SENSE

SENSE: Semantic-based Explanation of Cyber-physical Systems

Deliverable 6.2:
Results of PoC testing in the smart buildings domain

Authors	:	Christoph Moser, Dagmar Jähnig
Dissemination Level	:	Public
Due date of deliverable	:	31.3.2025
Actual submission date	:	31.7.2025
Work Package	:	WP6
Type	:	Report
Version	:	1.0

Abstract

Deliverable D6.2 reports on the technical evaluation of the SENSE system in the smart building domain. A proof-of-concept (PoC) was implemented to demonstrate how semantic-based explanations can support the detection of anomalous events in building operation. The initial setup focused on one room and a single event (excessive heating demand), which was further extended to two rooms and additional scenarios.

Based on this initial setup and a series of additional tests, the scalability, extensibility and resilience of the SENSE system for smart buildings have been evaluated. In addition, key requirements for the SENSE system from the point of view of a facility manager have been identified.

History

Version	Date	Reason	Revised by
0.5	22.06.2023	Initial draft	DJ
1.0	10.9.2025	Final version	DJ, CM

Author List

Project Partner	Name (Initial)	Contact Information
AEE INTEC	Dagmar Jähnig (DJ)	d.jaehnig@aee.at
AEE INTEC	Christoph Moser (CM)	c.moser@aee.at

Executive Summary

The SENSE project aims to explain events occurring in technical systems regarding the area of Smart Grid and Smart Buildings. The goal is to contribute to Austria's sustainability goals by making complex systems that underlie key (and often highly polluting) infrastructures more efficient and user-friendly through explanations of (anomalous) events occurring in those systems. The SENSE system developed in this project aims to make complex cyber-physical systems (CPS) more transparent and thereby improve the performance and user acceptance of such systems.

This deliverable reports on the technical evaluation of the SENSE system in the smart building domain. A proof-of-concept (PoC) was implemented to demonstrate how semantic-based explanations can support the detection of anomalous events in building operation. This initial setup focused on one room and a single event (excessive heating demand), which was further extended to two rooms and additional scenarios.

The evaluation showed that the SENSE system is capable of reliably detecting anomalies such as open windows or sensor malfunctions and providing explanations that correspond to expert assessments. Usability aspects from a facility manager's perspective highlighted the potential of integrating explanations into daily building operation, while underlining the need for automation and simple interfaces.

Scalability tests indicated that extending the system to more rooms is technically feasible but currently requires significant manual effort. Steps such as data preparation, semantic modeling, and database configuration are time-consuming and prone to errors. However, once standardized data formats and automated workflows are introduced, the implementation time can be significantly reduced. It is estimated that a whole building could be set up within one day under optimized conditions.

Extensibility tests demonstrated that the system can be adapted to handle additional events, but that creating robust explanatory logic is complex due to the variety of influencing factors (building physics, user behavior, weather conditions). Additional sensor data or simulation-based reference models could further improve the accuracy of explanations.

Resilience testing confirmed that the system can handle temporary outages and out-of-range values but struggles with missing or invalid data formats. This points to the need for improved error-handling mechanisms in real-world applications.

From the facility management perspective, three key requirements were identified: (1) automated implementation to reduce manual setup, (2) reliance on existing sensor infrastructure without costly new installations, and (3) inclusion of statistical or simulation-based analysis to benchmark building behavior.

The SENSE system provides a promising approach to increasing transparency in smart building operation by combining event detection with semantic explanations. The PoC results demonstrate both the feasibility and the challenges of scaling such a system to real-world building environments. Future work will focus on automation, robustness, and integration

into facility management workflows, thereby contributing to energy efficiency and sustainability goals.

More detailed information on the PoC definitions, the SENSE architecture and the SENSE technology stack implementation can be found in the respective deliverables ([1], [2], [3] and [4]).

Table of Content

List of Figures

1 Introduction

The SENSE project aims to explain events occurring in technical systems regarding the area of Smart Grid and Smart Buildings. In the case of smart buildings, technical systems can be quite complex consisting of many different systems such as heating, cooling, ventilation and air-conditioning systems as well as shading systems for the windows.

This deliverable reports on the technical evaluation of the SENSE system in the smart building domain. A proof-of-concept (PoC) was implemented to demonstrate how semantic-based explanations can support the detection of anomalous events in building operation. The initial setup focused on one room and a single event (excessive heating demand), which was further extended to two rooms and additional scenarios to test the workflow and the scalability.

In the following chapters, the status of the developed system to explain events in cyber-physical systems is evaluated for the case of smart buildings.

In chapter 2, the implemented PoC is evaluated. Chapter 3 is dedicated to the question of scalability. The PoC has been extended to more rooms and more sensors also giving an estimation of the effort that would be needed for a large office building.

In chapter 4, the extensibility of the system to different explanations and different events to be explained is discussed. Resilience tests against sensor errors have been executed and are described in chapter 5.

Finally in chapter 6, the usability of the system from the perspective of a user (the facility manager) is examined and key requirements for the application of the system in a real-world environment are identified.

2 Evaluation of the Implemented PoC

Because buildings are very complex and are influenced by the environmental conditions as well as the user behavior, the SENSE system has been applied to a simple case as a proof of concept (PoC). The PoC consisting of one meeting room and one event to detect has been implemented in the SENSE system. The event to be detected is the fact that in the room significantly more heating energy is consumed than usual. In parallel, two different types of events were detected that can be possible explanations for too much energy being consumed in that room.

- A window has been left open. This can be detected by means of a window contact. But this is only considered an event if the window stays open for more than two hours. Opening the window for shorter periods for ventilation purposes should not trigger an event.
- The temperature sensor in the room that is used to control the heating system is broken. It shows a constant value over a long period of time (4 hours). In normal operation, temperatures never stay exactly constant, but they vary slightly.

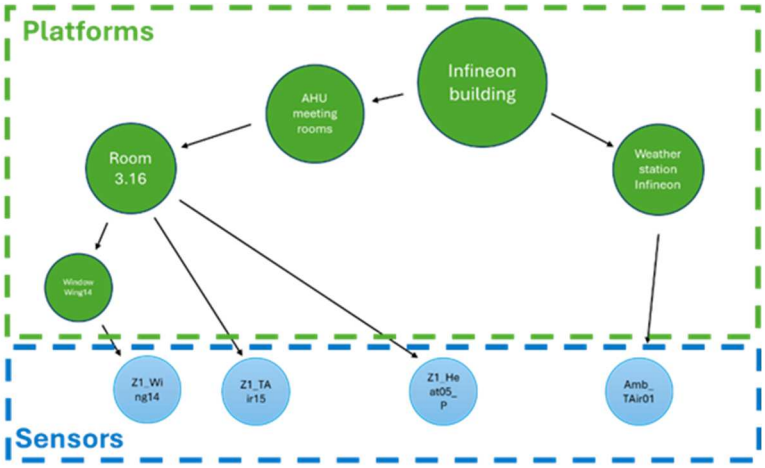

Figure 1: Schematic of the sensors and platforms that exist in the PoC Smart Building

2.1 General Performance of the Event Detection

For the development of the SENSE stack in the smart-building use case, a three-months period of data from SmartBuilding was thoroughly analyzed regarding the events mentioned above by domain experts. All events in the three-month period were labeled manually.

The first step of the evaluation was therefore to verify that all the events which were detected via this manual inspection were also detected by the SENSE system

The SENSE system successfully identified most events that had been detected during manual inspection, including cases of excessive heating demand and open windows. The detection logic proved to be robust for the defined use case, although limitations were observed in cases where sensor data were incomplete or ambiguous. Overall, the results demonstrate

that the system can provide reliable event detection in a real building environment, thereby validating the chosen methodological approach.

2.2 General Performance of the Chatbot to Explain Events

The chatbot developed within the SENSE framework was evaluated regarding its ability to provide meaningful explanations of detected events. The explanations generated by the system were compared with those of domain experts. The results show that the chatbot can reproduce expert-level reasoning for simple cases and helps to bridge the gap between technical data analysis and user understanding. This functionality is essential for supporting facility managers, as it enables them to quickly access explanations in natural language without requiring detailed technical knowledge of the underlying data processing or semantic models. The chatbot therefore contributes significantly to the usability and acceptance of the SENSE system.

However, in the current state of development, the user must ask the chatbot pre-defined questions. Changing the wording a little bit can lead to wrong answers. Also, when asking for events in a certain period of time rather than just the last events, results were not always correct. Further development is needed to facilitate communication between the user and the chatbot.

2.3 Usability From a Facility Manager's Perspective

Facility managers are primarily focused on restoring normal building operation when faults occur. Their backgrounds are diverse but typically grounded in technical or craft-based training. While they are familiar with building systems and tools such as MS Office, their IT expertise is usually limited, and knowledge of semantic technologies is absent. Facility managers are accustomed to routinely filling out spreadsheets, which they perceive as a manageable task. However, they often lack sensitivity to the implications of incorrect data entries.

For this user group, the usability of the SENSE system is essential. Interfaces must remain simple and robust, with automated checks to minimize the impact of errors. Explanations should be provided in clear, non-technical language, enabling facility managers to benefit from the system without requiring advanced IT training.

3 Scalability

The proof-of-concept scenario described in the previous chapter was extended to two rooms to test the scalability of the system. The structure including the additional sensors and platforms is shown in Figure 2.

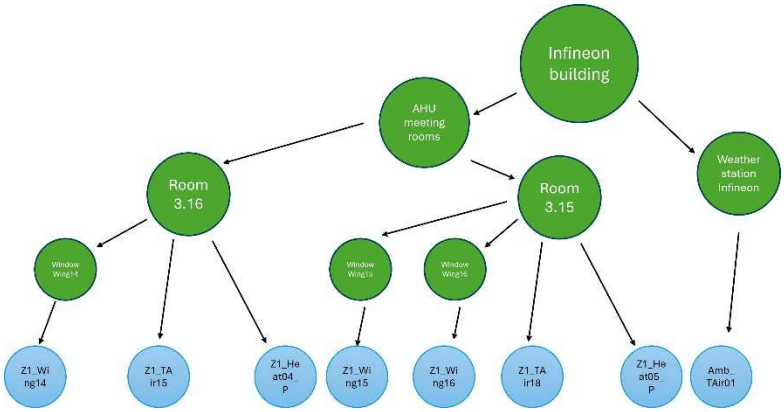

Figure 2: Schematic of the sensors and platforms that exist in the extended PoC Smart Building

3.1 Steps to add a room to the PoC

3.1.1 Step 1 – measurement data file (output_line_protocol.txt)

Get the measured values of the desired additional room, e.g. room 3.16. The room has one temperature sensor (Z1Tair18), two windows (Z1Wing15 and Z1Wing16) and a heat meter (Z1Heat05P). Create a new datafile "output_line_protocol.txt" starting with the indicator "smartbuilding" in the first column, the measured values and the unix timestamp in the last column. The expected duration to get the measured data, running a python script to create the described data formatting and to copy the new datafile to the smart building folder is estimated to 15 minutes. Figure 3 shows an excerpt of the new data file.

```
smartbuilding AmbTAir01=-0.267,Z1TAir15=23.037,Z1TAir18=23.014,Z1Wing14=0,Z1Wing15=0,Z1Wing16=0,Z1Heat04P=0,Z1Heat05P=0 1640995200000000000
smartbuilding AmbTAir01=-0.3,Z1TAir15=23.064,Z1TAir18=23.003,Z1Wing14=0,Z1Wing15=0,Z1Wing16=0,Z1Heat04P=0,Z1Heat05P=0 1640995260000000000
smartbuilding AmbTAir01=-0.367,Z1TAir15=23.049,Z1TAir18=22.981,Z1Wing14=0,Z1Wing15=0,Z1Wing16=0,Z1Heat04P=0,Z1Heat05P=0 1640995320000000000
smartbuilding AmbTAir01=-0.4,Z1TAir15=23.062,Z1TAir18=23.007,Z1Wing14=0,Z1Wing15=0,Z1Wing16=0,Z1Heat04P=0,Z1Heat05P=0 1640995380000000000
smartbuilding AmbTAir01=-0.4,Z1TAir15=23.085,Z1TAir18=22.978,Z1Wing14=0,Z1Wing15=0,Z1Wing16=0,Z1Heat04P=0,Z1Heat05P=0 1640995440000000000
smartbuilding AmbTAir01=-0.4,Z1TAir15=23.072,Z1TAir18=23.006,Z1Wing14=0,Z1Wing15=0,Z1Wing16=0,Z1Heat04P=0,Z1Heat05P=0 1640995500000000000
smartbuilding AmbTAir01=-0.4,Z1TAir15=23.074,Z1TAir18=23.001,Z1Wing14=0,Z1Wing15=0,Z1Wing16=0,Z1Heat04P=0,Z1Heat05P=0 1640995560000000000
smartbuilding AmbTAir01=-0.4,Z1TAir15=23.06,Z1TAir18=22.974,Z1Wing14=0,Z1Wing15=0,Z1Wing16=0,Z1Heat04P=0,Z1Heat05P=0 1640995620000000000
smartbuilding AmbTAir01=-0.4,Z1TAir15=23.037,Z1TAir18=22.986,Z1Wing14=0,Z1Wing15=0,Z1Wing16=0,Z1Heat04P=0,Z1Heat05P=0 1640995680000000000
smartbuilding AmbTAir01=-0.4,Z1TAir15=23.006,Z1TAir18=22.995,Z1Wing14=0,Z1Wing15=0,Z1Wing16=0,Z1Heat04P=0,Z1Heat05P=0 1640995740000000000
smartbuilding AmbTAir01=-0.4,Z1TAir15=22.996,Z1TAir18=23.002,Z1Wing14=0,Z1Wing15=0,Z1Wing16=0,Z1Heat04P=0,Z1Heat05P=0 1640995800000000000
smartbuilding AmbTAir01=-0.4,Z1TAir15=22.982,Z1TAir18=23.018,Z1Wing14=0,Z1Wing15=0,Z1Wing16=0,Z1Heat04P=0,Z1Heat05P=0 1640995860000000000
smartbuilding AmbTAir01=-0.4,Z1TAir15=22.977,Z1TAir18=22.974,Z1Wing14=0,Z1Wing15=0,Z1Wing16=0,Z1Heat04P=0,Z1Heat05P=0 1640995920000000000
smartbuilding AmbTAir01=-0.4,Z1TAir15=22.968,Z1TAir18=22.948,Z1Wing14=0,Z1Wing15=0,Z1Wing16=0,Z1Heat04P=0,Z1Heat05P=0 1640995980000000000
```

Figure 3: Excerpt of the modified datafile

3.1.2 Step 2 – Create a new semantic data file (system.ttl)

The additional room and the sensors names from the previous step have to be added to the spreadsheet SystemData.xlsx, in the Platforms sheet (Figure 4) and the Sensor sheet (Figure 5).

	A	B	C
1	**Platform**	**PlatformType**	**hostedBy_Platform**
2	Infineonbuilding	Building	
3	AHUmeetingrooms	AHU	Infineonbuilding
4	weatherStationInfineon	weatherStation	Infineonbuilding
5	Room3.16	Room	AHUmeetingrooms
6	Room3.15	Room	AHUmeetingrooms
7	windowWing14	Window	Room3.16
8	windowWing15	Window	Room3.15
9	windowWing16	Window	Room3.15
10			

Figure 4: Adjustment in Platform sheet (marked in grey colour)

	A	B	C	D	E
1	**Sensor**	**hostedBy_Platform**	**observes_ObservableProperty**	**TimeseriesId**	
2	Z1Heat04P	Room3.16	Heatradiatorroom	measurement=smartbuilding,field=Z1Heat04P	
3	Z1Heat05P	Room3.15	Heatradiatorroom	measurement=smartbuilding,field=Z1Heat05P	
4	AmbTAir01	weatherStationInfineon	AmbientTemperature	measurement=smartbuilding,field=AmbTAir01	
5	Z1Wing14	windowWing14	WindowState	measurement=smartbuilding,field=Z1Wing14	
6	Z1Wing15	Room3.15	RoomTemperature	measurement=smartbuilding,field=Z1Wing15	
7	Z1Wing16	Room3.15	RoomTemperature	measurement=smartbuilding,field=Z1Wing16	
8	Z1TAir15	Room3.16	RoomTemperature	measurement=smartbuilding,field=Z1TAir15	
9	Z1TAir18	Room3.15	RoomTemperature	measurement=smartbuilding,field=Z1TAir18	
10	DiffHeatingThreshold1	Room3.16	HeatDiffRadiator	measurement=smartbuilding,field=DiffHeatingThreshold1	=Z1Heat04P (4 hour average) - (-0,02 AmbTAir01(4 hour average +0,7))
11	DiffHeatingThreshold2	Room3.15	HeatDiffRadiator	measurement=smartbuilding,field=DiffHeatingThreshold2	=Z1Heat05P(4 hour average) - (-0,0288461 AmbTAir01(4 hour average +1,2692312))

Figure 5: Adjustment in Sensor sheet (marked in grey colour)

Move to the folder: smart-building/infrastructure/knowledgebase
Check that the Python-script "XLSXtoTTL.py" is available in this folder.
Open the virtual environment: smart-building/infrastructure/knowledgebase$ source .venv/bin/activate
Run the following command in the terminal:

```
python3 XLSXtoTTL.py "http://example.org/smartbuilding#" \
   ./SystemData.xlsx \
   ./data/system-data.ttl \
   --shacl-path ./shacl/additional-validation-rules.ttl \
   --shacl-reasoning-path ./shacl/additional-event-reasoning.ttl
```

The resulting message is shown in

```
(.venv) [redacted]:~/smart-building/infrastructure/knowledgebase$ python3 XLSXtoTTL.py "http://example.org/smartbuilding#" \
    ./SystemData.xlsx \
    ./data/system-data.ttl \
    --shacl-path ./shacl/additional-validation-rules.ttl \
    --shacl-reasoning-path ./shacl/additional-event-reasoning.ttl
The ObservableProperty HeatDiff_Radiator has not been defined as an Observable Property yet!
The Platform Type window has not been defined as a Platform Type yet!
Conforms: True
Results Graph:
@prefix sh: <http://www.w3.org/ns/shacl#> .
@prefix xsd: <http://www.w3.org/2001/XMLSchema#> .

[] a sh:ValidationReport ;
    sh:conforms true .


Results Text:
Validation Report
Conforms: True

Ontology serialized to ./data/system-data.ttl
```

Figure 6. Then close the virtual environment (cmd: deactivate).

```
(.venv) [redacted]:~/smart-building/infrastructure/knowledgebase$ python3 XLSXtoTTL.py "http://example.org/smartbuilding#" \
    ./SystemData.xlsx \
    ./data/system-data.ttl \
    --shacl-path ./shacl/additional-validation-rules.ttl \
    --shacl-reasoning-path ./shacl/additional-event-reasoning.ttl
The ObservableProperty HeatDiff_Radiator has not been defined as an Observable Property yet!
The Platform Type window has not been defined as a Platform Type yet!
Conforms: True
Results Graph:
@prefix sh: <http://www.w3.org/ns/shacl#> .
@prefix xsd: <http://www.w3.org/2001/XMLSchema#> .

[] a sh:ValidationReport ;
    sh:conforms true .


Results Text:
Validation Report
Conforms: True

Ontology serialized to ./data/system-data.ttl
```

Figure 6: Result from execution of python script

3.1.3 Step 3 – InfluxFB files

The first file 4hrmovingaverage.flux creates the 4-hour moving average values of the ambient air (AmbTair01) and the heating power (Z1Heat04P). Here the value of the heating power of the new room (Z1Heat05_P) is added.

```
from(bucket: "smartbuilding")
  |> range(start: 2022-01-01T00:00:00Z, stop: 2022-06-01T00:00:00Z)
  |> filter(fn: (r) => r["_measurement"] == "smartbuilding")
  |> filter(fn: (r) => r["_field"] == "AmbTAir01" or r["_field"] == "Z1Heat04P" or r["_field"] == "Z1Heat05P")
  |> aggregateWindow(every: 1m, fn: mean, createEmpty: false, period: 4h)
  |> map(fn: (r) => ({r with
      _time: r._time,          // Preserve _time for "to()"
      _value: r._value,        // Use the mean value from aggregateWindow
      _field: r._field + "_4hr_avg",
      _measurement: "smartbuilding",
  }) )
  |> to(
    bucket: "smartbuilding",
    org: "smartbuilding"
  )
```

Figure 7: Modified file - 4hrmovingaverage.flux

The second file VirtualSensorafter-import.virt.flux creates the virtual sensors for the heating system, which calculates the threshold value of a too high or too low heating power of the room. Therefore, the coefficients of the equations must be adapted for room 3.16. The resulting script is shown below.

```
threshold1 = from(bucket: "smartbuilding")
  |> range(start: 2022-02-01T00:00:00Z, stop: 2022-06-01T00:00:00Z)
  |> filter(fn: (r) => r._measurement == "smartbuilding" and (r._field == "AmbTAir01_4hr_avg" or r._field == "Z1Heat04P_4hr_avg"))
  |> pivot(rowKey:["_time"], columnKey: ["_field"], valueColumn: "_value")
  |> map(fn: (r) => ({
      _time: r._time,
      _value: r.Z1Heat04P_4hr_avg + 0.02 * r.AmbTAir01_4hr_avg - 0.7,
      _field: "DiffHeatingThreshold1",
      _measurement: "smartbuilding"
  }))

threshold2 = from(bucket: "smartbuilding")
  |> range(start: 2022-02-01T00:00:00Z, stop: 2022-06-01T00:00:00Z)
  |> filter(fn: (r) => r._measurement == "smartbuilding" and (r._field == "AmbTAir01_4hr_avg" or r._field == "Z1Heat05P_4hr_avg"))
  |> pivot(rowKey:["_time"], columnKey: ["_field"], valueColumn: "_value")
  |> map(fn: (r) => ({
      _time: r._time,
      _value: r.Z1Heat05P_4hr_avg + 0.0288461 * r.AmbTAir01_4hr_avg - 1.2692312,
      _field: "DiffHeatingThreshold2",
      _measurement: "smartbuilding"
  }))

union(tables: [threshold1, threshold2])
  |> to(bucket: "smartbuilding", org: "smartbuilding")
```

Figure 8: Modified file - VirtualSensorafter-import.virt.flux

3.1.4 Step 4 – Generate a new docker container

First the file "config.docker.json" must be modified to consider the increased measurement period starting from January till June. In Figure 9 an excerpt of this file, with the changed values, is shown.

```
        "data-ingestion": {
        "mode": "replay",
        "startAt": "2022-01-01T00:00:00",
        "stopAt": "2022-06-01T00:00:00",
        "stopAction": "repeat",
        "deltaInSeconds": 6000,
        "sleepInSeconds": 1
    },
```

Figure 9: Excerpt from config.docker.json

Move to the folder: smartbuilding (cd smart-building/)
And rebuild the container influxdb and influxdb-writer (sudo docker-compose build)

3.1.5 Step 5 – Changes in influxDB – database

After starting the SENSE system (sudo docker-compose up), the influxdb database (http://localhost:8086) is created and the modified scripts are executed. This might take some minutes until the 4-hour moving-average values (ending with "4hr_avg") and the threshold values (DiffHeatingThreshold) appear (Figure 10), as well as in Figure 11 which show the measured and calculated data.

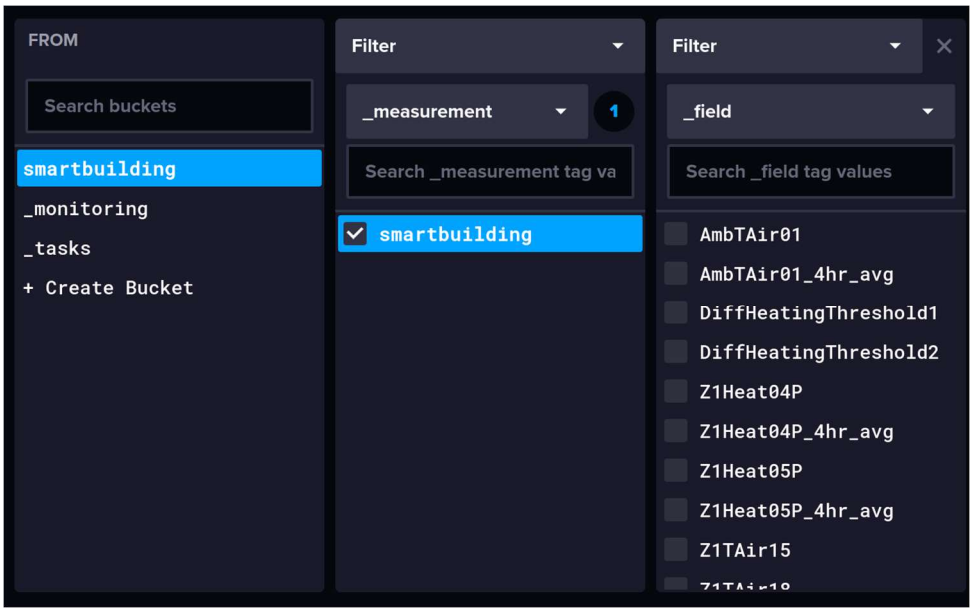

Figure 10: Variable list in influxdb

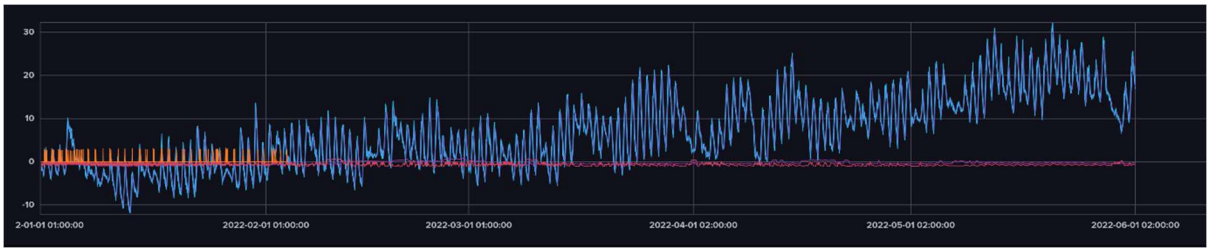

Figure 11: Measured and calculated values from January till June

<u>Open Window case:</u>

The open window case was evaluated in both rooms (Room3.15 & Room3.16)

- Wing14 (Room 3.16) 03.01.2022 08:00-12:00 → Recognised
- Wing 15 (Room 3.15) 03.01.2022 13:00-17:00 → Not Recognised

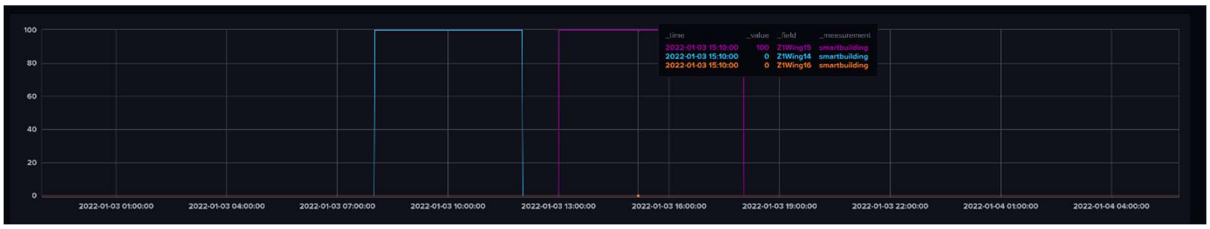

Figure 12: Open window case in January 2022

Figure 13 shows all window cases from January till June 2022.

- Wing 16 (Room 3.15) 09.02.2022-11.02.2022 → Not Recognised

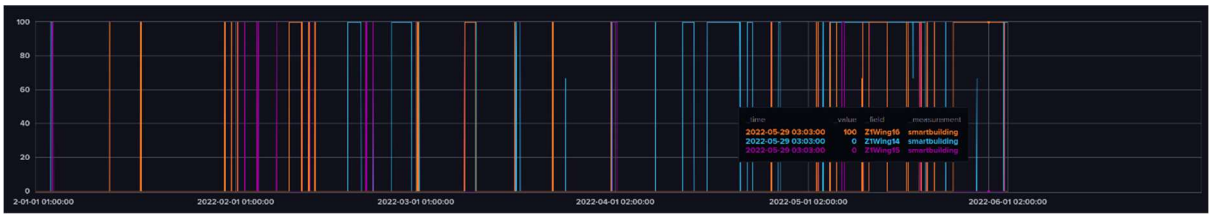

Figure 13: Open window cases from January till June 2022

High heating demand event

The event of a high heating demand, indicated by a change of the Threshold to positive or negative values are shown in Figure 14 (February till March 2022)

- DiffHeatingThreshold1 (Room3.16): 11.02.2022 11:30 to 03.03.2022 09:45
- DiffHeatingThreshold2 (Room3.15): 09.02.2022 10:40 to 11.02.2022 14:00

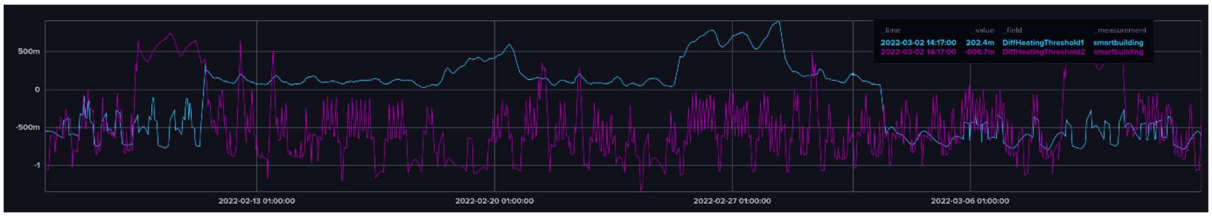

Figure 14: Threshold1 (Room3.16) and Threshold2 (Room3.15)

3.1.6 Step 6 – Check results in GraphDB

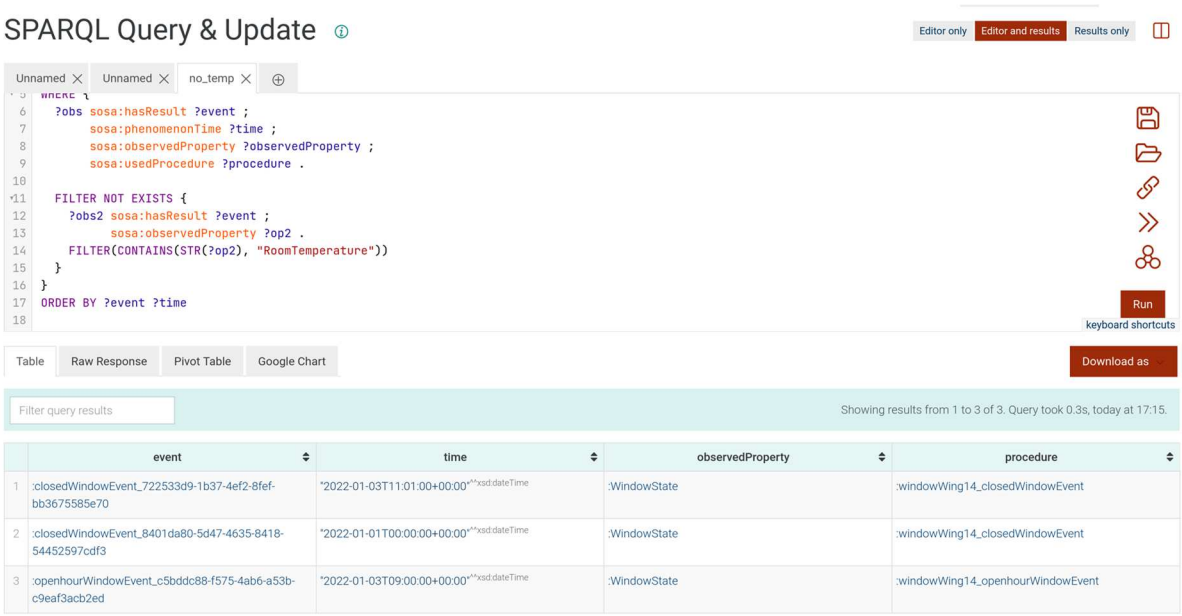

Figure 15: Results from the GraphDB-query (http://localhost:7200/sparql)

3.1.7 Step 7 - Visualisation in Python

A python script was used to evaluate the results from GraphDB.

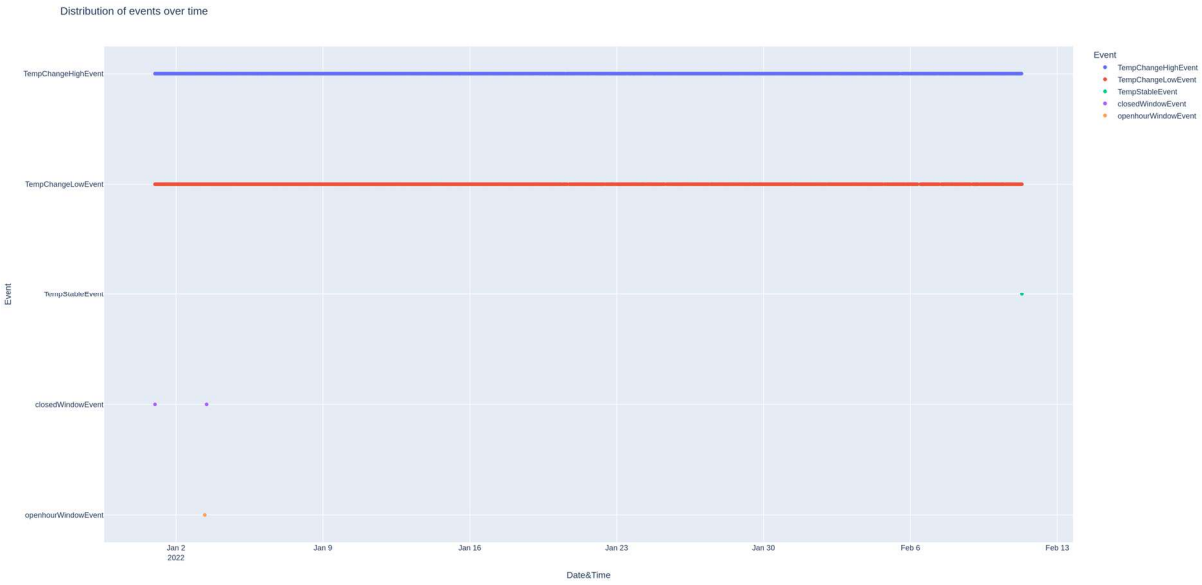

Figure 16: Visualization of GraphDB results with Plotly (Python), period from January 1st till February 12th 2022

3.2 Complexity

For the extension, a total of 60 minutes is required to complete steps 1 through 4. The most time-consuming task is step 1. In this step, measurement data must be retrieved, relevant sensor values selected, a Unix timestamp created, the dataset prepared in the required format and finally copied to the correct location. The following time estimates assume a trained person working within established workflows and environments.

Step 1 (30 minutes):
This step is the most demanding because it requires multiple tools: architectural drawings to identify the correct sensors, ThingsBoard to download the measurement values, Python to create the Unix timestamp, and spreadsheets to organize the dataset for export.

Step 2 (15 minutes):
A spreadsheet is filled with data from each sensor along with equations to detect unreasonable heat demand. This task again requires architectural drawings and measurement data to ensure correct value assignments (care must be taken to avoid typos). Once complete, a system description file is generated via a script.

Step 3 (10 minutes):
Specific flux files are created for data preparation within the InfluxDB database. Only the file entries need to be extended (e.g., additional 4-hour averages and threshold values), making the task manageable within 10 minutes. However, there is no feedback about if a typo occurs, which can be problematic.

Step 4 (5 minutes):
This step involves changes and builds, independent of the number of measurement values. Adjustments are made once in the Docker configuration file (e.g., measurement duration), and the build process (“sudo docker-compose build”) is straightforward. In total, 5 minutes are required.

For expanding the demonstrator setup from 2 rooms to 4 rooms, we estimate a total of 90 minutes:

- Step 1: +15 minutes
- Step 2: +10 minutes
- Step 3: +5 minutes

To scale the process to the entire building with a total of 350 rooms and 14.951 m² building space, it can be assumed that especially Step 2 (spreadsheet input) and Step 3 (InfluxDB data files) can be optimized. Given standardized measurement data, these files should be automatically generated. For the demonstrator, a detailed spreadsheet with sensor types, locations, and measurement ranges was available, containing all required information. Such documents are typically created in complex projects to provide an overview of the various sensor values and types.

These documents are also convenient for facility managers, as spreadsheets are widely used for documentation. Furthermore, 4-hour average values can be pre-calculated and imported together with other data during Step 1.

If such measures are implemented and integrated into Step 1, the entire building could be set up within one day.

3.3 General Performance of Extended PoC

The extended PoC with the additional room requires double the amount of time during the initialization and the data ingestion is much slower. In combination with the longer time span, the run time is around three times more (6 hours compared to 2 hours) and depends also on the computational power of the used device.

4 Extensibility

Another crucial aspect is the extension of the system to handle new events and explanations. In the PoC, only a very simple case was tested. However, the two explanations that were handled in the PoC are, of course, not the only possible explanations when there is significantly more heating energy consumed than usual.

In general, the thermal behavior of buildings is very complex. There are many different factors that can cause room temperatures to be too high or too low, or lead to excessive energy consumption. In addition to malfunctions in the building services themselves, the weather and user behavior (e.g., the presence of more or fewer people than usual, significant deviations in household electricity consumption in the rooms) also have a major influence.

In buildings that are heated in winter and cooled in summer, heating and cooling can also neutralize each other, so that users enjoy a comfortable indoor climate, but too much energy is still consumed.

The SENSE method can be used to support facility managers in detecting malfunctions and identifying possible sources of error. However, it is essential that the explanations are checked by technical personnel.

The implementation of new explanations for malfunctions is quite complex due to the complexity of the possible causes. In most cases, it is not sufficient to evaluate the values from one or two sensors; instead, a whole series of values must be consulted and compared with historical values when the building services are functioning normally.

The most complex part of this is that a precise logic must be developed to determine which parameters decide which explanation for an undesirable phenomenon is most likely. This often requires not only the current measurements from sensors, but also, for example, an average value over a certain period, historical measurements, or even calculations or simulations. A major problem here is that the sensor technology normally implemented in buildings may not be sufficient. For example, buildings do not normally have window contacts that indicate whether a window is open or closed. If this were the case, facility management could be notified directly if a window was left open for too long, without the need for the SENSE algorithm.

Buildings are also usually unique. The materials used, the orientation, and the size of the windows have a major impact on energy consumption and user comfort. Therefore, the methodology used to explain malfunctions must also be adapted to each single building.

In some cases, single-family homes, terraced houses, or sometimes even apartment buildings that are built in large numbers are an exception. But even here, not every building can be treated the same, as the orientation of the building (direction), the location (weather), and user behavior will not be identical. Self-learning algorithms would be ideal for buildings.

5 Resilience

Evaluating system resilience is a crucial step in ensuring the reliable operation of modern cyber-physical systems, particularly in buildings where large amounts of sensor data are continuously collected and processed. Real-world environments are inherently uncertain: sensors can fail, deliver values outside their expected range, or stop transmitting altogether. In addition, communication networks may introduce delays or data loss, leading to incomplete or outdated information for control systems.

In the scope, five representative test cases were carried out to assess resilience:

- Temporary outage of data acquisition – the complete loss of sensor data over a defined period, simulating outages or communication breakdowns.
- Out-of-range values – the sensor first delivered values below the defined minimum (e.g., –100 instead of 0), followed by values exceeding the maximum (e.g., 200 instead of 100), to test the system's ability to handle physically implausible inputs.
- Missing sensor value – the sensor value didn't provide data for a period of time
- Missing sensor value and key – the sensor value didn't show up for a period of time
- Invalid data type – the sensor provided an alphanumeric value (e.g., s1%) instead of a numerical measurement, to evaluate how the system reacts to structurally corrupted data.

For each case an adapted measurement datafile (output_line_protocol.txt) was created. The docker container of the influxDB database were newly built and checked in the influxDB database. Then, the events were checked via GraphDB and visualised in a Python graph.

5.1 Temperature sensor failure

The measurement data from step 3.1 were modified in such a way that during the window opening of Wing14 between 7am and 11am all measured values between 08am and 10am are deleted. Since influxDB connects the last valid measurement points with a line, the missing values are not directly visualised (see Figure 17). However, when opening the influxDB instance and hovering over the line the nearest available measurement point is highlighted making the two-hour gap indirectly visible.

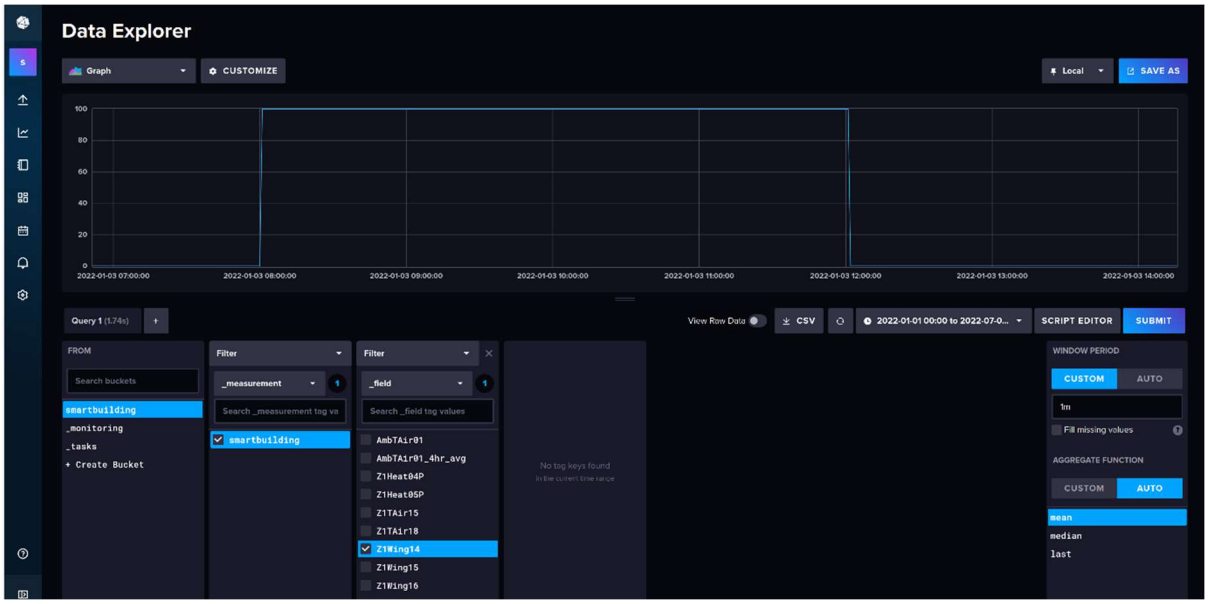

Figure 17 Measurement value of Wing14 in influxDB, test case missing sensors values from 08am to 10am.

This temporary outage doesn't affect the SENSE system. Additionally, no change of the opening state of Wing14 was observed in Figure 18.

Figure 18: Visualization of GraphDB results with Plotly (Python) with the measurement outage

5.2 Out of range

In this setup, the measurement value of Wing14 (measurement range from 0 to 100) changes during an opening from 07am to 11am. The change at 09am to a negative value of 100 lasts 30 minutes and the change later to positive 200 last 30 minutes. This change is clearly shown in influxDB (Figure 19).

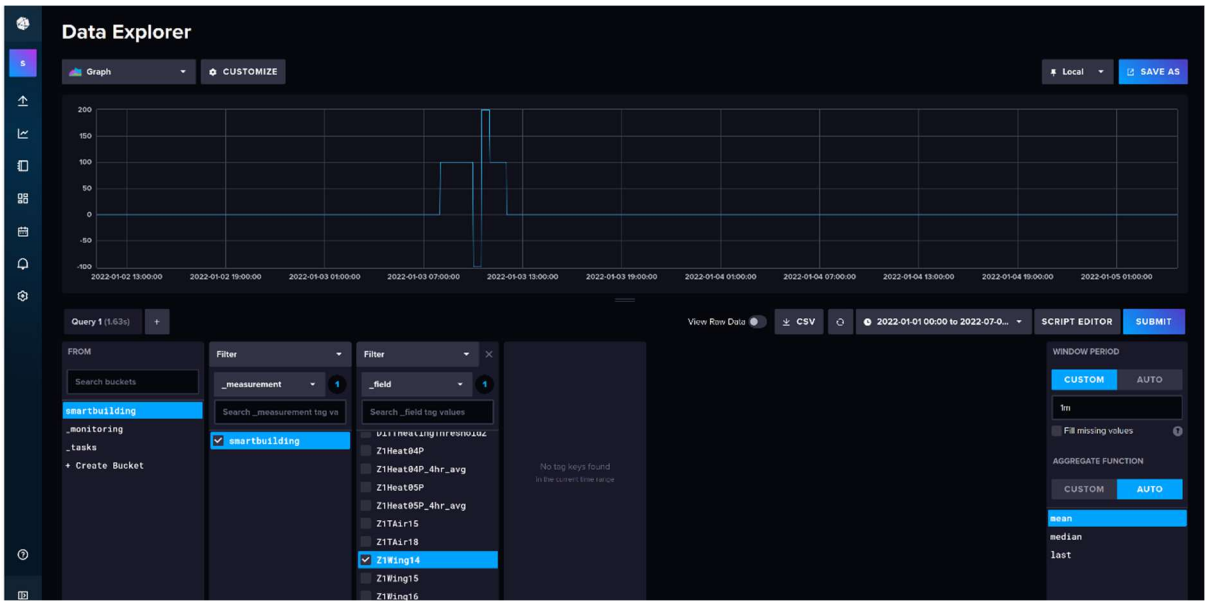

Figure 19: Measurement value of Wing14 in influxDB with sensors out of range.

Now, we can see a change of the window opening state of Wing14 (closed window event), when the value is changed to negative 100. The value change later is not shown in "openhourwindowEvent" in Figure 20 because it lasts only for 1 hour and 30 minutes. At 11am an additional closed window event was detected.

Figure 20: Visualization of GraphDB results with Plotly (Python) with the condition sensor out of range

5.3 Missing a single sensor value

In this case, the measurement value of Wing14 is empty from 09am for 30 minutes. After this period, the measurement value is set back to the original value. In the spreadsheet, this is indicated with "Z1Wing14 =". During the initialization, it appears that the SENSE system cannot cope with this error. In Figure 21 the system shows the error: missing field value.

```
influxdb-writer | unable to parse 'smartbuilding AmbTAir01=-1.633,Z1TAir15=22.59,Z1TAir18=22.626,Z1Wing14=,Z1Wing15=0,Z1Wing16=0,Z1Heat04P=0,Z1Heat05P=0 1641200820000000000': missing fi
eld value
influxdb-writer | unable to parse 'smartbuilding AmbTAir01=-1.6,Z1TAir15=22.575,Z1TAir18=22.625,Z1Wing14=,Z1Wing15=0,Z1Wing16=0,Z1Heat04P=0,Z1Heat05P=0 1641200880000000000': missing fie
ld value
influxdb-writer | unable to parse 'smartbuilding AmbTAir01=-1.6,Z1TAir15=22.575,Z1TAir18=22.604,Z1Wing14=,Z1Wing15=0,Z1Wing16=0,Z1Heat04P=0,Z1Heat05P=0 1641200940000000000': missing fie
ld value
influxdb-writer | unable to parse 'smartbuilding AmbTAir01=-1.6,Z1TAir15=22.573,Z1TAir18=22.586,Z1Wing14=,Z1Wing15=0,Z1Wing16=0,Z1Heat04P=0,Z1Heat05P=0 1641201000000000000': missing fie
ld value
influxdb-writer | unable to parse 'smartbuilding AmbTAir01=-1.533,Z1TAir15=22.572,Z1TAir18=22.574,Z1Wing14=,Z1Wing15=0,Z1Wing16=0,Z1Heat04P=0,Z1Heat05P=0 1641201060000000000': missing f
ield value
influxdb-writer | unable to parse 'smartbuilding AmbTAir01=-1.5,Z1TAir15=22.581,Z1TAir18=22.58,Z1Wing14=,Z1Wing1S=0,Z1Wing16=0,Z1Heat04P=0,Z1Heat05P=0 1641201120000000000': missing fiel
d value
influxdb-writer | unable to parse 'smartbuilding AmbTAir01=-1.433,Z1TAir15=22.562,Z1TAir18=22.549,Z1Wing14=,Z1Wing15=0,Z1Wing16=0,Z1Heat04P=0,Z1Heat05P=0 1641201180000000000': missing f
ield value
influxdb-writer | unable to parse 'smartbuilding AmbTAir01=-1.4,Z1TAir15=22.553,Z1TAir18=22.562,Z1Wing14=,Z1Wing15=0,Z1Wing16=0,Z1Heat04P=0,Z1Heat05P=0 1641201240000000000': missing fie
ld value
influxdb-writer | unable to parse 'smartbuilding AmbTAir01=-1.4,Z1TAir15=22.575,Z1TAir18=22.547,Z1Wing14=,Z1Wing15=0,Z1Wing16=0,Z1Heat04P=0,Z1Heat05P=0 1641201300000000000': missing fie
ld value
influxdb-writer | unable to parse 'smartbuilding AmbTAir01=-1.333,Z1TAir15=22.588,Z1TAir18=22.567,Z1Wing14=,Z1Wing15=0,Z1Wing16=0,Z1Heat04P=0,Z1Heat05P=0 1641201360000000000': missing f
ield value
influxdb-writer | unable to parse 'smartbuilding AmbTAir01=-1.233,Z1TAir15=22.576,Z1TAir18=22.532,Z1Wing14=,Z1Wing15=0,Z1Wing16=0,Z1Heat04P=0,Z1Heat05P=0 1641201420000000000': missing f
ield value
influxdb-writer | unable to parse 'smartbuilding AmbTAir01=-1.333,Z1TAir15=22.575,Z1TAir18=22.533,Z1Wing14=,Z1Wing15=0,Z1Wing16=0,Z1Heat04P=0,Z1Heat05P=0 1641201480000000000': missing f
```

Figure 21: Error during initialization with missing value

Because of this error, the influxDB database cannot initialize fully and shows in Figure 22 only the query smartbuilding but without any data. Consequently, no results are stored in the GraphDB database.

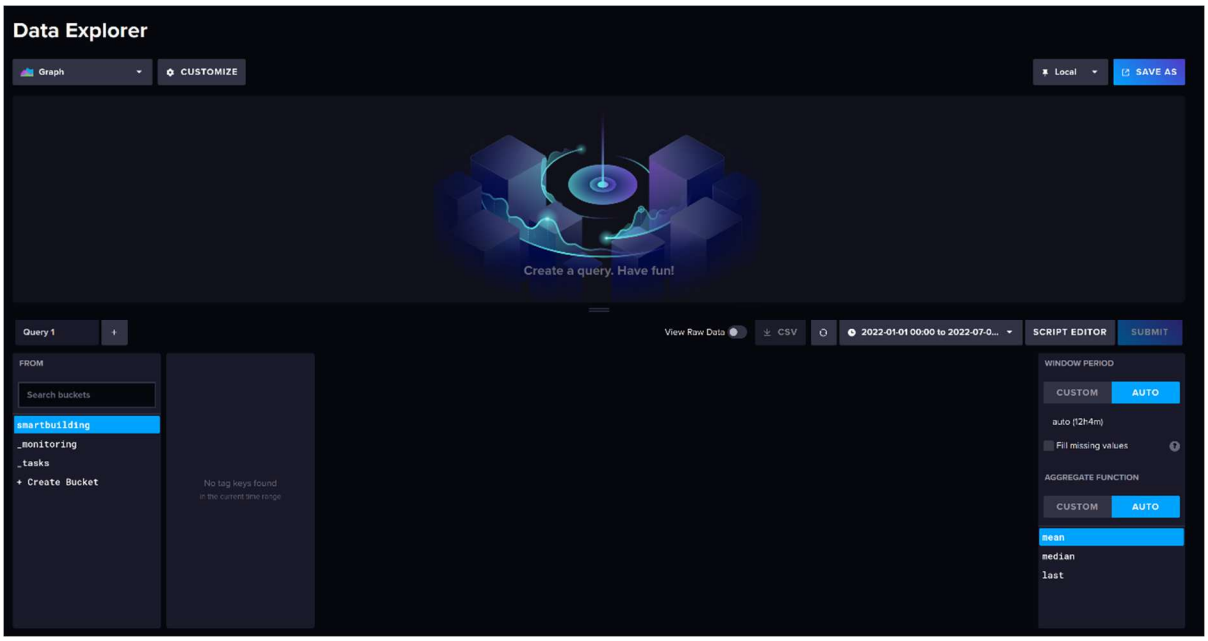

Figure 22: Empty influxDB database due to the missing measurement value

5.4 Missing of value and key

Now the measurement value and the key of Wing14 are empty from 09am for 30 minutes. After this period the measurement value is set back to the original value. In the spreadsheet this is indicated with “”. During the initialization, it appears that the SENSE system cannot cope with this error. In Figure 23 the system shows the error: invalid field format and no results are present in the GraphDB database.

Figure 23: Error during initialization with missing value and key

5.5 Invalid data type

This test was performed, but due to the wrong data type no data can be found in the influxDB database. This affects also the event-detection later on (due to missing data) and no results are shown in the GraphDB database.

5.6 Conclusion

The resilience tests demonstrated that the SENSE system remains robust when facing temporary outages and out-of-range values, with event detection continuing to function reliably. However, missing values, missing keys, or invalid data types proved critical: in these cases, InfluxDB could not be initialized, resulting in no data being stored or analyzed in GraphDB. This underlines the importance of ensuring data quality and consistency before the data is ingested in the SENSE system. Future versions should include a quality check to warn the user if the used measurement data are structurally corrupted or incomplete, to ensure reliable event detection in real-world, error-prone environments.

6 Desirable Scenarios From a Facility Manager's Perspective

In larger office buildings, it is usually the facility manager's job to ensure that the building services equipment is operating properly. The focus is often not on energy efficiency, but rather on ensuring that users have an acceptable indoor climate and that there are as few complaints as possible.

It would be helpful for facility managers if, when problems arise, they were provided with the most likely explanation so that the problem can be solved as quickly as possible. In terms of energy savings, it would also be helpful if excessive energy consumption were automatically detected and explanations for the excessive energy consumption were provided at the same time.

6.1 Automated Implementation

In order to enable widespread use of SENSE technology, it is important that operation is made as simple as possible for the facility manager. The implementation of the system and the programming of explanations cannot be carried out by the facility manager. Ready-made algorithms, which are then adapted to the respective building, should be created during the planning process and implemented in the building automation system.

The implementation of a chatbot that the facility manager can communicate with in natural language can be very helpful. It could be used to ask questions about the problems detected. However, the malfunctions themselves should be reported automatically and sent to the facility manager (e.g., by email or chat message).

6.2 Only Use Existing Measurement Technology

As mentioned above, no additional sensors should be necessary to generate malfunctions and explanations.

For example, it would be helpful to develop an algorithm that can determine whether a window is open and for how long it is based on changes in room temperature and outside temperature. It should also be determined whether a window has already been closed again. Brief openings for ventilation should not immediately trigger an alarm.

6.3 Include Statistical Analysis

To support the detection of malfunctions and the generation of explanations, it would be helpful if a statistical evaluation of “normal building behavior,” i.e., energy consumption and comfort parameters, were to run in the background. This would allow the current values to be compared with statistical values, leading to better results in fault detection.
In new buildings, a simulation of the building with building services could replace the statistical values. Simulations could be used to determine the energy consumption to be expected in the current weather conditions.

7 Summary

This deliverable reports on the technical evaluation of the SENSE system in the smart building domain. A proof-of-concept (PoC) was implemented to demonstrate how semantic-based explanations can support the detection of anomalous events in building operation. The initial setup focused on one room and a single event (excessive heating demand), which was further extended to two rooms and additional scenarios.

The evaluation showed that the SENSE system is capable of reliably detecting anomalies such as open windows or sensor malfunctions and providing explanations that correspond to expert assessments. Usability aspects from a facility manager's perspective highlighted the potential of integrating explanations into daily building operation, while underlining the need for automation and simple interfaces.

Scalability tests indicated that extending the system to more rooms is technically feasible but currently requires significant manual effort. Steps such as data preparation, semantic modeling, and database configuration are time-consuming and prone to errors. However, once standardized data formats and automated workflows are introduced, the implementation time can be significantly reduced. It is estimated that a whole building could be set up within one day under optimized conditions.

Extensibility tests demonstrated that the system can be adapted to handle additional events, but that creating robust explanatory logic is complex due to the variety of influencing factors (building physics, user behavior, weather conditions). Additional sensor data or simulation-based reference models could further improve the accuracy of explanations.

Resilience testing confirmed that the system can handle temporary outages and out-of-range values but struggles with missing or invalid data formats. This points to the need for improved error-handling mechanisms in real-world applications.

From the facility management perspective, three key requirements were identified: (1) automated implementation to reduce manual setup, (2) reliance on existing sensor infrastructure without costly new installations, and (3) inclusion of statistical or simulation-based analysis to benchmark building behavior.

The SENSE system provides a promising approach to increasing transparency in smart building operation by combining event detection with semantic explanations. The PoC results demonstrate both the feasibility and the challenges of scaling such a system to real-world building environments. Future work will focus on automation, robustness, and integration into facility management workflows, thereby contributing to energy efficiency and sustainability goals.

References

[1] Poelmans et al. “Deliverable 2.3: Definition of PoC's”, SENSE project 2024
[2] Frühwirth et al. "SENSE Deliverable 3.1 Auditable SENSE Architecture," 2024.
[3] Frühwirth et al. “Deliverable 5.1: Technology Stack Implementation”. SENSE project 2025
[4] Jaehnig D., Moser C. “Deliverable 5.3: PoC implementation in smart buildings domain”, SENSE project 2025.