NodeRed Smart Home

Petros Ellinas
Undergraduate student
Nat. Tech. University of Athens
Athens, Greece

Theodoros Sotirou
Undergraduate student
Nat. Tech. University of Athens
Athens, Greece

Nicolas Pigadas
Undergraduate student
Nat. Tech. University of Athens
Athens, Greece

***Abstract*—Nowadays, technological advances in the area of Information Systems enable us to better manage energy crises by providing more efficient and sophisticated tools for energy management and optimization. These tools can help energy companies and consumers to monitor and control their energy usage in real-time, identify energy waste and inefficiencies, and make more informed decisions about energy consumption and production. In this report, we present the steps to build such tools, that interact with each other, constituting a novel smart home system, via the use of open-source tools including node-red, apache kafka, grafana, telegram and mysql. We present the steps needed to replicate our project, alongside experimental results of a synthetic real life simulation. To implement our smart home solution we used Python to build our synthetic data, hypothetizing that they follow a random Markov Decision Process Distribution.**

***Index Terms*—Node-Red, Apache Kafka, Grafana, Telegram, mySQL, Python, Nodejs**

I. Introduction

Smart homes are the future of residential living. With the proliferation of the Internet of Things (IoT) devices and connected technologies, engineers have a significant role to play in designing and building smart home systems. A smart home system allows residents to automate various tasks such as controlling the temperature, lighting, security systems, and more, thereby enhancing their comfort and convenience.

One of the primary reasons why engineers need to build smart home systems is the growing demand for sustainability and energy efficiency. Smart home systems enable residents to monitor and control their energy consumption, leading to significant reductions in energy bills and greenhouse gas emissions. Engineers can also design and develop smart home systems that are energy-efficient, using renewable energy sources such as solar power, wind power, and geothermal energy.

Another significant advantage of building smart home systems is the enhanced security features they offer. Smart home systems incorporate advanced security features such as motion sensors, door sensors, and cameras, which enable residents to monitor their homes remotely. Engineers can also design smart locks and access control systems that enable homeowners to grant or deny access to their homes remotely. This technology not only enhances security but also provides homeowners with peace of mind and a sense of control over their homes.

In conclusion, smart homes are the future of residential living, and engineers have a crucial role to play in building smart home systems. Smart home systems offer numerous benefits, including energy efficiency, enhanced security, and increased comfort and convenience for residents. As the demand for smart home technology continues to grow, engineers need to design and develop smart home systems that are both functional and affordable, making this technology accessible to a wider range of people.

The rest paper is organised as follows:

- In section II we present the steps to replicate and setup our project
- In section III we present how we generated the simulation data and constructed the system of devices used in the simulation
- In section IV we show experimental results and we describe the scenario that describes the simulation
- In section V we conclude our project with some possible future applications and extensions of our project

II. Setup

A. Clone Project

First of all, git clone must be run using our repository's link [1]

B. NodeJS and server set-up

1) Install NVM with: curl https://raw.githubusercontent.com /creationix/nvm/master /install.sh — bash
2) Install Node JS by: nvm install 16
3) Navigate to project-folder/kafka-api and enter the command "npm install"

C. Apache Kafka

1) Downloading Apache Kafka: copy and paste this link to your browser [4]
2) Unzip Kafka to the desired folder with the command: tar zxvf kafka_2.13-3.0.2.tgz (cd /Kafka)
3) Create aliases to use kafka easier. Navigate to cd and vim .bashrc. Find the comment "some more aliases" and add the following lines:

kafka-aliases
start zookeeper
alias zookeeper="bin/zookeeper-server-start.sh config/zookeeper.properties"
start kafka

alias kafka=”bin/kafka-server-start.sh config/server.properties” topics commands alias list-topics=”bin/kafka-topics.sh –bootstrap-server localhost:9092 –list” alias describe-topic=”bin/kafka-topics.sh –bootstrap-server localhost:9092 –describe –topic” alias consume-topic=”bin/kafka-console-consumer.sh –bootstrap-server localhost:9092 –topic” alias create-topic=”bin/kafka-topics.sh –bootstrap-server localhost:9092 –create –topic”

alias delete-topic=”bin/kafka-topics.sh –bootstrap-server localhost:9092 –delete –topic” grafana alias grafana=”sudo systemctl start grafana-server localhost:9092 –describe –topic” alias consume-topic=”bin/kafka-console-consumer.sh –bootstrap-server localhost:9092 –topic” alias create-topic=”bin/kafka-topics.sh –bootstrap-server localhost:9092 –create –topic” alias delete-topic=”bin/kafka-topics.sh –bootstrap-server localhost:9092 –delete –topic” grafana alias grafana=”sudo systemctl start grafana-server”

Next we will configure Kafka and .bashrc with the following process:

1) Open nano /Kafka/kafka_2.13-3.0.2/config/zookeeper.properties , and change dataDir to the directory where data folder is. In our case dataDir= /Kafka/data
2) Open nano /Kafka/kafka_2.13-3.0.2/config/server.properties,
and change log.dirs to the directory where logs folder is. In our case log.dirs= /Kafka/logs
3) Create 2 folders in the Kafka folder you have just unzipped: ”data” and ”logs”

Then we will start zookeeper and kafka, and then we will create our topic. In order to do these actions we need to:

1) Open a terminal and navigate to ”cd /Kafka/kafka_2.13-3.0.0”. Execute the command ”zookeeper”
2) Open a new terminal and navigate again to ”cd /Kafka/kafka_2.13-3.0.0”. Execute the command ”kafka” when zookeeper is finished
3) Next we will create two topics with the commands ”create-topic smart-home –partitions 8 –replication-factor 1” and ”create-topic lamp –partitions 1 –replication-factor 1”

D. Database installation

We will use docker. It will host our MySQL database by executing the command: ”sudo snap install docker”.
We will follow the following process to install and configure the database:

1) Execute the command: docker run -p 3311:3306 –name smart-home-db -e MYSQL_ROOT_PASSWORD=root -d mysql:5.7
2) We will then install the database management tool $dbeaver-ce$, to simplify the process and visualize our database. We install it by: ”sudo snap install dbeaver-ce”
3) We connect with localhost’s database at port 3311, with user and password root using the tool and we create a database called smart_home
4) We then install flyway, with : ”wget -qO- https://repo1.maven.org/maven2/org/flywaydb/flyway-commandline/9.8.1/flyway-commandline-9.8.1-linux-x64.tar.gz — tar xvz sudo ln -s ‘pwd‘/flyway-9.8.1/flyway /usr/local/bin” . This is done in order to add our sql-table creation-code to the new database directly
5) We navigate into our project’s folder and we execute: ”/home/theo/flyway-8.4.1/flyway migrate -user=root -password=root -url=jdbc:mysql://localhost:3311/smart_home”

E. Python 3.8 installation

To install python and pip we runthe following commands in terminal:

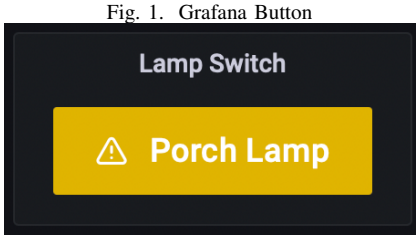
1) sudo apt update
2) sudo apt install software-properties-common
3) sudo add-apt-repository ppa:deadsnakes/ppa
4) We install python by ”sudo apt install python3.8”
5) Check if python is successfully installed with ”python3.8 –version” which should output ”Python 3.8.0”
6) Install pip with: ”sudo apt install python3-pip” in order to be able to install extra libraries
7) run ”pip3 install datetime”

F. Grafana

To install and use grafana follow the steps below:

1) enter the following command in terminal: ”wget https://dl.grafana.com/enterprise/release/grafana-enterprise_8.3.3_amd64.deb sudo dpkg -i grafana-enterprise_8.3.3_amd64.deb”
2) Then execute ”grafana” in terminal
3) Navigate to ”http://localhost:3000/” in browser
4) Input for username and password, ”admin” (default)
5) Import the dashboard json file from cloned project directory named ”$grafana - dashboard.json$”, by clicking the ”+” icon
6) The dashboard is illustrated in the General Tab

Fig. 1. Grafana Button



G. Node-Red Configuration

1) To install Node-Red we execute ”$sudo snap node - red$”
2) To start it we use ”$node - red.desktop - launch$”
3) When you navigate to ”$http : //localhost : 1880/$” in your browser you should see the node-red running
4) In order to import our flow in Node-Red, open ”hamburger menu” from the top-right of the page, then ”select

a file to import" and import the file "$node - red - flow.json$"

5) Navigate to "Manage palette" from "hamburger menu". Then go to install tap and install the following dependencies: "node-red-contrib-chatbot", "node-red-contrib-kafka-manager", "node-red-node-mysql", "node-red-dashboard"

1) Nodes Explained: In figure II-G1 we consume messages from a topic in Kafka and generate it into a node-red message. Each one of our sensors listen to a specific partition.

Fig. 2. Kafka consumer

The mysql node seen in II-G1 grants us basic access to our database. The msg.topic holds the query and the result is returned in msg.payload.

Fig. 3. MySQL

H. Telegram Installation

To install telegram you can follow the steps listed below:

1) Download telegram with snap install telegram-desktop
2) To start it we use "node-red.desktop-launch"
3) Follow the tutorials in [2] [3] as follows
4) Download telegram Desktop
5) Login to account
6) Create the bot: Send BotFather the message /newbot
7) Name the bot: Give it any name. This is how you will see it in Telegram.
8) Give the bot a username: Must be unique. Doesn't have to match the bot's name.
9) Grab the token: You will need this for configuring the nodes.
10) Send the bot a message: Just say hi or click the start button in the chat window.
11) Grab the chat ID: curl "https://api.telegram.org/bot$TOKEN/getUpdates" replacing $ TOKEN with your bot token.
12) Get chat ID from JSON response
13) Put chat ID to telegram conversation node.

1) Telegram nodes explained: The Telegram Receiver 4 node will receive every message sent in a specified chat. After the message is received the node chatbot-rules 5 will choose what action should be preformed based on the contents of the message (this node is analogous to a switch node).

The chatbot-message node in figure 6 receives a message object with various information about the chat and the message that needs to be sent. To send this message, the object chatbot-telegram-send is needed 7.

Fig. 4. Telegram Receiver)

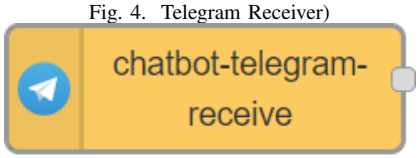


Fig. 5. Chatbot rules

2) Telegram-commands: Next we present the telegram commands

1) "/help": List available commands.
2) "/TH1" or "/TH2": Display latest temperature measurement from TH1 or TH2.
3) "/HVAC1" or "/HVAC2": Display latest energy measurement from HVAC1 or HVAC2 and the status of the device.
4) "/MiAC1" or "/MiAC2": Display latest energy measurement from MiAC1 or MiAC2 and the status of the device.
5) "/Etot": Display latest total energy measurement from Etot.
6) "/W1": Display latest water measurement from W1 and the status of the device.
7) "/alarm": Display status of the alarm.
8) "/HVAC1_comp" or "/HVAC2_comp": Change the status of the aircondition devices.
9) "/MiAC1_comp" or "/MiAC2_comp": Change the status of the other electric devices.
10) "/W1_comp": Change the status of the water consumption device.
11) "/alarm_comp": Arm or disarm alarm.

By typing this commands instructions are done by the bot

III. Simulation

In this section we will demonstrate the use case used to validate our setup:

Fig. 6. Chatbot message

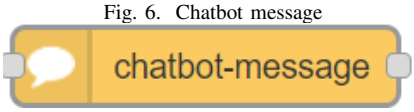
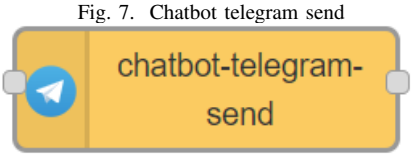


Fig. 7. Chatbot telegram send



A. Devices used

Device	Name	Value Type	Range
Temperature sensor	TH1	int	12-35
Temperature sensor	TH2	int	12-35
HVAC Energy Sensor	HVAC1	int	0-100
HVAC Energy Sensor	HVAC2	int	0-200
Energy Measurement	MiAC1	int	0-150
Energy Measurement	MiAC2	int	0-200
Total Energy Measurement	Etot	int	free
Movement	Mov1	binary	0,1
Water consumption	W1	float	0-1
Alarm	alarm	binary	0-1
Total water consumption	Wtot	int	free

B. Markov chains

Markov chains, also known as Markov processes, are a specific type of mathematical model that describes a system that transitions from one state to another based on the Markov property.

In a Markov chain, the future state of the system depends only on its current state and not on any previous states. This means that the probability of transitioning from one state to the next is fixed and determined by the current state, and is independent of the history of the system.

Markov chains are often represented using a directed graph, where the nodes represent the states and the edges represent the transitions between the states. Each edge is labeled with a probability, which represents the probability of transitioning from one state to the next.

The behavior of a Markov chain can be analyzed using a transition matrix, which is a square matrix that contains the probabilities of transitioning from each state to every other state. The entries of the matrix represent the probabilities of the transitions, and the rows and columns represent the current and next states, respectively.

C. Synthetic Data creation

Synthetic data were produced with python scripts. We used the concept of Markov chains described in the previous subsection, as every change of the sensors' state doesn't change with respect to the state of the previous timeslot. We also discretize time of horizon T, for every 15 minutes. Then data that belong to the timeslot are sent every 1 second. Concurrently, data are sent to an api which sends them to the correct topic of kafka, depending on which device is the sender. Then node-red process them in real time and take the required actions, saving them in our database

Finally, to simulate extreme events, we used conditions to trigger them in the exact time we planned, a topic that will be discussed next.

IV. Numerical Results

A. Scenario

1) The family is in the living room from 8:00 to 9:00 am in the morning so the movement sensor is triggered every time a person is moving
2) There is a malfunction in TH2 at day 6 at 16:00 to 16:30 and the alarm is triggered
3) There is a malfunction in MiAC2 at day 4 at 16:00 to 16:30 and the alarm is triggered
4) There is a malfunction in HVAC2 at day 3 at 16:00 to 16:30 and the alarm is triggered
5) There is a malfunction in W1 at day 6 at 16:00 to 16:30 and the alarm is triggered
6) Every 5 hours the W1 sensor is sending data from 1 day before
7) Every 30 hours the W1 sensor is sending data from 10 days before
8) On day 4 there is a burglary between 2:00 and 2:30 and the alarm is triggered
9) On day 2 load-shedding/power outage is spotted and the alarm goes on.
10) On day 6 at 16:00 to 16:50 sensors TH1,TH2,W1 have a malfunction and send out of range measurements
11) On day 4 at 16:00 to 16:50 sensors MiAC1,MiAC2 have a malfunction and send out of range measurements
12) On day 3 at 16:00 to 16:50 sensors HVAC1,HVAC2 have a malfunction and send out of range measurements

Alarm goes off when an extreme event is spotted and the family is alerted via telegram.

V. Conclusion

In this work, we built an end-to-end smart-home system with a testing simulation. The simulation was created in order to be as close to the real world as possible, accounting for many extreme situations and possible malfunctions of the system's components. This system can be used from individuals to monitor their water and electricity consumption. Additionally, demand response aggregators can use this system with some modifications to control and offer flexibility to the energy market.

A. Future work

There is still a lot of work to do to deploy the system in the real world. First of all, we must test the system using

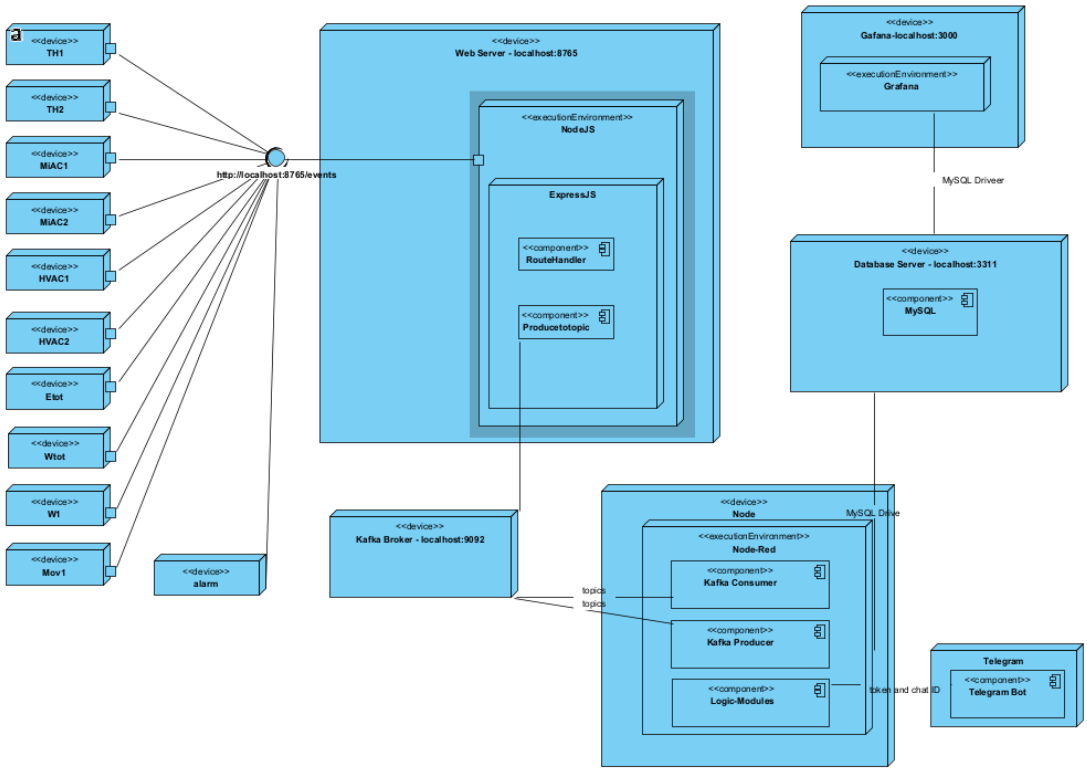


Fig. 8. Deployment Diagram for our project indicating devices and components of technology used

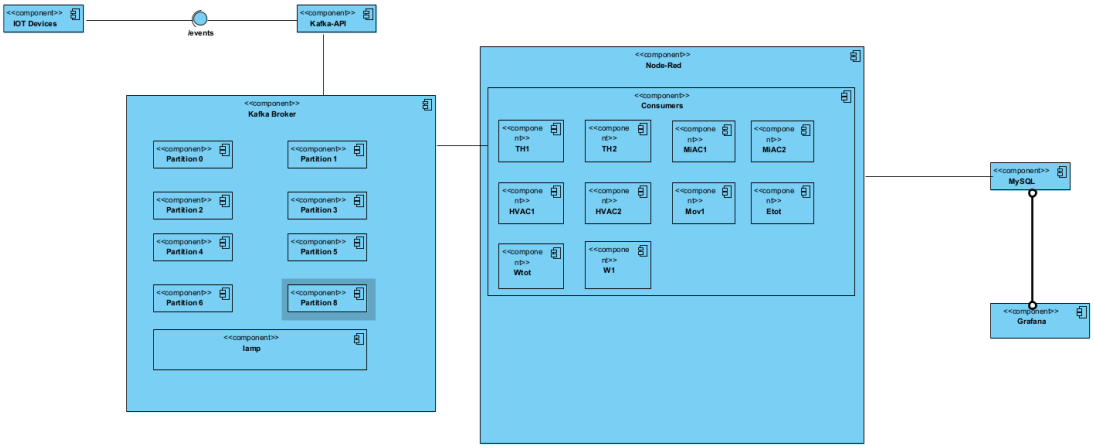


Fig. 9. Component Diagram for our project indicating components of technology used

real sensors. This can be done by building an Arduino-based system. Moreover, actuators and re-enforcement learning models can be used to control and ”shave” unnecessary power consumption by turning on and off devices, according to the user preferences. This can be integrated into node red. In order to commercialize our system, we must also investigate the regulations about smart-home systems. Finally, a really interesting topic is the scalability of the project and the way it can be organized and deployed in order to monitor and control effectively multiple smart houses across the world.

REFERENCES

[1] https://github.com/elpetros99/Node-Red-Smarthome
[2] https://sendpulse.com/knowledge-base/chatbot/create-telegram-chatbot
[3] https://core.telegram.org/bots6-botfather
[4] https://dlcdn.apache.org/kafka/3.0.2/kafka_2.13-3.0.2.tgz

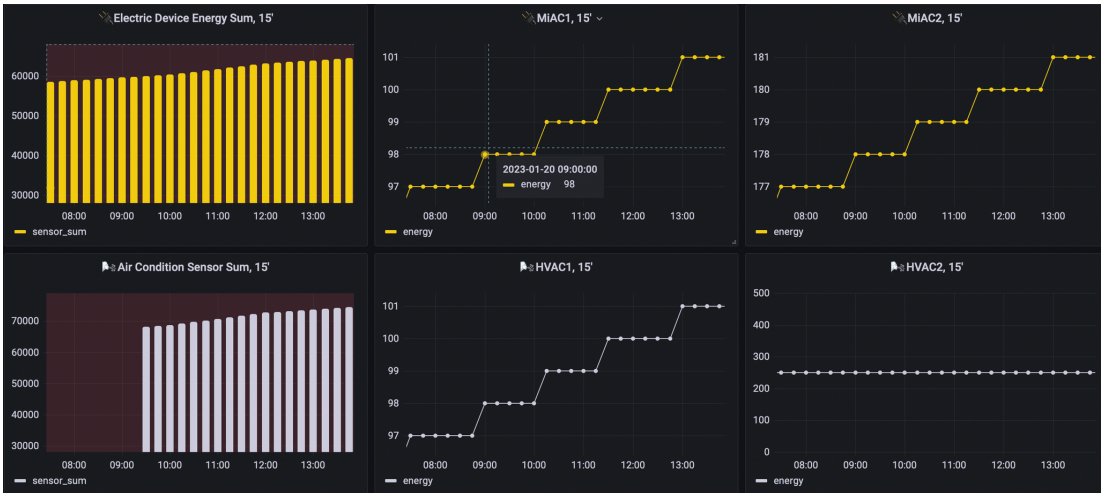


Fig. 10. Real time Grafana Plots for 15'-interval sensors

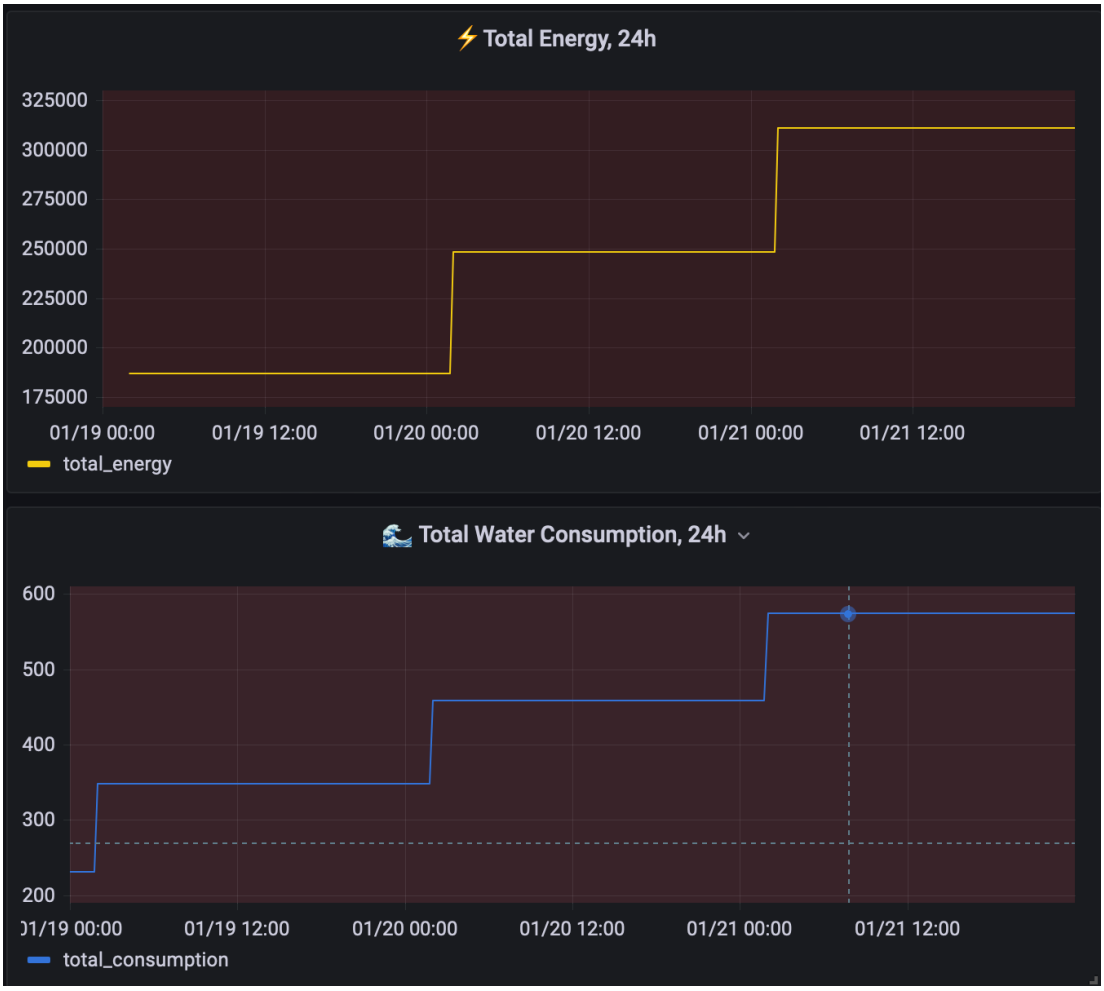


Fig. 11. Total Energy and Water Consumption in a day

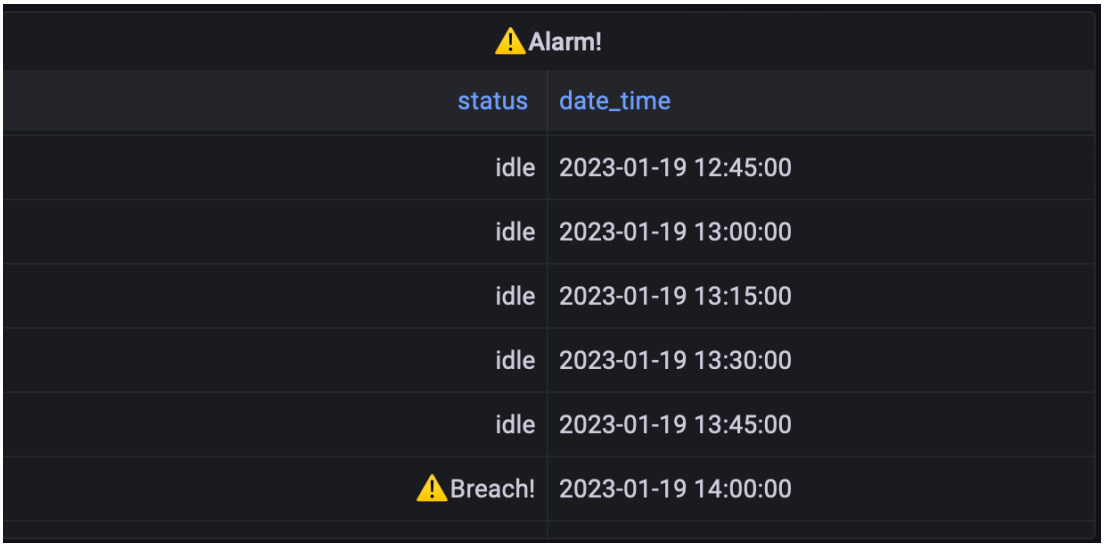


Fig. 12. Alarm notifications in Grafana

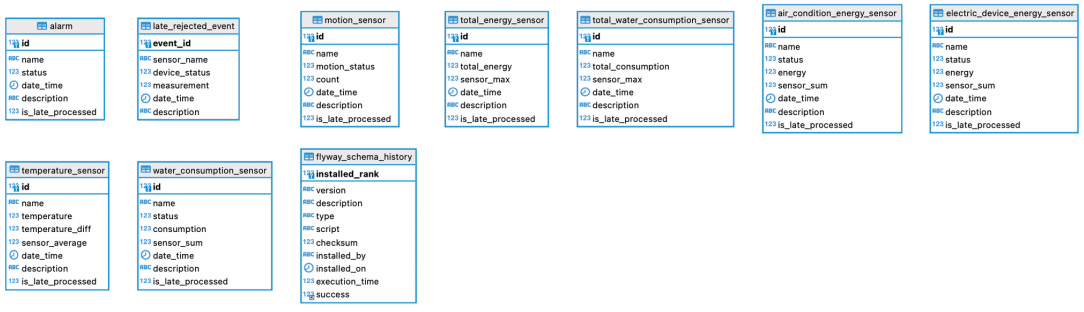


Fig. 13. Database ER Diagram