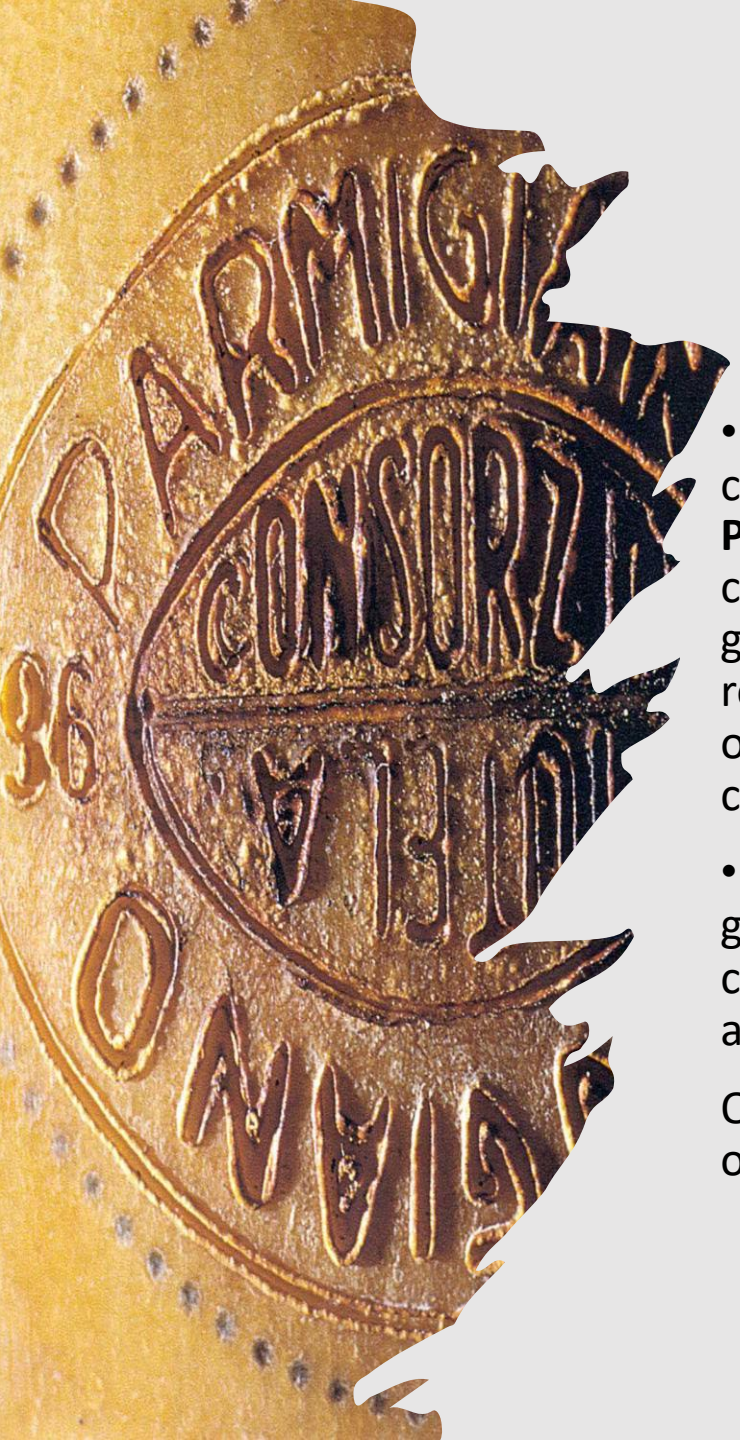
CheeseGuardian

A food managing system by Davide Mecugni



Business Case – Food Regulation

- Parmigiano Reggiano is a heavily regulated cheese product that **must follow the Production Regulation** made by the consortium in accordance with Italian government. There are several indications regarding the temperature, humidity and overall productions steps for the wheels of cheese.
- If any piece of cheese falls below the guidelines, **it loses the legal status** given by the consortium and must be sold under an anonymous brand, causing loss of revenue.

On the left is the stamp that the wheel receives once it is certified as following the guidelines.

Some Regulations

- The milk may be chilled immediately after milking and kept at a temperature not lower than 18ºC.
- ... In this case, the milk shall be kept at the dairy in special stainless steel containers at temperatures that are not below 10°C.
- In summer the temperature in the maturation room may not be lower than 16°C

Business Case – Natural disaster management

Emilia Romagna is the region where the cheese is produced. The area is considered seismic and subject to many floods, given that more than 30 rivers are present.

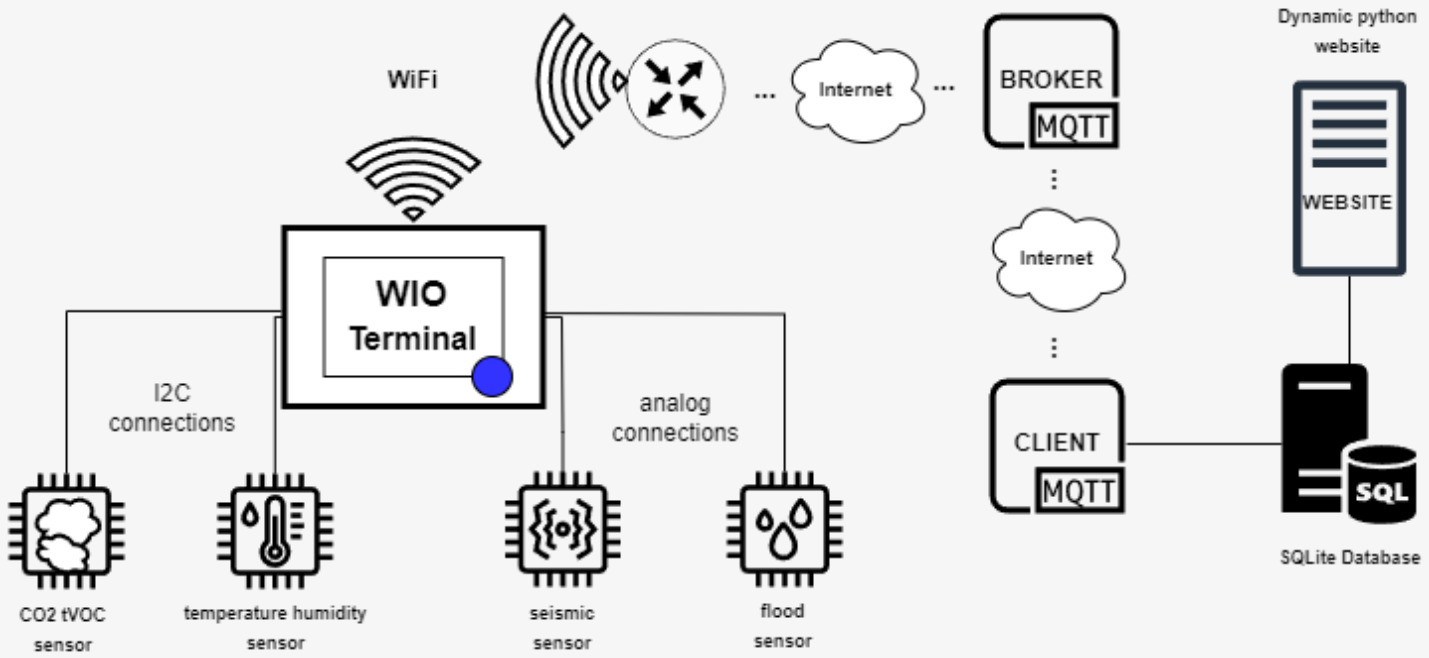
During the 2012 earthquake, more than 150M €(1.2B ¥) of damages[1] has been recorded by the consortium.

One of the main loss of revenue is the direct damage the wheels receive from falling from the shelves, or the shelves bending under the stress.

The shelves used for storing the cheese are also made of wood, making the storing area a fire hazard.

An alert system for earthquakes, floods and fires would be a cost-effective way of preventing damages. It can also ease insurance claims after the incident.

[1]University of Parma at https://agriregionieuropa.univpm.it/it/content/article/31/47/parmigiano-reggiano-terremotato-possono-fiducia-e-solidarieta-sconfiggere-le



Design

Sensors:

- LIS3DHTR Wio Terminal accelerometer
- SGP30 CO2 and tVOC air sensor
- SHT40 Temperature and humidity sensor
- Moisture Sensor v1.1

Servers:

- MQTT subscriber
- SQLite Database
- Dynamic python web server based on Streamlit

5-Way Switch on the Wio Terminal used to modify refresh rate and acoustic alarm sound

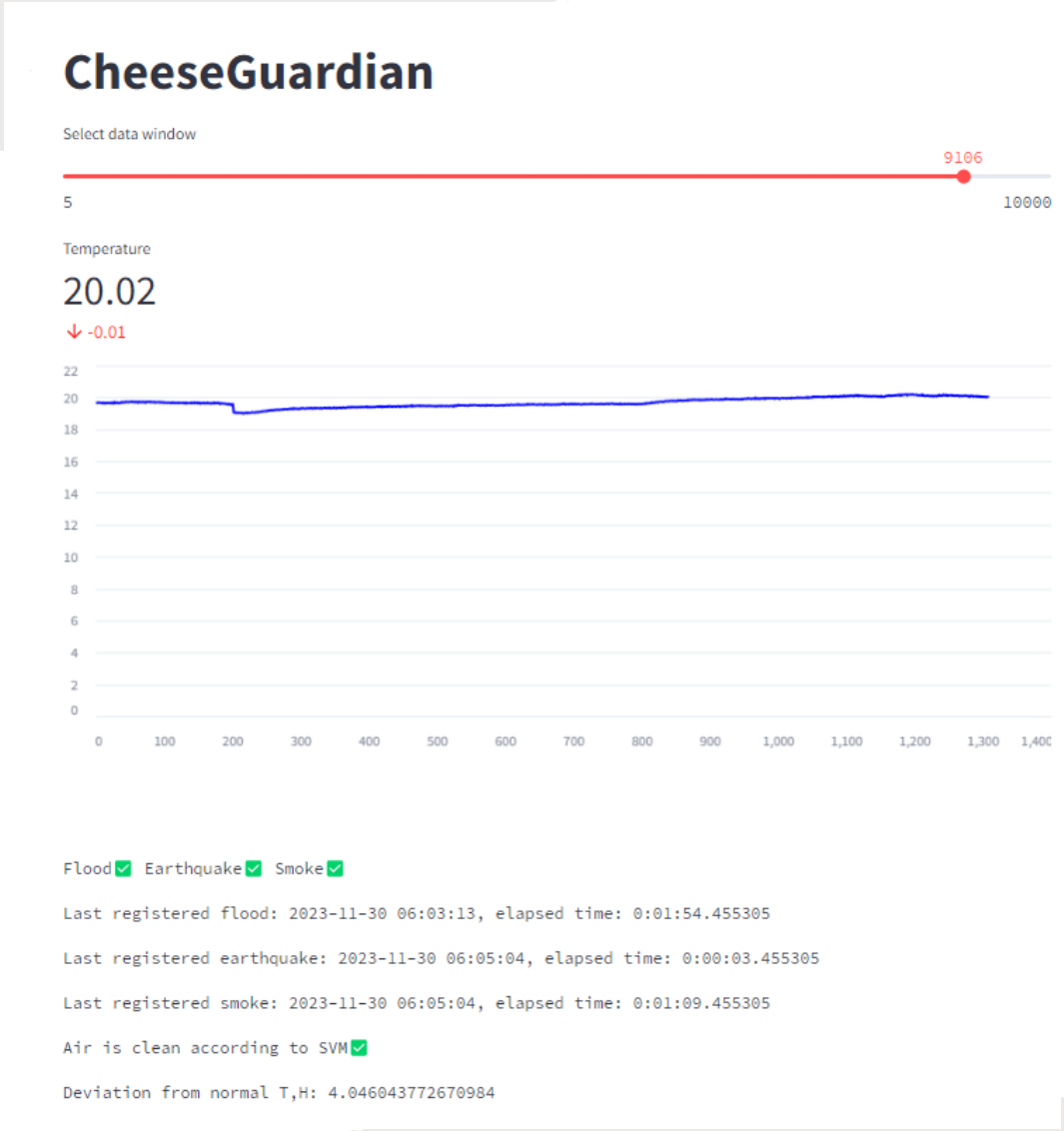
Design

This scalable infrastructure enables the installation of many devices on one or multiple sites. The MQTT subscriber receives the messages published by the CheeseGuardians through the hiveMQ broker. The message is a string containing all the values recorded. Different topics can be used for more rooms and cheese factories.

The content is split and saved on the SQLite Server(note that the server could be on the same machine or remote enabling maximum flexibility).

Each CheeseGuardian has its own table(ex. sensor_data_0001).

The database data is then displayed on a website by the final web server using batch processing, this server could also be remote. The server aggregates and graphs the data.



Development

The first task was collecting the data from the sensors. Given the imprecise air quality values, I calibrated the air sensor using the absolute humidity derived from the relative humidity reading(sensor fusion).

Given that a seismic sensor wasn't available, I used the accelerometer to calculate seismic activity. The value sensed in one reading is compared to the previous 2 measurements using Mean Square Error.

The fire detection part was more challenging. Given that a single sensor or measurement for fire is not available, I had to detect potential fires by the air quality sensor. I generated a dataset of clean and smokey air by exposing the sensor to different environments. I then labeled the data and used K-means for identifying fires. Once the centroids where found I implemented the algorithm in C on the device.

All the disaster analysis is done on the device practicing edge computing.

The following formula was fitted to Wexler's results, extrapolated for $T < 0°C$, to an accuracy of 0.1% for $-30°C \leqslant T \leqslant 35°C$:

$$\dot{e}_s(T) = 6.112 \exp\left(\frac{17.67\,T}{T + 243.5}\right). \qquad (10)$$

The Computation of Equivalent Potential Temperature
David Bolton
01 Jul 1980

	CO2	tVOC
0	459	2
1	460	0
2	458	0
3	449	0
4	444	3
..	...	...
109	675	203
110	15998	60000
111	1777	15614
112	1304	4763
113	643	1049

[114 rows x 2 columns]

Dataset used for smoke detection

Development

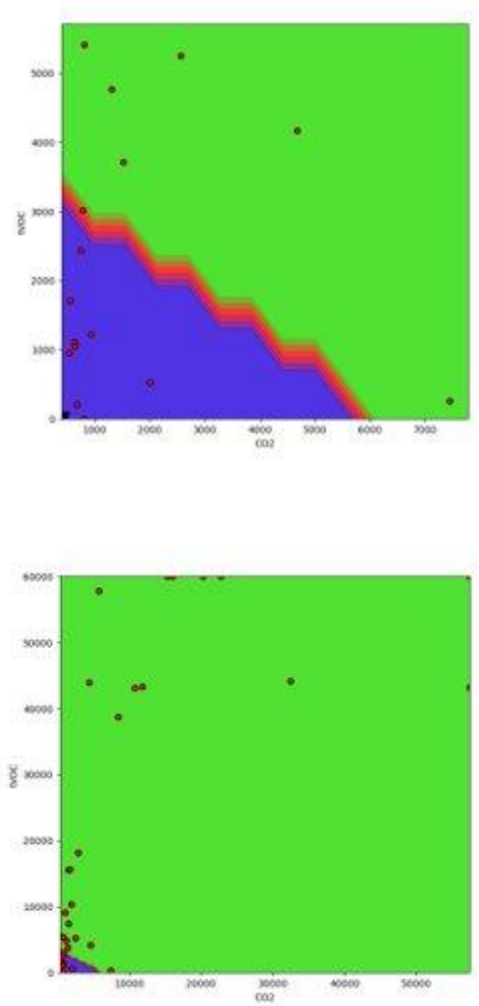
Once the data was organized and classified , the next task was displaying the data.

I chose to use the Streamlit Python framework. The final website retrieves the data from the SQLite server and displays it. It also calculates the time difference between the last natural disaster registered, useful for precise evaluation of when the incident took place.

The website graphs data recorded in a modifiable time window. It can also modify the refresh rate of the site.

Given that the K-means proved to be worst compared to other algorithms, I tried other solutions. The best algorithm found was Support Vector Machine(SVM) using Radial basis function kernel and a scalable gamma. The smoking detection using SVM is done on the final server.

On the side the SVM decision boundary is showed.

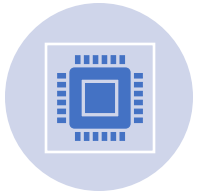


SVM classification

Deployment

Each CheeseGuardian in installed on a shelf where the cheese is stored. The flood sensor is placed close to the ground.

A single or multiple WiFi access points connect the devices. In case the cheese factory is very remote, and the internet connection is slow or unstable, a local DB server could be installed. The data is then shared with the central server once the connection is more stable or could be aggregated locally and then shared(fog computing).



Each factory could choose a more limited amount for sensors(ex. A single sensor controls the floodings, while multiple control the air quality). The data storing can then be modified on the SQLite server changing the CheeseGuardian's table.

The thresholds set for the natural disaster values could be modified once data is obtained after installation. The ML algorithms could then work on a real-world dataset, scoring better accuracy.

The fire detection system could be connected to the fire sprinkler system for automatic fire management.

An offsite team could be automatically alerted by the system when a natural disaster is detected by the sensors.

Sustainability

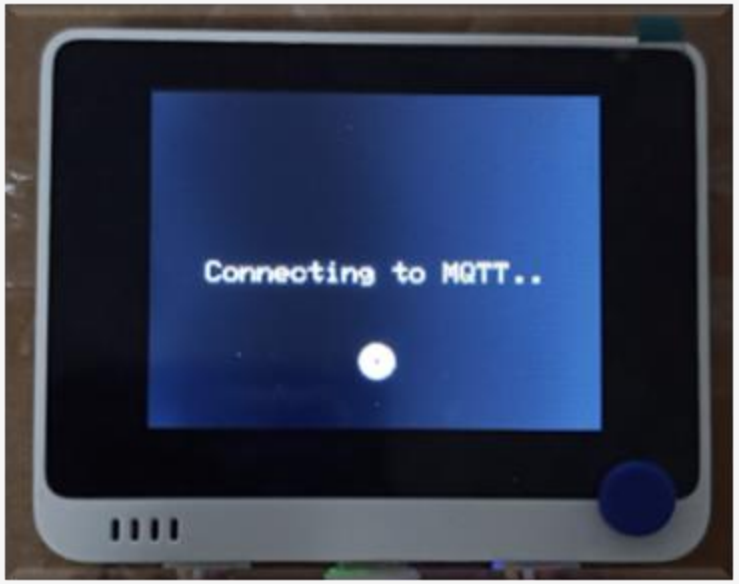
The CheeseGuardians are very reliable. If any connection is lost(WiFi connection to the router or MQTT connection) the CheeseGuardian automatically retries to connect.

The MQTT infrastructure allows to use many sensors on different sites just modifying the channel where the measurements are published. This allows an agile infrastructure where sensors can be moved in different rooms.

The database can accommodate a table for every CheeseGuardian's data, allowing a very big network of sensors.

If new calculations on the data are needed, the current infrastructure doesn't change. The data retrieved from the SQLite database can be used for whatever analysis is needed(ex. Days where the temperature exceeded the thresholds or average temperature between the rooms).

Other machine learning algorithms can be implemented for further data understanding and better business decisions.




```
Wio-CheeseGuardian/CheeseFactory001/Sensor0001
Wio-CheeseGuardian/CheeseFactory001/Sensor0002
Wio-CheeseGuardian/CheeseFactory001/Sensor0003
Wio-CheeseGuardian/CheeseFactory999/Sensor0001
Wio-CheeseGuardian/CheeseFactory500/#
```

```
sqlite> .tables
sensor_data_0001  sensor_data_0004  sensor_data_0007  sensor_data_9997
sensor_data_0002  sensor_data_0005  sensor_data_0008  sensor_data_9998
sensor_data_0003  sensor_data_0006  sensor_data_0009  sensor_data_9999
```

Security

Threat Example	Threat Type	Security Control	Threat Target	Possible Attack Techniques	Security Measures
The database is populated with malicious data	Data tampering	Authentication Integrity	SQLite Database	DB server compromise, social engineering	Use of authentication and different profiles on the DB
Data is read or modified during transfer by an attacker	Data tampering	Integrity Confidentiality	Data in motion (sensor -> DB) (DB -> website)	MITM attack	DB and broker hosted locally behind a firewall. Communication between website and DB though secure SSH protocol
Data is sent to the DB as if the attacker is a sensor	Identity spoofing	Authentication Integrity	SQLite Database	Access to physical assets Exposed DB	Each CheeseGuardian has a certificate used to sign the data that has been recorded
The website is slowed down through DoS attack	Denial of Service	Availability	Web server SQL DB	Dos or DDoS on web server or DB	Services hosted locally behind firewall or anti-DoS technology like Cloudflare(L3,L4 and L7)
Sensors are physically compromised	Bypassing physical security	Integrity	CheeseGuardians	Unplug the devices from power or the sensors from the devices	Flag on the DB when a sensor is offline. Better physical security(ex. CCTV)
DB is physically compromised	Bypassing physical security	Integrity Confidentiality	SQLite Database	Stealing the hard drives with the data	Asymmetric key encryption on the data at rest. SHA3 check on the data

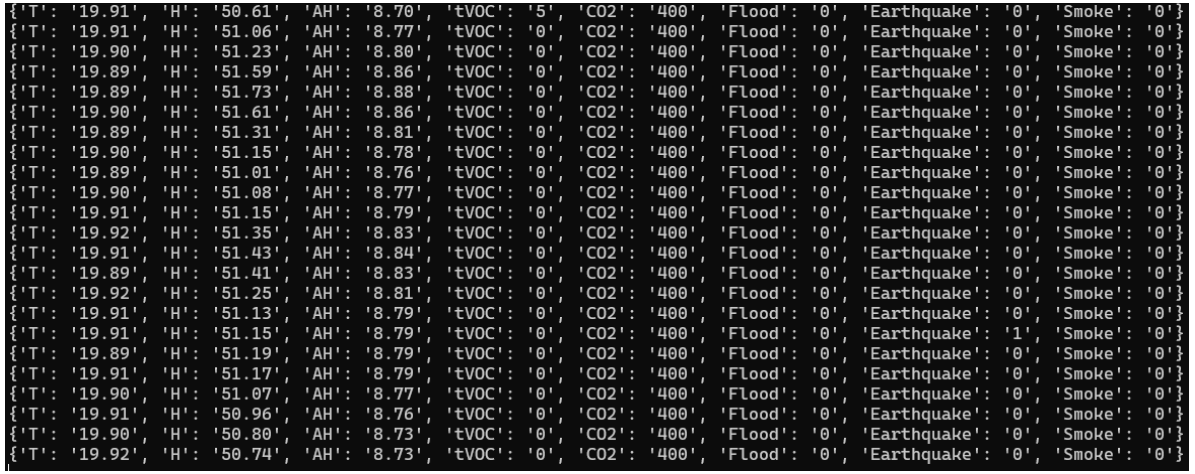
Demo

Video demo is attached

Demo

- graphData.py manages the webserver using Streamlit
- MakeSmokeDecision.py has been used to setup the ML methods
- readData.py is the MQTT listener that saves data on the DB
- sensor_data.db and sensor_data_0001 are the DB containing the experiment's data
- svn.mod is the SVM model obtained from the training data

Demo



```
{'T': '19.91', 'H': '50.61', 'AH': '8.70', 'tVOC': '5', 'CO2': '400', 'Flood': '0', 'Earthquake': '0', 'Smoke': '0'}
{'T': '19.91', 'H': '51.06', 'AH': '8.77', 'tVOC': '0', 'CO2': '400', 'Flood': '0', 'Earthquake': '0', 'Smoke': '0'}
{'T': '19.90', 'H': '51.23', 'AH': '8.80', 'tVOC': '0', 'CO2': '400', 'Flood': '0', 'Earthquake': '0', 'Smoke': '0'}
{'T': '19.89', 'H': '51.59', 'AH': '8.86', 'tVOC': '0', 'CO2': '400', 'Flood': '0', 'Earthquake': '0', 'Smoke': '0'}
{'T': '19.89', 'H': '51.73', 'AH': '8.88', 'tVOC': '0', 'CO2': '400', 'Flood': '0', 'Earthquake': '0', 'Smoke': '0'}
{'T': '19.90', 'H': '51.61', 'AH': '8.86', 'tVOC': '0', 'CO2': '400', 'Flood': '0', 'Earthquake': '0', 'Smoke': '0'}
{'T': '19.89', 'H': '51.31', 'AH': '8.81', 'tVOC': '0', 'CO2': '400', 'Flood': '0', 'Earthquake': '0', 'Smoke': '0'}
{'T': '19.90', 'H': '51.15', 'AH': '8.78', 'tVOC': '0', 'CO2': '400', 'Flood': '0', 'Earthquake': '0', 'Smoke': '0'}
{'T': '19.89', 'H': '51.01', 'AH': '8.76', 'tVOC': '0', 'CO2': '400', 'Flood': '0', 'Earthquake': '0', 'Smoke': '0'}
{'T': '19.90', 'H': '51.08', 'AH': '8.77', 'tVOC': '0', 'CO2': '400', 'Flood': '0', 'Earthquake': '0', 'Smoke': '0'}
{'T': '19.91', 'H': '51.15', 'AH': '8.79', 'tVOC': '0', 'CO2': '400', 'Flood': '0', 'Earthquake': '0', 'Smoke': '0'}
{'T': '19.92', 'H': '51.35', 'AH': '8.83', 'tVOC': '0', 'CO2': '400', 'Flood': '0', 'Earthquake': '0', 'Smoke': '0'}
{'T': '19.91', 'H': '51.43', 'AH': '8.84', 'tVOC': '0', 'CO2': '400', 'Flood': '0', 'Earthquake': '0', 'Smoke': '0'}
{'T': '19.89', 'H': '51.41', 'AH': '8.83', 'tVOC': '0', 'CO2': '400', 'Flood': '0', 'Earthquake': '0', 'Smoke': '0'}
{'T': '19.92', 'H': '51.25', 'AH': '8.81', 'tVOC': '0', 'CO2': '400', 'Flood': '0', 'Earthquake': '0', 'Smoke': '0'}
{'T': '19.91', 'H': '51.13', 'AH': '8.79', 'tVOC': '0', 'CO2': '400', 'Flood': '0', 'Earthquake': '0', 'Smoke': '0'}
{'T': '19.91', 'H': '51.15', 'AH': '8.79', 'tVOC': '0', 'CO2': '400', 'Flood': '0', 'Earthquake': '1', 'Smoke': '0'}
{'T': '19.89', 'H': '51.19', 'AH': '8.79', 'tVOC': '0', 'CO2': '400', 'Flood': '0', 'Earthquake': '0', 'Smoke': '0'}
{'T': '19.91', 'H': '51.17', 'AH': '8.79', 'tVOC': '0', 'CO2': '400', 'Flood': '0', 'Earthquake': '0', 'Smoke': '0'}
{'T': '19.90', 'H': '51.07', 'AH': '8.77', 'tVOC': '0', 'CO2': '400', 'Flood': '0', 'Earthquake': '0', 'Smoke': '0'}
{'T': '19.91', 'H': '50.96', 'AH': '8.76', 'tVOC': '0', 'CO2': '400', 'Flood': '0', 'Earthquake': '0', 'Smoke': '0'}
{'T': '19.90', 'H': '50.80', 'AH': '8.73', 'tVOC': '0', 'CO2': '400', 'Flood': '0', 'Earthquake': '0', 'Smoke': '0'}
{'T': '19.92', 'H': '50.74', 'AH': '8.73', 'tVOC': '0', 'CO2': '400', 'Flood': '0', 'Earthquake': '0', 'Smoke': '0'}
```

Data received by the MQTT listener

```
sqlite> select * from sensor_data_0001 order by id desc limit 15;
id    temperature  humidity  abs_humidity  tVOC  CO2   flood  earthquake  smoke  timestamp
----  -----------  --------  ------------  ----  ----  -----  ----------  -----  -------------------
8747  19.98        51.97     8.97          141   1060  0      0           0      2023-11-30 08:36:14
8746  19.98        51.85     8.95          5     400   0      0           0      2023-11-30 08:36:12
8745  19.98        51.84     8.95          5     400   0      0           0      2023-11-30 08:36:11
8744  19.98        51.65     8.91          14    400   0      0           0      2023-11-30 08:36:10
8743  19.99        51.16     8.84          27    400   0      0           0      2023-11-30 08:36:09
8742  19.97        50.95     8.79          28    400   0      1           0      2023-11-30 08:36:08
8741  19.98        50.7      8.75          47    400   0      0           0      2023-11-30 08:36:07
8740  19.96        50.48     8.7           52    400   0      0           0      2023-11-30 08:36:06
8739  19.97        50.59     8.72          59    400   0      0           0      2023-11-30 08:36:04
8738  19.98        50.85     8.78          77    400   0      1           0      2023-11-30 08:36:03
8737  19.98        51.23     8.84          94    400   0      1           0      2023-11-30 08:36:02
8736  19.96        51.67     8.91          106   400   0      1           0      2023-11-30 08:36:01
8735  19.96        52.14     8.99          123   400   0      0           0      2023-11-30 08:36:00
8734  20.0         52.43     9.06          147   400   0      0           0      2023-11-30 08:35:59
8733  19.99        52.23     9.02          173   400   0      0           0      2023-11-30 08:35:58
```

Data organized and saved in the SQLite Database