



SENSE: Semantic-based Explanation of Cyber-physical Systems

Deliverable 5.3: PoC implementation in smart buildings domain

Authors	:	Dagmar Jaehnig, Christoph Moser
Dissemination Level	:	Public
Due date of deliverable	:	01.11.2024
Actual submission date	:	July 2025
Work Package	:	WP5
Type	:	Report
Version	:	1.0

Abstract

Deliverable D5.3 reports on the implementation of the smart buildings PoC using the SENSE technology stack. It provides brief descriptions of all steps necessary for implementation using a period of three months of sample data.

The information in this document reflects only the author's views and neither the FFG nor the Project Team is liable for any use that may be made of the information contained therein. The information in this document is provided "as is" without guarantee or warranty of any kind, express or implied, including but not limited to the fitness of the information for a particular purpose. The user thereof uses the information at his/ her sole risk and liability.

History

Version	Date	Reason	Revised by
0.9	25.3.2025	Initial version	DJ, CM
1.0	1.8.2025	Final Version	

Author List

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

Executive Summary

This deliverable briefly outlines the smart buildings PoC developed within the SENSE system. The PoC and all data required to initialize and run it can be found in the respective gitlab repository.

A very simple PoC of smart building with one 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.

The deliverable documents how the SENSE system is implemented by filling information about the existing components in the building and their logical connections in an excel template sheet. Next, the system needs to be set up on a computer to be able to run it and detect events. Finally, a chatbot is used to extract information from the detected events.

By instantiating the SENSE Stack in the smart building domain, we have shown that the stack has a high degree of generality and can be applied to various domains where cyber-physical systems exist.

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]).

Table of Content

History	2
Author List.....	2
Executive Summary.....	3
Table of Content	4
List of Figures	5
1 Introduction	6
2 Steps to Implement the SENSE SYSTEM for PoC Smart Building	6
3 Event detection	15
4 Explanations from Chatbot	19
4.1 Ask for periods with high energy demand.....	19
4.2 Identify the cause of too much energy being used in a specific event	20
4.3 Identify where an event happened	21
5 Summary	21
References	21

List of Figures

Figure 1: Schematic representation of the sensors and platforms that exist in the PoC Smart Building	7
Figure 2: List of sensors.....	8
Figure 3: List of observable properties	8
Figure 4: Platforms tab in the excel sheet	9
Figure 5: PlatformTypes tab in Excel sheet.....	9
Figure 6: Increase the speed of data ingestion.....	10
Figure 7: Influx-DB- Show included measurement data	11
Figure 8: State types	12
Figure 9: EventStateMapping	13
Figure 10: StateTypeCausality tab	14
Figure 11: Add SSH-key to Gitlab	14
Figure 12: Successful Login	15
Figure 13: SENSE instance running	15
Figure 14: Detected broken sensor on the 11 th of February	16
Figure 15: Open Window from the 25 th till 28 th of February.....	17
Figure 16: Detected open window event after 2 hours opening	17
Figure 17: Window close event.....	18
Figure 18: Anomaly in the heating demand detected.	18
Figure 19: Too high energy demand event.....	19
Figure 20: Screenshot of chatbot (is there too much energy used in February).....	19
Figure 21: Screenshot of chatbot (Is there too much energy used in March 2022)	20
Figure 22: Screenshot of chatbot (Example answer for the case of a broken sensor).....	20
Figure 23: Screenshot of chatbot (Example answer for the case of an open window).....	21
Figure 24: Screenshot of chatbot (Example answer for the location of an error)	21

1 Introduction

The SENSE project aims to enhance the efficiency and user-friendliness of complex technical systems, by applying neurosymbolic AI approaches in combination with AI agents to provide meaningful insights, explanations and suggestions of system events to key stakeholders.

SENSE planned and implemented a proof of concept (PoC) in the smart building domain, which was defined and planned in great detail in the SENSE deliverable “D2.3 PoC Definition”[1].

This deliverable describes in detail the steps that have been performed in order to set up the PoC for a very small scenario with one problem to detect and two possible explanations. As an example, measured data from an office building at Infineon-Villach were available. To facilitate things, data from only one meeting room for a period of three months were used

To implement the SENSE system in the smart building PoC, the SENSE User Guideline was followed to fill in the SystemData-Excel file which is needed as input for the SENSE system (Chapter 2).

In chapter 3, it has been described how it is checked that all events in the sample data have actually been detected.

Finally, in chapter 4 the use of the chatbot to get explanations about the events is demonstrated.

2 Steps to Implement the SENSE SYSTEM for PoC Smart Building

Step 1 Common Conceptualization

The first step was to get familiar with the vocabulary necessary to fill in the Excel table.

Step 2 Know Your Goals

In the smart buildings PoC, the goal is to explain why there is too much heat consumption in a room. For the PoC a very small system has been defined which includes only one room and there are only two possible explanations: The window has been left open or there is a problem with a room temperature sensor value.

The SENSE system should be able to identify whether one of these explanations is valid or not.

Step 3 Know Your System

To clarify the topology of the system, a scheme showing the sensors needed for the explanation task and how they are located in different parts of the building (platforms) has been drawn (see Figure 1). In addition, the hierarchy of the different parts of the building is shown.

It was kept in mind, that later on there would be more than one room and also other sensors that could be used to generate additional explanations. Therefore, the topology is defined to allow for easy implementation of other sensors and rooms.

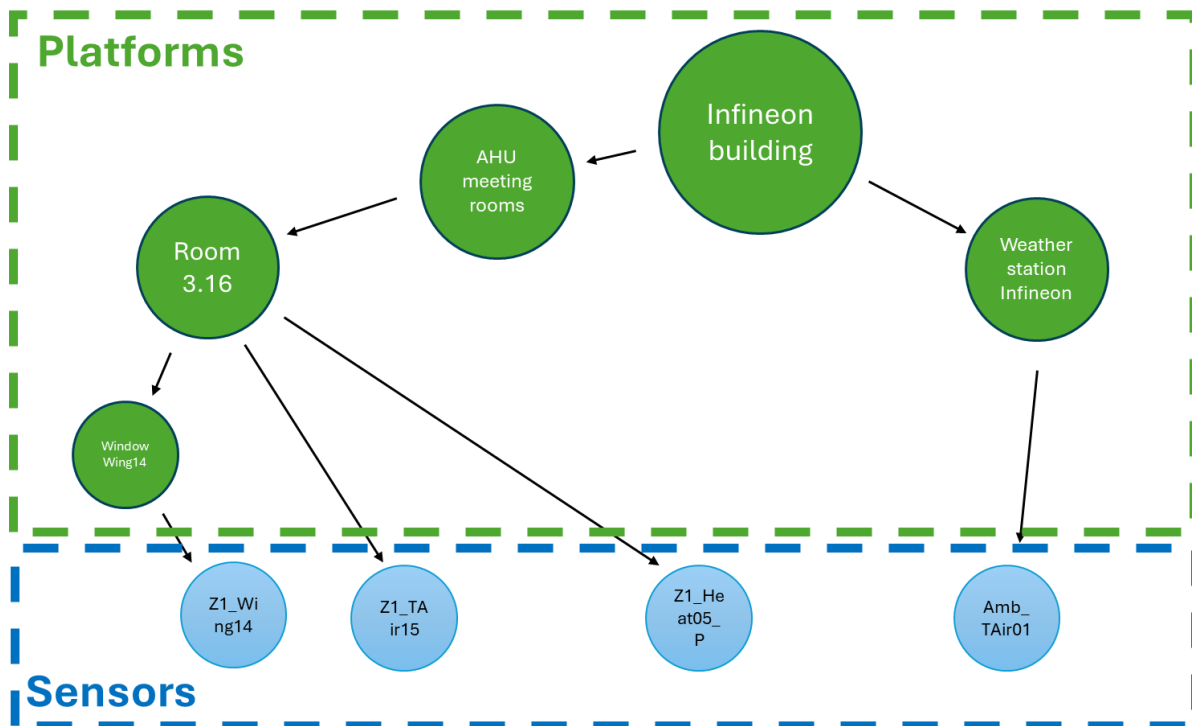


Figure 1: Schematic representation of the sensors and platforms that exist in the PoC Smart Building

It was found to be easier to go from the real sensors and platforms that will be used to the more general types of sensors and platforms. Therefore, we started with the sensors. In the PoC the following real sensors are used:

- the window contact sensor with the sensor name Z1_Wing14
- the room air temperature sensor in room 3.16 with the sensor name Z1_Tair15
- the heat meter measuring the heat consumed by the radiators in room 3.16 with the sensor name Z1_Heat05_P
- and the ambient air temperature with the sensor name Amb_TAir01.

In addition, there will be one virtual sensor. It is defined by the difference between the heat meter and a function that depends on the 4 hour-average of the ambient temperature. The function is shown in cell E6 of the Excel sheet and describes the average heat consumption of the room if no window is open as a function of the ambient temperature. This function needs to be specified for each individual room to be analyzed. The function has been identified by analyzing a whole heating season of data.

The resulting list of sensors, including the virtual sensor, is shown in Figure 2.

	A	B	C	D	E
1	Sensor	hostedBy_Platform	observes_Observable	TimeseriesId	
2	Z1Heat04P	Room3.16	Heatradiatorroom	measurement=smartbuilding,field=Z1Heat04P	
3	AmbTAir01	weatherStationInfini	AmbientTemperature	measurement=smartbuilding,field=AmbTAir01	
4	Z1Wing14	windowWing14	WindowState	measurement=smartbuilding,field=Z1Wing14	
5	Z1TAir15	Room3.16	RoomTemperature	measurement=smartbuilding,field=Z1TAir15	
6	DiffHeatingThreshold1	Room3.16	HeatDiffRadiator	measurement=smartbuilding,field=DiffHeatingThreshold1=Z1Heat05P(4 hour average) - (-0,02 Amb_TAIR01(4 hour average +0,7)	
7					
8					
9					
10					
11					
12					
13					
14					
15					
16					

Figure 2: List of sensors

Figure 2 also shows the observable property for each sensor. In our case we have the heat measured by the heat meter (**heatradiatorroom**), the **AmbientTemperature**, the state of the window (**WindowState**), the **RoomTemperature** and finally, the heat difference that is calculated by the virtual sensor (**HeatDiffRadiator**). The names of the observable properties are entered in the tab '**ObservableProperties**' as shown in Figure 3.

	A	B	C	D	E	F	G	H	I	J
1	ObservableProperty									
2	Heatradiatorroom									
3	AmbientTemperature									
4	WindowState									
5	RoomTemperature									
6	HeatDiffRadiator									
7										
8										
9										
10										
11										
12										
13										
14										

Figure 3: List of observable properties

In the second column of Figure 2, the platform that hosts the sensor is shown. That means where the sensor is located. In our case, two of the sensors are located in **Room3.16**. The virtual sensor can also be allocated to **Room3.16**. The ambient temperature sensor is located in the **weatherStationInfini** and the window contact is located at the **windowWing14**. These platforms are entered in the tab '**platforms**' (Figure 4). There are also a few additional platforms that clarify the hierarchy between the different platforms. The window platform wing 14 is allocated to the room 3.16 platform. The room 3.16 platform is in turn allocated to the AHU meeting rooms platform to allow for later addition of sensors that are located in the AHU. Both the AHU meeting rooms platform and the weather station are hosted by the overall platform called Infineon building.

	A	B	C	D	E	F	G
1	Platform	PlatformType	hostedBy_Platform				
2	Infineonbuilding	Building					
3	AHUmeetingrooms	AHU	Infineonbuilding				
4	weatherStationInfineon	weatherStation	Infineonbuilding				
5	Room3.16	Room	AHUmeetingrooms				
6	windowWing14	Window	Room3.16				
7							
8							
9							
10							
11							
12							
13							
14							
15							
16							
17							

Figure 4: Platforms tab in the excel sheet

From there, we can see that five platform types exist in the system: **Building**, **AHU**, **weatherStation**, **room** and **window**.

These must be entered in the tab '**PlatformTypes**' of the Excel sheet. Because all of them describe some part of the building or heating/cooling systems, they were all specified to be a subclass of the platform type '**facility**'. The resulting Excel tab is shown in Figure 5.

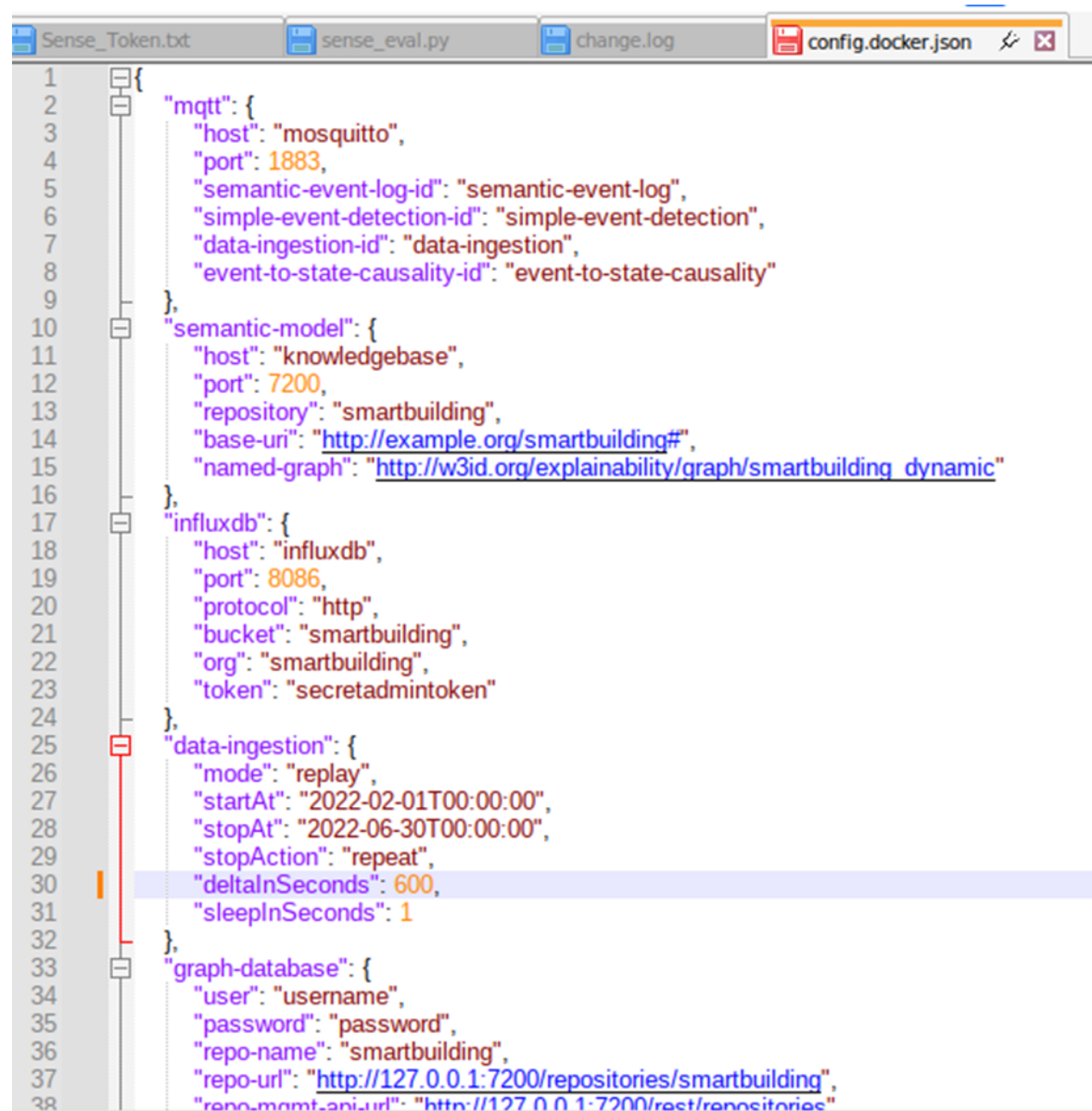
	A	B	C	D	E	F	G	H
1	PlatformType	subClassOf_PlatformType						
2	Facility							
3	Building	Facility						
4	Room	Facility						
5	Window	Facility						
6	AHU	Facility						
7	weatherStation	Facility						
8								
9								
10								
11								
12								
13								
14								
15								
16								

Figure 5: PlatformTypes tab in Excel sheet

Step 4 Know Your Data

In the 'sensors' tab shown in Figure 2 information on how to identify the sensor data in the time series database must be added to the column **TimeseriesId**.

Measurement data can be inserted in the SENSE system as "live data" or "demo-data". In the pre-defined configuration present on gitlab (<https://git.ai.wu.ac.at/sense/smart-building>), measurement data from February 2022 till April 2022 are included in an internal Influx-DB instance (<http://localhost:8086>; username: smartbuildings; password: secretpassword). This database is available after the SENSE system is started (see later Step 6 for more details). The SENSE system inserts small data chunks repeatedly. To increase the speed of the data ingestion, the following values must be changed in the file: config.docker.json: „deltaInSeconds“: 6000 (original value 60) and „sleepInSeconds“: 1 (original value 2).



```

1  {
2    "mqtt": {
3      "host": "mosquitto",
4      "port": 1883,
5      "semantic-event-log-id": "semantic-event-log",
6      "simple-event-detection-id": "simple-event-detection",
7      "data-ingestion-id": "data-ingestion",
8      "event-to-state-causality-id": "event-to-state-causality"
9    },
10   "semantic-model": {
11     "host": "knowledgebase",
12     "port": 7200,
13     "repository": "smartbuilding",
14     "base-uri": "http://example.org/smartbuilding#",
15     "named-graph": "http://w3id.org/explainability/graph/smartbuilding_dynamic"
16   },
17   "influxdb": {
18     "host": "influxdb",
19     "port": 8086,
20     "protocol": "http",
21     "bucket": "smartbuilding",
22     "org": "smartbuilding",
23     "token": "secretadmintoken"
24   },
25   "data-ingestion": {
26     "mode": "replay",
27     "startAt": "2022-02-01T00:00:00",
28     "stopAt": "2022-06-30T00:00:00",
29     "stopAction": "repeat",
30     "deltaInSeconds": 600,
31     "sleepInSeconds": 1
32   },
33   "graph-database": {
34     "user": "username",
35     "password": "password",
36     "repo-name": "smartbuilding",
37     "repo-uri": "http://127.0.0.1:7200/repositories/smartbuilding",
38     "repo-mgmt-uri": "http://127.0.0.1:7200/rest/repositories"

```

Figure 6: Increase the speed of data ingestion

Start the Sense system and open <http://localhost:8086> and type in the credentials above. Open the data explorer (on the left) and select “smartbuilding”. Now select the sensor values (Figure 7).

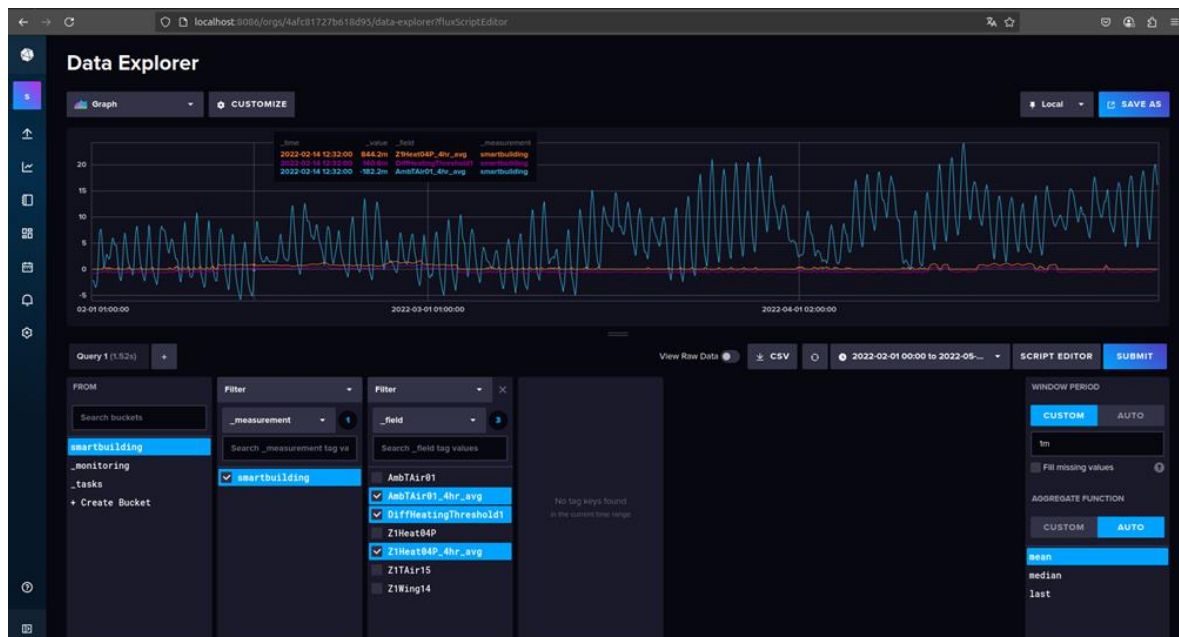


Figure 7: Influx-DB- Show included measurement data

Be aware of changing the correct time period and change the window period to the timestep of the included data (1 Minute = 1m) and click SUBMIT to update the screen. Check every event which you want to detect for availability in the measurement data set first.

Step 5 Explore Causal Relations in the System

In this step, the causal relations within the system should be defined. The procedure consists of several substeps.

Step 5.1 Define what you are interested in, what situation should be explained?

In the PoC scenario, there is only one situation to be explained. That is that there is too much heating energy consumed by the radiators in a certain room.

Step 5.2 Discuss scenarios of potential causes

In this first, very simple proof of concept we investigated only two possible explanations:

- The window has been left open.
- The room temperature sensor shows a constant value over a long period of time, possibly because the sensor itself is broken or there is some data transmission problem.

Step 5.3 Define corresponding States, Events

First, the undesired state was defined. In our case, that is the state type ‘**highEnergy**’. It uses the observable property of the virtual sensor (**HeatDiffRadiator**) and is associated with the platform type ‘room’. That means each room could have the state ‘**highEnergy**’. Too much energy is consumed in that room.

This state is a trigger state. That means if the system is in that state, an explanation will be generated. This is shown in column D of Figure 8.

After that, we have two states related to the two possible explanations.

- The windowOpen state can be detected by a **WindowState** sensor on the **Window** platform.
- The ConstantTemperatureValue state can be detected by a **RoomTemperature** sensor on the **Facility** platform because it can be applied to all of the room temperatures in the building.

	A	B	C	D	E	F
1	StateType	associatedObservableProperty	associatedPlatformType	isTriggerState	Description	
2	highEnergy	HeatDiffRadiator	Room	WAHR	energy consumption for space heating is high (compared to normal values)	
3	windowOpen	WindowState	Window	FALSCH		
4	ConstantTemperatureValue	RoomTemperature	Facility	FALSCH		
5						
6						
7						
8						
9						
10						
11						
12						
13						
14						
15						

Figure 8: State types

Next, it was necessary to define the events that activate and deactivate the previously defined states in the **EventStateMapping** tab of the Excel sheet. A total of seven events have been defined. The resulting tab is shown in Figure 9 split into two parts because the table is too wide to be shown on the page.

The first two events concern the **highEnergy** state. The first event marks the start of the **highenergy** state, the second one the end of it. The monitored platform is in both cases **Room** with the monitored signal **HeatDiffRadiator**, i.e. the virtual sensor. For starting the state the signal needs to overshoot a certain value, for ending to undershoot. The threshold value is defined in column I and is set to 0. That means if the difference between 4-hour average of the heat meter and the threshold function is above zero, the **highEnergy** state starts and it ends if the difference goes below zero again.

The next pair of events concerns the **windowOpen** state. However, in this case the threshold value is 50 (the sensor signal can go from 0 to 100). But for the starting event, there is an additional condition specified. The overshoot interval is set to 7200 seconds. That means that the event **windowOpen** starts only if the overshoot condition is met for 7200 seconds (=2 hours). For the **closeWindow** event there is no such condition specified. That means as soon as the window contact delivers a value below 50, the **windowOpen** state ends.

Finally, there is a set of three events defined that concern the **ConstantTemperatureValue** state. For starting the state there are two properties specified. The first property is **maximum_delta** with a value of 0.01. That means, that if the monitored temperature doesn't vary by more than 0.01 K, the **ConstantTemperatureValue** state starts. But only if the second condition is met: The **stable_time** of the value is more than 14400 seconds (= 4 hours).

For ending the **ConstantTemperatureValue** state, there are two options specified as separate events. The first one is called **TempChangeHighEvent** and it means that the monitored

temperature increases by at least a value of 0.01 K within 300 seconds (**rise_time**). The other one is the opposite, if the monitored temperature decreases by the same value in 300 seconds (**fall_time**).

1	A	B	C	D	E	F
1	EventType	StateType_starts	StateType_ends	MonitoredPlatform	MonitoredSignal	SignalProperty
2	start defining your events in the next line					
3	ThresholdViolatedEvent	highEnergy		Room	HeatDiffRadiator	Overshoot
4	ThresholdNormalEvent		highEnergy	Room	HeatDiffRadiator	Undershoot
5	openhurWindowEvent	windowOpen		Window	WindowState	Overshoot
6	closedWindowEvent		windowOpen	Window	WindowState	Undershoot
7	TempStableEvent	ConstantTemperatureValue		Room	RoomTemperature	StableTime
8	TempChangeHighEvent		ConstantTemperature	Room	RoomTemperature	RiseTime
9	TempChangeLowEvent		ConstantTemperature	Room	RoomTemperature	FallTime
10						
11						

1	A	G	H	I	J	K	L
1	EventType	SignalPropertyParameter1			SignalPropertyParameter2		
2		Name	Type	Literal OrSensor	Name	Type	LiteralOrSensor
3	ThresholdViolatedEvent	threshold	SensorValue	0			
4	ThresholdNormalEvent	threshold	SensorValue	0			
5	openhurWindowEvent	threshold	LiteralValue	50	overshoot_interva	LiteralValue	7200
6	closedWindowEvent	threshold	LiteralValue	50			
7	TempStableEvent	maximum_delta	LiteralValue	0.01	stable_time	LiteralValue	14400
8	TempChangeHighEvent	delta	LiteralValue	0.01	rise_time	LiteralValue	300
9	TempChangeLowEvent	delta	LiteralValue	0.01	fall_time	LiteralValue	300
10							
11							
12							
13							
14							
15							
16							
17							

Figure 9: EventStateMapping

Then, it was necessary to define the causal relations between states in the **StateTypeCausality** tab. There is a line for each of the two possible explanations (see Figure 10).

The first line defines that the **windowOpen** state causes the **highEnergy** state if the two states have a temporal overlap. Column E signifies that the platform where the effect state occurs (room/ highEnergy) is a parent platform of the platform where the causing state occurs (window/windowOpen).

In the second line, the same is defined for the **ConstantTemperatureValue** state which causes a **highEnergy** state. In this case, both states occur at the same platform.

	A	B	C	D	E	F	G	H
1	Number	StateType_cause	causalRelation	temporalRelation	topologicalRelation	StateType_effect	Justification/Description	
2		windowOpen	causes	overlaps	parentPlatform	highEnergy	Window left open for more than 2 hours	
3		ConstantTemperatureValue	causes	overlaps	samePlatform	highEnergy	Room temperature sensor is constant for more than 4 hours	
4								
5								
6								
7								
8								
9								
10								
11								
12								
13								
14								
15								
16								
17								

Figure 10: StateTypeCausality tab

Step 6 Run Your SENSE Instance

The following steps have to be taken to run the SENSE instance:

Pre-requisites

First create a SSH-Key on the PC.

```
ssh-keygen -o -a 100 -t ed25519
```

And add the generated public key to your Gitlab account



Figure 11: Add SSH-key to Gitlab

Check if docker is already installed with the command: Docker

Otherwise create an account (Windows) and install Docker

Check if Docker is correctly installed by the command:

```
Sudo docker run hello-world
```

Install git if not already installed

Create a personal Access Token in Gitlab (tick all boxes).

git clone <https://git.ai.wu.ac.at/sense/smart-building.git>

Type-in: User-name: Email address and Password: Access Token

If SENSE-system is already installed

After the initial prerequisites, the SENSE Instance can be started with the following commands via a terminal. This is also the starting point

```
cd smart-building/
```

```
sudo docker login -u "Email address" -p "Access Token" registry.ai.wu.ac.at
```



```

[redacted]:~$ cd smart-building/
[redacted]:~/smart-building$ sudo docker login -u [redacted] -p [redacted] registry.ai.wu.ac.at
[sudo] Passwort für cmoser:
WARNING! Using --password via the CLI is insecure. Use --password-stdin.
WARNING! Your password will be stored unencrypted in /root/.docker/config.json.
Configure a credential helper to remove this warning. See
https://docs.docker.com/engine/reference/commandline/login/#credential-stores
Login Succeeded

```

Figure 12: Successful Login

sudo docker-compose up

```

[redacted]:~/smart-building$ sudo docker-compose up
[+] Running 9/0
  ✓ Container mosquito Created 0.0s
  ✓ Container semantic-event-log-bridge Created 0.0s
  ✓ Container data-ingestion Created 0.0s
  ✓ Container influxdb Created 0.0s
  ✓ Container knowledgebase Created 0.0s
  ✓ Container explanation-interface Created 0.0s
  ✓ Container simple-event-detection Created 0.0s
  ✓ Container event-to-state-causality Created 0.0s
  ✓ Container influxdb-writer Created 0.0s
Attaching to data-ingestion, event-to-state-causality, explanation-interface, influxdb, influxdb-writer, knowledgebase, mosquito, semantic-event-log-bridge, simple-event-detection
influxdb | 2025-02-26T09:08:41.243669717Z Info found existing bolt db file, skipping setup wrapper {"system": "docker", "bolt_path": "/var/lib/influxdb2/influxdb.bolt"}
mosquitto | 1740560921: mosquitto version 2.0.18 starting
mosquitto | 1740560921: Config loaded from /mosquitto-no-auth.conf.
mosquitto | 1740560921: Opening ipv4 listen socket on port 1883.
mosquitto | 1740560921: Opening ipv6 listen socket on port 1883.
mosquitto | 1740560921: mosquitto version 2.0.18 running
influxdb | 2025-02-26T09:08:41.409543491Z Info found existing bolt db file, skipping setup wrapper {"system": "docker", "bolt_path": "/var/lib/influxdb2/influxdb.bolt"}
knowledgebase | + touch /opt/graphdb/graphdb.log
knowledgebase | This container runs GRAPHDB
knowledgebase | When graphdb has been started, it creates a repository, and imports the SENSE Semantic Model and the System Data into the corresponding named graphs
knowledgebase | + iptables -A INPUT -p tcp -s 127.0.0.1 --dport 7200 -j ACCEPT
knowledgebase | +
knowledgebase | iptables
knowledgebase | -A
knowledgebase | INPUT
knowledgebase | -p
knowledgebase | tcp
knowledgebase | --dport
knowledgebase | 7200
knowledgebase | -j
knowledgebase | DROP
knowledgebase |

```

Figure 13: SENSE instance running

After the SENSE-system is running, the INFLUX-DB database (<http://localhost:8080>), as well as a GraphDB instance (<http://localhost:7200>) is started.

3 Event detection

Three different state types have been defined in the Excel template. All of them are present in the example data used to test the PoC. Therefore, the corresponding events (for starting and ending of a state) should be detected.

To evaluate the event detection from the SENSE-system, a PYTHON script was written, which analysed the results from the GraphDB database. Therefore, a SPARQL-query is triggered and visualized with plottly via browser.

The following SPARQL-query was used to extract the data from GraphDB:

SPARQL-Endpoint

sparql_endpoint = "http://localhost:7200/repositories/smartbuilding"

SPARQL-Abfrage für Events mit zusätzlichen Eigenschaften

```
sparql_query = """
PREFIX sosa: <http://www.w3.org/ns/sosa/>
SELECT DISTINCT ?event ?time ?observedProperty ?procedure WHERE {
  ?obs sosa:hasResult ?event ;
    sosa:phenomenonTime ?time ;
    sosa:observedProperty ?observedProperty ;
    sosa:usedProcedure ?procedure .
} ORDER BY ?event ?time
"""
```

Event Detection – Broken sensor

The state type corresponding to a broken sensor is called **ConstantTemperatureValue**. It starts with the TempStableEvent which was successfully detected on 11th of February (green description in Figure 14). During the measurement period the temperature sensor Z1Tair15 was broken. The top two lines in the figure show that before and after the sensor was broken, both TempChange events are detected almost continuously. That makes sense because in normal operation, the room temperatures always change slightly.

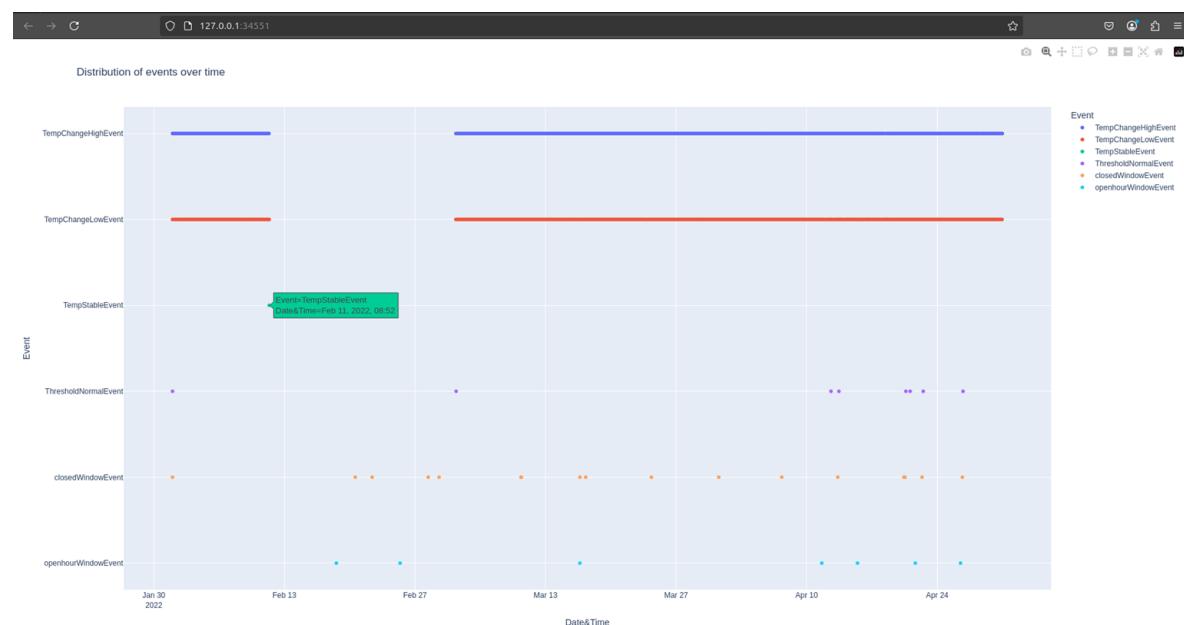


Figure 14: Detected broken sensor on the 11th of February

Event Detection – Open Window

Window openings have also been detected. For example, an open window is present from 25.02.2022 07:45 till 28.02.2022. The state of an open window is indicated by a rise of the sensor value Z1Wing14 to 100 (0 indicates a closed window), which is shown in Figure 15.

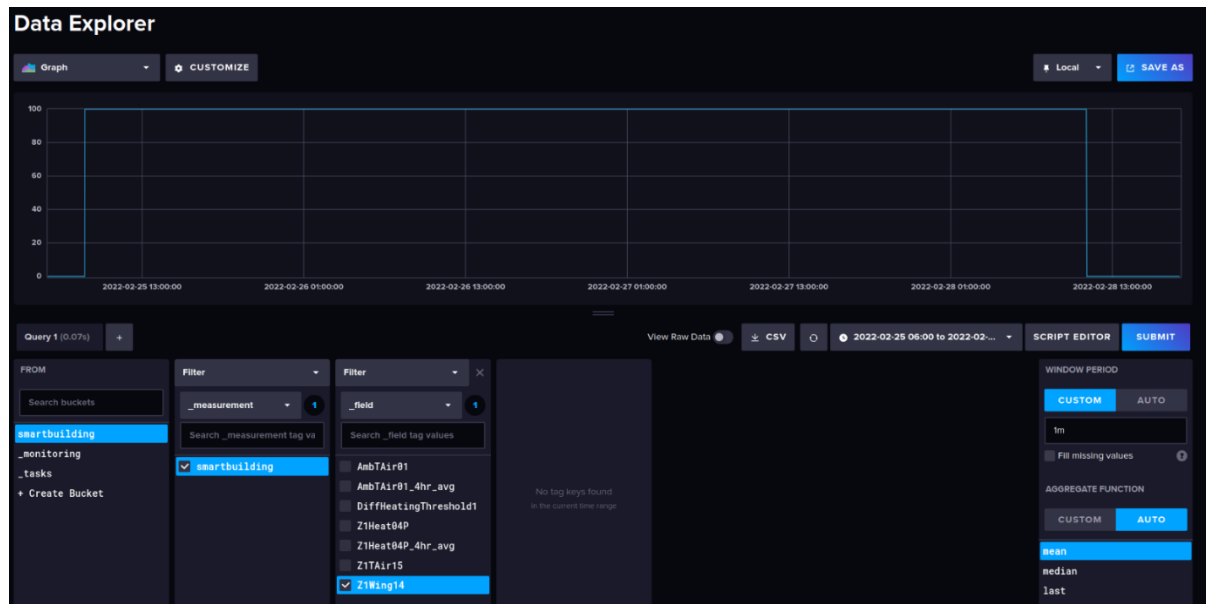


Figure 15: Open Window from the 25th till 28th of February

The SENSE system triggers in this case an open window event after 2 hours or 7200 seconds (predefined in Excel sheet) at 25.02.2022 09:45 (Figure 16).



Figure 16: Detected open window event after 2 hours opening

The system also detects the closing of the window on the 28th of February (Figure 17).



Figure 17: Window close event

Event Detection – Too high energy demand

If the variable **DiffHeatingThreshold1** changes its value from positive to negative values, a deviation from the proposed heating demand is detected. This happens on the 3rd of March between 10 and 11 AM (Figure 18).

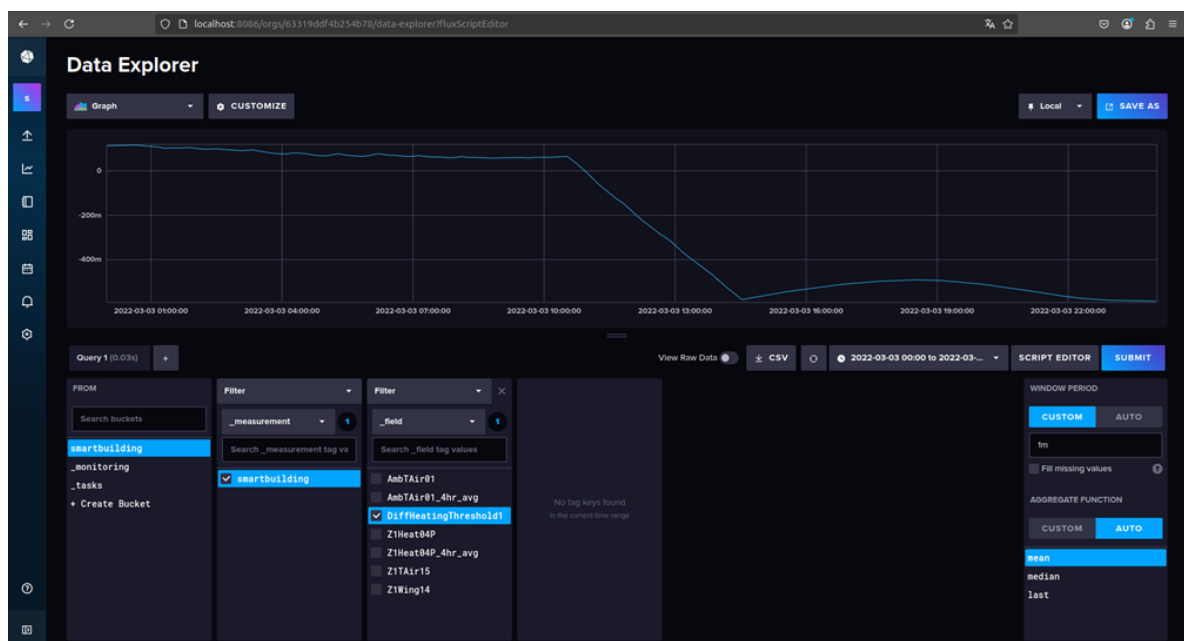


Figure 18: Anomaly in the heating demand detected.

This change of the value of DiffHeatingThreshold1 from positive to negative values is detected from the SENSE (Figure 19).



Figure 19: Too high energy demand event

4 Explanations from Chatbot

The functionalities of the chatbot, implemented to provide an interface for facility managers to ask questions about the detected events and possible explanations, are demonstrated in this chapter.

4.1 Ask for periods with high energy demand

The facility manager can ask the chatbot for the times when there has been too much energy consumed in a certain time period. For example, in February

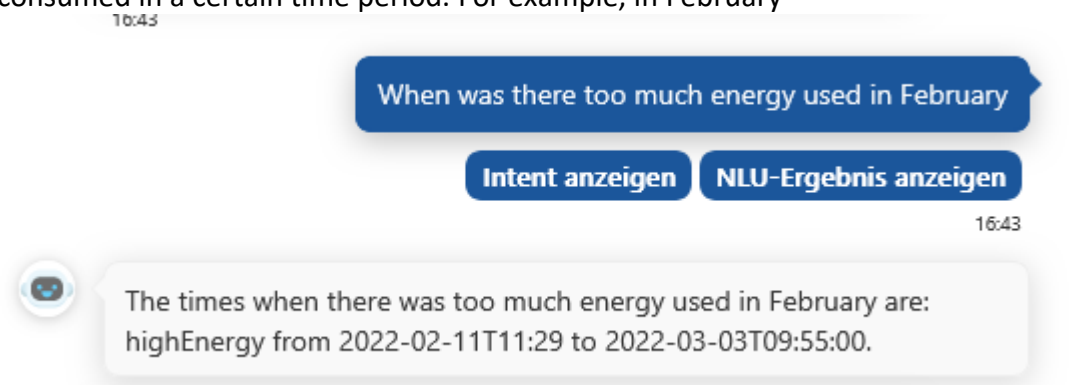


Figure 20: Screenshot of chatbot (is there too much energy used in February)

If there are no high energy events, the answer looks like this:

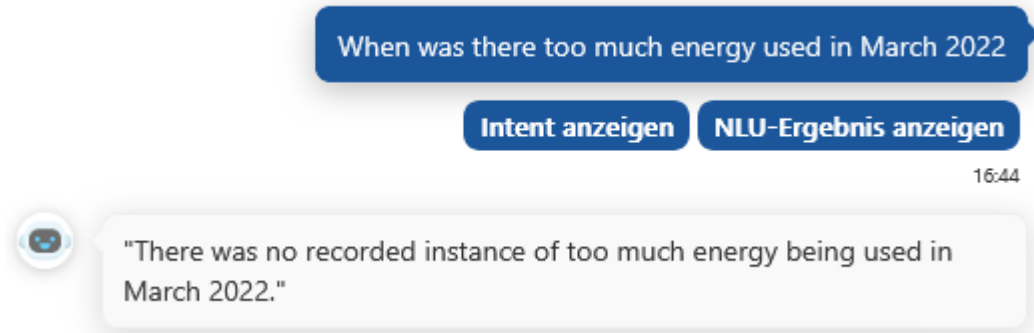


Figure 21: Screenshot of chatbot (Is there too much energy used in March 2022)

4.2 Identify the cause of too much energy being used in a specific event

Next, the facility manager would want to know why there was too much energy consumed at that time. The answer would look like this:

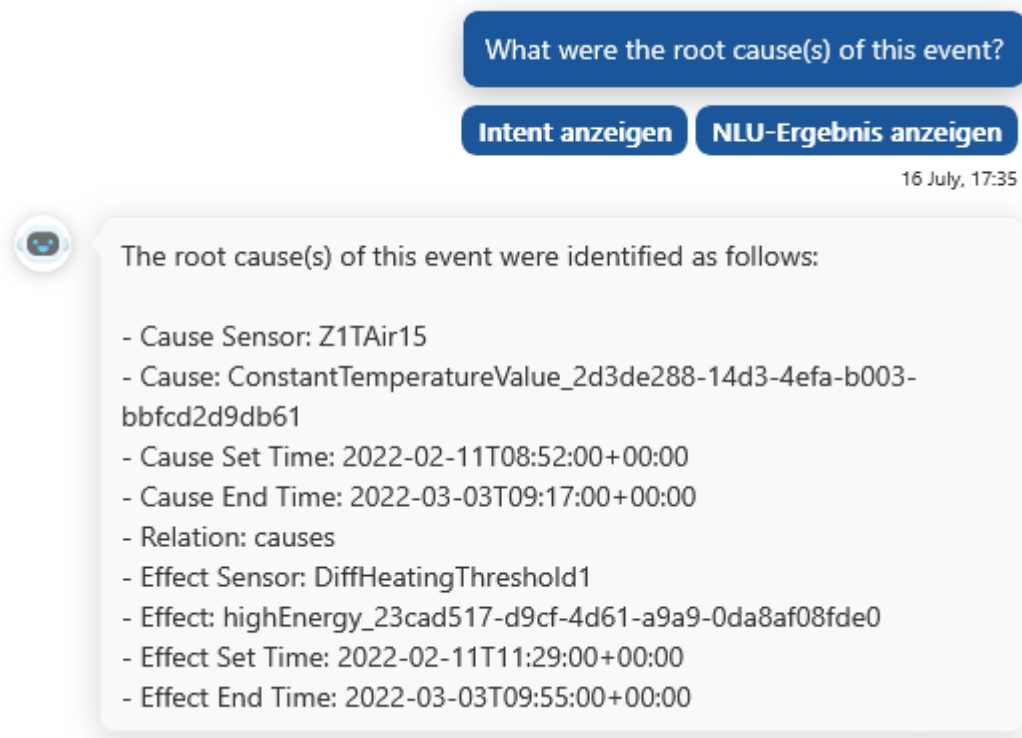


Figure 22: Screenshot of chatbot (Example answer for the case of a broken sensor)

The chatbot shows the sensor that caused the problem and the cause (=ConstantTemperatureValue). The sensor Z1TAir15 stayed at a constant temperature for a long period of time. The start and the end of the constant temperature event is also displayed, as is the period of the high energy event.

For the case of an open window, it would look like this:

```
2. Cause: windowOpen_5c6bbb3c-2882-4efa-8b69-966e72a8cee1
- Cause Sensor: Z1Wing14
- Set Time: 2022-04-11T21:11:00
- Cause Effect Time: 2022-04-13T08:04:00
- Effect Sensor: DiffHeatingThreshold1
- Effect: highEnergy_c095ff8f-e350-44b4-8a72-333c92297e51
```

Figure 23: Screenshot of chatbot (Example answer for the case of an open window)

4.3 Identify where an event happened

In addition, the facility manager could ask the chatbot where an event occurred. It will then identify the platform corresponding to the sensor where the event happened. In the example shown below, this is room 3.16 where a window has been left open.

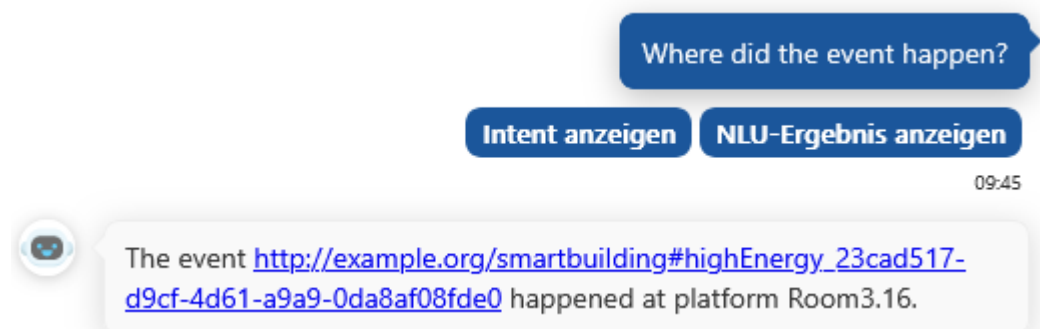


Figure 24: Screenshot of chatbot (Example answer for the location of an error)

5 Summary

This deliverable summarizes the smart building PoC which was developed within the SENSE project. The main deliverable in this context can be seen as the PoC itself.

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