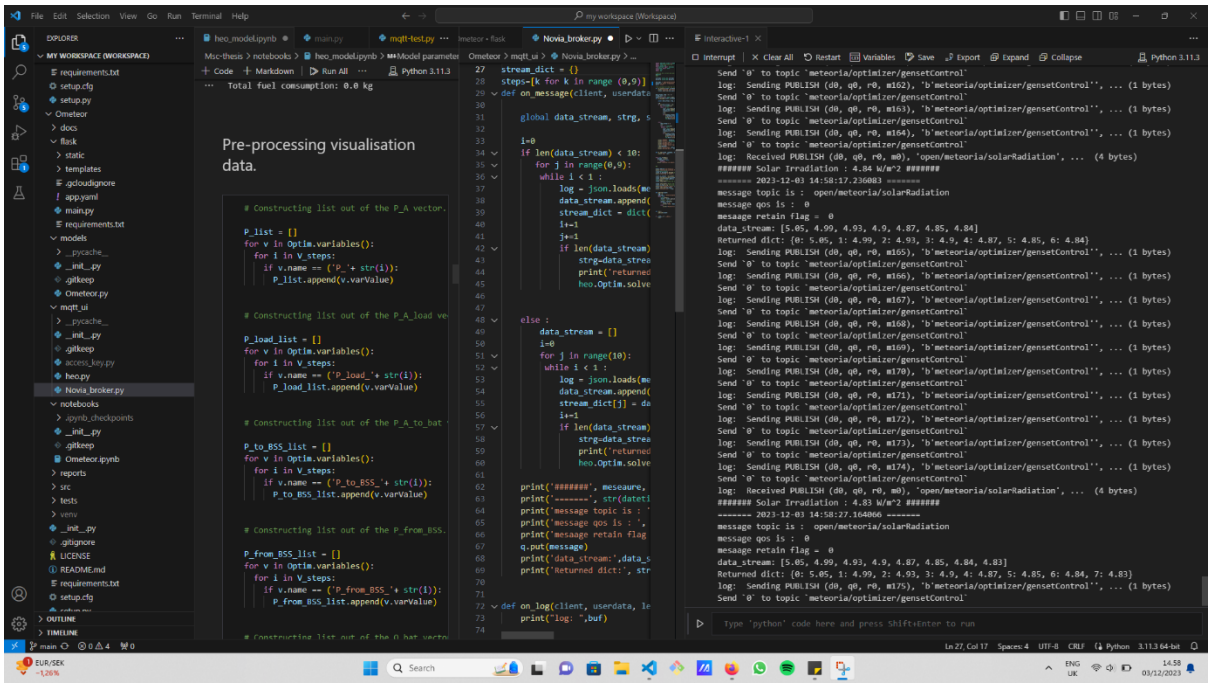

Figure 9: Running the Ometoria MQTT client

6.5 Success to publish on Novia broker

While running the full client, to see if the command is sent physically to the broker, I used the MQTT Explorer tool to manage and supervise the publishing process for the Genset control commands and send the commands through the topic 'meteoria/optimizer/gensetControl'. The system's battery capacity has been also considered, while observing the SoC decrease, the optimizer was always triggering the Genset to get the charge in the battery within the 90% even if there is considerable RES power available because the BESS is always the priority to fulfill before considering power resources constraints.

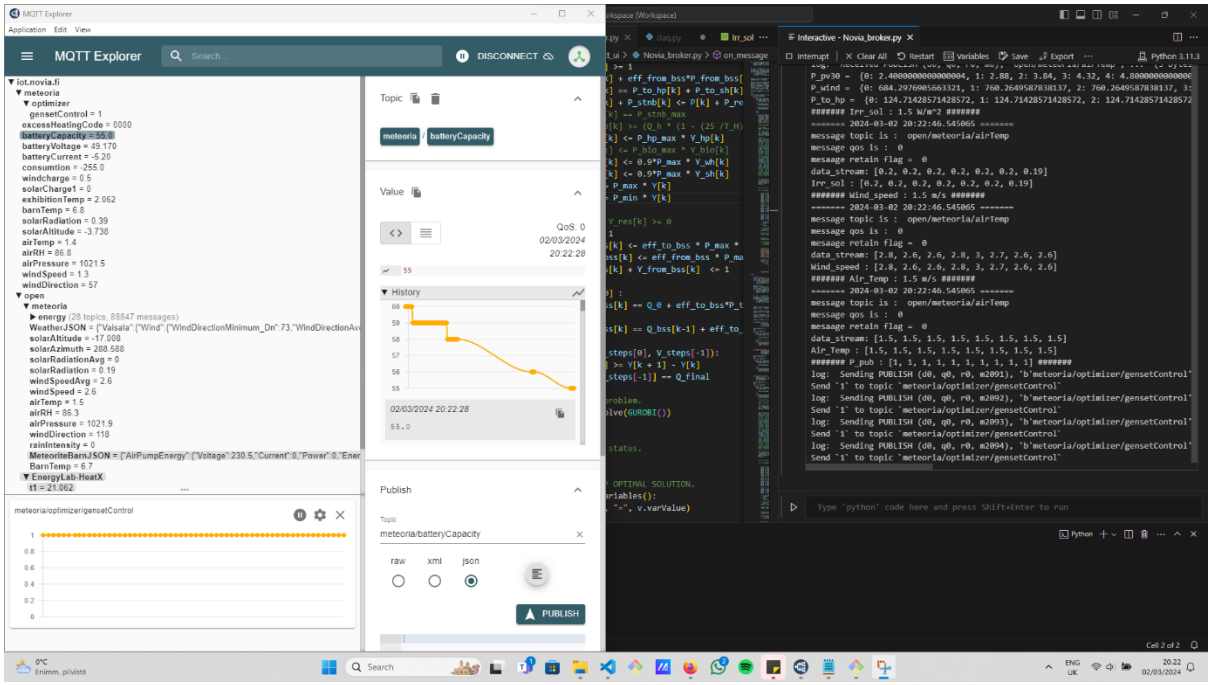

Figure 10: Optimizer sending '1' commands while SoC is decreasing.

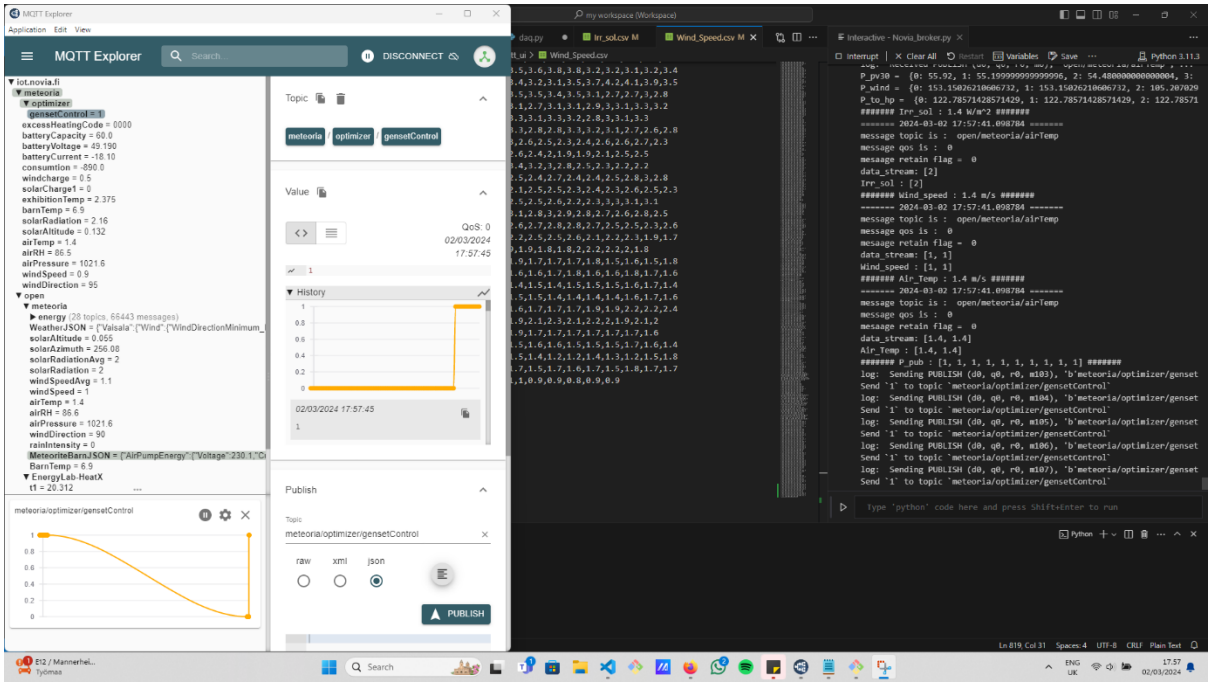

Figure 11: State transition from '0' command to '1'

7. Data visualization

7.1 Flask app

Using Flask as the base framework for visualizing the data and the optimization results is based on its reliability and maintainability which make it well documented over time.

In addition, Flask is a good choice if Novia University wants to deploy this solution in its host domain. To build the App, I built 2 HTML files that define the front end of the graphs we see when typing the domain/'name of the route'.

Two routes have been built: /L_P which will present the Load vs Genset power function of time in real-time, and the second is /*Power_Flow*, which will be the overview of the power generation and consumption in real-time, it will emphasize the following metrics in one chart:

- Load Requirements
- Power Generated by the Genset
- Power transferred from the Genset to the load.
- Power transferred from the BSS to the load.
- Power from RES.
-

Inside the Flask app I adopted Plotly as the main plotting library for its well vivid charts and ways of visualization compared to traditional libraries like Matplotlib. Moreover, Plotly is more web-oriented, unlike Matplotlib which is mostly used in static Notebooks to visualize the data.

When running the client, the programmed dashboard will be live on the address: '127.0.0.1' using port 8080.

Since we have two applications programmed natively, which I named in the code *L_P* (for the power generated and the actual power demand) and the second flask application named *Power_Flow* (Power flow and Generation/Consumption), this will require targeting which information I want to show. Therefore, the address will be always followed by the name of a typical application in the format 127.0.0.1:8080/'name of the application'.

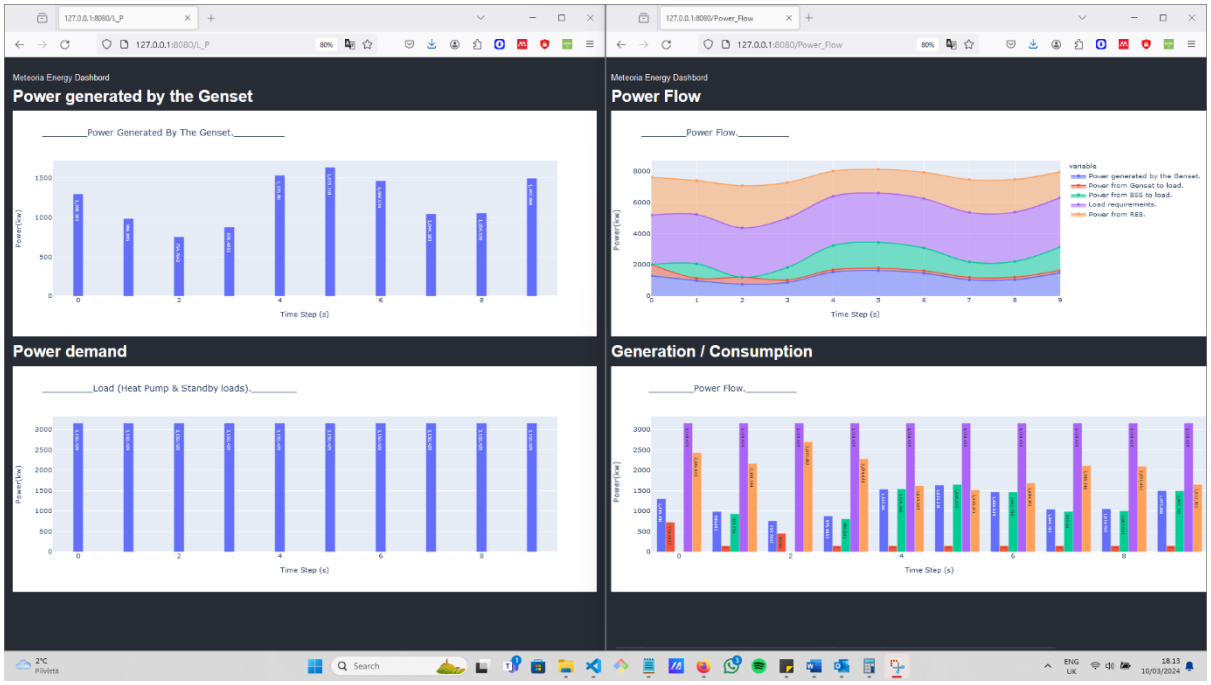

Figure 12: Metoeria Energy Optimizer at 18:13, 10.03.2024

In figure 12, I am presenting the two applications at the same time on reel time which results on the shown dashboard. The plots present the actual energy balance of the Meteorian Soderfjarden on March 10, 2024, at 18h13 Helsinki time zone. We can see that at that time, enough RES was being produced based on the site power capacity. The excess of the energy produced is stored in the battery system.

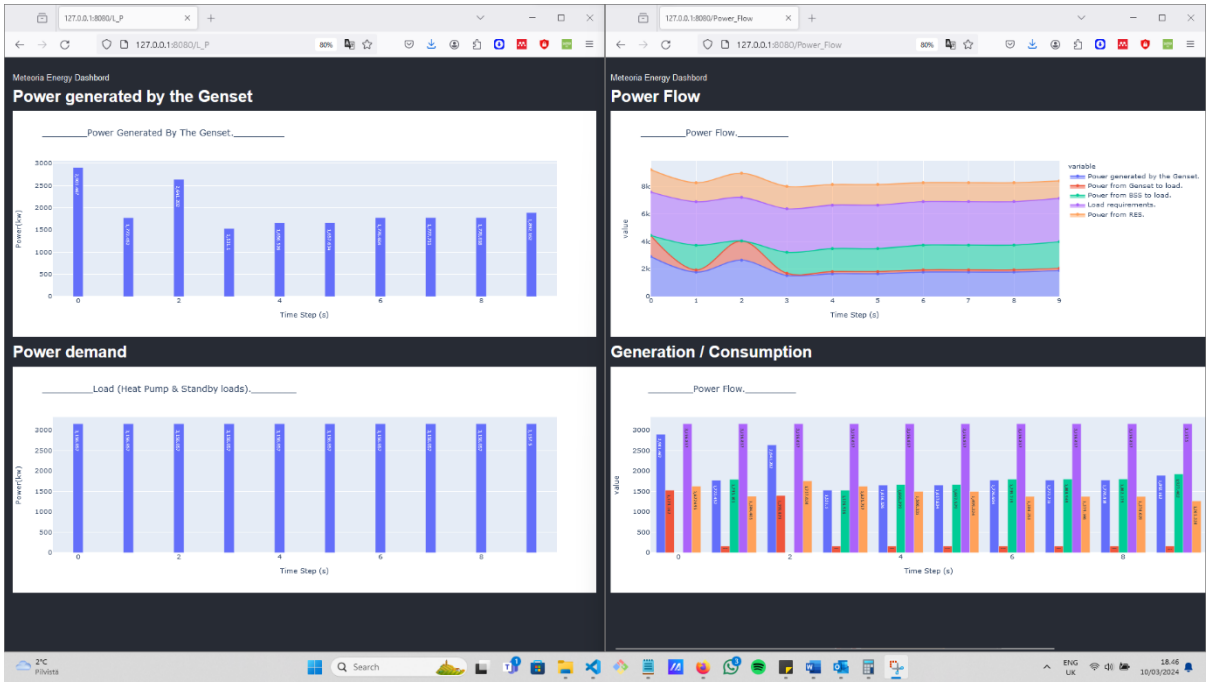

Figure 13: Flask App: Meteoria Energy Optimizer at 18:46, 10.03.2024

In Figure 13, we see the energy balance has been changed from the time of 18h13 shown on the balance state in Figure 12 previously. It's clear that by starting to be dark outside, the solar potential has diminished (but not that much), and the surplus is always being shifted to the storage while we see the load is almost constant which implies keeping the Genset running.

In other time further, exactly on May 04th, Figure 14 shows how the optimizer is operating and managing the energy balance, we see that there is a decent amount of RES (the sun at least is shining finally in spring) so this implies that the micro-grid will rely mainly at that time on green resources while keeping the genset off.

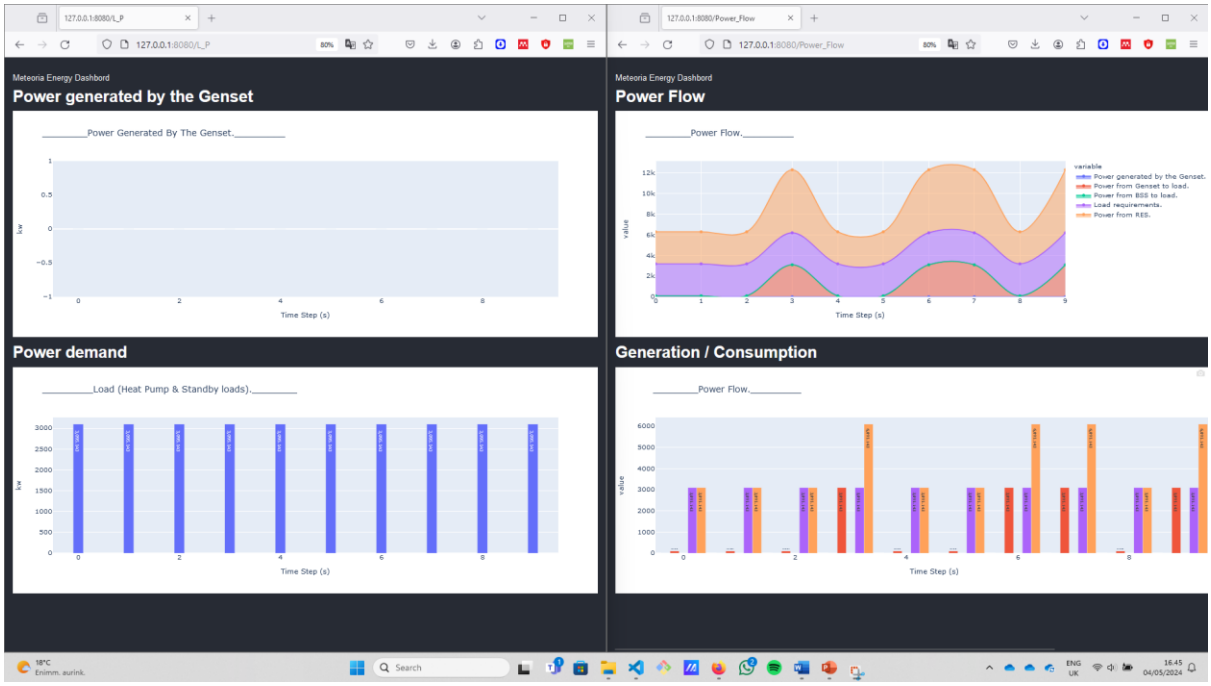

Figure 14: Flask App: Meteoria Energy Optimizer at 16:46, 04.05.2024

8. Results and Discussion

At the end of this thesis, the delivered solution worked very well and satisfied the objectives of this thesis work. However, in the world of high tech and automation there are big challenges when it comes to industrialization and solution-to-market deployment is not as easy as it is in the phase of research and development, they might even be a looping back after testing. To research and development phase to fine-tune further or change some ideas that aligned better with the industry but not necessarily with the academia.

During this thesis work, which included somehow a decent part of testing, and practical work on the software side and automation, there have been many challenges to overcome and times where the perfect solution is not found but needs more time and research to bring it to light.

Things to improve:

- Code maintenance.
- Code reliability.
- Error and exception handling.
- Optimizer status handling:

  Model Feasibility/ Infeasibility and solving parameters (Gurobi). Suggestion to add an extra layer of heuristics in case of infeasibility to estimate a solution near to optimum without flagging infeasibility and causing the crush of the client. Eg. In some cases, the problem cannot be feasible, but a heuristic solution was found.

- DAQ Data frames consistency (NaN values, Na value, empty values) > pandas data frames.
-
- What are the best practices to guarantee data being pre-processed reliably without causing overloadability of the client and keep the computation light which will help in the continuity of service of the client.
- Flask App (Visualization)

  Flask was chosen in this thesis work for visualization because of its lightweight framework and well-documented procedures. However, there are many other alternatives to do the same application or with even more features (eg. Dynamic updates of reel time values without actualization) but they might need to be licensed.

9. Future work

Further to this thesis, which has revealed the possibility of optimizing a real-life large-scale HMG and provided one approach from many other approaches to the same thing, I would be very interested to see future works that use metaheuristics techniques instead of the deterministic approaches in the same topic of energy optimization controlled via an IoT system.

As a follow-up work and complementary task to this thesis, I would like to implement the optimization MQTT client ahead of time for a scheduled optimization for a period ahead for example: 3 days ahead, 5 days, or 1 week. This is achievable through a simple integration of a weather API through which the client will forecast weather data in the region of Sundom in Vaasa and use the data to calculate the entry points being fed to the optimizer.

AI has gained popularity with its powerful ability to improve things in this area, however, AI itself is still in its core relies on fundamental mathematical and statistical models. I have been working on the AI-based energy optimization topic in the past and I know that this would be very likely to be happening in the next decades. But at least some future research work guided by academia on that topic will be of great benefit and would be a complementary part to this thesis work so that a benchmarking would take place and we could understand what the benefits of AI are in optimizing off-grid power systems over the deterministic approaches without compromising safety and service continuity and promoting green resources harvesting.

Furthermore, it would be very appreciated if future work would be grounded on developing an optimizer based on an IoT client completely hosted on the cloud and able to manage HMGs; as a cloud-based system are starting to replace local and centralized systems, nowadays edge computing is highly associated with the future IoT systems that need reliability, low latency and high coverage for edge devices which are all characteristic of an IoT connected HMGs.

Another aspect is cybersecurity; IoT devices use protocols to send and receive data and act in real-time bases, the few last years as IoT systems started to construct the image of the future this subject became one of the priorities of a successful solution. On the other hand, self-optimized HMGs will be required to be operating IIoT to have a robust scanning system to prevent any attacking damages and foresee the vulnerabilities before even to be known; therefore, embedding an optimizer within and IIoT based HMG will be also a good topic to explore as it will help to have better understanding of the behavior of IoT devices in an industrial environment.

Furthermore, one of the building blocks of the IoT based HMGs is 'Data'; hence, these systems use data inputs for every decision action and release other datatypes as outputs which means this data need to be stored and managed appropriately; in this thesis work I have been relying on many CSV files as storage database, however in a robust case scenario the data would be stored in servers and encrypted as needed as the smallest infection can cause a complete blackout of the HMGs in the worst scenarios. One suggestion for future work on that topic is to investigate the cloud-based system to store critical data related to the local grid or HMGs.

Finally, and for futuristic cutting-edge HMGs, the future generation of these systems will have the ability to connect with many systems like electric vehicles (V2G); giving a lot of room for the new generation of optimization and maybe adding more complexities, thus an interesting topic would be how to optimize HMGs while connecting to variable resources (as vehicles would be considered as a storage system and load at same time, but not constantly available).

10. References list

[1] M. Yu, “Assessment on the Environmental Impact of Conventional Energy Forms,” in *IOP Conference Series: Earth and Environmental Science*, IOP Publishing Ltd, Mar. 2021. doi: 10.1088/1755-1315/680/1/012002.

[2] P. R. Shukla *et al.*, *Climate Change 2022 Mitigation of Climate Change Working Group III Contribution to the Sixth Assessment Report of the Intergovernmental Panel on Climate Change Summary for Policymakers Edited by*. 2022. [Online]. Available: www.ipcc.ch

[3] Danish, R. Ulucak, and S. Erdogan, “The effect of nuclear energy on the environment in the context of globalization: Consumption vs production-based CO2 emissions,” *Nuclear Engineering and Technology*, vol. 54, no. 4, pp. 1312–1320, Apr. 2022, doi: 10.1016/J.NET.2021.10.030.

[4] Y. Parag and M. Ainspan, “Sustainable microgrids: Economic, environmental and social costs and benefits of microgrid deployment,” *Energy for Sustainable Development*, vol. 52, pp. 72–81, Oct. 2019, doi: 10.1016/J.ESD.2019.07.003.

[5] A. Hirsch, Y. Parag, and J. Guerrero, “Microgrids: A review of technologies, key drivers, and outstanding issues,” *Renewable and Sustainable Energy Reviews*, vol. 90, pp. 402–411, Jul. 2018, doi: 10.1016/J.RSER.2018.03.040.

[6] P. Nema, R. K. Nema, and S. Rangnekar, “A current and future state of art development of hybrid energy system using wind and PV-solar: A review,” *Renewable and Sustainable Energy Reviews*, vol. 13, no. 8, pp. 2096–2103, Oct. 2009, doi: 10.1016/J.RSER.2008.10.006.

[7] T. S. Ustun, C. Ozansoy, and A. Zayegh, “Recent developments in microgrids and example cases around the world—A review,” *Renewable and Sustainable Energy Reviews*, vol. 15, no. 8, pp. 4030–4041, Oct. 2011, doi: 10.1016/J.RSER.2011.07.033.

[8] C. V. Nayar, “Recent developments in decentralised mini-grid diesel power systems in Australia,” *Appl Energy*, vol. 52, no. 2–3, pp. 229–242, Jan. 1995, doi: 10.1016/0306-2619(95)00046-U.

[9] E. Rosales-Asensio, M. de Simón-Martín, D. Borge-Diez, J. J. Blanes-Peiró, and A. Colmenar-Santos, “Microgrids with energy storage systems as a means to increase power resilience: An application to office buildings,” *Energy*, vol. 172, pp. 1005–1015, Apr. 2019, doi: 10.1016/J.ENERGY.2019.02.043.

[10] J. Mitali, S. Dhinakaran, and A. A. Mohamad, “Energy storage systems: a review,” *Energy Storage and Saving*, vol. 1, no. 3, pp. 166–216, Sep. 2022, doi: 10.1016/J.ENSS.2022.07.002.

[11] A. H. Fathima and K. Palanisamy, “Optimization in microgrids with hybrid energy systems – A review,” *Renewable and Sustainable Energy Reviews*, vol. 45, pp. 431–446, May 2015, doi: 10.1016/J.RSER.2015.01.059.

[12] I. Jiménez-Vargas, J. M. Rey, and G. Osma-Pinto, “Sizing of hybrid microgrids considering life cycle assessment,” *Renew Energy*, vol. 202, pp. 554–565, Jan. 2023, doi: 10.1016/J.RENENE.2022.11.103.

[13] N. Y. Aydin, E. Kentel, and H. Sebnem Duzgun, “GIS-based site selection methodology for hybrid renewable energy systems: A case study from western Turkey,” *Energy Convers Manag*, vol. 70, pp. 90–106, Jun. 2013, doi: 10.1016/J.ENCONMAN.2013.02.004.

[14] L. Dai, S. Sun, T. Li, and S. G. Farkoush, “Probabilistic model for nondispatchable power resource integration with microgrid and participation in the power market,” *Energy Strategy Reviews*, vol. 33, p. 100611, Jan. 2021, doi: 10.1016/J.ESR.2020.100611.

[15] Y. Gao, Y. Matsunami, S. Miyata, and Y. Akashi, “Operational optimization for off-grid renewable building energy system using deep reinforcement learning,” *Appl Energy*, vol. 325, p. 119783, Nov. 2022, doi: 10.1016/J.APENERGY.2022.119783.

[16] Z. Ullah, M. R. Elkadeem, K. M. Kotb, I. B. M. Taha, and S. Wang, “Multi-criteria decision-making model for optimal planning of on/off grid hybrid solar, wind, hydro, biomass clean electricity supply,” *Renew Energy*, vol. 179, pp. 885–910, Dec. 2021, doi: 10.1016/J.RENENE.2021.07.063.

[17] S. Makhdoomi and A. Askarzadeh, “Daily performance optimization of a grid-connected hybrid system composed of photovoltaic and pumped hydro storage (PV/PHS),” *Renew Energy*, vol. 159, pp. 272–285, Oct. 2020, doi: 10.1016/J.RENENE.2020.06.020.

[18] M. A. Nasr, S. Nikkhah, G. B. Gharehpetian, E. Nasr-Azadani, and S. H. Hosseinian, “A multi-objective voltage stability constrained energy management system for isolated microgrids,” *International Journal of Electrical Power & Energy Systems*, vol. 117, p. 105646, May 2020, doi: 10.1016/J.IJEPES.2019.105646.

[19] C. D. Iweh and E. R. Akupan, “Control and optimization of a hybrid solar PV – Hydro power system for off-grid applications using particle swarm optimization (PSO) and differential evolution (DE),” *Energy Reports*, vol. 10, pp. 4253–4270, Nov. 2023, doi: 10.1016/J.EGYR.2023.10.080.

[20] Z. Belboul *et al.*, “Multiobjective Optimization of a Hybrid PV/Wind/Battery/Diesel Generator System Integrated in Microgrid: A Case Study in Djelfa, Algeria,” *Energies 2022, Vol. 15, Page 3579*, vol. 15, no. 10, p. 3579, May 2022, doi: 10.3390/EN15103579.

[21] V. H. Nguyen, Q. T. Tran, and Y. Besanger, “SCADA as a service approach for interoperability of micro-grid platforms,” *Sustainable Energy, Grids and Networks*, vol. 8, pp. 26–36, Dec. 2016, doi: 10.1016/J.SEGAN.2016.08.001.

[22] U. Yaqub, A. Al-Nasser, and T. Sheltami, “Implementation of a hybrid wind-solar desalination plant from an Internet of Things (IoT) perspective on a network simulation tool,” *Applied Computing and Informatics*, vol. 15, no. 1, pp. 7–11, Jan. 2019, doi: 10.1016/J.ACI.2018.03.001.

[23] G. Murugan and S. Vijayarajan, “IoT based secured data monitoring system for renewable energy fed micro grid system,” *Sustainable Energy Technologies and Assessments*, vol. 57, p. 103244, Jun. 2023, doi: 10.1016/J.SETA.2023.103244.

[24] J. Tuuf, H. Lindén, S. Lieskoski, and M. Björklund-Sänkiaho, “Development of a nano-size off-grid energy system using renewables and IoT technologies at the Meteoria visitor center: A Finnish case study,” *Heliyon*, vol. 9, no. 11, p. e21473, Nov. 2023, doi: 10.1016/J.HELIYON.2023.E21473.

11. Appendices

Full data storage files and source files could be found in my personal GitHub page that I built during the work on the thesis:

https://github.com/AmedBrook/Optimized-IoT_connected-NanoGrid/tree/main/mqtt_ui

https://github.com/AmedBrook/Optimized-IoT_connected-NanoGrid/tree/full_clt/mqtt_ui

11.1 Simplified Optimization source code* (Full code in GitHub).

```
### Introducing problem parameters
-----------------------------------------------------

Q_max = 5000
Q_0 = 0.1*Q_max
Q_final = 0.2*Q_max
eff_to_bss = 0.9
eff_from_bss = 0.9
P_max = 10000
P_min = 0
dt = 1
t_max = 10
t = np.atleast_2d(np.arange(0,t_max,dt)).T.conj()
n = len(t)
m = 1
fc_offset = 190
V_steps = [x for x in range(0,n)]
P_pub = [0 for x in range(0,n)]
V_steps_z = V_steps[:-1]
Q_h = { x:135000 for x in V_steps}
T_H = {x:21 for x in V_steps}
S_pv30 = 24
Ones = {x:1 for x in V_steps}

### Upper and lower bounds
-----------------------------------------

P_pv30_max = 2000
# (Irr_sol[k]*S_pv30 for k in V_steps)  Maximal power generated by Solar PV modules tilted by
30° (for maximal irradiance (W/m^2).
P_pv30_min = 0
    # Minimal power generated by Solar PV modules tilted by 30° (at night irradiance = 0 W/m^2).
P_pv60_max = 2000
```

```
    # (Irr_sol[k]*S_pv60 for k in V_steps)  Maximal power generated by Solar PV modules tilted
by 60° (for maximal irradiance (W/m^2).
P_pv60_min = 0
     # Minimal power generated by Solar PV modules tilted by 60° (at night irradiance = 0
W/m^2).
P_wind_max = 10000
     # (1/2*V_wind[k]*Cp*rho*math.pi*R**2 for k in V_steps) Maximal power generated by the wind
turbine (cut-out wind speed (kwh).  P = 0.5 Cp ρ π R2 V3
P_wind_min = 0
     # Minimal power generated by the wind turbine (cut-in wind speed (kwh).
P_hp_max = 1000.1
     # (Q_h / 1 / (1 - T_L[k] / T_H) for k in V_steps) Maximal power transferred to Meteoria
barn's heat pumps (kwh).
P_hp_min =  0
     # Minimal power transferred to Meteoria barn's heat pumps (kwh).
P_sh_max =  5000.1
      # Maximal power transferred to the space heater (kwh).
P_sh_min =  0
      # Minimal power transferred to the space heater (kwh).
P_bio_max = 1000.2
    # Maximal power generated by biomeiler compost heater (kwh).
P_bio_min = 0
      # Minimal power generated by biomeiler compost heater (kwh).
P_wh_max = 6000
       # Maximal power transferred to water tank heater (kwh).
P_wh_min = 0
    # Minimal power transferred to water tank heater (kwh).
P_stnb_max = 3000                                                    # Maximal standby
power to keep essential equipment on standby mode (kwh).
P_stnb_min = 3000
       # Minimal standby power to keep essential equipment on standby mode (kwh).

### Computing RES and loads
-----------------------------------------------------


P_pv30 = {k:Irr_sol_dict[k] * S_pv30 for k in V_steps}
P_wind = {k:Wind_vlct_dict[k] ** 3 * (6 ** 2) * np.pi *  1.225 * 0.45 * (1/2) for k in V_steps}

P_to_hp = {k:(Q_h[k] / Ones[k] / (Ones[k] - T_L_dict[k]) / T_H[k])*10**(-3) for k in V_steps}


### Optimization Problem modelling
--------------------------------------
Optim = LpProblem('Energy_Opt',LpMinimize)

### Slope, intercept, and maximum fuel bound calculation.
a_j = (fuelCon(0.9*P_max, P_max) - fuelCon(0.2*P_max, P_max)) /(0.9*P_max)
b_j = fuelCon(0.2*P_max, P_max) - a_j*0.2*P_max #Intercept.
```

```
FOC_max = fuelCon(0.9*P_max,P_max) # Max fuel bound.




### Setting-up decision Variables.
-------------------------------------------------
Q_bss = LpVariable.dicts("Q_bss", V_steps, lowBound=0.2*Q_max, cat=LpContinuous)
# Battery charge at time step k.
P_from_bss = LpVariable.dicts("P_from_bss", V_steps, lowBound=P_min, cat=LpContinuous)
# Power transferred from the battery to the load.
P = LpVariable.dicts("P", V_steps, lowBound=P_min, upBound=0.9*P_max, cat=LpContinuous)
# Power generated by the Genset A.
P_res = LpVariable.dicts("P_res", V_steps, lowBound=P_min, upBound=0.9*P_max, cat=LpContinuous)

# Total Power from RES.
P_load = LpVariable.dicts("P_load", V_steps, lowBound=P_min, upBound=0.9*P_max,
cat=LpContinuous)
# Power transferred from the Genset A to the load at time step k.
P_to_sh = LpVariable.dicts("P_to_sh", V_steps, lowBound=P_sh_min, upBound=.9*P_max,
cat=LpContinuous)
# Power to space heater (excess heat (kw).
P_to_wh = LpVariable.dicts("P_to_wt", V_steps, lowBound=P_wh_min, upBound=.9*P_max,
cat=LpContinuous)
# Power to water tank heater.

Z =  LpVariable.dicts("Z", V_steps_z, lowBound=0, upBound=1, cat=LpInteger)
# Additional cost fuel oil consumption when starting Genset j.
FOC = LpVariable.dicts("FOC", V_steps, lowBound=0, cat=LpContinuous)
# Fuel oil consumption of genset j.
P_to_bss =  LpVariable.dicts("P_to_bss", V_steps, lowBound=P_min, cat=LpContinuous)
Y_to_bss = LpVariable.dicts("Y_to_bss", V_steps, lowBound=0, upBound=1, cat=LpBinary)
# Genset selector to charge the battery at time step k.
Y_from_bss = LpVariable.dicts("Y_from_bss", V_steps, lowBound=0, upBound=1, cat=LpBinary)
# Battery selecter to transfert to the Genset j st time step k.
Y = LpVariable.dicts("Y", V_steps, lowBound=0, upBound=1, cat=LpBinary)
# Genset selecter : turned on : Y=1, turned off : Y=0.
Y_sh =  LpVariable.dicts("Y_sh", V_steps, lowBound=0, upBound=1, cat=LpBinary)
# Binary state of space heater.
Y_hp = LpVariable.dicts("Y_hp", V_steps, lowBound=0, upBound=1, cat=LpBinary) # Binary state of
heat pumps.
Y_wh = LpVariable.dicts("Y_wh", V_steps, lowBound=0, upBound=1, cat=LpBinary) # Binary state of
water heater.
#Y_wind = LpVariable.dicts("Y_wind", V_steps, lowBound=0, upBound=1, cat=LpBinary) # Binary
state of wind turbine.
#Y_pv30 = LpVariable.dicts("Y_pv30", V_steps, lowBound=0, upBound=1, cat=LpBinary)
```

```
# Binary state of solar panels tilted by 30°.
#Y_pv60 = LpVariable.dicts("Y_pv60", V_steps, lowBound=0, upBound=1, cat=LpBinary)
# Binary state of solar panels tilted by 60°.
Y_res = LpVariable.dicts("Y_res", V_steps, lowBound=0, upBound=1, cat=LpBinary)
# Binary state RES.

### Setting-up problem constraints
------------------------------------------------------_
FC = sum(FOC[k] for k in V_steps) * dt/1000
# sum of the fuel oil consumption for all gensets over all k steps.
L_added_cost = sum(Z[i] for i in V_steps_z)
# Sum of all the additional costs including starting costs.
Optim += lpSum (FC + L_added_cost), " Minimization fuel oil consumption objective "




for k in V_steps:

    Optim += FOC[k] == P[k]*a_j + (b_j - fc_offset)*Y[k]
    Optim += P_res[k] == P_pv30[k] + P_pv60[k] + P_wind[k]
    Optim += P_res[k] >= P_min * Y_res[k]
    Optim += P_load[k] + eff_from_bss*P_from_bss[k] + P_res[k] >= P_to_hp[k] +
             P_stnb[k]
    Optim += P_load[k] == P_to_hp[k] + P_to_sh[k]
    Optim += P_load[k] + P_stnb[k] <= P[k] + P_res[k]
    Optim += P_to_hp[k] <= P_hp_max * Y_hp[k]
    Optim += P_to_wh[k] <= 0.9*P_max * Y_wh[k]
    Optim += P_to_sh[k] <= 0.9*P_max * Y_sh[k]
    Optim += P[k]  <= P_max * Y[k]
    Optim += P[k]  >= P_min * Y[k]
    Optim += Y[k] <= 1
    Optim += P_to_bss[k] <= eff_to_bss * P_max * Y_to_bss[k]
    Optim += P_from_bss[k] <= eff_from_bss * P_max * Y_from_bss[k]
    Optim += Y_to_bss[k] + Y_from_bss[k]  <= 1
    if k == V_steps[0] :
        Optim += Q_bss[k] == Q_0 + eff_to_bss*P_to_bss[k]*dt -
                             P_from_bss[k]*dt
    else :
        Optim += Q_bss[k] == Q_bss[k-1] + eff_to_bss*P_to_bss[k]*dt -
                             P_from_bss[k]*dt

for k in range(V_steps[0], V_steps[-1]):
     Optim += Z[k] >= Y[k + 1] - Y[k]
     Optim += Q_bss[V_steps[-1]] == Q_final

### Solving the problem.
--------------------------------
status = Optim.solve(GUROBI())
```

11.2 Client implantation

```
username= access_key.username
password= access_key.password


sub_topic1 = 'open/meteoria/solarRadiation'
sub_topic2 = 'open/meteoria/windSpeed'
sub_topic3 = 'open/meteoria/airTemp'
mes1 = 'Irr_sol'
mes2 = 'Wind_speed'
mes3 = 'Air_Temp'


unit1 = 'W/m^2'
unit2 = 'm/s'
unit3 = '°C'



# The callback for when the client receives a CONNACK response from the server.
def on_connect(client, userdata, flags, rc):
    print("Connected with result code "+str(rc))
    client.subscribe([(sub_topic1, 2), (sub_topic2, 2), (sub_topic3, 2)])


data_stream_Irr, data_stream_wind, data_stream_Temp, strg_Irr, strg_wind, strg_Temp = [], [],
[], [], [], []
steps=[k for k in range (0,9)]


def on_message(client, userdata, message):
    Irr_list = []
    df_irr = pd.read_csv('Irr_sol.csv')
    df_irr_drop = df_irr.dropna()
    for i in df_irr_drop.iloc[-1,:]:
        Irr_list.append(float(i))
        Irr_sol_dict = dict(zip(V_steps, Irr_list))


    Wind_vlct = []
    df_vlct = pd.read_csv('Wind_speed.csv')
    df_vlct_drop = df_vlct.dropna()
    for i in df_vlct_drop.iloc[-1,:]:
```

```
    Wind_vlct.append(float(i))
    Wind_vlct_dict = dict(zip(V_steps, Wind_vlct))

T_L = []   # Temperature outside
df_temp = pd.read_csv('Air_temp.csv')
df_temp_drop = df_temp.dropna()
for i in df_temp_drop.iloc[-1,:]:
T_L.append(float(i))
T_L_dict = dict(zip(V_steps, T_L))
global data_stream_Irr, data_stream_wind, data_stream_Temp, strg_Irr, strg_wind,
strg_Temp, P_pub
P_stnb = {x:3000 for x in V_steps}

i=0
if len (data_stream_Irr or data_stream_wind or data_stream_Temp) < 10  :
   for j in range(0,9):
    while i < 1 :
        log = json.loads(message.payload.decode("utf-8"))
        if message.topic == sub_topic1 :
            data_stream_Irr.append(log)

        elif message.topic == sub_topic2 :
            data_stream_wind.append(log)

        elif message.topic == sub_topic3 :
            data_stream_Temp.append(log)

        i+=1
        j+=1

         if len(data_stream_Irr) == 10 or len(data_stream_wind) == 10 or
         len(data_stream_Temp) == 10 :
            strg_Irr = data_stream_Irr
            strg_wind = data_stream_wind
            strg_Temp = data_stream_Temp
            print('returned list', strg_Irr)
            print('returned list', strg_wind)
            print('returned list', strg_Temp)
             Ometeor.Optim.solve()
```

```
else :

        Irr_list = []
        df_irr = pd.read_csv('Irr_sol.csv')
        df_irr_drop = df_irr.dropna()
        for i in df_irr_drop.iloc[-1,:]:
            Irr_list.append(float(i))
            Irr_sol_dict = dict(zip(V_steps, Irr_list))

        Wind_vlct = []
        df_vlct = pd.read_csv('Wind_speed.csv')
        df_vlct_drop = df_vlct.dropna()
        for i in df_vlct_drop.iloc[-1,:]:
            Wind_vlct.append(float(i))
            Wind_vlct_dict = dict(zip(V_steps, Wind_vlct))

        T_L = []   # Temperature outside
        df_temp = pd.read_csv('Air_temp.csv')
        df_temp_drop = df_temp.dropna()
        for i in df_temp_drop.iloc[-1,:]:
            T_L.append(float(i))
            T_L_dict = dict(zip(V_steps, T_L))



        P_pv30 = {k:Irr_sol_dict[k] * S_pv30 for k in V_steps}
        print('P_pv30 = ',P_pv30)

         P_wind = {k:Wind_vlct_dict[k] ** 3 * (6 ** 2) * np.pi *  1.225 * 0.45 * (1/2)
          for k in V_steps}        # = Irr_sol[k]*S_pv30 for k in V_steps
        print('P_wind = ', P_wind)

         P_to_hp = {k:(Q_h[k] / Ones[k] / (Ones[k] - T_L_dict[k]) / T_H[k])*10**(-3)
          for k in V_steps}
        print('P_to_hp = ', P_to_hp)

        P_stnb = {x:3000 for x in V_steps}
        with open("Irr_sol.csv", "a+") as Irr_sol:
                    mystring_irr = ','.join(map(str,data_stream_Irr))
                    Irr_sol.write("%s" %(mystring_irr)+ "\n")
```

```
            Irr_sol.close()


data_stream_Irr = []
i=0
for j in range(10):
 while i < 1 :
    log = json.loads(message.payload.decode("utf-8"))
    if message.topic == sub_topic1 :


        data_stream_Irr.append(log)


    #Irr_sol[j] = data_stream[j]
    i+=1
    if len(data_stream_Irr) == 10 :
        strg_Irr=data_stream_Irr
        print('returned list', strg_Irr)



with open("Wind_speed.csv", "a+") as wind_speed:
            mystring_wind = ','.join(map(str,data_stream_wind))
            wind_speed.write("%s" %(mystring_wind)+ "\n")
            wind_speed.close()


data_stream_wind = []
i=0
for j in range(10):
 while i < 1 :
    log = json.loads(message.payload.decode("utf-8"))
    if message.topic == sub_topic2 :


        data_stream_wind.append(log)


    i+=1
    if len(data_stream_wind) == 10:
        strg_wind=data_stream_wind
        print('returned list:', strg_wind)



with open("Air_temp.csv", "a+") as Air_temp:
```

```
        mystring_temp = ','.join(map(str,data_stream_Temp))
        Air_temp.write("%s" %(mystring_temp)+ "\n")
        Air_temp.close()

    data_stream_Temp = []
    i=0
    for j in range(10):
     while i < 1 :
        log = json.loads(message.payload.decode("utf-8"))
        if message.topic == sub_topic3 :

            data_stream_Temp.append(log)

        i+=1
        if len(data_stream_Temp) == 10:
            strg_Temp=data_stream_Temp
            print('returned list:', strg_Temp)
              Ometeor.Optim.solve()


print('#######', mes1, ':',str(message.payload.decode('utf-8')),unit1,'#######')
print('=======', str(datetime.now()), '=======')
print('message topic is : ',message.topic)
print('message qos is : ', message.qos)
print('mesaage retain flag = ', message.retain)
q.put(message)
print('data_stream:',data_stream_Irr)
print(mes1,':', data_stream_Irr)

print('#######', mes2, ':',str(message.payload.decode('utf-8')),unit2,'#######')
print('=======', str(datetime.now()), '=======')
print('message topic is : ',message.topic)
print('message qos is : ', message.qos)
print('mesaage retain flag = ', message.retain)
q.put(message)
print('data_stream:',data_stream_wind)
print(mes2,':', data_stream_wind)

print('#######', mes3, ':',str(message.payload.decode('utf-8')),unit2,'#######')
```

```
    print('=======', str(datetime.now()), '=======')
    print('message topic is : ',message.topic)
    print('message qos is : ', message.qos)
    print('mesaage retain flag = ', message.retain)
    q.put(message)
    print('data_stream:',data_stream_Temp)
    print(mes3,':', data_stream_Temp)


def on_log(client, userdata, level, buf):
    print("log: ",buf)

while not q.empty():
   message = q.get()
   if message is None:
       continue
   print("received from queue",str(message.payload.decode("utf-8")))

broker_address='iot.novia.fi'
print('Creating new instance of the client')
client=mqtt.Client()
client.on_message=on_message #attach function to callback
print("connecting to Novia broker..")

client.on_connect = on_connect
client.username_pw_set(username, password)
client.connect(broker_address, 1883, 60)
sleep(4) # wait
client.loop_stop() #stop the loop

client.loop_start() #start the loop
client.on_log=on_log #printing log information
print("Subscribing to topic", sub_topic1)
print("Subscribing to topic", sub_topic2)
print("Subscribing to topic", sub_topic3)
client.subscribe[[(sub_topic1, 2),(sub_topic2, 2), (sub_topic3, 2)]]
#client.subscribe(sub_topic2)
print('returned list:', strg_Irr)
print('returned list:', strg_wind)
```

```
print('returned list:', strg_Temp)


def publish(client):
    for i in P_pub:
        msg = P_pub[i]
    pub_topic = 'meteoria/optimizer/gensetControl'

    while True:
        sleep(1)
        result = client.publish(pub_topic, msg)
        # result: [0, 1]
        status = result[0]
        if status == 0:
            print(f"Send `{msg}` to topic `{pub_topic}`")
        else:
            print(f"Failed to send message to topic {pub_topic}")



def run():
    publish(client)

if __name__ == '__main__':
    run()
```

11.3 Sample of the Optimizer's outputs in runtime*

```
Connected to Python 3.11.3
...
Z_6 = 1.0
Z_7 = -0.0
Z_8 = -0.0
Actual total fuel comsumption: 858.1902011713191 g
Optimization completed !
################################################
P_pv30 :  {0: 8.16, 1: 8.16, 2: 7.92, 3: 8.16, 4: 7.68, 5: 7.4399999999999995, 6: 6.959999999999999, 7: 6.959999999999999, 8: 6.48, 9: 5.76}
P_wind :  {0: 3236.4175987458234, 1: 3667.407935378938, 2: 2478.428229038692, 3: 3896.556638155591, 4: 2655.3942453171 67, 5: 3034.2018954521, 6: 3667.407935378938, 7: 2148.4366404665707, 8: 1995.0369987356623, 9: 1454.381972078298}
P_to_hp {0: 1.1688311688311688, 1: 1.1688311688311688, 2: 1.1688311688311688, 3: 1.1688311688311688, 4: 1.1688311688311688, 5 : 1.1688311688311688, 6: 1.1688311688311688, 7: 1.1688311688311688, 8: 1.1688311688311688, 9: 1.1688311688311688}
################################################
Gurobi Optimizer version 10.0.0 build v10.0.0rc2 (win64)

CPU model: 11th Gen Intel(R) Core(TM) i5-1135G7 @ 2.40GHz, instruction set [SSE2|AVX|AVX2|AVX512]
Thread count: 4 physical cores, 8 logical processors, using up to 8 threads

Optimize a model with 225 rows, 156 columns and 539 nonzeros
Model fingerprint: 0x09f1cc03
Variable types: 92 continuous, 64 integer (0 binary)
Coefficient statistics:
  Matrix range     [9e-01, 2e+05]
  Objective range  [1e-03, 1e+00]
  Bounds range     [1e+00, 9e+03]
  RHS range        [1e+00, 4e+03]
Presolve removed 225 rows and 156 columns
Presolve time: 0.00s
Presolve: All rows and columns removed

Explored 0 nodes (0 simplex iterations) in 0.04 seconds (0.00 work units)
Thread count was 1 (of 8 available processors)

Solution count 2: 1206.25 1206.25

Optimal solution found (tolerance 1.00e-04)
Best objective 1.206249751691e+03, best bound 1.206249751691e+03, gap 0.0000%
Gurobi status= 2
FOC_0 = 0.0
FOC_1 = 0.0
FOC_2 = 277807.5052486564
FOC_3 = 0.0
FOC_4 = 251620.3054899056
FOC_5 = 0.0
FOC_6 = 0.0
FOC_7 = 325762.3904327571
FOC_8 = 348059.55051956547
FOC_9 = 0.0
P_0 = 0.0
P_1 = 0.0
P_2 = 514.8206021301339
P_3 = 0.0
P_4 = 334.09458585166067
P_5 = 0.0
P_6 = 0.0
P_7 = 845.7721907022578
P_8 = 999.6518324331646
P_from_bss_0 = 0.0
P_from_bss_1 = 0.0
P_from_bss_2 = 0.0
P_from_bss_3 = 0.0
P_from_bss_4 = 0.0
P_from_bss_5 = 0.0
P_from_bss_6 = 0.0
P_from_bss_7 = 0.0
P_from_bss_8 = 0.0
P_load_0 = 1.1688311688285467
P_load_1 = 1.1688311688285467
P_load_2 = 1.1688311688285467
P_load_3 = 1.1688311688285467
P_load_4 = 1.1688311688285467
P_load_5 = 1.1688311688285467
P_load_6 = 1.1688311688285467
P_load_7 = 1.1688311688285467
P_load_8 = 1.1688311688285467
P_res_0 = 3244.5775987458233
P_res_1 = 3678.567935378938
P_res_2 = 2486.348229038692
P_res_3 = 3904.7166381555908
P_res_4 = 2667.074245317167
P_res_5 = 3041.6418954521
P_res_6 = 3675.867935378938
P_res_7 = 2155.3966404665707
P_res_8 = 2001.5169987356624
P_stnb_0 = 3000.0
P_stnb_1 = 3000.0
P_stnb_2 = 3000.0
P_stnb_3 = 3000.0
P_stnb_4 = 3000.0
P_stnb_5 = 3000.0
P_stnb_6 = 3000.0
P_stnb_7 = 3000.0
P_stnb_8 = 3000.0
P_to_bss_0 = 9000.000000000002
P_to_bss_1 = 0.0
P_to_bss_2 = 0.0
P_to_bss_3 = 0.0
P_to_bss_4 = 0.0
P_to_bss_5 = 0.0
P_to_bss_6 = 0.0
P_to_bss_7 = 0.0
P_to_bss_8 = 0.0
P_to_sh_0 = 0.0
P_to_sh_1 = 0.0
P_to_sh_2 = 0.0
P_to_sh_3 = 0.0
P_to_sh_4 = 0.0
P_to_sh_5 = 0.0
P_to_sh_6 = 0.0
P_to_sh_7 = 0.0
P_to_sh_8 = 0.0
P_to_wt_0 = 0.0
P_to_wt_1 = 0.0
P_to_wt_2 = 0.0
P_to_wt_3 = 0.0
P_to_wt_4 = 0.0
P_to_wt_5 = 0.0
P_to_wt_6 = 0.0
P_to_wt_7 = 0.0
P_to_wt_8 = 0.0
Q_bss_0 = 8600.0
Q_bss_1 = 8600.0
Q_bss_2 = 8600.0
Q_bss_3 = 8600.0
Q_bss_4 = 8600.0
Q_bss_5 = 8600.0
Q_bss_6 = 8600.0
Q_bss_7 = 8600.0
Q_bss_8 = 8600.0
Q_bss_9 = 1000.0
Y_0 = -0.0
Y_1 = 0.0
Y_2 = 1.0
```

Y_3 = -0.0
Y_4 = 1.0
Y_5 = -0.0
Y_6 = 0.0
Y_7 = 1.0
Y_8 = 1.0
Y_9 = -0.0
Y_from_bss_0 = 0.0
Y_from_bss_1 = -0.0
Y_from_bss_2 = -0.0
Y_from_bss_3 = -0.0
Y_from_bss_4 = -0.0
Y_from_bss_5 = -0.0
Y_from_bss_6 = -0.0
Y_from_bss_7 = -0.0
Y_from_bss_8 = -0.0
Y_hp_0 = 1.0
Y_hp_1 = 1.0
Y_hp_2 = 1.0
Y_hp_3 = 1.0
Y_hp_4 = 1.0
Y_hp_5 = 1.0
Y_hp_6 = 1.0
Y_hp_7 = 1.0
Y_hp_8 = 1.0
Y_sh_0 = 1.0
Y_sh_1 = 1.0
Y_sh_2 = 1.0
Y_sh_3 = 1.0
Y_sh_4 = 1.0
Y_sh_5 = 1.0
Y_sh_6 = 1.0
Y_sh_7 = 1.0
Y_sh_8 = 1.0
Y_to_bss_0 = 1.0
Y_to_bss_1 = -0.0
Y_to_bss_2 = -0.0
Y_to_bss_3 = -0.0
Y_to_bss_4 = -0.0
Y_to_bss_5 = -0.0
Y_to_bss_6 = -0.0
Y_to_bss_7 = -0.0
Y_to_bss_8 = -0.0
Y_wh_0 = 1.0
Y_wh_1 = 1.0
Y_wh_2 = 1.0
Y_wh_3 = 1.0
Y_wh_4 = 1.0
Y_wh_5 = 1.0
Y_wh_6 = 1.0
Y_wh_7 = 1.0
Y_wh_8 = 1.0
Z_0 = 0.0
Z_1 = 1.0
Z_2 = -0.0
Z_3 = 1.0
Z_4 = -0.0
Z_5 = 0.0
Z_6 = 1.0
Z_7 = -0.0
Z_8 = -0.0
Actual total fuel comsumption: 1206.2497516908848 g
Optimization completed !
##
P_pv30 : {0: 8.16, 1: 8.16, 2: 7.92, 3: 8.16, 4: 7.68, 5: 7.4399999999999995, 6: 6.959999999999999, 7: 6.959999999999999, 8: 6.48, 9: 5.76}
P_wind : {0: 3236.4175987458234, 1: 3667.407935378938, 2: 2478.428229038692, 3: 3896.556638155591, 4: 2655.394245317167, 5: 3034.2018954521, 6: 3667.407935378938, 7: 2148.4366404665707, 8: 1995.0369987356623, 9: 1454.381972078298}
P_to_hp {0: 1.1688311688311688, 1: 1.1688311688311688, 2: 1.1688311688311688, 3: 1.1688311688311688, 4: 1.1688311688311688, 5 : 1.1688311688311688, 6: 1.1688311688311688, 7: 1.1688311688311688, 8: 1.1688311688311688, 9: 1.1688311688311688}
##
Gurobi Optimizer version 10.0.0 build v10.0.0rc2 (win64)

CPU model: 11th Gen Intel(R) Core(TM) i5-1135G7 @ 2.40GHz, instruction set [SSE2|AVX|AVX2|AVX512]
Thread count: 4 physical cores, 8 logical processors, using up to 8 threads

Optimize a model with 250 rows, 169 columns and 599 nonzeros
Model fingerprint: 0x2e3ca40b
Variable types: 100 continuous, 69 integer (0 binary)
Coefficient statistics:
  Matrix range   [9e-01, 2e+05]
  Objective range [1e-03, 1e+00]
  Bounds range   [1e+00, 9e+03]
  RHS range   [1e+00, 4e+03]
Presolve removed 226 rows and 136 columns
Presolve time: 0.00s
Presolved: 24 rows, 33 columns, 64 nonzeros
Variable types: 25 continuous, 8 integer (8 binary)
Found heuristic solution: objective 1632.7545436

Explored 0 nodes (0 simplex iterations) in 0.01 seconds (0.00 work units)
Thread count was 8 (of 8 available processors)

Solution count 1: 1632.75

Optimal solution found (tolerance 1.00e-04)
Best objective 1.632754543573e+03, best bound 1.632754543573e+03, gap 0.0000%
Gurobi status= 2
FOC_0 = 0.0
FOC_1 = 0.0
FOC_2 = 277807.5052486564
FOC_3 = 0.0
FOC_4 = 251620.3054899056
FOC_5 = 0.0
FOC_6 = 0.0
FOC_7 = 325762.3904327571
FOC_8 = 348059.55051956547
FOC_9 = 426504.7918822176
P_0 = 0.0
P_1 = 0.0
P_2 = 514.8206021301339
P_3 = 0.0
P_4 = 334.09458585166067
P_5 = 0.0
P_6 = 0.0
P_7 = 845.7721907022578
P_8 = 999.6518324331646
P_9 = 1541.026859090529
P_from_bss_0 = 0.0
P_from_bss_1 = 0.0
P_from_bss_2 = 0.0
P_from_bss_3 = 0.0
P_from_bss_4 = 0.0
P_from_bss_5 = 0.0
P_from_bss_6 = 0.0
P_from_bss_7 = 0.0
P_from_bss_8 = 0.0
P_from_bss_9 = 0.0
P_load_0 = 1.1688311688285467
P_load_1 = 1.1688311688285467
P_load_2 = 1.1688311688285467
P_load_3 = 1.1688311688285467
P_load_4 = 1.1688311688285467
P_load_5 = 1.1688311688285467
P_load_6 = 1.1688311688285467
P_load_7 = 1.1688311688285467
P_load_8 = 1.1688311688285467
P_load_9 = 1.1688311688285467
P_res_0 = 3244.5775987458233
P_res_1 = 3678.567935378938
P_res_2 = 2486.348229038692
P_res_3 = 3904.7166381555908
P_res_4 = 2667.074245317167
P_res_5 = 3041.6418954521
P_res_6 = 3675.8679353578938
P_res_7 = 2155.3966404665707
P_res_8 = 2001.5169987356624

P_res_9 = 1460.141972078298
P_stnb_0 = 3000.0
P_stnb_1 = 3000.0
P_stnb_2 = 3000.0
P_stnb_3 = 3000.0
P_stnb_4 = 3000.0
P_stnb_5 = 3000.0
P_stnb_6 = 3000.0
P_stnb_7 = 3000.0
P_stnb_8 = 3000.0
P_stnb_9 = 3000.0
P_to_bss_0 = 555.5555555555555
P_to_bss_1 = 0.0
P_to_bss_2 = 0.0
P_to_bss_3 = 0.0
P_to_bss_4 = 0.0
P_to_bss_5 = 0.0
P_to_bss_6 = 0.0
P_to_bss_7 = 0.0
P_to_bss_8 = 0.0
P_to_bss_9 = 0.0
P_to_sh_0 = 0.0
P_to_sh_1 = 0.0
P_to_sh_2 = 0.0
P_to_sh_3 = 0.0
P_to_sh_4 = 0.0
P_to_sh_5 = 0.0
P_to_sh_6 = 0.0
P_to_sh_7 = 0.0
P_to_sh_8 = 0.0
P_to_sh_9 = 0.0
P_to_wt_0 = 0.0
P_to_wt_1 = 0.0
P_to_wt_2 = 0.0
P_to_wt_3 = 0.0
P_to_wt_4 = 0.0
P_to_wt_5 = 0.0
P_to_wt_6 = 0.0
P_to_wt_7 = 0.0
P_to_wt_8 = 0.0
P_to_wt_9 = 0.0
Q_bss_0 = 1000.0
Q_bss_1 = 1000.0
Q_bss_2 = 1000.0
Q_bss_3 = 1000.0
Q_bss_4 = 1000.0
Q_bss_5 = 1000.0
Q_bss_6 = 1000.0
Q_bss_7 = 1000.0
Q_bss_8 = 1000.0
Q_bss_9 = 1000.0
Y_0 = -0.0
Y_1 = 0.0
Y_2 = 1.0
Y_3 = -0.0
Y_4 = 1.0
Y_5 = -0.0
Y_6 = 0.0
Y_7 = 1.0
Y_8 = 1.0
Y_9 = 1.0
Y_from_bss_0 = 0.0
Y_from_bss_1 = 1.0
Y_from_bss_2 = 1.0
Y_from_bss_3 = 1.0
Y_from_bss_4 = 1.0
Y_from_bss_5 = 1.0
Y_from_bss_6 = 1.0
Y_from_bss_7 = 1.0
Y_from_bss_8 = 1.0
Y_from_bss_9 = 1.0
Y_hp_0 = 1.0
Y_hp_1 = 1.0
Y_hp_2 = 1.0
Y_hp_3 = 1.0
Y_hp_4 = 1.0
Y_hp_5 = 1.0
Y_hp_6 = 1.0
Y_hp_7 = 1.0
Y_hp_8 = 1.0
Y_hp_9 = 1.0
Y_sh_0 = 1.0
Y_sh_1 = 1.0
Y_sh_2 = 1.0
Y_sh_3 = 1.0
Y_sh_4 = 1.0
Y_sh_5 = 1.0
Y_sh_6 = 1.0
Y_sh_7 = 1.0
Y_sh_8 = 1.0
Y_sh_9 = 1.0
Y_to_bss_0 = 1.0
Y_to_bss_1 = -0.0
Y_to_bss_2 = -0.0
Y_to_bss_3 = -0.0
Y_to_bss_4 = -0.0
Y_to_bss_5 = -0.0
Y_to_bss_6 = -0.0
Y_to_bss_7 = -0.0
Y_to_bss_8 = -0.0
Y_to_bss_9 = 0.0
Y_wh_0 = 1.0
Y_wh_1 = 1.0
Y_wh_2 = 1.0
Y_wh_3 = 1.0
Y_wh_4 = 1.0
Y_wh_5 = 1.0
Y_wh_6 = 1.0
Y_wh_7 = 1.0
Y_wh_8 = 1.0
Y_wh_9 = 1.0
Z_0 = 0.0
Z_1 = 1.0
Z_2 = -0.0
Z_3 = 1.0
Z_4 = -0.0
Z_5 = 0.0
Z_6 = 1.0
Z_7 = -0.0
Z_8 = -0.0
Actual total fuel comsumption: 1632.7545435731024 g
Optimization completed !
##

11.4 Flask App back-end development*

```
app = Flask(__name__)

            @app.route('/L_P')
            def L_P():

            # POWER GENERATED BY THE GENSET.(Chart 1)
                P_dframe = {
                        'Power generated by the Genset.': P_list,
                        'Scale' : P_list}

                P_A_df = pd.DataFrame(P_dframe)
                fig = px.bar(P_A_df, x = V_steps,
                        y = P_list,
                        title = '________Power Generated By The Genset._________',
                        labels = dict(x = "Time Step (s)", y = "kw"), text_auto = True)
                fig.update_traces(width=0.2)
                graphJSON1 = json.dumps(fig, cls=plotly.utils.PlotlyJSONEncoder)


            #-------------------------------------------------------------------------------#


            # POWER FROM THE GENSET TO LOAD.(Chart 2)
                L_dframe = {
                        "Load (Heat Pump & Standby loads).": L_list,
                        'Scale' : L_list}

                L_df = pd.DataFrame(L_dframe)


                fig = px.bar(L_df, x = V_steps,
                        y = L_list,
                        title = '_________Load (Heat Pump & Standby loads).________',
                        labels = dict(x = "Time Step (s)", y = "kw"), text_auto = True)
                fig.update_traces(width=0.2)
                graphJSON2 = json.dumps(fig, cls=plotly.utils.PlotlyJSONEncoder)
```

```
    return render_template('L_P.html', graphJSON2=graphJSON2, graphJSON1=graphJSON1)


@app.route('/Power_Flow')


def Power_Flow():
    Powr_flow_area = {
                'Power generated by the Genset.': P_list,
                "Power from Genset to load.": P_load_list,
                "Power from BSS to load.": P_bss_list,
                "Load requirements.": L_list,
                "Power from RES.": P_res_list
   }


    area_df = pd.DataFrame(Powr_flow_area)
    fig = px.area(area_df, x = V_steps,
                y = area_df.columns,
                title = '________Power Flow._________',
                labels = dict(x = "Time Step (s)", y = "kw"),
                line_shape='spline',
                markers= True,
                width=1200)


    graphJSON3 = json.dumps(fig, cls=plotly.utils.PlotlyJSONEncoder)



    Powr_flow_bar = {
                'Power generated by the Genset.': P_list,
                "Power from Genset to load.": P_load_list,
                "Power from BSS to load.": P_bss_list,
                "Load requirements.": L_list,
                "Power from RES.": P_res_list
   }


    bar_df = pd.DataFrame(Powr_flow_bar)
    fig = px.bar(bar_df, x = V_steps,
                y = bar_df.columns,
                barmode = 'group',
                width = 1500,
                title = '________Power Flow._________',
```

```
                            labels = dict(x = "Time Step (s)", y = "kw"), text_auto = True)


                graphJSON4 = json.dumps(fig, cls=plotly.utils.PlotlyJSONEncoder)



                return render_template('Power_Flow.html', graphJSON3=graphJSON3,
graphJSON4=graphJSON4)

            app.run(port=8080
```

11.5 Flask App front-end development

```
<head><meta http-equiv="refresh" content="10"></head>
<!doctype html>
<html>

    <style>
        * {
                -webkit-box-sizing: border-box;
                box-sizing: border-box;
        }
        body {
                background-color: #272b34;
                font-family: 'Khula', sans-serif;
                font-weight: 300;
                color: white;
                line-height: 1em;
                margin: 0;
            padding: 2em 1em;
        }

        script.chart3 {
            float: right;
            align-content: center;
            padding: 5px;
            height: 200px;
            width: 500px;
            display: flex;
            flex-wrap: wrap;

        }

        script.chart4 {
            float: left;
            align-content: center;
            padding: 5px;
            height: 200px;
```

```
        width: 500px;
        display: flex;
        flex-wrap: wrap;

    }

    div.chart3 {
        size: 3cm;
        align-self: center;
    }
    div.chart4 {
        size: 3cm;
    }
    body.header {
        font-size: large;
        align-items: center;
    }
</style>


<body>
    <header>Meteoria Energy Dashbord</header>

    <h1 class="chart3">Power Flow</h1>
    <div id='chart3' class='chart3'"></div>
    <h1 class="chart4">Generation / Consumption</h1>
    <div id='chart4' class='chart4'"></div>
</body>


<script src='https://cdn.plot.ly/plotly-latest.min.js'></script>
<script type='text/javascript'>
    var graph3 = {{ graphJSON3 | safe }};
    Plotly.plot('chart3',graph3,{});
</script>


<script src='https://cdn.plot.ly/plotly-latest.min.js'></script>
<script type='text/javascript'>
```

```
        var graph4 = {{ graphJSON4 | safe }};
        Plotly.plot('chart4',graph4,{});
    </script>


</html>
```

*Source code and data may be subject to update.