UNIVERSITÀ DEGLI STUDI DELL'AQUILA

DISIM Dipartimento di Ingegneria e Scienze dell'Informazione e Matematica

Department of Information Engineering,
Computer Science and Mathematics

University of L'Aquila, Italy

Final Report

Smart Home IoT System

Course:
Software Engineering for Internet of Things
(SE4IOT)

Supervisor:
Professor Davide Di Ruscio

Members:
Mete Harun Akcay
Thanh Phuc Tran
Pragati Manandhar

2025/2026, Fall Semester
January 12, 2026

Contents

1 Introduction

Smart homes use interconnected devices to improve comfort, safety, and energy efficiency through automation and real-time monitoring. In this project, we developed an IoT-based smart home system that simulates sensor data for multiple rooms, enables real-time monitoring and historical analysis, and notifies the homeowner when abnormal situations such as smoke detection or unusual environmental values occur.

2 Objectives

The objectives of this project were to:

- Simulate smart home sensors, including temperature, humidity, window status, and smoke detection, with configurable numbers of rooms and devices.
- Enable reliable communication between simulated devices and the system using a publish–subscribe messaging approach.
- Collect and store all sensor measurements in a time-series data store to support real-time observation and historical analysis.
- Provide a user-friendly, web-based dashboard that displays live sensor values and trends over time.
- Detect abnormal conditions based on configurable thresholds and automatically notify the homeowner through instant messaging.
- Design the entire system so that it can be easily deployed and executed using containerized components in a modular architecture.

3 Functional Requirements

ID	Name	Description
FR-1	Sensor Simulation	The system simulates smart home sensors including temperature, humidity, window state, and smoke detection using a Python-based simulator. Sensor values follow realistic ranges, evolve over time, and each sensor is uniquely identified by room ID, sensor type, and sensor ID.
FR-2	Communication Protocol	The system uses MQTT as the main communication protocol. Sensor data is published to structured topics following the format `/home/<room_id>/<sensor_type>/<sensor_id>`.
FR-3	Data Processing	Incoming sensor data is processed using Telegraf, which subscribes to MQTT topics and parses data automatically based on the topic structure. Node-RED performs higher-level processing such as anomaly detection and triggering notifications.
FR-4	Data Storage	All sensor measurements are stored in a time-series database with appropriate metadata, including timestamps, room ID, and sensor ID, enabling historical analysis and monitoring.
FR-5	Data Visualization	The system provides interactive dashboards that display real-time and historical sensor data using charts and graphs. Users can filter data by room, sensor type, and time range.
FR-6	Alerting Mechanism	The system detects abnormal conditions based on predefined thresholds, such as high temperature or smoke detection, and sends alerts to the homeowner via Telegram with information about the affected room and sensor.

Table 1: Functional Requirements

4 Non-Functional Requirements

ID	Name	Description
NFR-1	Portability	The system is fully containerized using Docker Compose, ensuring consistent deployment and execution across different environments and machines.
NFR-2	Scalability	The system supports adding new rooms and sensors by modifying configuration files, without requiring changes to the simulator source code.
NFR-3	Resilience	The system includes error handling mechanisms to manage temporary communication failures and ensure continuous operation of the simulation and data pipeline.
NFR-4	User-Friendly Design	The dashboards provide clear and intuitive real-time visualizations that allow users to easily understand sensor behavior and system status.
NFR-5	Security	Access to the MQTT broker, time-series database, and visualization dashboard is protected using authentication credentials to prevent unauthorized access.

Table 2: Non-Functional Requirements

5 Technologies Used

5.1 Python (Simulator)

Python is used to implement the smart home simulator responsible for generating synthetic sensor data. The simulator produces realistic values for temperature, humidity, window state, and smoke detection for multiple rooms and sensors, where the number of rooms and sensors can be configured without modifying the source code. The `paho-mqtt` library is used as an MQTT client to publish sensor measurements to the MQTT broker using structured topics of the form: `/home/<room_id>/<sensor_type>/<sensor_id>`.

5.2 Mosquitto (MQTT Broker)

Mosquitto is used as the MQTT broker for message exchange within the system. It enables lightweight, reliable, and asynchronous communication between the simulated sensors and the downstream components. All system components subscribe to the relevant MQTT topics in order to receive sensor data in real time.

5.3 Telegraf (Data Collector)

Telegraf acts as the data ingestion layer of the system. Using its MQTT consumer input plugin, Telegraf subscribes to sensor topics published by the simulator, automatically parses the topic structure to extract metadata such as room ID and sensor ID, and forwards the collected measurements to the time-series database for storage and analysis.

5.4 InfluxDB (Database)

InfluxDB is used as the time-series database for storing all sensor measurements generated by the system. It is optimized for high write throughput and efficient querying of time-stamped data, making it suitable for IoT workloads. Each measurement is stored together with timestamps and tags such as room ID and sensor ID, enabling both real-time monitoring and historical analysis.

5.5 Grafana (Dashboard)

Grafana is used to visualize the sensor data stored in InfluxDB. Interactive dashboards are created to display real-time and historical sensor values using charts and graphs. Users can filter data by room, sensor type, and time range, providing a clear and intuitive overview of the smart home environment. An example dashboard is given below.

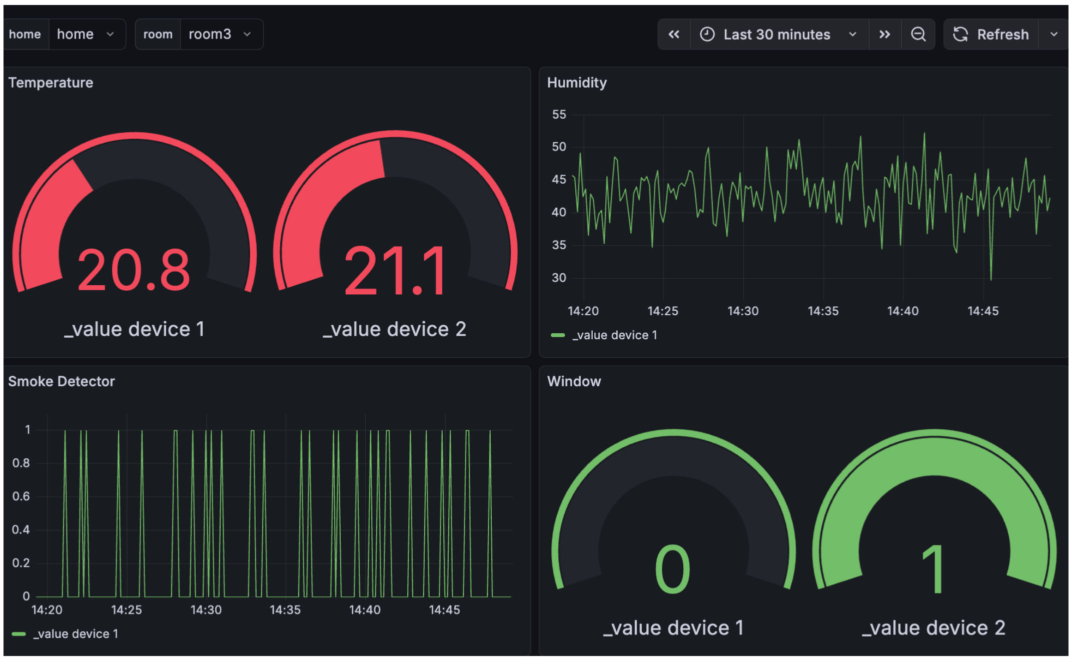

Figure 1: Data Visualization (Grafana)

5.6 Node-RED and Telegram (Automation and Notification)

Node-RED is used for higher-level data processing and automation. It retrieves sensor data from the database and evaluates it against predefined thresholds to detect abnormal situations, such as smoke detection or unusually high temperatures. When such conditions are detected, Node-RED triggers notifications that are sent to the homeowner via Telegram. Telegram bots are used to deliver alert messages containing information about the affected room, sensor type, and detected condition, enabling timely user awareness and response.

Node-RED

6 System Architecture

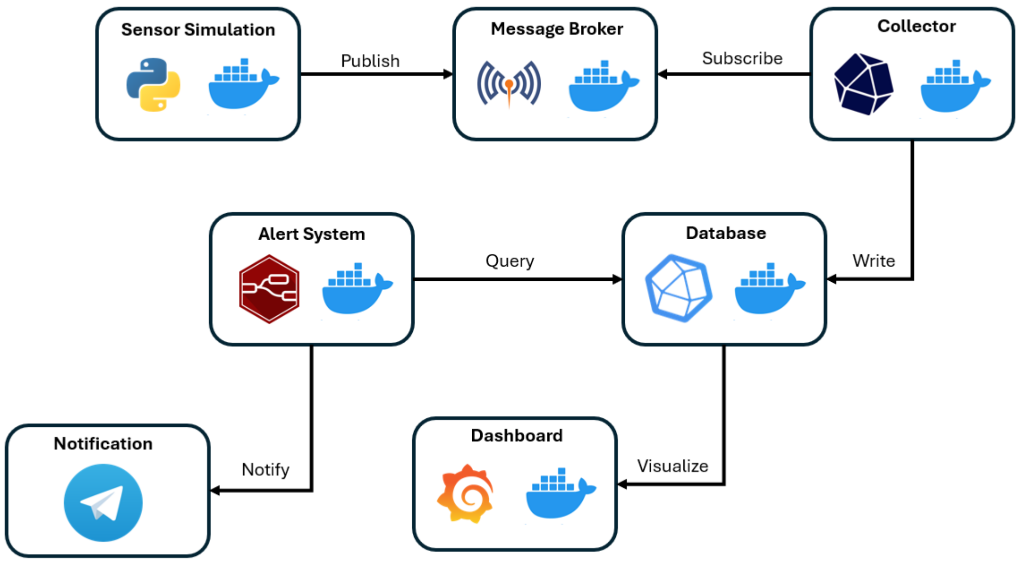

Figure 2: System Architecture

7 Conclusion

In this project, we designed and implemented a complete smart home IoT system that demonstrates how sensor data can be generated, transmitted, stored, visualized, and monitored in an integrated environment. The system shows how modular and containerized components can be combined to build a scalable and practical smart home solution with real-time monitoring and alerting capabilities.