# IoT-Based Indoor Air Quality Monitoring System

Presentation of Internet of Things Project
A.A. 2021/22
Andrea Gurioli and Mario Sessa

Introduction

*Overview of the Project*

Architecture

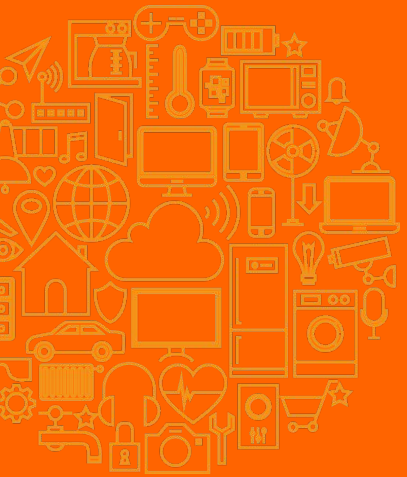
*Components, intercommunications, services and features*

Implementation

*Libraries usage and implementation of project functionalities*

Results

*Metrics, Evaluations and final observations*



# Introduction

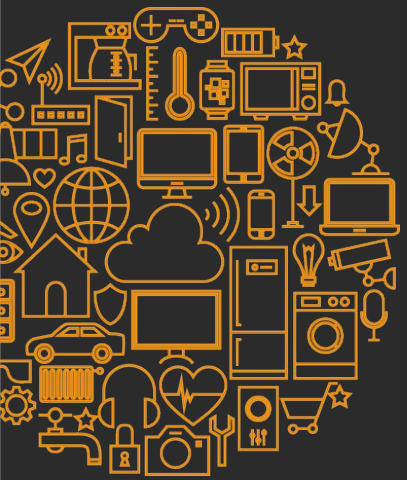
*Overview of the Project*

## Overview of the Project

- Internet of Things has a significant importance in Indoor and Outdoor environments: Agriculture, Urban System and Smart Home.

- **Goal:** Implementation of a Air Quality monitoring system based on IoT environment using multiple ESP-32 micro-controllers  sensing with DHT-22 and MQ-2.

- More in detail, our aim is to implement a system to retrieve sensors measurement, process them in a proxy server, display on a monitoring dashboard and store results on a storage layer. Furthermore, we add some features related to the runtime configurations, devices registration and real time forecasting system.



# Architecture

*Components, intercommunications, services and features*

## Micro-Controllers and Sensors Management

- We used ESP-32 micro-controllers with DHT-22 sensor for temperature and humidity measurements and MQ-2 sensor for gas concentration index.

- In addition, ESP-32 measures RSS of the Wi-Fi signal and AQI on gas concentration related to the following formula:

$$AQI = \begin{cases} 2 & \text{if avg} \leq MAX\_GAS\_VALUE \\ 1 & \text{if MIN\_GAS\_VALUE} \leq avg < MAX\_GAS\_VALUE \\ 0 & \text{otherwise} \end{cases}$$

- ESP-32 communicates data to the Proxy Server via MQTT or CoAP protocol chosen runtime. Remote commands to set configuration parameters and behaviors are communicated by the Proxy Server using MQTT embedded topics.

## Protocol Management

### MQTT Usage

- MQTT protocol usage is related to the configuration parameters with Quality of Service equals to 1 due to the absence of side-effects in the retransmissions cases, 2 for testing procedure.
- It is also used to send data from ESP-32 to the Proxy Servers.
- Data transmission is related to a single topic with QoS to 0, meanwhile setting operations have multiple related topics with different setups.

### CoAP Usage

- Exploited as a second data transmission protocol.
- In this case, the server become the board and the client the proxy.
- As we need a URI identifier, we exploit the MQTT protocol as a discovery mechanism for CoAP usage
- GET  requests drive our data retrival

## Proxy Server

- Proxy Server has the gateway role in the IoT proposed network. It has many functions:
    - Sensors Registration according to Monitoring dashboard requests
    - Hyperparameters Setup to change sample frequency and bound of gas concentration for AQI estimation.
    - Alive Service to monitor online microcontrollers status
    - Switching Service for runtime protocol changing over micro-controllers connections and communications mechanisms
    - Testing Services:
        - Manage internally CoAP Testing during intercommunication.
        - Sending a MQTT testing request to micro-controllers and receive the metrics in response via multiple topics communication.
    - Data Storing for store sensors measurements in an InfluxDB
    - Forecasting Requests to retrieves predictions on future data behaviors
    - Outdoor Monitoring System related to external APIs for outdoor temperature.

$$P_{loss} = 100 - (\frac{5}{N_{req}}) \times 100$$

$$L_d = \frac{\sum_{i=0}^{5}(T_{stop_i} - T_{start_i})}{5}$$

## Forecasting Server

- It provides prediction of future values for one of the following attributes:
  - Indoor Temperature
  - Humidity
  - Gas Concentration
- During the prediction, server checks a dump model in the file system:
  - If the model is not present, use a base model for the next prediction and train a new one.
  - Otherwise, it loads the model from the file system and update it with the last real measurements stored in the Influx Database.
- Forecasting Server retrieves training data using Flux queries.
- Every type of attribute has an embedded ARIMA model related to each micro-controller.
- Training and Forecasting procedures are parallelized using **Repeated Time Threads.**
- Threads usage increment runtime performances due to faster response processing.

## Influx Database and Grafana Dashboard

- Proxy, Forecasting Server and even Telegram bot communicate with **Influx Database** using Flux queries in an instance of InfluxManager class which handles intercommunication with the storing layer.

- Influx Manager can:
  - Write new points related to new sensors measurements
  - Write Points with embedded timestamp for prediction value
  - Submit general queries
  - Manage mean and values distribution for Telegram Bot functions

- **Grafana** visualizes stored data and manage an alert system connected to Telegram Bot in case of AQI values above to  one.

- It is also accessible via Monitoring Dashboard using a simple link system.

## Monitoring Dashboard

- It is a front-end component with 4 different functions:

1. Provides a Registration System for authentication management related to micro-controllers connected to the monitoring network

2. It has a setting management to change micro-controllers settings for:
   a. Gas bounds for AQI calculus
   b. Sample Frequency for reading sensors data

3. Dashboard has also a state management system related to active sensors connected to the system. It carries out different procedures:
   a. Protocol Switching
   b. Testing over the chosen protocol
   c. Visualize GPS location related to the latitude and longitude hard-coded in the micro-controller firmware

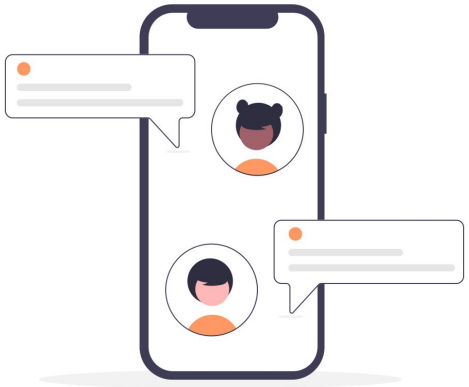
4. Finally, it has also an anchor for Grafana Dashboard

## Telegram Bot

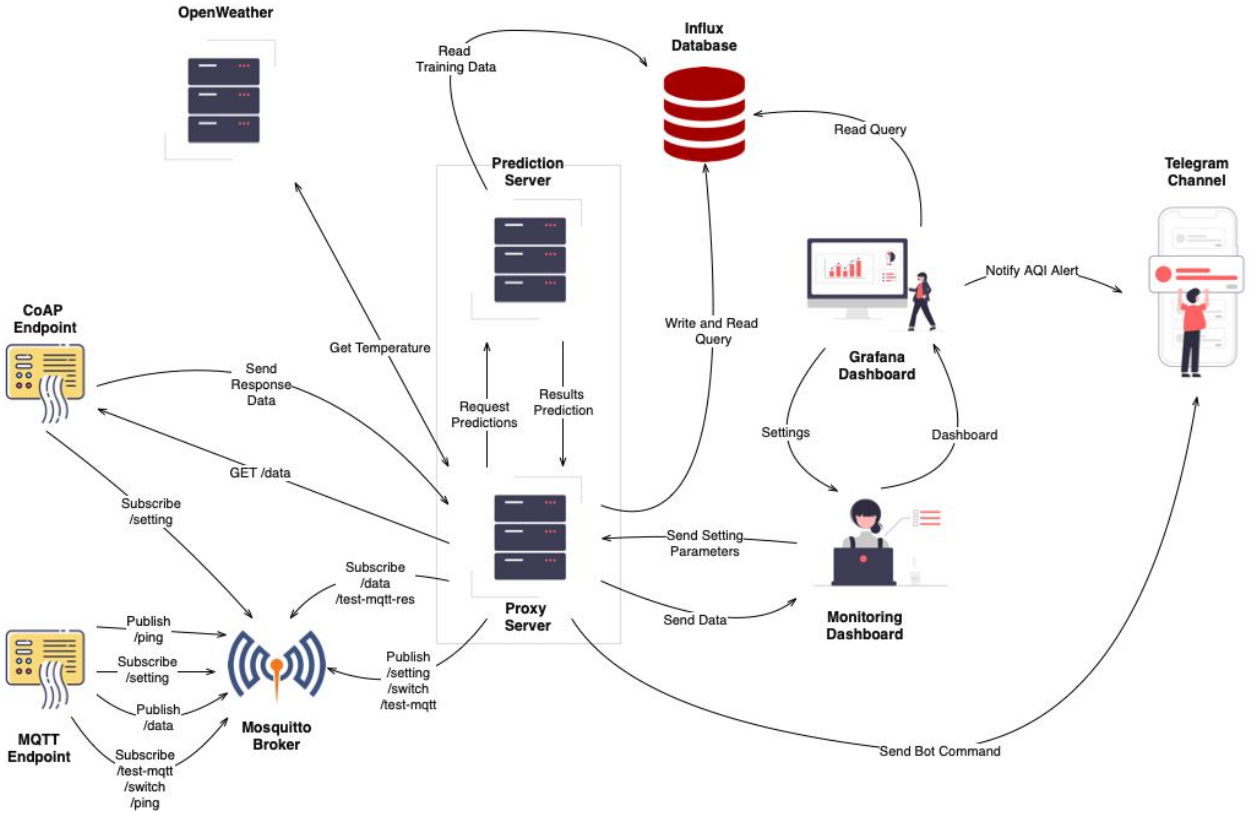
- It is accessible via Telegram Application for some functions:

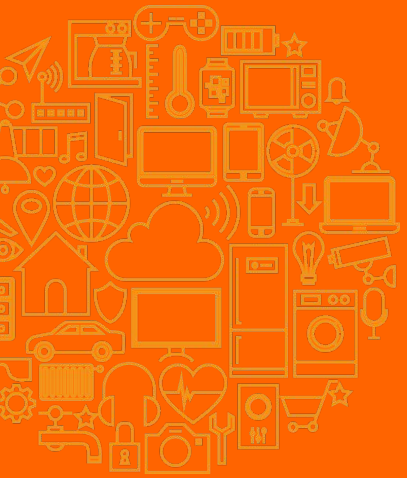
1. Provides mean values related to one or more Influx Database buckets

2. It sent a notification in case of alert trigger on Grafana Dashboard related to AQI values greater than 1.



# Architecture





# Implementation

*Libraries usage and implementation of project functionalities*

III

## ESP-32, DHT-22 and MQ-2 Management Implementation

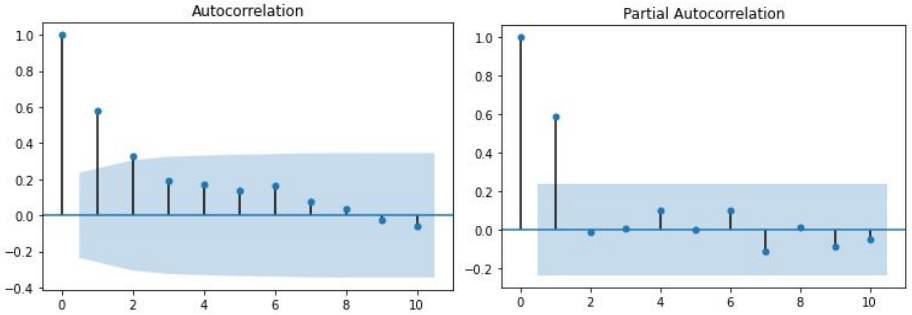
- We used *PubSubClient.h* and *Thing.Coap.h* libraries for protocols managements.
- *ArduinoJson.h* for data packaging during transmission
- *DHT.h* for DHT-22 sensor reading. MQ-2 data retrieval is done using analogic reading directly on the microcontroller pins.

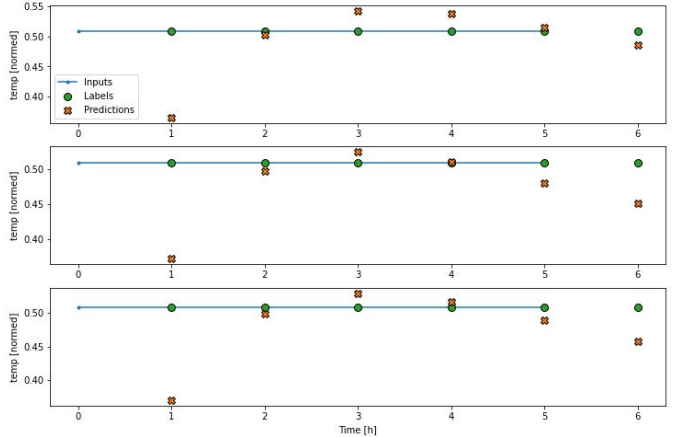
```
AQI:1
Gas sensor: 4095
Temperature in Celsius: 29.20
Humidity value: 39.70
Protocol: MQTT
-------------------------
--------- Data -----------
WiFi RSS Strength: -80.00
ID: 409151bfa0cc
AQI:1
Gas sensor: 4095
Temperature in Celsius: 29.20
Humidity value: 39.70
Protocol: MQTT
-------------------------
--------- Data -----------
WiFi RSS Strength: -82.00
ID: 409151bfa0cc
AQI:1
Gas sensor: 4095
Temperature in Celsius: 29.20
Humidity value: 39.70
Protocol: MQTT
-------------------------
```

# Tested Forecasting Models

- ARIMA Models:
  - Controls of Not-stationary using Dickey-Fuller Test.
  - Autocorrelation and Partial autocorrelation tests
  - Confidence Intervals and RMSE evaluation.

- LSTM
  - Window-based models
  - Seasonality handled by sine and cosine projections
  - Training on normalized dataset

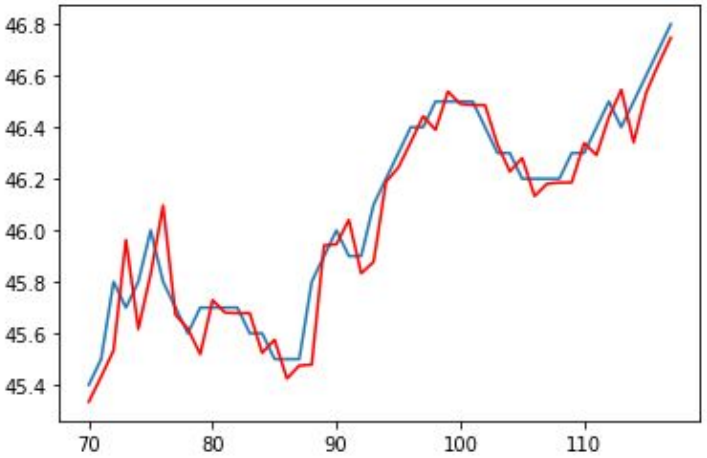


## Servers

- Proxy Server is implemented in *app.js and protocol.js* scripts using:
  - *Express.js and http* libraries for HTTP intercommunication
  - *mqtt* library for MQTT management
  - *coap* library for CoAP requests
  - *request* library for external API invocation
  - *influx-db-client* library for *InfluxManager.js* management

- Forecasting Server is implemented in *app.py* using *flask* package. In addition:
  - We used *InfluxDBClient* for training-data retrieval
  - *pickle* for models dumps and loading
  - *pandas* for data management
  - *statsmodels* for ARIMA building.

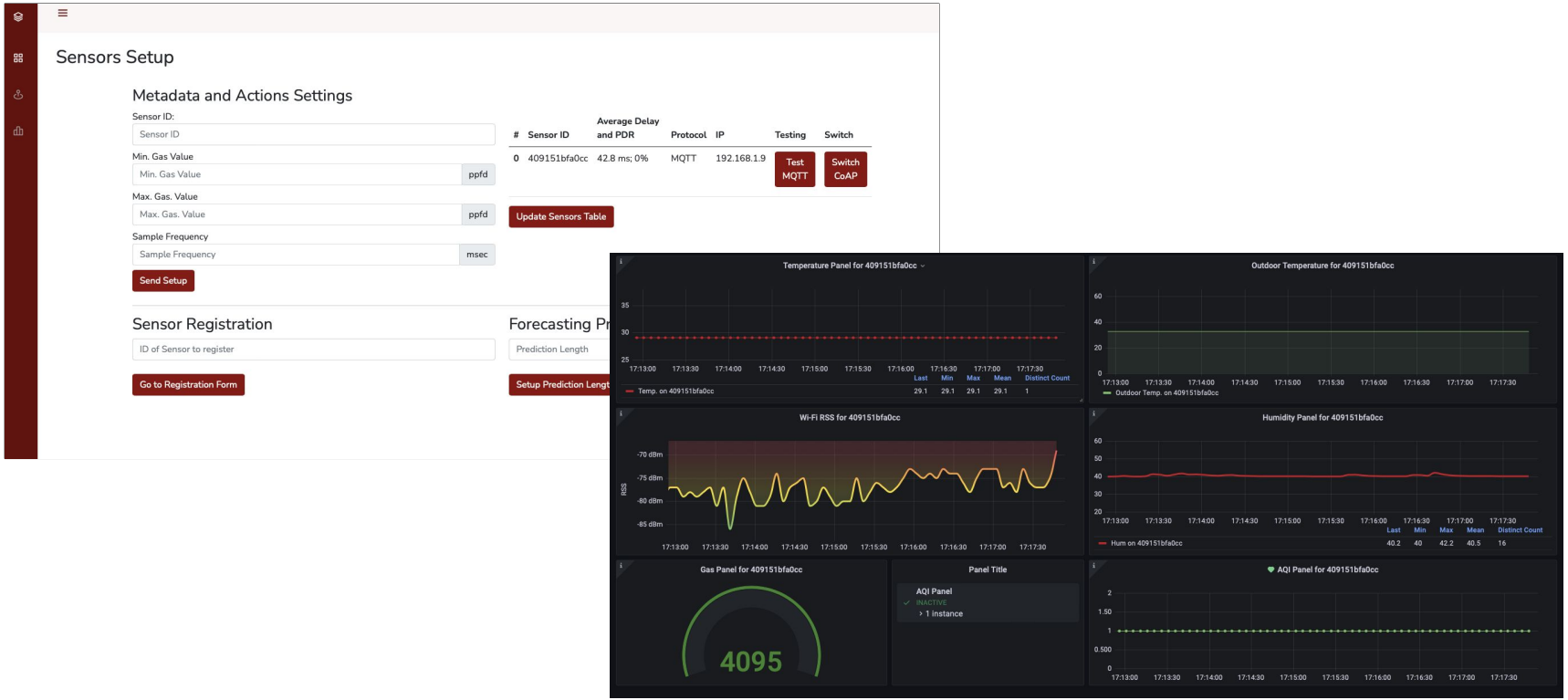



*Arima humidity model training*

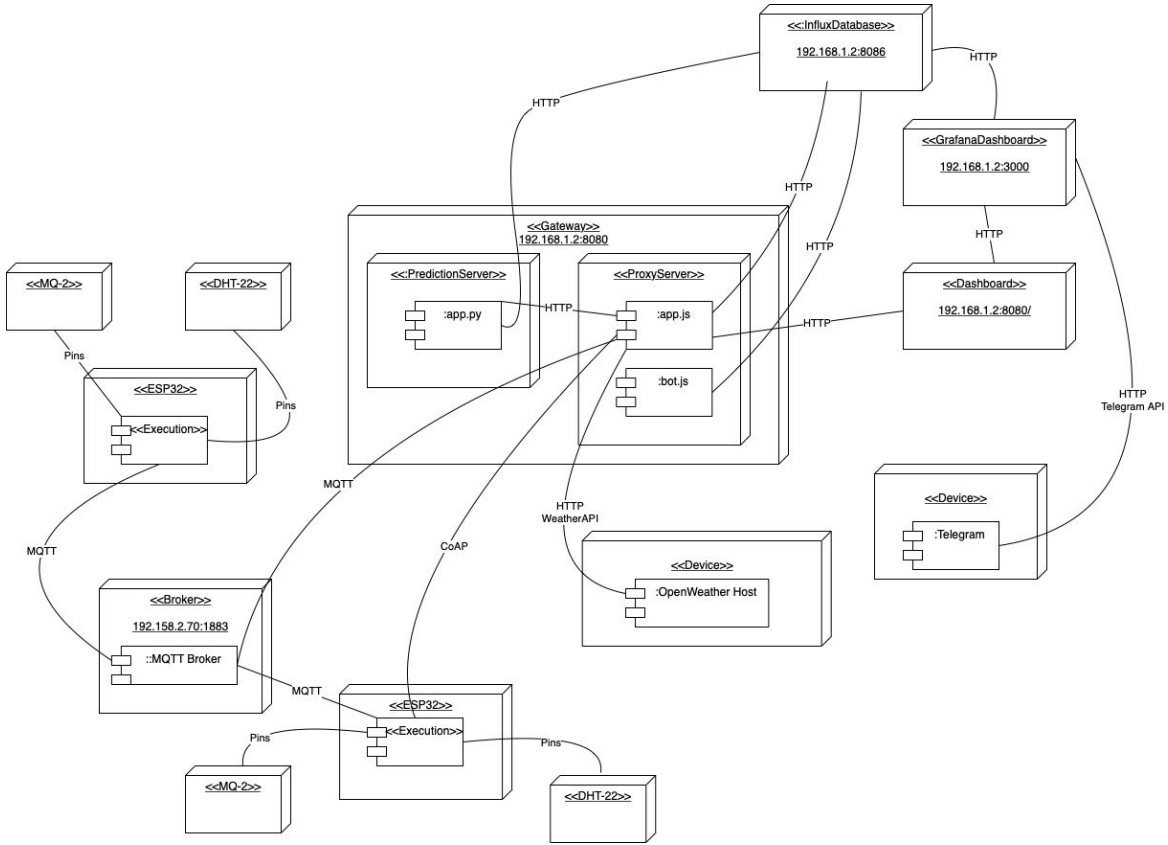
## Influx Database

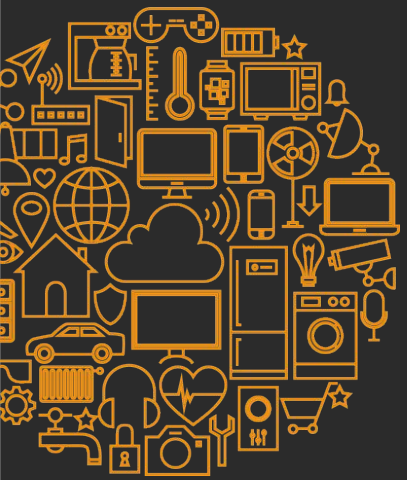
- We used *Flux* language to make queries on 6 different buckets:
  - Indoor Temperature
  - Outdoor Temperature
  - Humidity
  - Gas Concentration
  - RSS
  - AQI

# Monitoring Dashboard and Grafana Integration



# Deployment Diagram





# Results

*Metrics and Evaluations and final observations*

## Metrics and Evaluation

- Protocol Metrics:
  - Latency
  - Package Loss

- Forecasting Metrics:
  - RMSE
  - Prediction Mean
  - Interval of Confidence

| Protocol | Latency | Package Loss |
|---|---|---|
| **MQTT** | 41.30 ms (± 1.63) | 0% |
| **CoAP** | 146.08 ms (± 31.44) | 17% |

| Model Type | RMSE | Prediction Mean | Interval of Confidence |
|---|---|---|---|
| **Temperature** | 0.019 | 29.91 | [29.83, 29.99] |
| **Humidity** | 0.118 | 40.52 | [36.01, 45.03] |
| **Gas** | 180.939 | 4095 | [3567.68, 4622.31] |

# DEMO!