University of Pisa

Department of Information Engineering

Internet of Things

FireGUARD

Fire, Gas & Ultrafine Air Real-time Detector

Student

Fabio Malloggi

Academic Year: 2023-2024

Contents

1 Introduction **2**
1.1 FireGUARD: Project objectives and motivations . . . 2

2 Use Case Description **3**
2.1 System Components . . . 3
2.2 Air quality parameters and safe levels . . . 3
2.2.1 Detailed Sensor Parameters Description . . . 4

3 System Design and Architecture **5**
3.1 Wireless Sensor Network . . . 6
3.1.1 Working Principles . . . 6
3.1.2 Devices functionalities breakdown . . . 6
3.1.3 CoAP-Based Communication Framework . . . 8
3.2 Cloud Infrastructure . . . 10
3.2.1 Cloud Server . . . 10
3.2.2 User Interaction: Remote Control Application and Grafana . . . 10
3.2.3 Database . . . 11

4 Simulations **12**
4.1 Smart Smoke Detector Simulator . . . 12
4.2 Smart Vent Simulator . . . 12

5 Artificial Intelligence Model **13**
5.1 Data Collection . . . 13
5.2 Preprocessing . . . 14
5.2.1 Data Cleaning . . . 14
5.2.2 Exploratory Data Analysis . . . 15
5.2.3 Data Reduction . . . 17
5.2.4 Data Splitting and Normalization . . . 17
5.3 Deep Learning Model: FeedForward Neural Network . . . 18
5.4 Implementation on IoT Device . . . 19

6 Sensor Data Representation: SenML **20**
6.1 Smart Smoke Detector Measurement . . . 20
6.2 Smart Smoke Detector Environment Status . . . 21

7 Grafana Web Application **22**

8 References **23**

1 Introduction

Smoke detectors are nowadays a fundamental element in public and private buildings, improving safety in everyday life. They are devices designed to detect early signs of combustion such as smoke, heat, or gas emissions and activate alarms or automated responses to contain it as soon as possible before it becomes a life threat.

Over the last decades they have dropped fire-related fatalities, in particular in Europe and North America, despite an increasing and aging population and a significant rise in the use of combustible materials [1].

However, many devices are not working properly or even not working at all. In the U.S., "approximately 20% of homes with smoke alarms have non-operational smoke alarms. It is estimated that if every home had working smoke alarms, U.S. residential fire deaths could drop by 36% ($\tilde{1}$100 lives saved per year.)" [2]. In addition, false alarms have become a dramatic waste of resources and a serious problem for the few firefighters available on the territory. In 2020, between 40 and 44% of fire alarms in England were claimed to be false [3].

The advent of artificial intelligence and its easy and cheap worldwide access, has lifted the standard for automated recognition efficiency in a way that those statistics can no more be accepted.

As western countries try to develop the Internet of Things, and in particular Fog Computing, it is clear that those issues represent a test bench for the correctness and proper implementation of its founding principles.

1.1 FireGUARD: Project objectives and motivations

The aim of this project is to develop an IoT application intended to support the complete integration of a smart fire detector and smoke alarm into an air quality control system.

More accurately, it defines an Air Quality Control System offering monitoring and actuating services over a custom and mutable grid of devices that includes sensors, fog devices capable of locally process data coming from sensors and actuators able to activates specific mechanical ventilation as well as water fire suppression.

The main idea behind the project is to bring down, using an AI-powered IoT system, the percentage of false alarm requests generated by existing fire detector systems while improving the functioning of ventilation systems, particularly in hazardous environments. In fact, in the event of a fire, activation of the correct ventilation system is mandatory, determining whether it helps or worsens the situation.

Concerning this project, two kind of ventilation systems are considered:

1. A Smoke Control Ventilation System [4], that sucks out smoke and let low level fresh air inflow during a fire to keep escape routes clear and assist firefighting.
2. An Hazard Ventilation system, that activates when dangerous pollutants—like fine particles or gases—are detected, improving air quality through filtration and high fresh air exchange, without a fire being present.

Therefore, a smart fire detector equipped with AI not only monitors early signs of fire —such as smoke, heat, and gas emissions— but can also intelligently distinguish between an actual fire and a dangerous concentration of volatile compounds without combustion.

2 Use Case Description

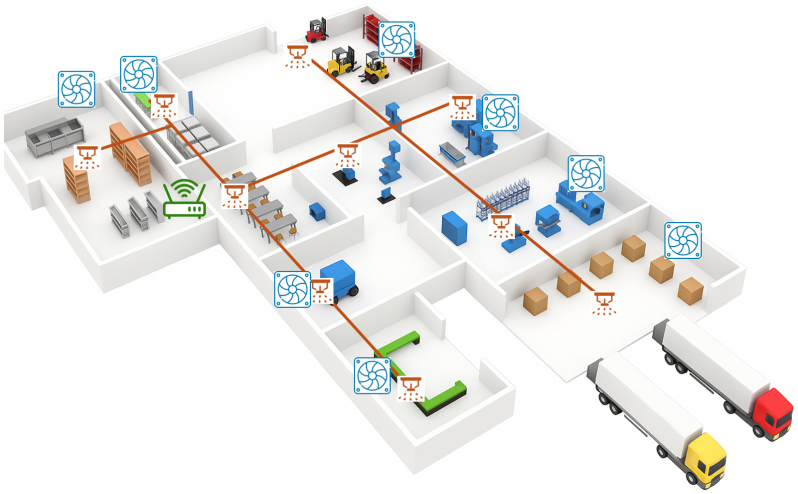


Figure 1: real-world scenario example use case.

The image depicts a real-world industrial scenario where an IoT-based environmental monitoring and safety system can be effectively deployed. Each room in the facility is equipped with two categories of smart devices, smart smoke detectors and air ventilation controls. Both are carefully designed and strategically installed to continuously assess air quality and ensure safe operating conditions.

For clarity, the operation of the system is illustrated considering a single room as an example monitoring area.

2.1 System Components

- **Border Router Device**
  This component connects the IoT sensor network to the Internet, enabling communication between the local devices and the remote user application.

- **Smart Smoke Detectors devices**
  low-power sensor devices with slightly smaller power and computational constraints. Equipped with specialized air quality sensors, they continuously monitor environmental parameters such as particulate concentration, gas levels, and volatile organic compounds (VOCs). Their primary function is to detect, supported by an Artificial Intelligence model, fire emergency or hazardous conditions, trigger appropriate response measures, and alert occupants within the monitored area.

- **Smart Ventilation devices**
  low-power actuator devices with stricter resource constraints. Unlike smoke detectors, they do not perform sensing or computation but instead receive environmental data and alerts from other devices. Based on these inputs, they activate safety response systems —such as alarms, smoke control or filtering ventilation mechanisms—, ensuring a coordinated reaction to detected fire or hazard risk.

2.2 Air quality parameters and safe levels

To ensure a safe and healthy indoor environment, several air quality parameters must be monitored and kept within recommended thresholds. The following table presents typical safe levels for particulate matter—the most hazardous airborne pollutant—based on established health guidelines.

Parameter	Safe Level	Unit
Air Temperature	–	°C
Air Humidity	–	% RH
Air Pressure	–	hPa
TVOC (Total Volatile Organic Compounds)	-	ppb
Raw H_2	-	ppm
Raw Ethanol	-	ppm
PM 1.0	<250	µg/m^3
PM 2.5	<250	µg/m^3
NC 0.5 (Particles <0.5 µm)	<10,000	particles/cm^3

Table 1: Thresholds for Air Quality Monitoring based on typical indoor standards.

2.2.1 Detailed Sensor Parameters Description

Air Temperature

Measures the ambient temperature of the surrounding air.

Relative Humidity (RH)

Measures water vapor in the air, relative to the air temperature. Specifically, it is the ratio, expressed as a percentage, between the current amount of water vapor in the air and the maximum amount the air can hold at the same temperature. It indicates how close the air is to saturation, with 100% representing fully saturated air.

Air Pressure

Measures the atmospheric pressure in the environment, reported in hectopascals (hPa). It represents the force exerted by the weight of the air above a given point and typically ranges from 980 hPa to 1050 hPa at sea level.

TVOC

Total Volatile Organic Compounds (TVOC) measures the total concentration of volatile organic compounds present in the air, reported in parts per billion (ppb). VOCs are emitted as gases from various sources such as paints, cleaning products, and industrial processes. High TVOC levels can indicate poor air quality and may affect human health.

Raw Hydrogen (H_2)

Raw sensor output representing molecular hydrogen levels, expressed in part per million (ppm). These values are unprocessed and not compensated for factors like temperature or sensor bias, making them useful for low-level diagnostics and sensor calibration.

Raw Ethanol

Indicates the raw measurement of ethanol gas present in the air. Like raw H_2, this is a raw output without environmental compensation, often used internally by algorithms to estimate air quality or for further processing.

PM 1.0 / PM 2.5

Refers to the mass concentration of particulate matter in the air, measured in micrograms per cubic meter (µg/m^3). PM1.0 includes particles smaller than 1.0 micrometer, while PM2.5 includes particles smaller than 2.5 micrometers. These particles can come from combustion, dust, smoke, and other pollutants, and pose health risks when inhaled.

NC0.5

Represents the number concentration of airborne particles smaller than 0.5 µm. It is measured in particles per cubic centimeter (particles/cm^3), and give a count-based view of particulate pollution.

3 System Design and Architecture

This project employs a Wireless Sensor Network (WSN) to implement the previously defined use case. At its core are three Nordic Semiconductor nRF52840 Dongles, each fulfilling a distinct role: Border Router, Smart Vent, and Smart Smoke Detector. All devices run Contiki-NG, a lightweight operating system tailored for low-power, resource-constrained environments.

The devices communicate using the Constrained Application Protocol (CoAP), which supports real-time updates through its observe mechanism.

In the cloud, a Cloud Server and a Remote Control Application provide the necessary monitoring, storage, and control functionalities. The former collects and stores sensor data from the Wireless Sensor Network in a structured MySQL database, which can be graphically visualized through Grafana. The latter provides a web interface for monitoring and control, allowing users to assess environmental conditions and issue commands to actuators when needed.

A diagram summarizing the complete system is presented below, followed by a detailed breakdown of each system component's design and function.

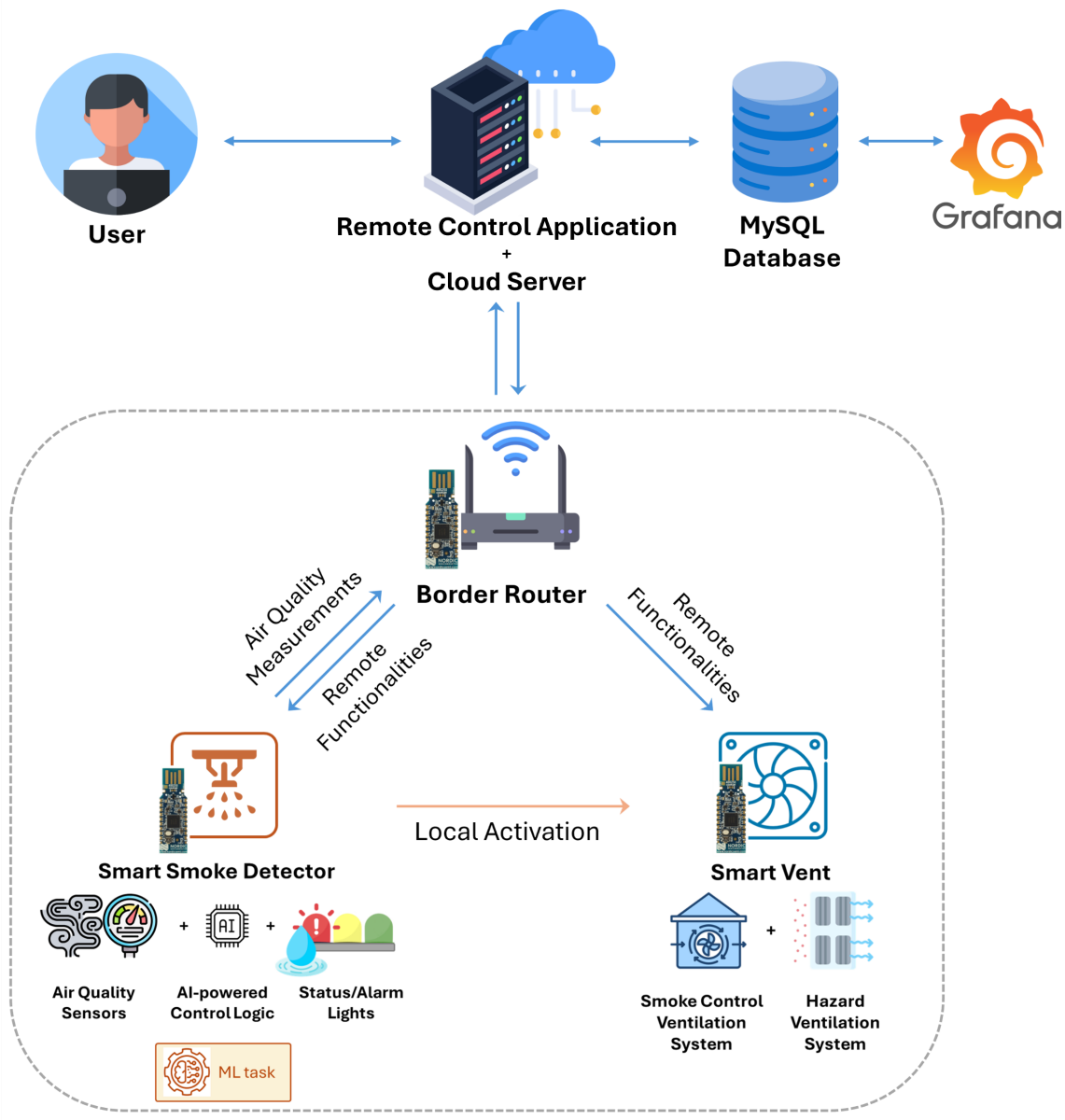


Figure 2: System architecture

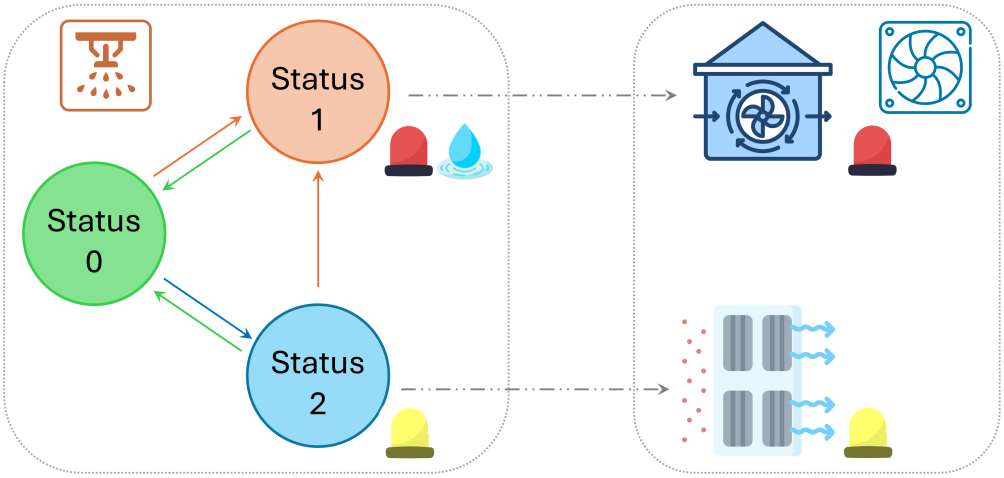
3.1 Wireless Sensor Network

This subsection provides a comprehensive overview of the internal architecture and functional logic of the Wireless Sensor Network (WSN), detailing the specific roles and operational behaviors of each device within the system. It concludes with a description of how the CoAP protocol is employed to implement communication across the network in support of the intended use case.

3.1.1 Working Principles

The Wireless Sensor Network operates through three distinct execution states, each reflecting a particular environmental condition (normal, fire, or hazard). These states determine the system's behavior, influence the response strategies of the devices, and trigger relevant control actions. The table below summarizes the functionality associated with each operational state.

Status	Alarm Lights	Description
0 (Normal)	None	All air quality parameters are within safe limits. It is safe to remain in the monitored area.
1 (Hazard Risk)	Yellow LED	One or more dangerous parameters (PMs) exceed safety peak limits. Short exposure is advised to allow the ventilation system to stabilize conditions.
2 (Fire Emergency)	Red LED	Fire detected. Follow fire emergency protocols and evacuate the building.



3.1.2 Devices functionalities breakdown

This section presents a detailed functionalities breakdown of the main devices involved in the WSN, highlighting their effect and behaviors depending on the network state.

Border Router Device

Functionality	Activation	Effect	Description
RPL Routing Root	always	Connectivity	It provides external connectivity for the CoAP network, enabling integration with the Remote Control Application and Cloud Server. All collected measurements are continuously transmitted to the latter, which stores them and made them available for real-time monitoring and further analysis.

Smart Smoke Detector device

Functionality	Activation	Effect	Description
1 Hz frequency sensing	Status: 0/1/2	environment parameters assessment	It performs continuous environmental monitoring at a fixed 1 Hz sampling rate, ensuring reliable, real-time updates for all air quality parameters every second. This high-frequency sampling is essential to guarantee the fastest possible response to potential fire detection scenarios
N-measurements circular buffer storage	Status: 0/1/2	N-sec frequency notification	To reduce energy consumption and extend device lifespan, each air quality sensor uses a circular buffer to store measurements. Instead of transmitting every reading, the system sends updates to CoAP subscribers (e.g. Cloud Server) at an N-second interval, where 'N' is determined based on constraints such as maximum packet size. This N-sec notification strategy minimizes transmission frequency while still supporting on-demand access to the latest measurement when requested.
Hazard Air Quality Detection	Status: 0/2	POS: Status = 2 NEG: Status = 0	Whenever any health-critical parameter exceeds its defined safety threshold, the system sets the status to 2, transitioning to the Hazard Risk state. If all parameters fall back below safe limits, the status is reset to 0, indicating a return to the Normal condition state. It is important to note that no such checks are performed during the Fire Emergency state, where the system's priority is solely focused on handling the detected fire.
Fast Change Detection	Status: 0/2	AI model activation	Regardless of whether the system is in a Normal or Hazard Danger state, each new sensor reading is compared against the most recent M stored measurements (with $M \leq N$) for each parameter. The AI-based fire detection model is triggered only if the difference for at least one parameter exceeds a predefined threshold specific to that parameter. This approach significantly reduces the number of times the AI model is executed on the sensor, conserving computational resources and minimizing energy consumption due to the model's high processing demands.
AI-powered Fire Ignition Detection	Status: 0/2 and fast change detected	POS: Status = 1 NEG: -	Whenever a rapid change is detected, the current environmental parameters are fed into the AI model to infer a fire detection result. If a fire is identified, the system sets the status to 1, transitioning to the Fire Emergency state.
Water Sprinkler	Status: 1	Fire Suppression	The water sprinkler system is activated immediately to suppress any detected fire ignition as quickly as possible.
Light Alarm System	IF State = 1 IF State = 2	Yellow Light Red Light	The alarm system activates the corresponding light based on the detected environmental condition.

Smart Vent device

Functionality	Activation	Effect	Description
Hazard Ventilation Control	Status: 1	Air filtering activation	Directly triggers the Hazard Ventilation System to activate immediately, bypassing remote control.
Smoke Ventilation Control	Status: 2	Smoke dissipation activation	Directly triggers the Smoke Ventilation System to activate immediately, bypassing remote control.
Light Alarm System	IF State = 1 IF State = 2	Yellow Light Red Light	The alarm system activates the corresponding light based on the detected environmental condition.

3.1.3 CoAP-Based Communication Framework

The CoAP protocol forms the backbone of the Wireless Sensor Network's communication layer. It enables efficient and low-latency data exchange specifically tailored for constrained devices, making it well-suited for this type of application.

CoAP's observe feature enables both devices and servers in the system to register for real-time updates without relying on continuous polling. This event-driven mechanism ensures efficient communication, reduces power consumption, and supports responsive interaction across the network.

The following section describes the exposed CoAP resources for each device and their respective roles within the system.

Smart Smoke Detector Device
Acting as a CoAP server, this device exposes observable resources that represent both environmental measurements and system status. These resources are accessible to the Smart Vent, the Remote Control Application, and the Cloud Server. The Cloud Server, in particular, observes all available resources to receive continuous updates.

Resource	Obs.	Methods	Parameter	Description
/status	Yes	GET	-	Returns current environment status code
/temp	Yes	GET	n (int)	Returns at most n latest temperature (°C) values
/hum	Yes	GET	n (int)	Returns at most n latest humidity (%) values
/pressure	Yes	GET	n (int)	Returns at most n latest pressure (hPa) values
/tvoc	Yes	GET	n (int)	Returns at most n latest TVOC (ppb) values
/raw_h2	Yes	GET	n (int)	Returns at most n latest raw H_2 (ppm) values
/raw_eth	Yes	GET	n (int)	Returns at most n latest raw Ethanol (ppm) values

Resource	Obs.	Methods	Parameter	Description
/pm1_0	Yes	GET	n (int)	Returns at most n latest PM1.0 (µg/m^3) values
		POST/PUT	limit (int)	Updates the stored peak threshold (µg/m^3) used to detect hazardous environmental conditions.
/pm2_5	Yes	GET	n (int)	Returns at most n latest PM2.5 (µg/m^3) values
		POST/PUT	limit (int)	Updates the stored peak threshold (µg/m^3) used to detect hazardous environmental conditions.
/nc0_5	Yes	GET	n (int)	Returns at most n latest NC0.5 (particles/cm^3) values
		POST/PUT	limit (int)	Updates the stored peak threshold (particles/cm^3) used to detect hazardous environmental conditions.

Table 2: Resources exposed by Smart Smoke Detector

Smart Vent device
Operating both as a CoAP server (for remote commands) and as a CoAP client (towards the Smart Smoke Detector), this device subscribes to observe notifications on the `status` resource to monitor the environmental state. Based on the received updates, it autonomously reacts to critical conditions by activating ventilation mechanisms.

Resource	Obs.	Methods	Parameter	Description
/vent	No	POST/PUT	system = filter|smoke mode = on|off	Controls the activation or deactivation of the Hazard or Smoke Control Ventilation System.
/obs_status	No	POST/PUT	mode = on|off	Simulate the start-up or shutdown of the entire Ventilation System by toggling its observation of the Smart Smoke Detector's `status` resource.

Table 3: Resources exposed by Smart Vent

3.2 Cloud Infrastructure

This section describes the top layer of the system architecture, responsible for centralized data management and user interaction. The cloud infrastructure consists of three main components: the Cloud Server, which handles communication and data collection from the Wireless Sensor Network; a structured MySQL relational database, which ensures persistent and organized storage of all sensor measurements; and the user interface tier, composed of the Remote Control Application and Grafana dashboard, which enable monitoring, control, and visualization.

3.2.1 Cloud Server

The Cloud Server acts as the core data aggregator for the Wireless Sensor Network. It establishes CoAP-based communication with Smart Smoke Detector devices, observing all available sensing and status resources. Sensor data is systematically collected and inserted into a MySQL relational database, enabling structured, long-term storage. This setup ensures data integrity, supports scalable historical analysis, and enables smooth integration with external visualization and control layers.

3.2.2 User Interaction: Remote Control Application and Grafana

This interface layer provides access to system data and remote control functionalities. The Remote Control Application offers a command-line interface (CLI) that allows users to interact with the system, monitoring air quality, querying historical data, updating safety thresholds, or controlling ventilation mechanisms. Meanwhile, Grafana connects directly to the underlying database to present both historical trends and live updates through intuitive dashboards. These tools together support real-time monitoring, enhance usability, and promote proactive environmental management.

```
================================================================================
                  Welcome to FireGUARD IoT System (Remote)
================================================================================

Commands:
  help                          - Show this help
  show devices                  - Show devices info and resources
  show safety                   - Show current safety levels
  query <sensor> (<n>)          - Query DB last 'n' sensor measurements
  query status (<n>)            - Query DB last 'n' status records
  dev <sensor> (<n>)            - Query dev last 'n' sensor measurements
  dev status                    - Query dev current environment status
  daily hazard levels           - Show the daily average of hazard parameters
  set safety <param> (<value>)  - Set levels by given (or default) parameters
  start <filter|smoke> vent     - Start ventilation
  stop  <filter|smoke> vent     - Stop ventilation
  exit                          - Exit the server

================================================================================
fireGUARD >
```

Figure 3: Remote Control Application CLI and Welcoming

Below is a list of the supported commands within the Remote Control Application Server:

Operation	**Description**
`show`	Display a list of all supported operations
`show devices`	List all registered devices in the WSN with their descriptions
`show safety`	Show current hazard threshold values for PM levels
`query <sensor> (<n>)`	Retrieve up to `n` latest measurements (default 10) of a given parameter from the database
`query status (<n>)`	Retrieve up to `n` latest environment status changes (default 10) from the database
`dev <sensor> (<n>)`	Send a GET request to a device to obtain up to `n` latest measurements (default 10)
`dev status`	Send a GET request to a Smart Smoke Detector device to retrieve its current environment status
`daily hazard levels`	Return the daily average values for each monitored hazard parameter
`set safety <param> (<value>)`	Send a POST request to update the safety threshold for a specified hazard parameter
`start|stop <filter|smoke> vent`	Trigger or stop the filter or smoke ventilation system via a POST request to Smart Vent devices

3.2.3 Database

To support efficient and persistent storage of sensor data, the system relies on a normalized MySQL relational database. This structure is optimized for time-series analysis, rapid queries, and easy integration with visualization tools like Grafana.
A master table stores metadata for each registered IoT device, such as device identifiers, base URIs, ports, and exposed resources. Separate tables are defined for each measurement type (e.g., temperature, humidity, particulate matter). These tables link to the device metadata via device id and include fields for measurement values, device-generated timestamps, and computed absolute timestamps for consistency across the system.
The tables below illustrate the schema structure along with a brief description of each field:

Table 4: Table: WSN IoT Devices

Field	**Data Type**	**Description**
id (PK)	INT	Unique identifier for the device
base_uri	VARCHAR	CoAP base URI of the device
port	INT	Communication port for CoAP requests
resources	TEXT	JSON-encoded list of resources exposed by the device

Table 5: Example Sensor Table: Monitored Area Temperature Records

Field	**Data Type**	**Description**
id (PK)	INT	Unique identifier for the measurement
time	BIGINT	Device-local timestamp of the measurement
timestamp	TIMESTAMP	Computed global timestamp for synchronization
device	INT	Reference to the device ID
value	FLOAT	Recorded temperature value

4 Simulations

4.1 Smart Smoke Detector Simulator

For the first smart smoke sensor device, the Dongle's button is used to simulate adverse air quality and fire-related hazards. This is achieved by triggering two distinct but potentially overlapping simulation events: hazardous air quality and fire ignition. These events are initiated through short and long button presses, respectively, and can occur independently or in sequence, depending on the current state of the simulation. The general usage logic is as follows:

- **Short button press:** If no hazard condition is currently active, a hazard scenario is simulated. If a hazard is already active, the event is terminated.
- **Long button press (more than 2 seconds):** If no fire ignition is present, one is simulated. If a fire is already active, the event is terminated.

4.2 Smart Vent Simulator

For the smart vent device, the button serves a different purpose: it is used to start or stop the observation of data received from the smart smoke sensor device. This simulates the shutdown or reset of the ventilation system.

- **button press:** it toggles the observation state—either enabling the monitoring of incoming data or suspending it.

5 Artificial Intelligence Model

In this project, artificial intelligence is deployed on the Smart Detector node to accurately distinguish between smoke-related hazards and actual fire ignition events. Based on the classification, it can trigger the most appropriate ventilation response and activate alarm systems when necessary.

5.1 Data Collection

The dataset used to train the model is sourced from Kaggle and available at the following link[5]. It is closely aligned with the intended use case. It contains 62,630 measurements sampled at a rate of 1 Hz, collected using real sensor hardware in both normal and fire conditions, specifically within firefighter training facilities and under firefighter supervision.

All data was acquired using an Arduino Nicla Sense ME board, which leverages an integrated sensor fusion chip to coordinate the readings. The core measurements were performed with the Sensirion SPS30, a high-precision particulate matter sensor. Surrounding this, a range of additional sensors were used to collect environmental "meta" data—such as temperature, humidity, pressure, and gas concentrations—to aid in classifying the surrounding conditions. Redundancy was deliberately built into the sensor array to improve system reliability and enable the detection of faulty readings.

Onboard Sensors (Arduino Nicla Sense ME)

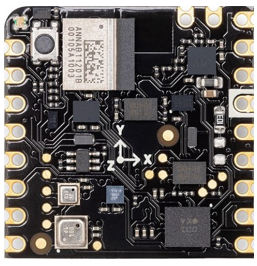
Bosch BHI260AP	6-axis IMU (3-axis Accelerometer + 3-axis Gyroscope) + MCU
Bosch BMP390	Pressure sensor
Bosch BMM150	Magnetometer
Bosch BME688	Humidity, Temperature, and Gas sensor (VOC)

External Sensors

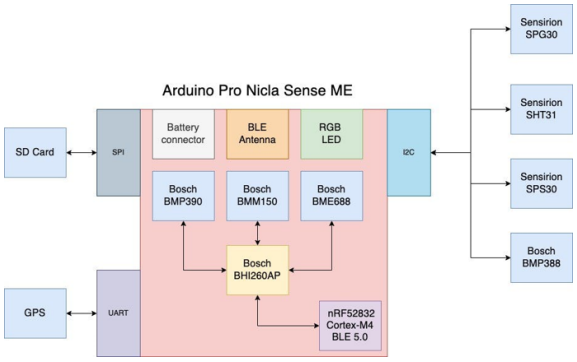
Bosch BMP388	Pressure sensor
Sensirion SPS30	Particulate Matter sensor (Smoke detector)
Sensirion SHT31	Humidity and Temperature sensor
Sensirion SPG30	Gas sensor (VOC)
GPS Module	Used for time synchronization of sensor readings

To build a robust training dataset, sensor measurements were collected across a variety of environments and fire sources. The following is a brief list of representative scenarios included in the data:

- Normal indoor
- Normal outdoor
- Indoor wood fire, firefighter training area
- Indoor gas fire, firefighter training area
- Outdoor wood, coal, and gas grill
- Outdoor high humidity



(a) Arduino Pro Nicla sense ME



(b) Sensors device architecture used to collect data

5.2 Preprocessing

The raw dataset, stored in CSV format, consists of 15 features along with the target class, as detailed below.

Dataset Features Description

Temperature [°C]	Ambient temperature
Humidity [%]	Relative humidity
TVOC [ppb]	Total Volatile Organic Compounds; measured in parts per billion
eCO_2 [ppm]	CO_2 equivalent concentration; estimated from TVOC and other factors
Raw H_2	Raw molecular hydrogen; not compensated (bias, temperature, etc.)
Raw Ethanol	Raw ethanol gas concentration
Pressure [hPa]	Air pressure
PM1.0	Particulate matter with diameter ¡ 1.0 µm
PM2.5	Particulate matter with diameter between 1.0 µm and 2.5 µm
Fire Alarm	Ground truth label: 1 indicates fire is present
CNT	Sample counter
UTC	Timestamp in UTC seconds
NC0.5	Number of particles ¡ 0.5 µm
NC1.0	Number of particles between 0.5 µm and 1.0 µm
NC2.5	Number of particles between 1.0 µm and 2.5 µm

5.2.1 Data Cleaning

The dataset was first inspected for missing values and duplicate entries. Since none were found, the metadata columns `Unnamed: 0`, `UTC`, and `CNT`—representing the sample index, timestamp, and counter, respectively—were safely removed, as they are not relevant for the subsequent analysis or model training.

Each feature was then validated against known physical limits to detect anomalies potentially caused by sensor errors or data corruption. As shown in the table below, all values fall within plausible ranges, confirming the dataset's overall integrity.

Feature	Min Feasible	Min	Max	Max Feasible
Temperature [°C]	-273	-22.01	59.93	–
Humidity [%]	0	10.74	75.20	100
TVOC [ppb]	0	0	60000	1,000,000,000
eCO2 [ppm]	0	400	60000	1,000,000
Raw H2	0	10668	13803	1,000,000
Raw Ethanol	0	15317	21410	1,000,000
Pressure [hPa]	876	930.85	939.86	1083.80
PM1.0	0	0	14333.69	–
PM2.5	0	0	45432.26	–
NC0.5	0	0	61482.03	–
NC1.0	0	0	51914.68	–
NC2.5	0	0	30026.44	–
Fire Alarm	0	0	1	1

Table 6: Feature Ranges: Minimum, Maximum, and Feasibility Limits

Subsequently, given their large dynamic range (0 to 60,000) and the distribution of values, all NC and PM measurements were rounded to the nearest integer. Decimal precision was deemed unnecessary for these parameters, and truncating the fractional part simplifies further processing without sacrificing information quality.

Finally, data balancing was considered for the target class, which distinguishes between the presence or absence of fire as the cause of smoke. As shown in the following graphs, the dataset is clearly imbalanced in favor of fire presence. However, no correction was applied. Indeed, this

imbalance—and the resulting opportunity for fine-tuning a deep learning model on this target—is desirable, since detecting fire must take higher priority over the absence class due to the greater severity of the threat.

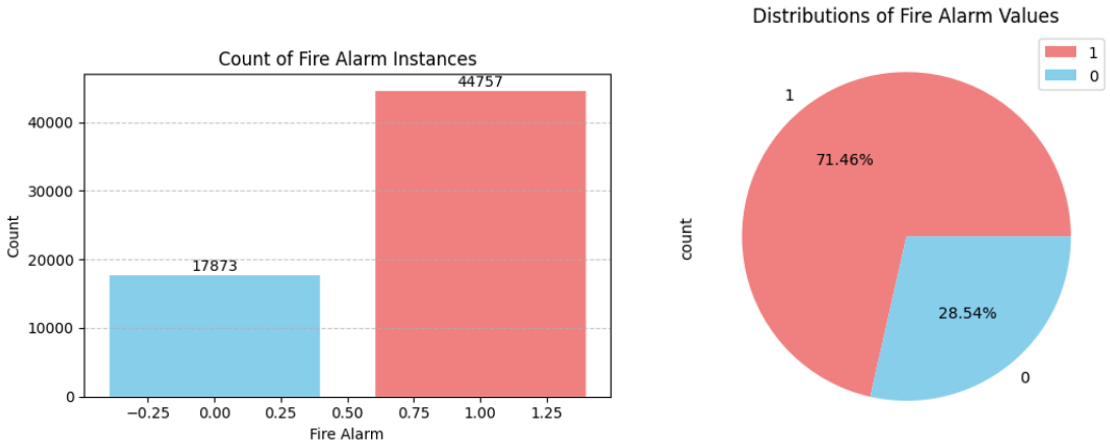


Figure 5: Distribution of the fire alarm target class represented as both a bar chart and a pie chart

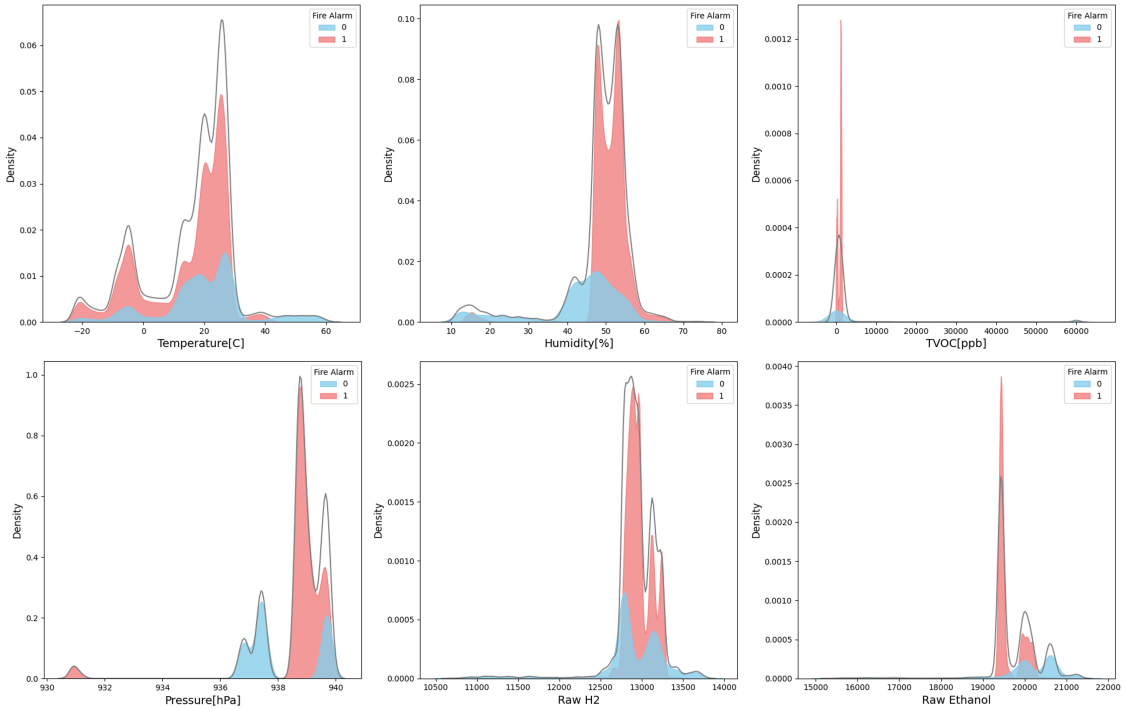
5.2.2 Exploratory Data Analysis

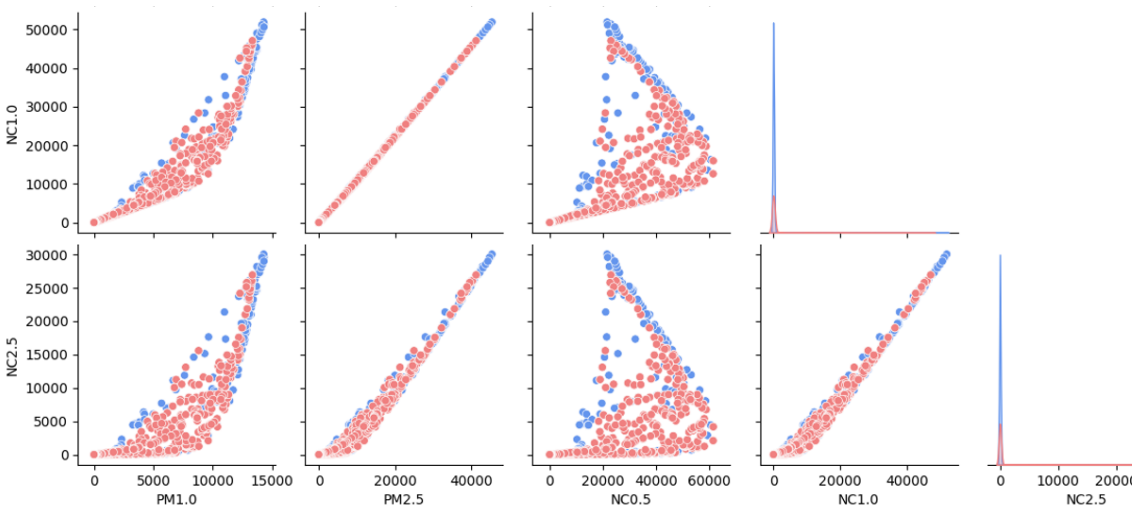
Since the primary objective of this artificial intelligence task is to classify the presence or absence of fire based on the available features, conducting a detailed correlation analysis is crucial.

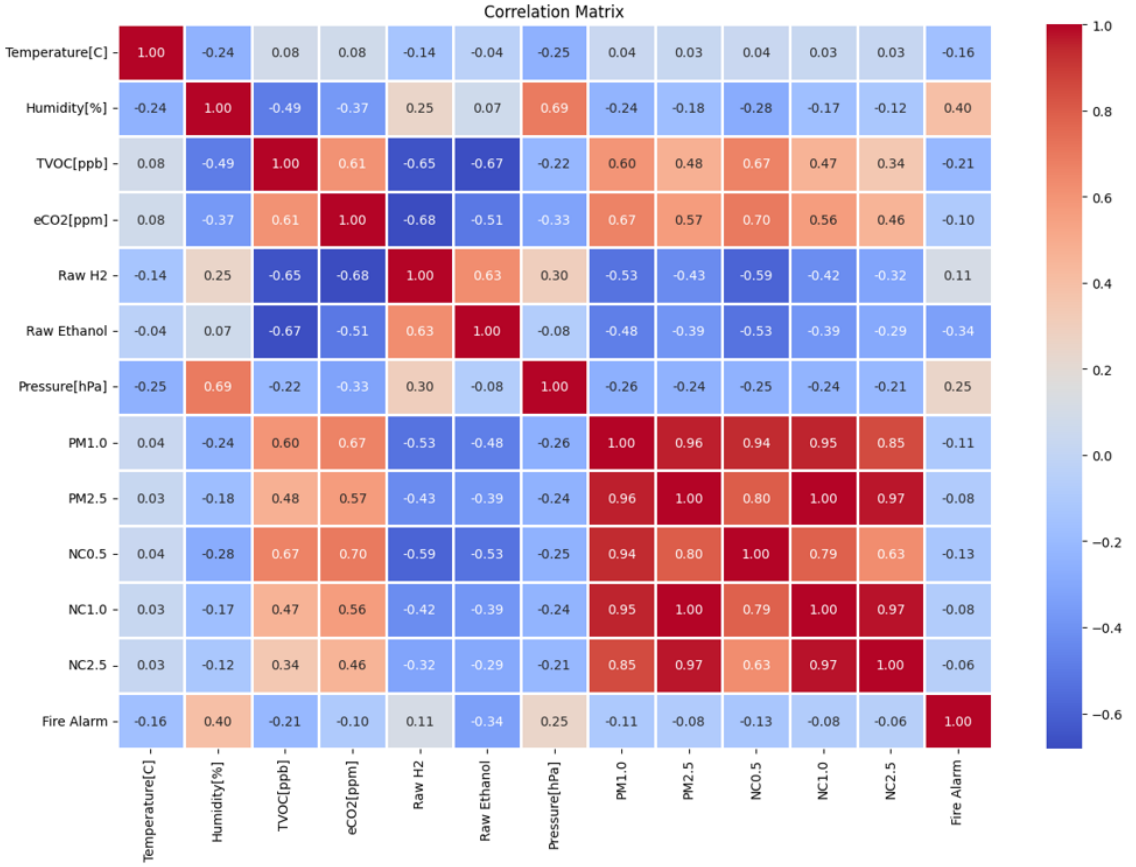
To evaluate the potential usefulness of each feature, individual feature distributions are plotted. These include kernel density estimation (KDE) plots for each target class, as well as an overall distribution that disregards class labels. This visualization helps reveal how each feature behaves relative to fire presence or absence. Additionally, box plots for each feature are analyzed to better highlight differences in feature behavior with respect to the target classes.

Thereafter, a pairwise correlation analysis is performed—both among the features themselves and between each feature and the target variable—to assess feature relevance and identify any redundancy. Finally a Correlation Matrix is computed and plotted.

In the following pictures a subsets of those plots are shown.







As clearly shown in the correlation matrix, all Particulate Matter features—PM1.0, PM2.5, NC0.5, NC1.0, and NC2.5—exhibit strong correlations according to Pearson's correlation coefficient. However, since Pearson's measure only assesses linear relationships, it does not fully capture more complex associations. Therefore, removing any of these features based solely on Pearson's correlation requires additional analysis to ensure no valuable information is lost.

Further inspection through pairplot analysis confirms that NC1.0 and NC2.5 exhibit a nearly perfect linear relationship with PM2.5. These strong linear correlations suggest that NC1.0 and NC2.5 do not provide additional independent information beyond what is already captured by the corresponding PM features. Similarly, eCO2 shows very low correlation with the Fire Alarm output while being highly correlated with TVOC and Raw H2, indicating redundancy and minimal contribution to fire prediction.

Among the various sensor features analyzed, humidity and atmospheric pressure emerged as the

most strongly correlated with the target class, providing valuable indicators for the early detection of fire events. Their distributions show distinct and physically interpretable shifts in the presence of fire, supporting their relevance in practical fire monitoring scenarios.

Starting with humidity, its correlation with the target class is clearly evident in the corresponding density plot. Under fire-present conditions, the humidity distribution shifts toward a tighter, higher-density cluster centered around 50–55%. While fire is often associated with a drying effect—due to heat increasing the air's capacity to hold moisture—this observed shift may also reflect the release of water vapor during the combustion of wood, which contains inherent moisture and chemically produces H_2O. It is reasonable to assume that firefighters primarily used wood as a fuel source in controlled fire tests, both for its accessibility and its ability to realistically simulate indoor fire conditions.

Similarly, atmospheric pressure demonstrates a strong correlation with the presence of fire, making it another key feature for detection. This relationship is physically plausible, as a drop in pressure can occur in confined spaces due to the heat-induced expansion of air and the resulting vertical displacement. Supporting this, fire-present samples show a reduction in both main bell shapes of the atmospheric pressure density plot. The more pronounced reduction likely corresponds to indoor environments, where hot air accumulates and causes greater pressure drops. In contrast, the smaller shift—showing minimal to no reduction—is likely associated with outdoor environments. This interpretation is consistent with the higher number of samples in the latter category, suggesting that firefighters preferred conducting controlled burns outdoors to ensure safety and better manageability. As a result, these tests involved smaller, safer flames that did not significantly impact ambient pressure. Nevertheless, this distinction aligns with the core objective of the application: to detect fires at their earliest stages—ideally when they are still minor ignitions—so that rapid and effective response measures can be implemented.

5.2.3 Data Reduction

Building on the insights from the previous section, the features `NC1.0`, `NC2.5`, and `eCO2` were removed to reduce redundancy and simplify the dataset without compromising predictive performance. This step enhances model efficiency, reduces potential multicollinearity, and discards features with limited relevance to the target variable.

5.2.4 Data Splitting and Normalization

As the final step in data preparation, the dataset was randomly partitioned into three subsets—a training set, a validation set, and a test set—while preserving the original class distribution through stratified sampling using sklearn's `train_test_split`. The data was divided according to the following proportions:

Dataset Split	Percentage	Samples
Training Set	72%	45,094
Validation Set	18%	11,273
Test Set	10%	6,263
Overall Dataset	100%	62,630

Table 7: Dataset partitioning percentages

Additionally all input features were normalized to have zero mean and unit standard deviation. Importantly, the normalization parameters (mean and standard deviation) were computed using only the training set, to avoid data leakage and ensure proper generalization.

5.3 Deep Learning Model: FeedForward Neural Network

As presented in the use case scenario, a deep feedforward neural network model is deployed on the NRF52840 Dongle edge device to enhance fire detection capabilities. This setup necessitates not only optimizing for standard performance metrics but also accounting for the device's constrained computational and memory resources.

To identify the most accurate and efficient architecture, multiple feedforward neural networks were explored. A total of six models were evaluated, varying in both depth and width, enabling a comparative assessment of architectural complexity.

All models used the ReLU activation function in each hidden layer. The output layer consisted of a single neuron with a sigmoid activation function, producing a binary classification output. Training was conducted using the Adam optimizer with a learning rate of 0.0001, over 100 epochs, using mini-batches of 32 samples. Model performance was evaluated with respect to validation accuracy. Additionally, an early stopping criterion based on validation loss was employed to prevent overfitting, and checkpoint callbacks were used to save the best-performing model during training.

Layer (activation function)	**Neurons per Layer**					
	NN1	**NN2**	**NN3**	**NN4**	**NN5**	**NN6**
Dense1(relu)	16	32	64	16	32	32
Dense2(relu)	64	128	256	64	128	64
Dense3(relu)				128	256	128
Dense4(relu)						256
DenseLast(sigmoid)	1	1	1	1	1	1

Table 8: Neuron configuration for each network architecture.

The results after training are the following:

Metric	**NN1**	**NN2**	**NN3**	**NN4**	**NN5**	**NN6**
Accuracy	0.9936	0.9965	0.9989	0.9990	0.9992	0.9994
Loss	0.0191	0.0103	0.0069	0.0053	0.0042	0.0056
Recall	0.9975	0.9989	0.9996	0.9996	0.9996	0.9996
Precision	0.9935	0.9962	0.9989	0.9991	0.9993	0.9996
F1 Score	0.9955	0.9975	0.9992	0.9993	0.9994	0.9996
Confusion Matrix						
TP	4465	4471	4474	4474	4474	4474
FN	11	5	2	2	2	2
FP	29	17	5	4	3	2
TN	1758	1770	1782	1783	1784	1785
Model Size (KB)	36 KB	75 KB	226 KB	139 KB	468 KB	546 KB

Table 9: Performance metrics for each neural network architecture.

The highest accuracy is achieved by the final and most complex architecture. However, considering the critical constraints on memory and computational resources, the penultimate model is preferred. It offers the best recall—an essential metric in this application—while maintaining a relatively compact size.

5.4 Implementation on IoT Device

In alignment with the objectives of the Smart Smoke Detector application, the selected model for on-device deployment is the NN4 neural network. To enable integration with the IoT device's firmware, the model is converted from Python to efficient C code using the `emlearn` library. This converted model is then embedded within the device's core logic, where it is used to infer whether detected smoke is likely caused by an actual fire event. It is important to note that all input features on the device are properly normalized before being fed into the model, ensuring consistent and reliable inference performance.

6 Sensor Data Representation: SenML

This project adopts an efficient and flexible data format to simplify integration with external applications. Sensor data is represented using SenML (Sensor Markup Language), a standardized format designed for constrained devices, offering simplicity, interoperability, and a semantic foundation for consistent data reporting. Combined with JSON encoding, this approach ensures compatibility across diverse platforms and technologies, enabling scalable, efficient, and interoperable IoT solutions.

According to the SenML specification, a SenML object represents a collection of measurements organized in a simple hierarchical structure. It comprises a set of optional base attributes along with an array of entries—each entry corresponding to a measurement that can include its own specific attributes, as detailed below (including their JSON representation).

SenML Base Object Field	JSON	Description
Base Name (optional)	bn	Acts as a prefix for the names in the measurement entries.
Base Time (optional)	bt	A reference timestamp added to each entry's relative time value.
Base Units (optional)	bu	Specifies a default unit of measurement that applies to all entries unless locally overridden.
Version (default = 1)	ver	Indicates the version of the SenML specification used.
Measurement Entries	e	A list of measurement records, each describing a specific observation.

Table 10: Description of SenML Base Fields

SenML Array Entry Field	JSON	Description
Name	n	Identifies the specific sensor or parameter.
Unit	u	Defines the unit of the recorded value.
Value	v/bv/sv	Measurement value (floating point, boolean or string support-only)
Time	t	Relative timestamp for the measurement.

Table 11: Description of SenML Measurement Entry Fields

6.1 Smart Smoke Detector Measurement

This section presents an example of measurement data collected from a smart smoke detector node, focusing on temperature readings captured over time. The example demonstrates how standard environmental parameters are represented using the JSON-encoded SenML format, ensuring that data remains interoperable and understandable across heterogeneous systems.

```
{
 "bn": "coap://[fd00::202:2:2:2]/temp",
 "bu": "C",
 "ver": 1,
 "bt": 88,
 "e": [
  { "v": 14.366, "t": 0 },
  { "v": 14.555, "t": 2 },
  { "v": 14.777, "t": 4 },
  { "v": 14.763, "t": 6 },
  { "v": 14.598, "t": 8 },
  { "v": 14.509, "t": 10 },
  { "v": 14.497, "t": 12 }
 ]
}
```

The `bn` (Base Name) field serves as a globally unique identifier for the data source. It typically combines the device's network address with the resource name (e.g., temperature), using a CoAP URI to ensure consistency and uniqueness across distributed IoT deployments. The `bu` (Base Unit) specifies the default measurement unit (in this case, degrees Celsius) that applies to all entries.

Each measurement entry in the `e` array includes a relative timestamp `t`, which—when added to the `bt` (Base Time)—yields the absolute time of observation. This representation reduces redundancy while enabling precise temporal alignment. Moreover, it facilitates detection of duplicate or repeated measurements by providing a deterministic structure for time-based comparisons.

6.2 Smart Smoke Detector Environment Status

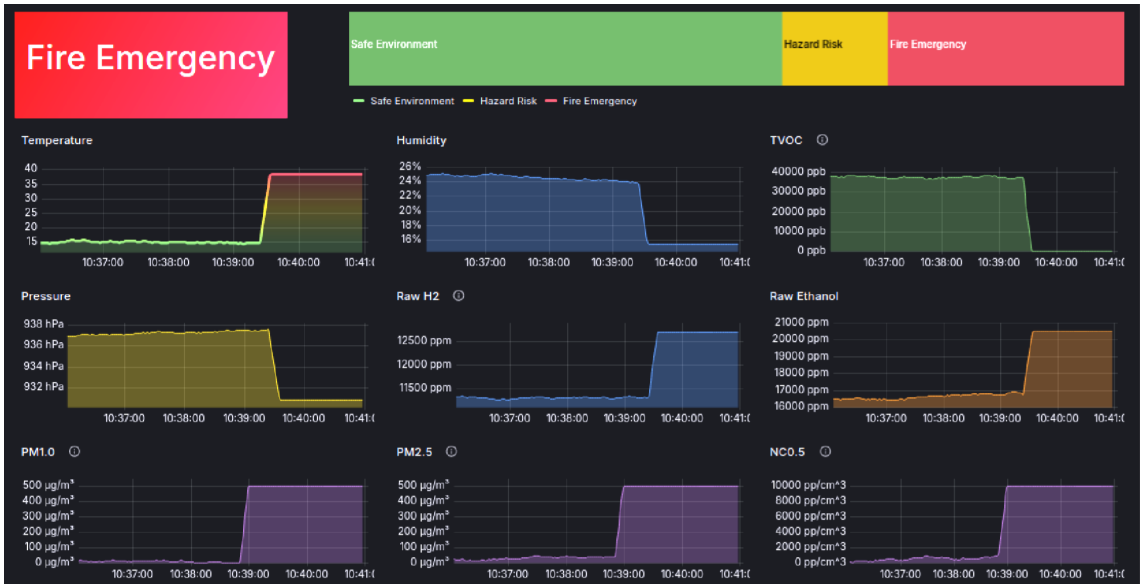
Non-sensor data, such as environment state or status, can also be expressed using a simplified variant of the SenML structure:

```
{
 "bn": "coap://[fd00::202:2:2:2]/status",
 "status": 0,
 "bt": 112
}
```

In this case, the payload does not contain a physical measurement but rather application-specific control information (e.g., an operational state). Because such data is not semantically self-describing, consumers of this information must possess minimal a priori knowledge to interpret the meaning of the `status` field correctly. As a result, these messages require lightweight contextual awareness to ensure correct handling.

7 Grafana Web Application

This section presents the dashboard developed using Grafana, tailored to support the proposed use case scenario. The visualizations provided enable users to intuitively assess both the current status and historical trends of the monitored environment. Through states charts and time series plots, users can easily evaluate system behavior, choose appropriate safety thresholds, and point out preferred limits for hazard detection or automatic ventilation activation based on their specific needs and risk tolerance.



References

[1] Swedish Civil Contingencies Agency (MSB), *"Fire Safety Statistics"*, https://www.modernbuildingalliance.eu/fire-safety-statistics/, Accessed: 2025-05-05.

[2] Cleary T.G., and Chernovsky, A., *"Smoke Alarm Performance in Kitchen Fires and Nuisance Alarm Scenarios,"*, NIST Technical Note 1784 (January, 2013), https://www.nist.gov/el/smoke-alarm-research-0, Accessed: 2025-05-06.

[3] BAFE Fire Safety Register - British Approvals for Fire Excellence, *"Home Office statistics highlight fire false alarms remain an issue"*, https://www.bafe.org.uk/bafe-news/home-office-statistics-highlight-fire-false-alarms-remain-an-issue, Accessed: 2025-05-06.

[4] Brian O'Connor, National Fire Protection Association (NFPA), *"Smoke Control Systems"*, https://www.nfpa.org/news-blogs-and-articles/blogs/2021/02/05/smoke-control-systems, Accessed: 2025-05-08.

[5] Stefan Blattmann, *Smoke Detection Dataset*, https://www.kaggle.com/datasets/deepcontractor/smoke-detection-dataset/data, Accessed: 2025-05-13.

[6] Fabio Malloggi, *fireGUARD IoT Project Github Repository*, https://github.com/FabioMalloggi/fireGUARD_IoT_Project