UNIVERSITY OF PISA

Master's degree in Computer Engineering

882II Internet of Things

SmartBreeze

Professors:

Giuseppe Anastasi

Francesca Righetti

Carlo Vallati

Student:

Tommaso Califano

ACADEMIC YEAR 2023/2024

Contents

1 Introduction **2**

2 Use Case: Cooling a Datacenter **3**
2.1 System Overview 3
2.1.1 Energy Area 3
2.1.2 Climate Area 4
2.2 System workflow 4

3 System Implementation **5**
3.1 System Components and Interaction 5
3.1.1 Energy Manager 6
3.1.2 Climate Manager 6
3.1.3 Border Router 8
3.1.4 CoAP Server 8
3.1.5 User Application 8
3.1.6 Database 8

4 Data Encoding **9**
4.1 JSON and SenML 9
4.2 SenML Structure 9
4.3 Floating point values 10

5 Machine Learning Model **11**
5.1 Model Training and Integration 11
5.1.1 Dataset features 11
5.1.2 Preprocessing 11
5.1.3 Training 12
5.1.4 Node Deployment 13

6 MySQL Database Schema **14**
6.1 Table: nodes 14
6.2 Table: solar_production 14
6.3 Table: temperature 15

7 Grafana Dashboard **16**

8 Conclusions **18**

Chapter 1

Introduction

The rapid growth of data centers, driven by the increasing demand for cloud computing, big data, and digital services, has led to significant challenges in managing energy consumption and maintaining optimal operational efficiency. In particular in a industry that require more and more energy to function the necessity of new type of power sources that are less costly is crucial.

The renewable energy source are the apparent solution to this problem; however the efficiency of the power generation is low and often linked to time and meteorological phenomenon. So a proper power plant require at least a battery for accumulating energy whenever is not needed. This approach is simple but does not fully take advantage of the energy production due to the capacity limit of the batteries and the passive loss of the accumulation. The ideal situation will be produce and consume on site the renewable energy production.

SmartBreeze wants to mitigate this problem adapting the energy consumption to the renewable variability production, minimizing the energy loss and even eliminate the necessity of a battery in a power plant system guaranteed that the first energy that will be consumed will be the renewable one.

To achieve this the system want to regulate the cooling process of a data center activating it only when it is necessary, guaranteeing the desired temperature.

Chapter 2

Use Case: Cooling a Datacenter

The SmartBreeze system (Figure 2.1) is designed for adapting the energy consumption of the cooling system to the power production of a renewable energy source. In this case is taken into account the requirement of cooling a data center using the minimum energy possible.

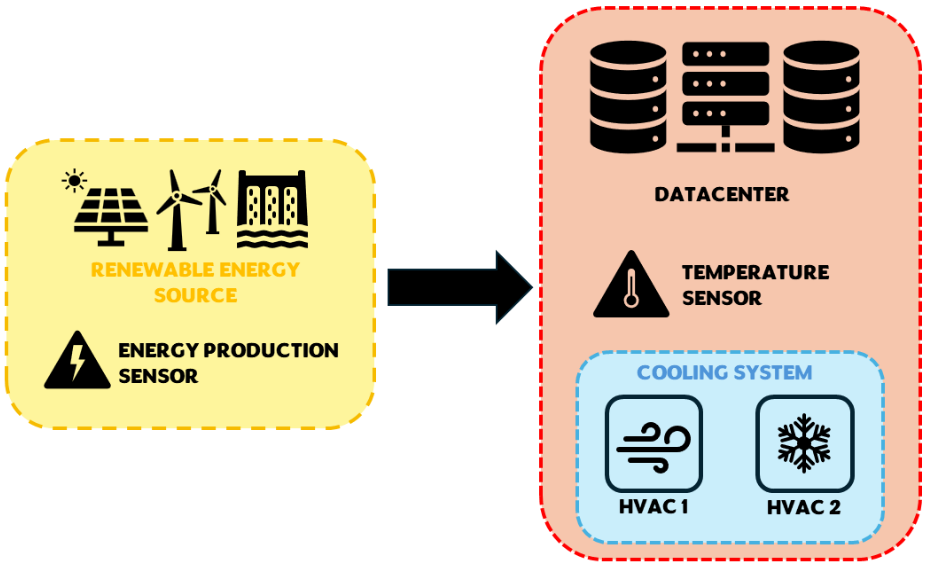

Figure 2.1: A simple scheme of the SmartBreeze system

2.1 System Overview

The system is divided in two major areas: Energy Area and Climate Area.

2.1.1 Energy Area

This area is composed by the **energy source** and a **energy sensor**. Thanks to the versatile of the system, the first can be even a fully functioning power plant. The latter instead is required for tracking the evolution of the production during time, this data will be used by a machine learning model to predict the future production. In this way will be possible to active the cooling system only in periods when the energy production is sufficient to sustain it.

2.1.2 Climate Area

This is the area where is installed the main infrastructure to cool down. This will produce heat that must be dissipated through the **cooling system**. In particular the system must be composed by two HVAC (Heating, Ventilation and Air Conditioning) that has different consumption and cooling efficiency:

- **HVAC 1** is less power hungry with limited cooling capacity; it can be mainly a ventilation system.
- **HVAC 2** is powerful cooling system with the disadvantage of a notably high energy consumption.

Based on the prediction of the energy production and the temperature tracking of the **temperature sensor** the cooling system will be automatically activated only when it is strictly necessary and its energy request is fulfilled.

2.2 System workflow

To sum up the system operation is cyclic and in every cycle it execute the following steps:

1. The energy production is sampled by the relative sensor.
2. The machine learning model produce a prediction of the future production and sent to the climate area.
3. The temperature is sampled by the relative sensor.
4. Using the prediction and the sampled temperature it is decided if turn on one of the two HVAC up to the next operation cycle.

Chapter 3

System Implementation

The SmartBreeze system (Figure 3.1) employs a simple wireless sensor network (WSN) to minimize the electric consumption of the cooling system still maintaining the requested temperature level. In this chapter is analyzed every single part of the system and its implementation.

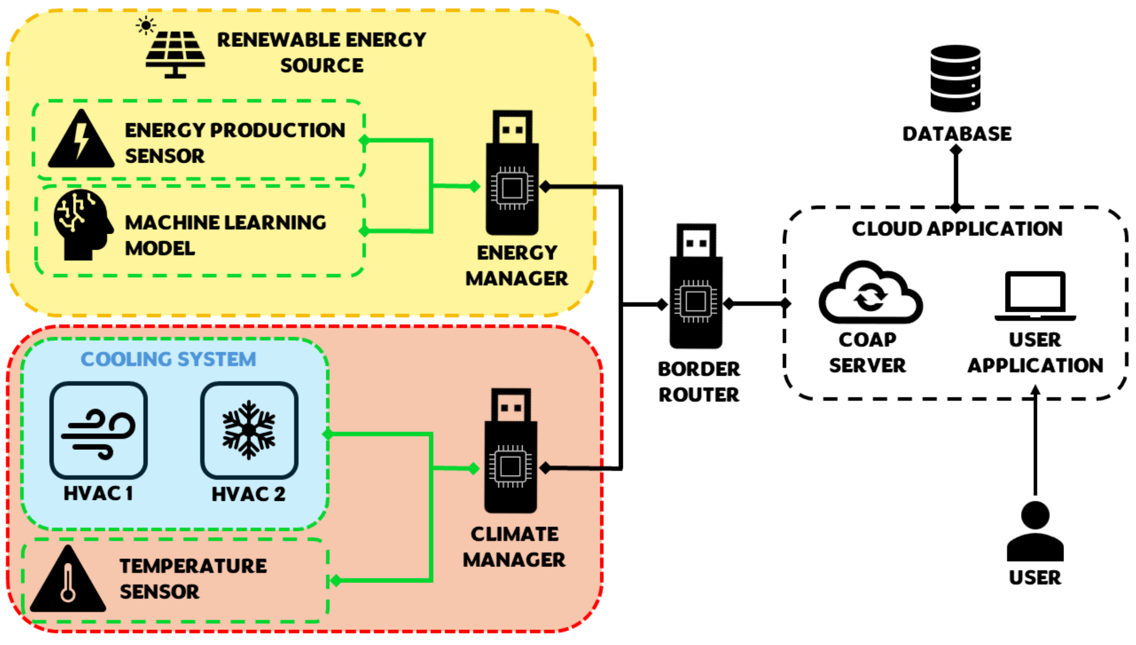

Figure 3.1: SmartBreeze system architecture that highlight the use of the nodes.

3.1 System Components and Interaction

Given the previous system division the role assignment of the nodes is clear. For each area there is one node that handle it, in addiction to that there is the border router that guarantee the communication from WSN to the Internet.

Every manager node has a personal exposed resource named *settings*, this resource is used by the user application for changing the parameters of the nodes.

3.1.1 Energy Manager

- **Resource Exposed:** `/solar_energy`, `/settings`.
- **Function:** Sample the energy production and compute the prediction.
- **Observations:** Does not observe any other resource.

This node is essentially a sensor since it does not take any action on the environment, however it has the assignment to establish when a new cycle of operation start. The operations of the node are divided in the following phases, each phase is described by a LED colour:

1. **Configuration Phase**: (YELLOW)
   This phase happen only once after the device is turned on. It is composed by:
   (a) **Main Resource Activation:** `/solar_energy` is activated for permitting others node to subscribe to it.
   (b) **Clock request:** the node make a request to the CoAP Server to retrieve the exact time. This is necessary for the operation of the **machine learning model** and the correct format of the data exchange (this will be discussed in the next chapter).
   (c) **CoAP Server registration:** the node register to the server, sending its initial settings and information.
   (d) **Secondary Resource Activation:** `/settings` is activated for permitting the user application to modify it.
   (e) **Initialize the timer:** `/settings` the time is set for the cyclic operation of the next phase.
2. **Operation Phase** (GREEN)
   This phase is called every time the timer expires, given that this is the operation at time t_i:
   (a) **Energy sampling:** thanks to the apposite sensor the sample En_i is generated.
   (b) **Energy prediction:** the node compute the timestamp t_{i+1} and compute energy prediction P_{i+1} using the machine learning model (MLM):

$$P_{i+1} = MLM(t_{i+1}) + CorrectionFactor_i \tag{3.1}$$

$$CorrectionFactor_i = (En_i - P_i) * 0.12 + CorrectionFactor_{i-1} * 0.88 \tag{3.2}$$

   (c) **Notify the observers:** notifying the observers of the `/solar_energy` resource.
3. **Sleep Phase** (RED)
   In this phase the node is waiting for a new event.

3.1.2 Climate Manager

- **Resource Exposed:** `/temperature_HVAC`, `/settings`.
- **Function:** Sample the temperature and manage the cooling system.

- **Observations:** `/solar_energy`.

This node is both a sensor, thanks to the temperature sensor, and an actuator, thanks to the role of switching the state of the HVAC. Indeed the colour of the LED mainly identifies which HVAC is operating:

- **RED:** the cooling system is off.
- **BLUE:** the HVAC 1 is active.
- **GREEN:** the HVAC 2 is active.

The other association with LED's color are express in the following phases:

1. **Configuration Phase** (YELLOW)
   This phase happen only once after the device is turned on. It is composed by:
   (a) **Main Resource Activation:** `/temperature_HVAC` is activated for permitting others node to subscribe to it.
   (b) **Clock request:** the node make a request to the CoAP Server to retrieve the exact time. Although this is not necessary for the execution of the main operation, because it will be used the timestamp received by the *Energy Manager*, it was decided to maintain the consistency of the messages; even in the first once that will arrive to the CoAP server.
   (c) **CoAP Server registration:** the node register to the server, sending its initial settings and information.
   (d) **CoAP Discovery and Subscribing:** the node ask to the CoAP server the ip of the *Energy Manager* for subscribing to the `/solar_energy` resource.
   (e) **Secondary Resource Activation:** `/settings` is activated for permitting the user application to modify it.
2. **Operation Phase** (HVAC color)
   On the contrary of the *Energy Manager* this phase is called only when arrive a notification from the `/solar_energy` resource. This choice is made for guaranteeing the synchronization of the two nodes. It is composed by:
   (a) **Energy Data Reception:** with the notification the new data is received, in particular the sampled and predicted energy production values.
   (b) **Temperature sampling:** thanks to the apposite sensor a new sample is generated.
   (c) **Managing the cooling System:** using the temperature and the predicted energy it is decide the new state of the cooling system (for the logic see the next subparagraph).
   (d) **Notify the observers:** notifying the observers of the `/temperature_HVAC` resource.
3. **Manual Phase** (BLUE $\rightarrow$ HVAC color)
   This phase is triggered when the button is pressed (the led will turn BLUE for a moment to signal that):
   (a) **Energy Data Reception:** with the notification the new data is received, in particular the sampled and predicted energy production values.

(b) **Temperature sampling:** thanks to the apposite sensor a new sample is generated.
(c) **Managing the cooling System:** using the temperature and the predicted energy it is decided the new state of the cooling system. The latter is activated only when there is sufficient solar power to maintain it or the temperature is near the max temperature set by the user. The system will tend to maintain the lower temperature possible exploiting the HVAC only when it could afford them.
(d) **Notify the observers:** notifying the observers of the `/temperature_HVAC` resource.

4. **Sleep Phase** (HVAC color)
   In this phase the node is waiting for a new event.

3.1.3 Border Router

- **Function:** Provides connectivity between the WSN and the internet.

The border router facilitates remote access and control of the system, enabling seamless communication between local sensors and cloud-based applications.

3.1.4 CoAP Server

- **Resource Exposed:** `/clock`, `/discovery`, `/registration`.
- **Function:** manage node registration and the insertion of new resource value in the database.
- **Observations:** `/solar_energy`, `/temperature_HVAC`.

The CoAP Server handles the initial registration of the nodes as they join the network, furthermore collects and stores data from temperature and energy production in a MySQL database. The latter is achieved with observing each main resource of a new registered node.

3.1.5 User Application

- **Function:** allows the dynamic control of some parameters of the nodes and its listing.
- **Observations:** Does not observe any other resource.

the user application implements two essential functions: list all registered nodes and dynamically modify some parameters of the nodes, such as the sampling interval, the max desired temperature...

3.1.6 Database

- **Function:** storing all the collected data.

The database is implemented in mySQL.

Chapter 4

Data Encoding

In the SmartBreeze system, it is used JSON (JavaScript Object Notation) for data encoding, this choice is driven by its simplicity, human readability, and widespread adoption, making it an ideal format for data interchange in IoT environments. Moreover, for the the sensors measurements, it adopt the SenML (Sensor Measurement Lists) standard to ensure interoperability and seamless integration with third-party applications.

4.1 JSON and SenML

The choice of JSON data encoding is justified by three aspects:

1. **Simplicity:** the JSON format is simple to generate and easy to read for a human operator; this makes the perfect choice for an IoT environment that needs continuos monitoring and debugging.
2. **Lightweight:** in an IoT environment where the computational resources are limited having a lightweight structure increase efficiency and ensures the minimum resource utilization.
3. **Globally used:** thanks to the widespread diffusion, using the JSON format facilitate the integration of the system with other third-party application.

On top of that the adoption of a more specific format like the **SenML** standard is taken for extending further the third point: indeed the sensors programmed in the SmartBreeze system can be utilized in a broader context thanks to this choice.

4.2 SenML Structure

SenML format is divided in two type of field: base and regular.

- <u>Base fields</u>
  - **Base Name (bn):** this is a string that is prepended to the names found in the entries.
  - **Base Unit (bu):** a base unit that is assumed for all entries, unless otherwise indicated. If a record does not contain a Unit value, then the Base Unit is used. Otherwise, the value found in the Unit (if any) is used.

- **Regular fields**
  - **Name(n):** name of the sensor or parameter. When appended to the Base Name field, this must result in a globally unique identifier for the resource.
  - **Time(t):** time when the value was recorded.
  - **Unit (u):** unit for a measurement value.
  - **Value (v):** values are represented using basic data types:
    - v: floating point numbers.
    - vd: integer value.
    - vb: boolean value.
    - vs: string.

An example of a *Energy Manager* SenML packet is the following:

```
[
    {"bn":"urn:dev:mac:7E4B5A3D2C1F","bu":"W"},
    {"n":"predicted","u":"W","t":"2024-07-13T19:44Z","v":899939},
    {"n":"sampled","u":"W","t":"2024-07-13T18:44Z","v":0}
]
```

4.3 Floating point values

Due to the limitation of the NRF52840 microcontrollers is not possible to print anywhere any floating point, despite the dongle support the floating point operations. To resolve this it is decided to multiply every float number by 10000 and to convert it in a integer right before the data is sent. The cloud application reconvert it to float after the reception. This give space for only four decimal numbers, but they were considered sufficient for the application. So for example a 90 W of production will be sent as 900000.

Chapter 5

Machine Learning Model

The SmartBreeze system utilize a machine learning model for predicting the solar production during the year; this ensure a more reactive and adaptive system. The chosen model is the **Random Forest Regressor**, known for its robustness and accuracy, and it is implemented within the *Climate Manager node.*

5.1 Model Training and Integration

The *Climate Manager node* uses the real time to predict the solar production of a photovoltaic plant during an entire year with a hour precision. As it has seen this is the limitation of the application that can be easily solved with the adopt of a lager dataset in the model training. In reality the application can function properly even in the order of minutes, due to the fact that the machine learning model produces the same value for every minute between two consecutive hours; resulting in a step behavior. This is exploited for testing the system, indeed through a preprocessor constant called *DEBUG* is possible to force the system to function with a smaller sampling period, maintaining an hourly advancing of time. This function is active by default and must be disabled from the *setting.c* energy manager resource file, *energy_manager.c* and *climate_manager.c* files.

The dataset that has been utilized is the "solar-powe-generation-data dataset" from Kaggle.

5.1.1 Dataset features

The chosen dataset contains 8760 records of environmental sensor data, including wind speed, air pressure, radiation, air temperature... All the data is taken from a solar power plant to see possible implementation of solar production prediction with machine learning. For the scope of this project the only feature required is the time (excluding the solar production, obviously), so the others features can be dropped.

5.1.2 Preprocessing

As stated before not all the feature are necessary to the scope of the application, so in this phase the irrelevant once's are removed. Moreover, all the records that have a 0 value on the solar production are removed in this way the regressor can learn only the significant variation of day production. In the next section there will be a verification of the behavior of the model during night time.

```python
# Load the dataset
data = pd.read_csv('./Solar Power Plant Data.csv')

# Drop the columns that are not required
columns_to_keep = ['Date-Hour(NMT)', 'SystemProduction']
columns_to_drop = [col for col in data.columns if col not in
    columns_to_keep]
data = data.drop(columns_to_drop, axis=1)

# Reformatting the date column, taking only the hours
data['Date-Hour(NMT)'] = pd.to_datetime(data['Date-Hour(NMT)'], format=
    '%d.%m.%Y-%H:%M')
data['Month'] = data['Date-Hour(NMT)'].dt.month
data['Day'] = data['Date-Hour(NMT)'].dt.day
data['Hour'] = data['Date-Hour(NMT)'].dt.hour

#  Removing the data points where the system production is zero
data = data[data['Hour'].between(6, 22)]
data = data[data['SystemProduction'] != 0]
```

5.1.3 Training

The training of the Random Forest Regressor involves four key steps:

1. **Data preparation:** the features (X) and labels (y) are defined.
2. **Dataset Splitting:** The data is divided into training and testing sets to ensure the model's performance can be evaluated accurately.
3. **Standardization:** The data is standardized for a better performance, the parameters used for the standardization will be integrated in the node's code as a preprocessor variable.
4. **Model training:** The Random Forest Regressor is trained on the training set, learning the relationships between the solar production and the progressing of time.

```python
from sklearn.model_selection import train_test_split
from sklearn.ensemble import RandomForestRegressor
from sklearn.metrics import mean_squared_error, r2_score
from sklearn.preprocessing import StandardScaler

# Prepare the data for modeling
x = data[['Month', 'Day', 'Hour']]
y = data['SystemProduction']

# Split the data into training and testing sets
x_train, x_test, y_train, y_test = train_test_split(x, y, test_size
    =0.2, random_state=42)
```

```

# Standardize the features
scaler = StandardScaler()
x_train_scaled = scaler.fit_transform(x_train)
x_test_scaled = scaler.transform(x_test)
print(scaler.mean_, scaler.scale_)

# Initialize and train the RandomForestRegressor
model = RandomForestRegressor(n_estimators=20, random_state=42)
model.fit(x_train_scaled, y_train)

# Predict on the test set
y_pred = model.predict(x_test_scaled)

# Evaluate the model
mse = mean_squared_error(y_test, y_pred)
r2 = r2_score(y_test, y_pred)

print(f"Mean Squared Error: {mse:.2f}")
print(f"R^2 Score: {r2:.2f}")
```

5.1.4 Node Deployment

The model can now be converted through *emlearn* in efficent C code, allowing it to run effectively on the node's hardware.

```
import emlearn

#Convert the model to C code for use in IoT devices
path = './solar_prediction.h'
cmodel = emlearn.convert(model, method='inline')
cmodel.save(file=path, name='solar_prediction')

print('Wrote model to', path)
```

Chapter 6

MySQL Database Schema

The SmartBreeze system uses a MySQL database to store sensor data and actuator statuses. This database is critical for maintaining historical records and guarantee a dynamic function of the system. In this chapter will be described the key schema of the key tables in the database.

6.1 Table: nodes

In this table are registered all the nodes with the correlated information.

Field	**Type**	**Null**	**Key**
ip	varchar(255)	NO	PRI
name	varchar(255)	NO	
resource	varchar(255)	NO	
settings	varchar(255)	NO	

Table 6.1: Nodes table schema.

As it is possible to see from Table 6.1 the fields are:

- **ip:** the ip of the node.
- **name:** the name of the node.
- **resouce:** an array of the exposed resources of the node.
- **settings:** a json formatted string that contains all the configurable parameters.

6.2 Table: solar_production

In this table are collected all the energy production sample and the relative prediction. In particular given a sample E_i at the time instant i, in a respective prediction P_{i-1} will be memorized in the same row to facilitate the comparison of the two values.

As it is possible to see from Table 6.2 the fields are:

- **timestamp:** time instant where the energy was sampled.
- **sampled:** the value of the sampled energy production.

Field	Type	Default	Null	Key
timestamp	TIMESTAMP	current_timestamp		PRI
sampled	FLOAT	NULL	YES	
predicted	FLOAT	NULL	YES	

Table 6.2: Solar production table schema.

- **predicted:** the value of the predicted energy production for the same timestamp.

6.3 Table: temperature

In this table are collected all temperature sample and the status of the cooling system in that time instant.

Field	Type	Default	Null	Key
timestamp	TIMESTAMP	current_timestamp		PRI
temperature	FLOAT		NO	
active_HVAC	TINYINT(1)		NO	

Table 6.3: Solar production table schema.

As it is possible to see from Table 6.3 the fields are:

- **timestamp:** time instant where the temperature was sampled.
- **temperature:** the value of the temperature in that instant.
- **active_HVAC:** indicate which of the two HVAC is active:
  - **OFF**: 0.
  - **HVAC 1**: 1.
  - **HVAC 2**: 2.

Chapter 7

Grafana Dashboard

To effectively monitor and analyze the evolution of the system, it is implemented a web interface using Grafana for data visualization. Below are some screenshots demonstrating the real-time dashboards and historical data analysis capabilities provided by Grafana. It must be remembered that all the following data has been generated through a raw sensor simulation, for this reason it can be seen a significant difference between the sampled and predicted energy.

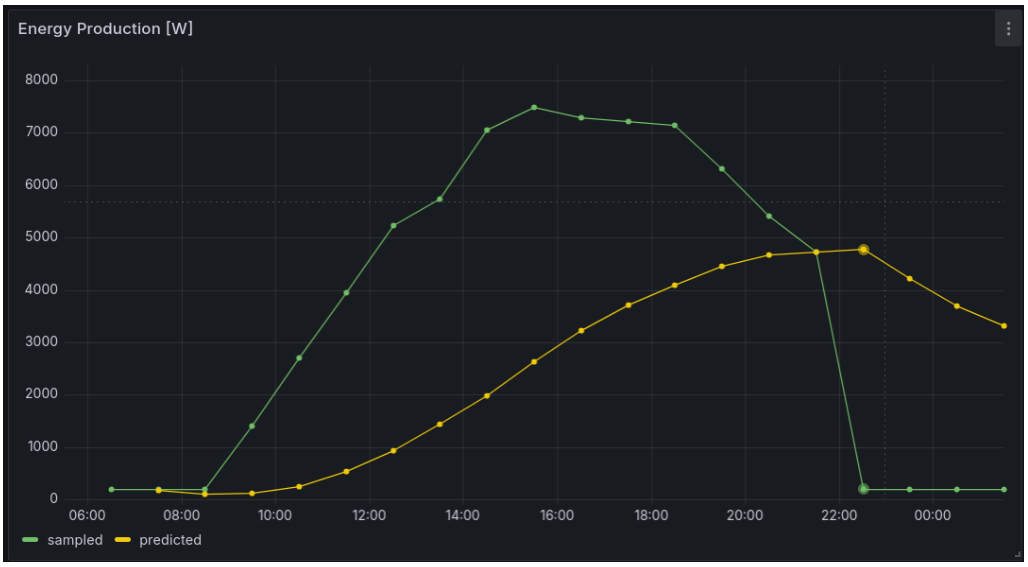

Figure 7.1: The sampled energy (in green) compared to the predicted value (yellow) during time

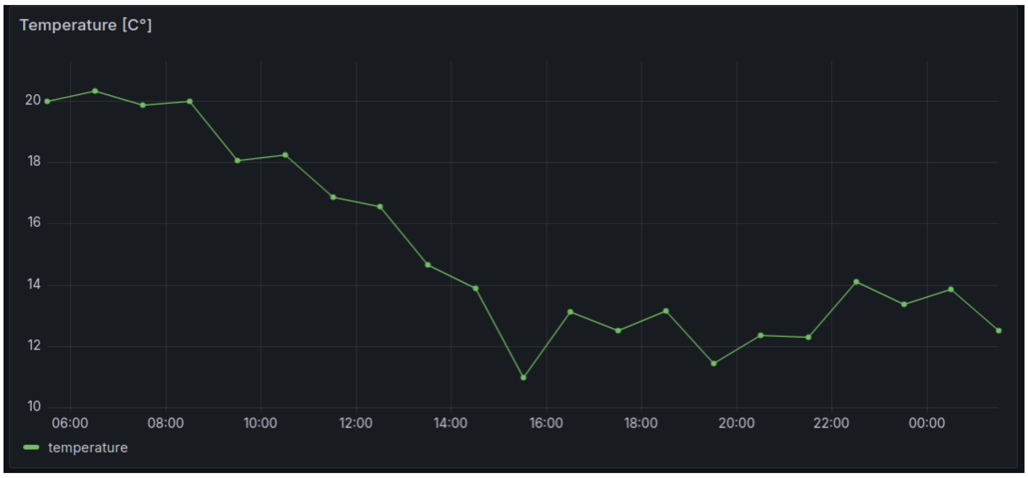

Figure 7.2: Temperature readings over time, showing a detailed trend of temperature changes.

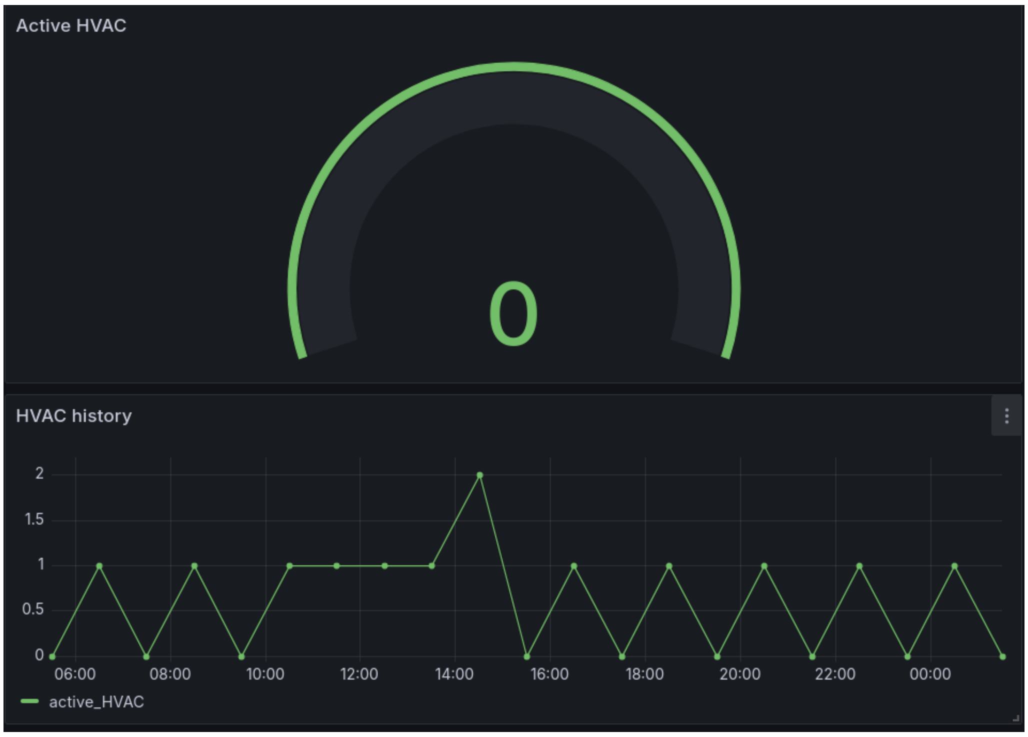

Figure 7.3: The cooling system status history, indicating the operational state of the HVACs system and which is active now.

Chapter 8

Conclusions

The SmartBreeze system has been successfully implemented with a functional regression model capable of predict the solar production. With this system it has been achieved a better energy utilization in a cooling setting.

Effective as it is, the current system could be improved and expanded. A future version could use more sensor and actuator to expand the energy sources and the possible energy consumers. It will even possible to integrate a battery system and try to minimize the charging/discharging cycle.

Having more sensor could consent to collect meteorological data for exploiting a more sophisticated machine learning model capable to predict the solar production more precisely; guaranteeing a more versatile version of the system.

In conclusion, SmartBreeze system offers a robust solution for the energy management and cooling of a data center, or whatever industrial structure have this necessity, with significant potential for future enhancements to further advance the smart grid industry.