Real-Time Monitoring of Environmental Conditions

1 Introduction

In a context where comfort, safety, and energy efficiency are major priorities, the ability to monitor indoor environmental conditions in real-time is essential. This project aims to provide a technical solution for the automatic monitoring of several key parameters of an indoor environment, using data collected from simulated sensors and processed in real-time using modern technologies such as Apache Kafka, Faust and Panda.

2 Objective and Functional Extension

The application has been extended to process six different data streams: temperature, CO2, motion, lighting, noise, and the number of people in the room. This more comprehensive approach allows for the evaluation of a wide range of ambient conditions.

The application not only collects this data but also interprets it. It calculates the average temperature per room, signals high CO2 levels, cold rooms, insufficient lighting, excessive noise, and overcrowding. These signals are essential for an alert or automated control system.

3 Application Structure and Real Data

The code is organized into two main components. The `app.py` file contains the Faust processing logic, where agents that receive and interpret the data are defined. Each agent is dedicated to a specific type of information (e.g., temperature, sound, etc.).

The data generator is based on the `sensor_data.csv` file, which contains real information about room conditions, collected from a sensor system. This data is sent to Kafka by a Python script (`producer.py`) that reads each row and sends the corresponding data for each room and parameter.

4 Processing and Alerts

The system processes the following in real-time:

- **Average temperature** – Constantly updated and displayed per room.
- **High CO2 level** – Alert above 1000 ppm.

- **Cold room** – Alert below 18°C.
- **Low lighting** – Alert below 50 lux.
- **Loud noise** – Alert above 1.0 dB.
- **Overcrowding** – Alert when more than 3 people are in a room.

These evaluations allow for an immediate response from a user or an automated system.

5 Technologies Used

5.1 Kafka

Kafka is the platform that manages the transmission of messages between the data generator and the processing system. Each data stream is represented as a Kafka topic, which can be listened to and processed separately.

5.2 Faust

Faust is used for processing these streams in Python. Each type of information has a dedicated agent, which listens to the Kafka topic, receives the data, processes it, and generates alerts or internal states using persistent tables.

5.3 Python and Pandas

Pandas is used for reading the CSV file containing the real data. The values are extracted for each room and sent to Kafka by the `producer.py` script, which uses the `kafka-python` library.

6 Python Virtual Environment (venv)

To maintain a clean and controlled work environment, all the libraries required for the project are installed in a Python virtual environment. It is created using the command:

```
python -m venv venv
```

Then, the virtual environment is activated with:

```
source venv/bin/activate
```

All dependencies (Faust, pandas, kafka-python) are installed using

```
pip install -r requirements.txt}
```

This isolation ensures that the project does not interfere with other projects or system libraries.

7 Execution and Testing

The complete initialization of the application is done using an automation script (`start.sh`) which:

1. Starts the Kafka and Zookeeper services in Docker containers.
2. Creates the necessary Kafka topics.
3. Activates the Python virtual environment.
4. Runs the Faust application.
5. Starts the data generator.

This process ensures a fast startup and easy testing in any environment.

8 Conclusions

This project demonstrates how an indoor environment monitoring system can be implemented using real data and real-time processing. The scalability and modularity of the solution allow its integration into complex applications in fields such as home automation, smart buildings, or industrial spaces.

By using Faust and Kafka, the system benefits from a modern, resilient, and easily extendable architecture, capable of handling large data streams with minimal latency.

9 Bibliography

- Apache Kafka documentation – `https://kafka.apache.org/documentation/`
- Faust Stream Processing for Python – `https://faust.readthedocs.io/en/latest/`
- Docker Documentation – `https://docs.docker.com/`
- Kafka Python – `https://kafka-python.readthedocs.io/`
- Pandas Library – `https://pandas.pydata.org/docs/`