RESEARCH ARTICLE OPEN ACCESS

# Design and Implementation of a Real-Time Weather Forecasting System Based On IOT

**Alphonse Binele Abana*, Brice Ekobo Akoa*, Patrick Dany Bavoua Kenfack*, Paul Salomon Ngohe Ekam*, Emmanuel Tonye*, Jean Daibes***

Department of Electrical and Telecommunication Engineering,
University of Yaounde 1, Yaounde, Cameroon

**ABSTRACT**

The objective of this project is to the design and simulate the implementation of a real-time weather forecasting system based on the Internet of Things (IoT). The solution proposes a network architecture that should collect, transmit, and display meteorological data, based on "smart" weather stations installed in different regions. These stations, based on microcontrollers, send the data via mobile networks to a central server. Efficient, reliable, and secure communication between the different components is carried out via the MQTT protocol. The collected data is stored in a relational database (BDD) and displayed on a dynamic website open and accessible 24 hours a day. This work highlights the integration between hardware, software, and network components in an IoT environment, and proposes a modern and efficient solution for environmental monitoring. The system can be extended to support additional sensors such as air quality, wind speed, and UV index, making it scalable and adaptable to various use case.

***Keywords*** — Weather stations ; IoT; Smart cities; MQTT; Network simulation

## I. INTRODUCTION

Meteorological phenomena have always aroused the interest of humanity. The first attempts to observe the weather were based on empirical methods, essentially based on direct observation of nature and atmospheric signs. Thus, in the face of dangerous climate change and the increasing frequency of extreme weather events, the need for accurate, reliable, and real-time environmental data is becoming an urgent need. These data play a vital role in sectors such as agriculture, transportation, tourism, public health, and natural resource management. We therefore propose an experimental IoT-based real-time weather forecasting system consisting of virtual weather stations (simulated in Cisco Packet Tracer) communicating with a real central server responsible for collecting and processing meteorological data. The proposed system will therefore allow organizations responsible for managing meteorological data, which still collect these data manually, to get an idea of the scope of IoT in the provision of their services

## II. CONTEXT AND ISSUES

An easy way to comply with the conference paper formatting requirements is to use this document as a template and simply type your text into it.

For a better understanding of our work, it is important to define some important elements.

### *A. Definitions*

#### *Weather station [1]*

It is a set of measuring devices, made up of different sensors that record and provide information concerning physical measurements related to climate variations. These physical quantities can be temperature, wind speed, rainfall, etc.

#### *Climate and weather [2]*

Climate and weather are different but related concepts. Weather describes the local weather situation, current or foreseeable, and climate provides information on average weather conditions in a specific place and over a specific period. We would therefore speak of "climate in the Mediterranean" if we wish to describe the weather curve in the Var, for example, and we would ask about "weather conditions" in Toulon if we wish to know whether it will rain there tomorrow or whether the weather will be nice...

#### *FACTORS THAT INFLUENCE CLIMATE*

In each region, the climate is characterized by a set of measurable climatic parameters that can be related to the weather configuration: pressure, precipitation, humidity, winds, temperature, etc. These climatic parameters are influenced by climatic factors, such as latitude, altitude, presence of significant bodies of water (rivers, oceans, etc.), layout and characteristics of the relief, vegetation, etc.

#### *Smart City [3]*

It is an urban area where technology and data collection help improve the quality of life as well as the sustainability and efficiency of municipal operations. Smart city technologies used by local governments include information and communication technologies (ICT) and the Internet of Things (IoT).

Areas of city activity where ICT, IoT, and other smart technologies are playing an increasingly important role including transportation, energy, and infrastructure. As a city updates its systems and structures to incorporate these technologies, it becomes smarter. However, there is debate about exactly which cities should be considered smart or should claim the title of "smartest" city.

Just as the human body's nervous system governs how humans react to the world around them, evolving technologies enable cities to respond to changes in their local and urban environments.

Data collection technologies, including real-time data, are central to smart city initiatives and the benefits they promise. The data collected helps local governments improve urban planning and the deployment of municipal services, from waste management to public transportation, resulting in a better quality of life for residents.

More efficient municipal services can also help reduce carbon emissions, contributing to global efforts to combat climate change while improving local air quality. Additionally, smart city solutions can drive economic growth, as improved infrastructure and technological innovation can foster job creation and business opportunities.

***Internet of Things (IoT) [4]***

The Internet of Things (IoT) represents a vision of a world where billions of objects, equipped with embedded intelligence, means of communication as well as detection and action capabilities, will be connected via IP (Internet Protocol) networks.

The Internet of Things (IoT) touches every aspect of our lives. IoT-based applications are found in a wide range of scenarios, including home and building automation, smart cities, smart grids, Industry 4.0, and smart agriculture. In each of these areas, the use of a common (IP-oriented) communication protocol stack enables the creation of innovative applications.

### *B. B.Brief presentation of the Lebanese meteorological service.*

Let us take the case of the Lebanese Meteorological Service, which is a public institution attached to the 'Ministry of Public Works and Transport'. This public service is the sole source of all information concerning weather forecasting and climate in Lebanon, including meteorological services. On its website (https://www.meteo.gov.lb ), it disseminates:

- The weather forecast: Delivered twice a day, one in the morning and the other at night, each forecast predicts the weather for the next three or four days. It is estimated that around 730 forecasts are published per year.
- The Daily Report: This is a report for all stations that cover almost all Lebanese territories; these stations send their weather measurements continuously to the airport data center, and employees, at the beginning of each day, display this data in a sheet called: "Daily Weather Report".

This service includes 15 weather stations, installed and distributed in the different Lebanese territories, and connected to the central office (at Beirut airport), with a connection via the PSTN network (a modem and a telephone line), and the data from each measuring instrument is sent to the central office 3 or 4 times a day.

### *C. Problematic*

We can identify 3 types of weather station implementations:

#### *1) Case 1: Use of conventional measuring devices*

The values are recorded manually and then entered on a form containing the date, time, type of measurement (temperature, humidity, atmospheric pressure, etc.) and the recorded value. This information is then entered into an appropriate computer system.

Risks:

- Possible errors when reading values.
- Input errors.
- Heavy reliance on staff to carry out these operations

#### *2) Case 2: Use of advanced measuring devices with memory card*

The measurements are automatically recorded on a memory card as a text file. The cards are then removed, the files extracted and sent to the main office to be imported into the database server.

- Risks:
- Confusion between files from different stations.
- Missing or lost some files.
- File corruption.

Continued reliance on personnel for data handling and transfer

#### *3) Case 3 : Use of IoT for automatic data collection and transfer real-time meteorological data to a central processing system*

This is the case we propose in this project. It constitutes a modern alternative to traditional methods by addressing the main limitations, especially in terms of data accuracy, real-time availability and reduced dependence on human intervention. In addition, it allows the integration of advanced management and configuration tools. For example, a dashboard can be set up to visualize the operational status of the stations (active or offline), to configure variables such as the measurement publication interval (for example, every five minutes), or to ensure, as far as possible, remote maintenance and troubleshooting operations.

Conventional weather systems, still widely used (Lebanon and several other countries), rely on the first two cases . These methods lead to high costs, risks of errors, limited geographical coverage and low real-time availability.

Examples include:

- A collection and sometimes manual entry of data from measuring devices.
- Manual manipulation of measurements recorded on memory cards integrated into the devices.
- A data transfer to a main server following limited schedules and exhausting procedures.
- Limited geographic coverage.
- Low frequency of time reports or bulletins.
- Communication channels are sometimes obsolete.
- Insufficient security measures.
- A lack of publication or online access.

These practices entail numerous constraints: risk of human error, high management costs, difficulty of control and maintenance, high dependence on the availability of human resources, delays in data transmission, and security vulnerabilities.

In a context marked by climate change and the need to integrate smart cities, it is becoming urgent to design a modern, automated and reliable solution capable of collecting, transmitting and disseminating meteorological data in real time, with extensive coverage and better security.

So, this project aims to design and implement (simulation) a real-time weather forecasting system based on IoT, which allows:

- Automatic collection of weather data parameters.
- The secure and reliable transmission of these parameters to a database on a web server.
- Dissemination of information in real time using a website.
- Wider geographical coverage including rural and wild areas.
- It therefore aims to show through simulation the steps to follow to set up a modern, economical, reliable and easy-to-manage meteorological system in terms of control and maintenance for engagement in a smart city context .

## III. METHODOLOGY

### A. *Overview*

In order to carry out this project the following techniques are integrated:

- A simulation with Cisco Packet Tracer: which highlights an entire network architecture with IPv4, including IoT equipment, routers, and appropriate servers...
- The WampServer [5] installed on a Windows 10 host machine, providing: Apache 2.4, -MySQL 5 & 8, PHP 5, 7 & 8, used to host the website in PHP, and for database management.
- The MQTT communication protocol [6]: the MQTT (Message Queuing Telemetry Transport) protocol is a publishing / subscribing messaging protocol . It is an application layer protocol in the OSI model, based on TCP, essential for low-bandwidth environments, and with unstable connections. It is a lightweight protocol (in terms of processing and energy consumption), which makes it more suitable for the Internet of Things.

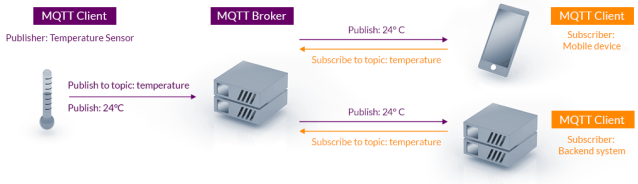


Fig. 1. MQTT Publish / Subscribe Architecture

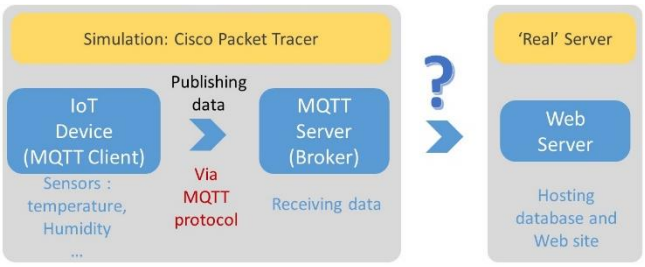


Fig. 2. Overview

This illustration represents the general architecture of a weather monitoring system using Cisco Packet Tracer for simulation, and a real server for data collection and visualization on the web.

#### *1) Simulation – Cisco Packet Tracer*

- IoT Devices (MQTT Client) : These are weather stations in the form of microcontroller boards with simulated sensors (temperature, humidity, etc.) that collect environmental data.
- These devices then publish their data via the MQTT protocol to a central server.
- Server (MQTT Broker) : it receives all data sent by the stations. This server acts as a messaging intermediary , temporarily storing MQTT messages.

#### *2) Transition to the real world*

The symbol " ? " ( question mark ) in Figure 2 here represents the link to be designed to enable communication between the virtual meteorological data collection and transfer system (implemented on the Packet Tracer simulator) and the real world.

#### *3) ' Real ' Server*

The "Web Server" hosts the database as well as the website intended for displaying weather data; it is created using the WampServer tool.

Packet Tracer simulation and the 'Real' server is made using a class named " RealHTTPClient ( )" in python with the following methods :

- http = RealHTTPClient ( )
- http.onDone ( onHTTPDone )
- http.post ( url ,register)

The python code below links to a PHP module on the 'real' server, named insert_form.php (its url is: http://127.0.0.1/meteo/insert_form.php ); it will insert the new values into the database at each time interval (set to 5 min) via the PHP module insert_form.php .

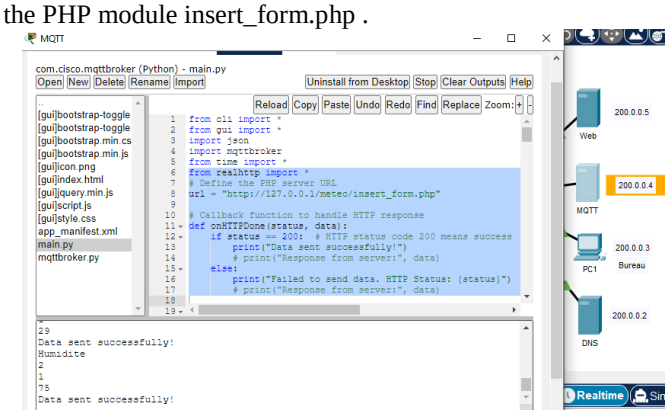

Fig. 3. Python code – ReallHTTP class

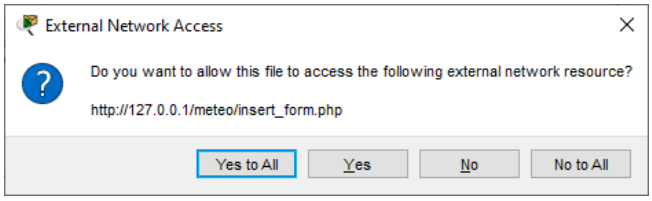


Fig. 4. Permission to access the external database

**Python code of the publication modules (case of the Beirut station, Station_ID = 1):**

```
def publishTemperature ( value ):
        temperature_data = {
        "Station ": "1",
        "value ": value
        }
        topic = " Temperature " #will be tested and replaced with code 1 upon receipt
        payload = json.dumps ( temperature_data )
        qos ="1"

        mqttclient.publish (topic, payload, qos )

        CLI.exit ()

def publishHumidity (value ):
        humidity _data = {
        "Station ": "1",
        "value ": value
        }
        topic = " Humidity " #will be tested and replaced with code 2 upon receipt
        payload = json.dumps ( humidity_data )
        qos ="1"

        mqttclient.publish (topic, payload, qos )

        CLI.exit ()
```

B. ***Network architecture***

The proposed architecture is designed to collect, transmit, and publish weather data in real time. Two stations in Beirut and Tripoli measure temperature and humidity using sensors connected to microcontrollers. This data is transmitted via cellular stations to the Central Office Server, which relays the information to the dedicated servers. Using the MQTT protocol, the data is published and received by the MQTT server, which in turn sends it to the dedicated web server, hosting the database, where it is saved.

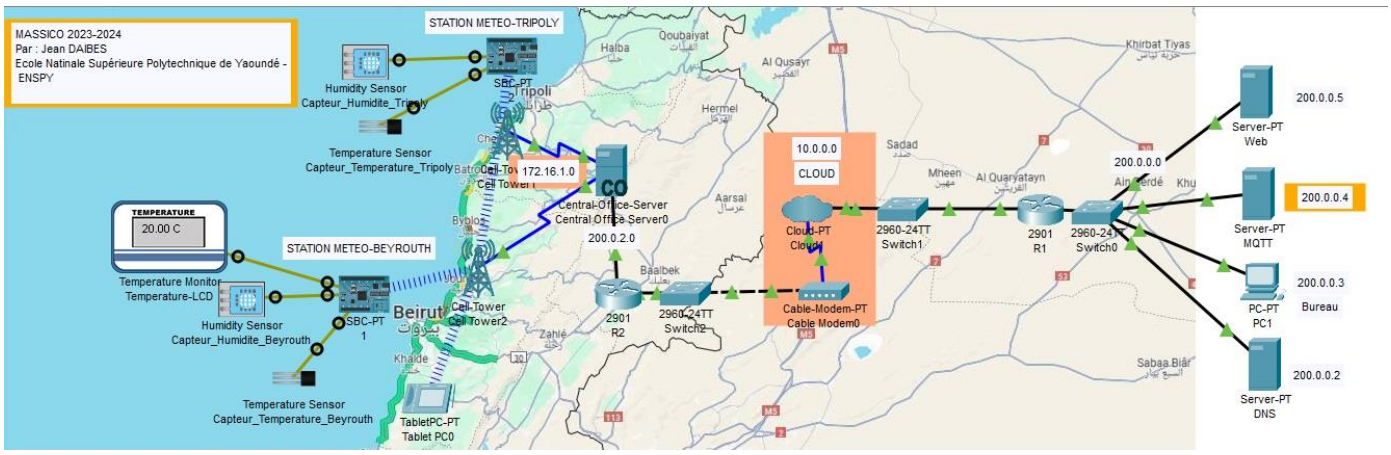


Fig. 5. Network simulation

C. ***Detailed description of the network***

***1) WEATHER STATION - BEIRUT***

As already mentioned, a station is formed by a microcontroller card to which are connected:

- Sensor_Temperature_Beirut : Measures temperature (randomly generated values between val_Min and val_Max using Javascript code).
- Sensor_Humidity_Beirut : Humidity sensor (Also randomly generated values).

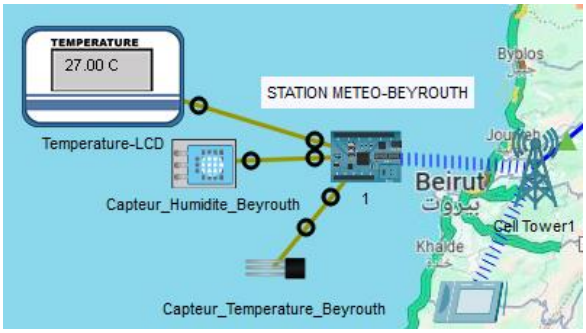


Fig. 6. Beirut Station

Remarks :

- In the real world, physical sensors will measure the exact parameters of the surroundings.

- The formula for random generation is: temp = ( Math.random () * ( val_Max - val_Min ) ) + val_Min );

We can also add other sensors. The map is driven by a mobile connection interface.

Note : The temperature and humidity sensors are connected to the SBC board on digital ports D0 and D1 respectively.

### 2) *Data transmission*

- Cell Tower1 (Beirut) and Cell Tower2 (Tripoli) : Represent the cellular telecommunications stations that provide wireless data transmission to the Central Office Server .
- Network IP Address: 172.16.1.0 /24

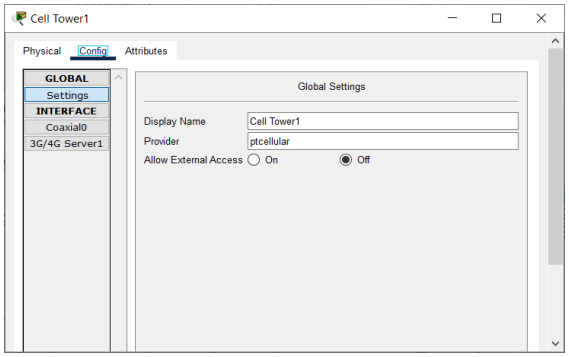


Fig. 7. Cell Tower Configuration

### 3) *Central Office Server0*

- Receives weather data from stations in Beirut and Tripoli.
- Cell Tower Interface : IP: 172.16.1.1 /24
- Backbone interface : IP: 200.0.2.2 /29
- Serves as a gateway to the rest of the network and relays data to the destination via router R2 .


Fig. 8. Central Office Server

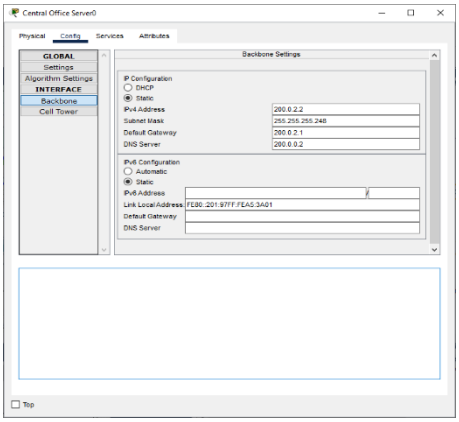


Fig. 9. Central Office Server - Backbone interface

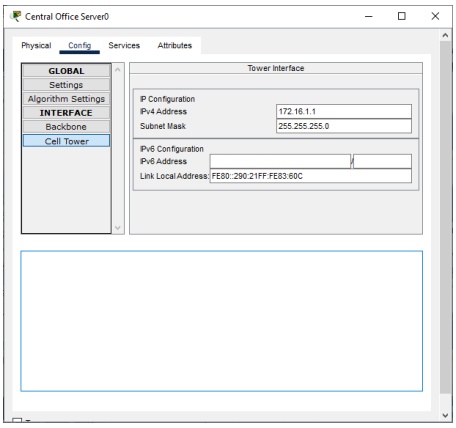


Fig. 10. Central Office Server - Cell Tower interface

### 4) *Cloud Infrastructure*

- Cloud (10.0.0.0) : Represents a complex Internet infrastructure, it is used here to complicate the network architecture a little and assume having varieties of subnets within it.
- Cable Modem0 : Provides connectivity between the core network (via R2) and the cloud.
- Switch1 : Connects the cloud to the rest of the network

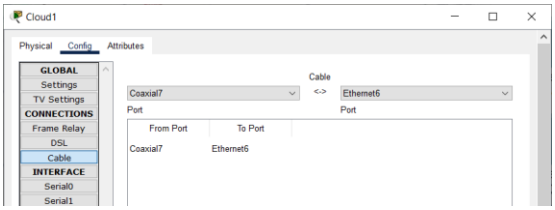


Fig. 11. Cloud Configuration

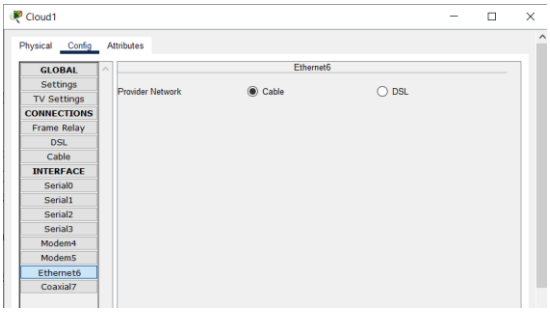


**Fig. 12.** Cable Modem Configuration

### *5) Data processing and dissemination infrastructure*

The network on the right in Figure 5 is the heart of data processing and dissemination .

- R1 : Main network router that connects the cloud to the various services via Switch1 and Switch3.
- Servers in the Data Center: In this project, several servers are used to ensure the proper functioning of the collection, processing, and presentation of meteorological data. Each server plays a specific role:
- Server IP = 200.0.0.2 , Mask = 255.255.255.248 : responsible for name resolution in the internal network.

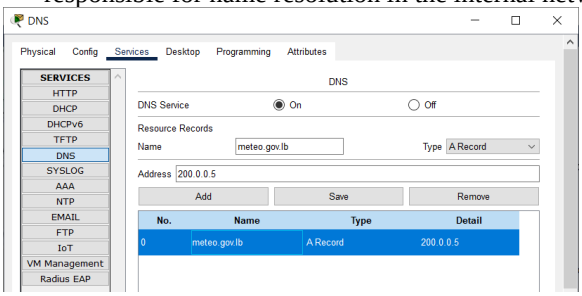


Fig. 13. DNS Server Configuration

- MQTT Server : IP = 200.0.0.4, Mask = 255.255.255.248: This server acts as an MQTT Broker. Its role is to receive data published by weather stations located in different regions. The stations, equipped with temperature, humidity, etc. sensors, send the weather parameters in the form of MQTT messages. The MQTT server collects all these messages and can then forward them to other systems or applications, including the 'real' server hosted on the Windows 10 host machine. (In the context of this project).

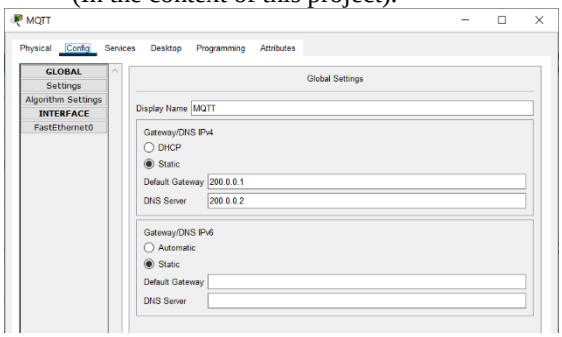


Fig. 14. MQTT server configuration

**Note:** Usually the MQTT server publishes the received data, so that the " Subscribers " can receive it, but in this case, this service is not used, since the data is collected in the database.

- R1 router configuration:
  - Enable HTTP/HTTPS (TCP port 80, 443) to the Web Server (200.0.0.5).
  - Allow only MQTT traffic (TCP port 1883) and (TCP port 8883 over TLS) to the MQTT Server (200.0.0.4).
  - Allow DNS queries (UDP and TCP at port 53) to the DNS Server (200.0.0.2).
  - Enable OSPF protocol
  - Ban all traffic
  -

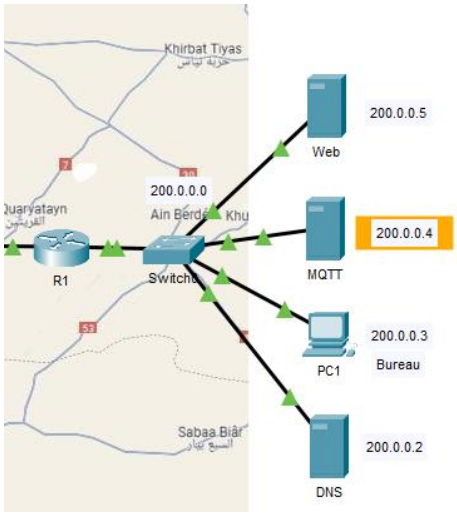


Fig. 15. Datacenter

### D. *Database Management BDD [7]:*

The Conceptual Data Model MCD [8] introduces the notion of entities, relationships and properties... The graphical representation, simple and accessible, allows a non-computer scientist to participate in its development. The basic elements constituting a conceptual data model are:

- The properties;
- The entities;
- Relationships.

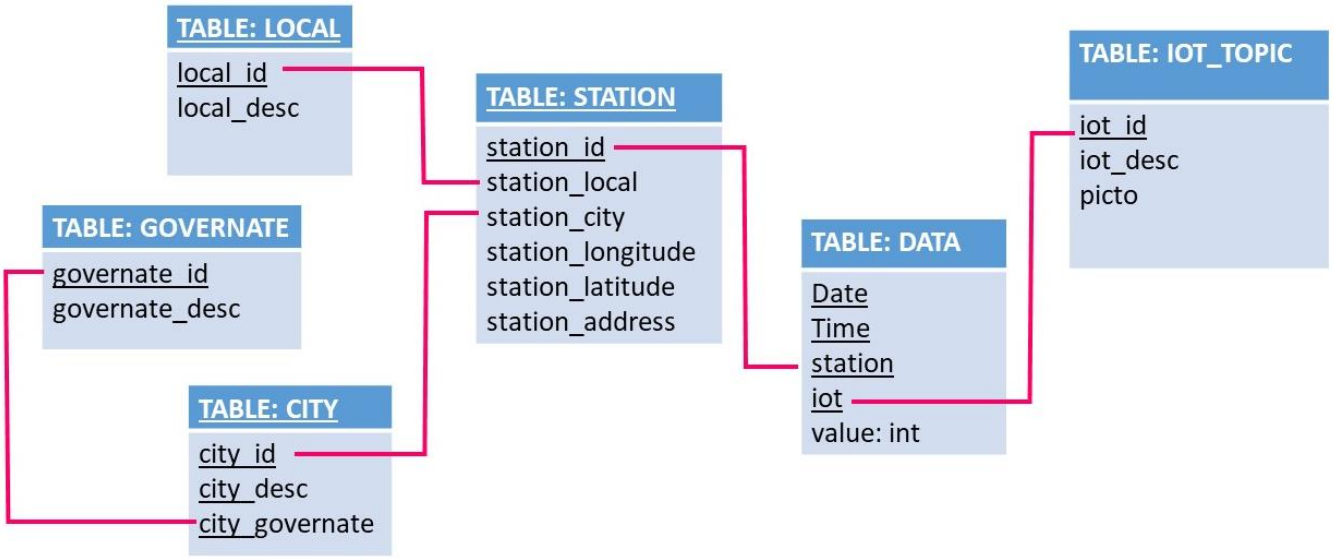


Fig. 16. Simplified Database diagram derived from the MCD diagram

The database used in this project allows saving all the information requested and planned for the operation of the weather system, such as the stations distributed over the different points, the types of objects, the values of the parameters as the data is received.

### *1) Description of database tables*

Here is a detailed description of the tables:

- LOCAL TABLE : managed in the back-end by the local_managment .php module .
  - local _id : Unique identifier of the location type (primary key).
  - local _desc : Text description of the location type (for example, this field indicates a given neighborhood or area).
- TABLE GOVERNATE: managed in the back-end by the governate_managment .php module .
  - governate _id : Unique identifier of the governorate (primary key).
  - governate _desc : Name or description of the governorate (eg: Beirut, Tripoli...).
- TABLE CITY: managed in the back-end by the ville_managment module . php
  - city _id : Unique identifier of the city (primary key).
  - city _desc : Name or description of city/village.
  - city _governate : Reference to the identifier of the governorate to which the city belongs ( foreign key to governate_id in the GOVERNATE table).
- TABLE STATION: managed in the back-end by the station_managment module . php
  - station _id : Unique identifier of the weather station (primary key).
  - station _local : Reference to the station location type ( foreign key to local_id in the LOCAL table).
  - station_city : Reference to the city where the station is located ( foreign key to city_id in the CITY table).
  - station _longitude : Longitude coordinate of the station.
  - station _latitude : Latitude coordinate of the station.
  - station_address : Descriptive or textual address of the station.
- TABLE IOT_TOPIC: managed in the back-end by the iot_managment module . php
  - iot _id : Unique identifier of the IoT sensor or subject (primary key).
  - iot _desc : Description of the data type or sensor (e.g. temperature, humidity).
  - picto : Pictogram or icon associated with this type of data for display.
- TABLE: DATA: managed by the insert_form.php module , but automatically, the data received from different stations are inserted automatically by adding the system date and time.
  - Date : The date of the measurement.
  - Time : The time of the measurement.
  - station : Reference to the station that issued the data ( foreign key to station_id in the STATION table).
  - iot : Reference to the sensor or measurement type ( foreign key to iot_id in the IOT_TOPIC table).
  - value : The measured value (integer type here, so temperature or humidity represented as an integer).

### *2) Main relationships :*

- CITY is linked to GOVERNATE via ville_governate.
- STATION is linked to CITY via station_ville , and to LOCAL via station_local .
- DATA is linked to STATION and IOT_TOPIC for recording measurements.

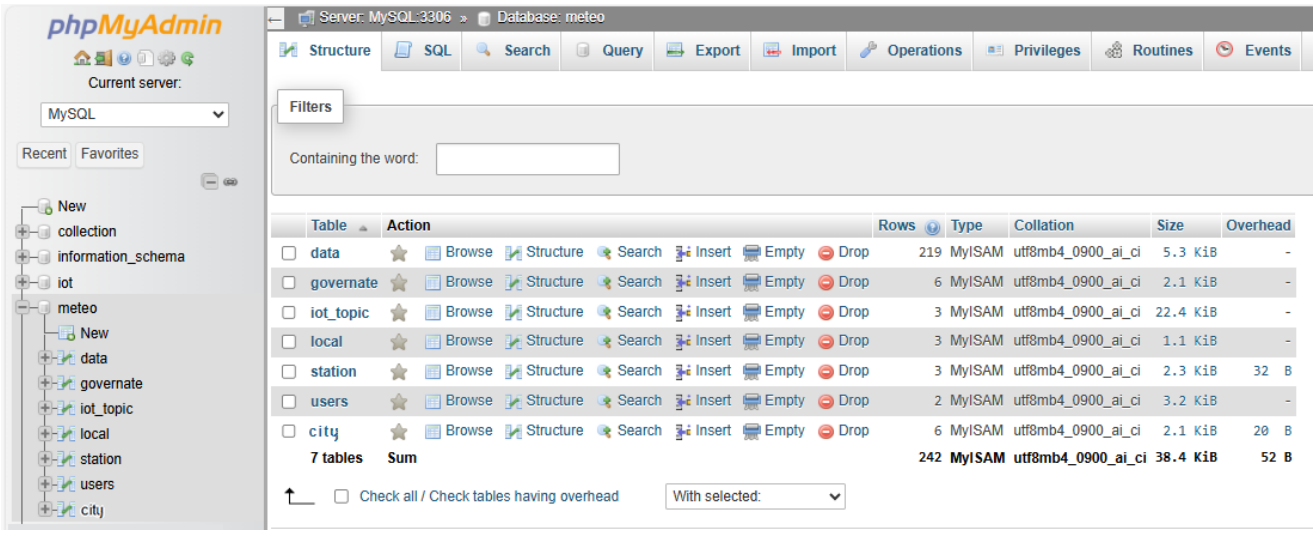


Fig. 17. Meteo Database in MySql

## E. *Organizational charts*

Cisco Packet Tracer, starting with version 7, has an integrated MQTT system, in two versions; one for the MQTT client and one for the MQTT broker. This is a source code library written in Python, with the appropriate classes and methods.

To give a general idea of how such a system works, we will simply describe simplified flowcharts for the transmission (figure 18) and reception (figure 19) of an MQTT message with a QoS set to 1.

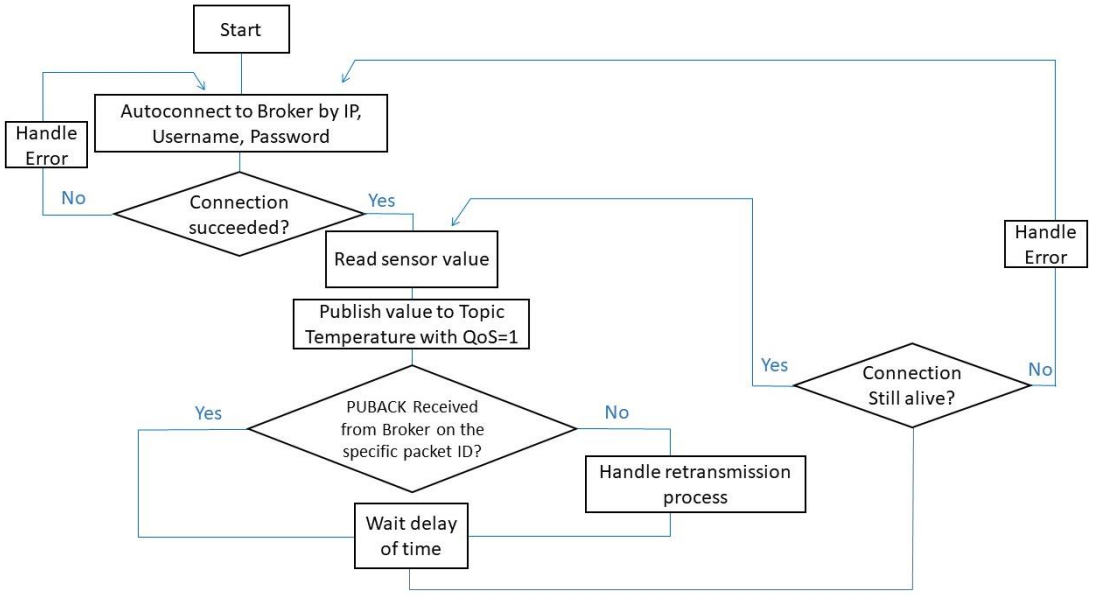


Fig. 18. Client Side

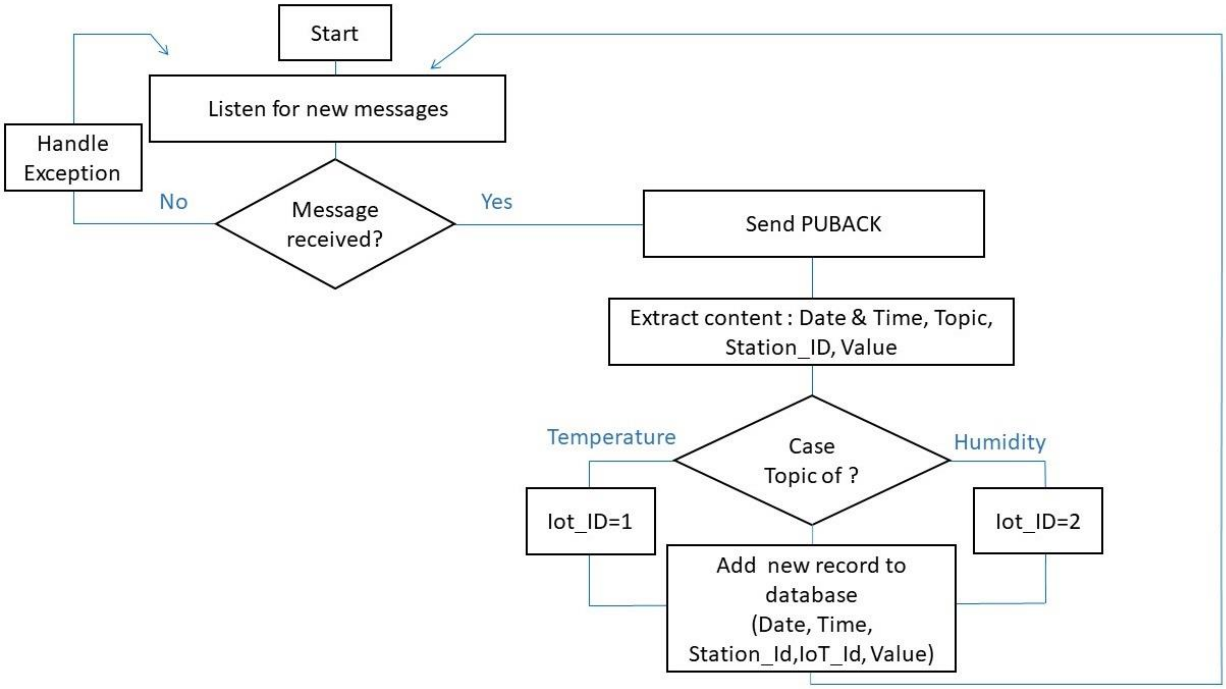


Fig. 19. Broker ( Server ) Side

## IV. RESULTS AND DISCUSSIONS

The tests carried out on the proposed system made it possible to:

- Check the connection between the weather stations and the MQTT server.
- Manipulate MQTT messages.
- Description of a received MQTT message.
- Manipulate the database.
- View weather parameters in real time on a web platform.

### A. *Checking connectivity between a station and the MQTT server.*

The 'ping' command clearly shows that the connection is established between the Beirut weather station and the MQTT server (Figure 21).

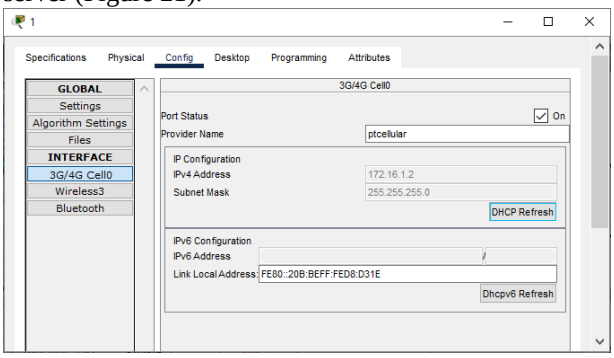


Fig. 20. Beirut Station of IP address: 172.16.1.2 assigned by the DHCP at the central office server

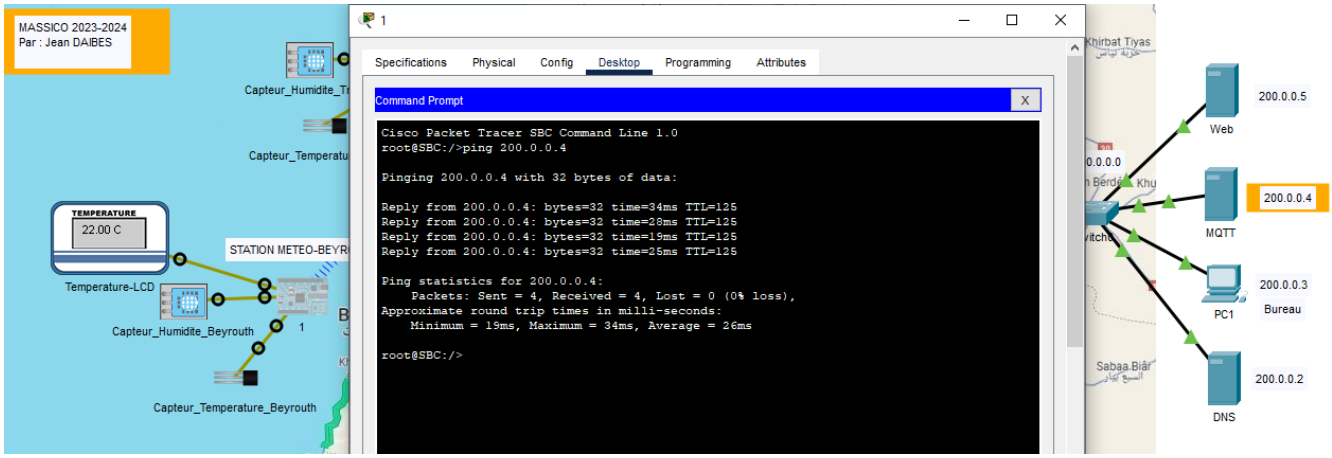


Fig. 21. PING Connection Test

### B. *Manipulating MQTT messages*

The MQTT message can be customized from the moment of transmission (figure 22) by checking its fields. Once it reaches the server, it is in the form of ' Rawdata ' from which the requested fields can be extracted, such as the Topic, the type

of measurement (temperature or humidity), the value and the number of the source station (figure 23), which will be sent again to the database hosted on the real server (figure 24).

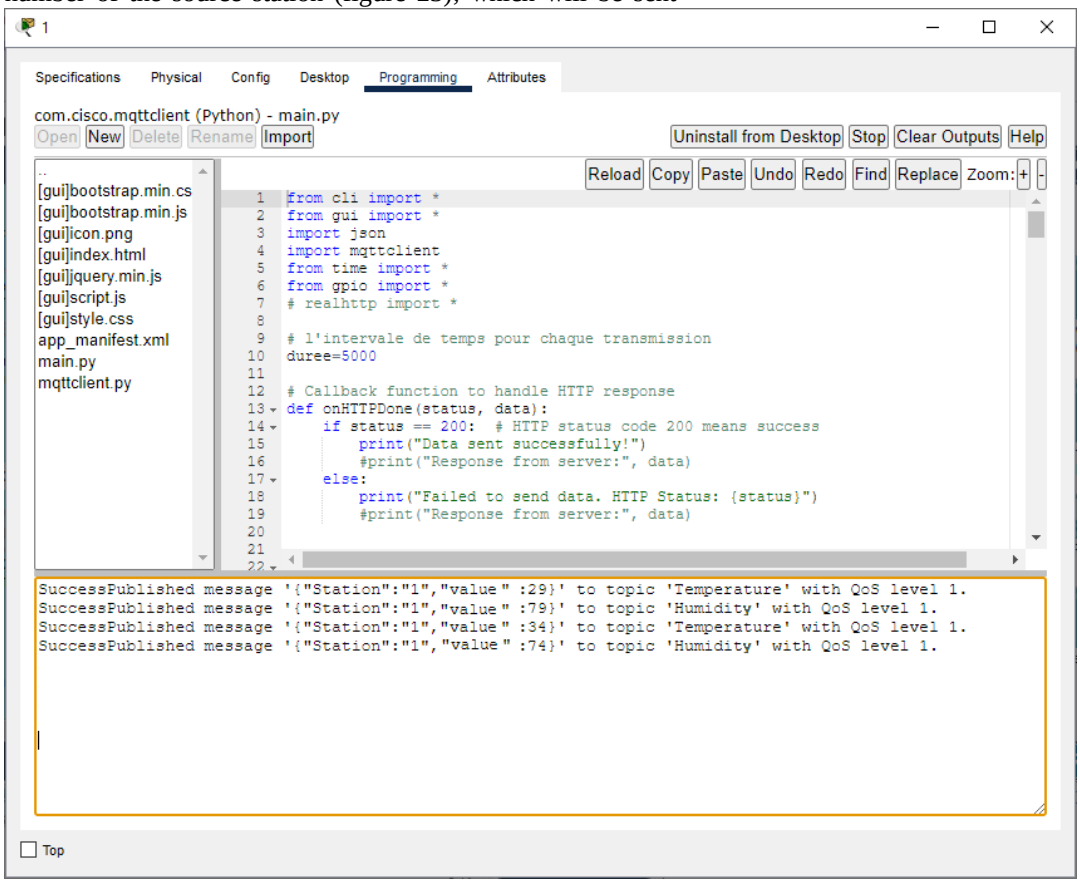


Fig. 22. Published messages by Beirut station

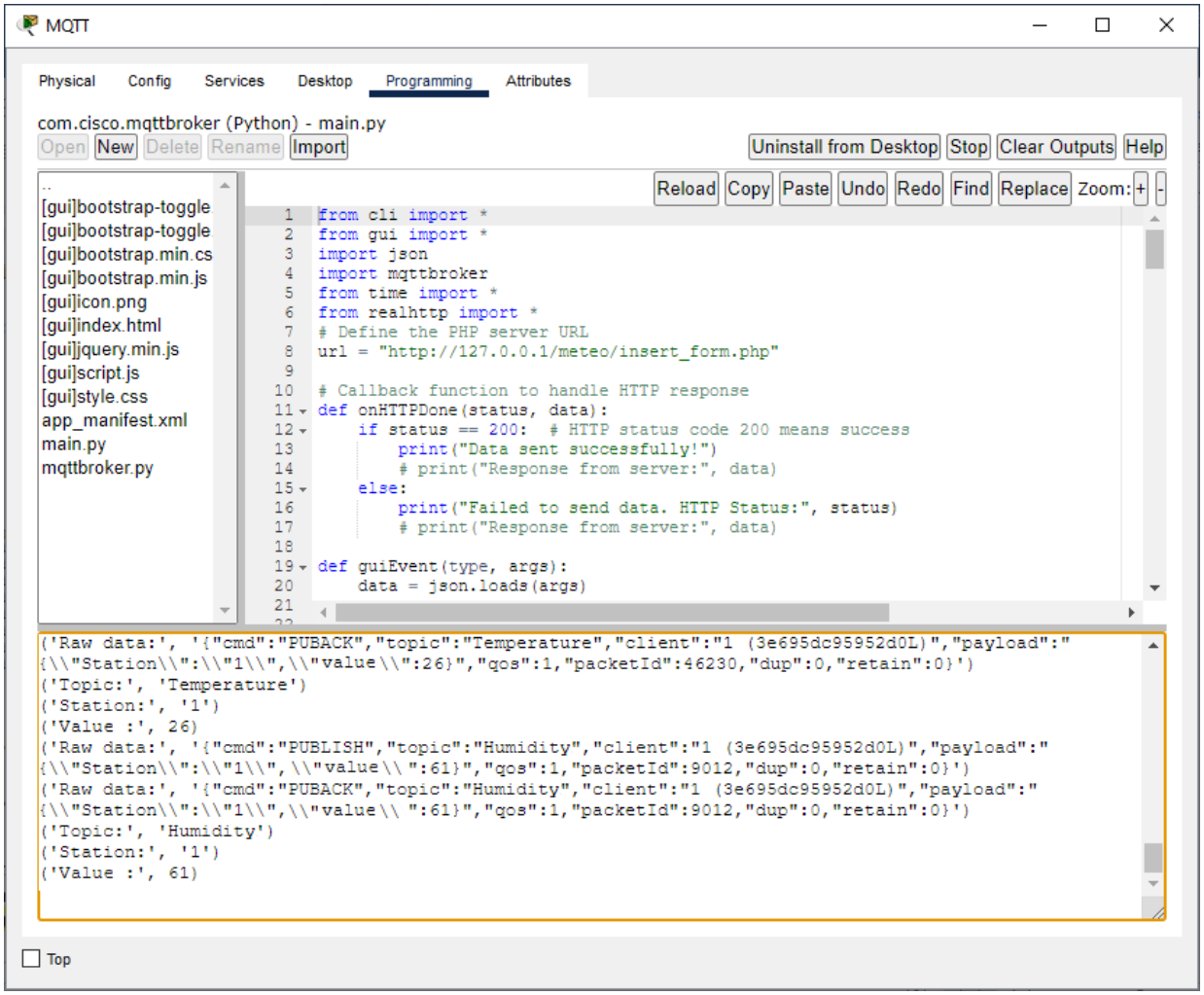


Fig. 23. Received Messages - server side (Broker)

The following assumptions will be retained in this project for configuration purposes in the database:

- Station = 1: City of Beirut
- Station = 2: City of Tripoly
- Iot = 1: Temperature (If Topic = ' Temperature ' Then Iot =1)
- Iot = 2 : Humidity (If Topic = ' Humidity ' Then Iot =2)

The detailed explanation of the contents of a received MQTT message is presented in the following section.

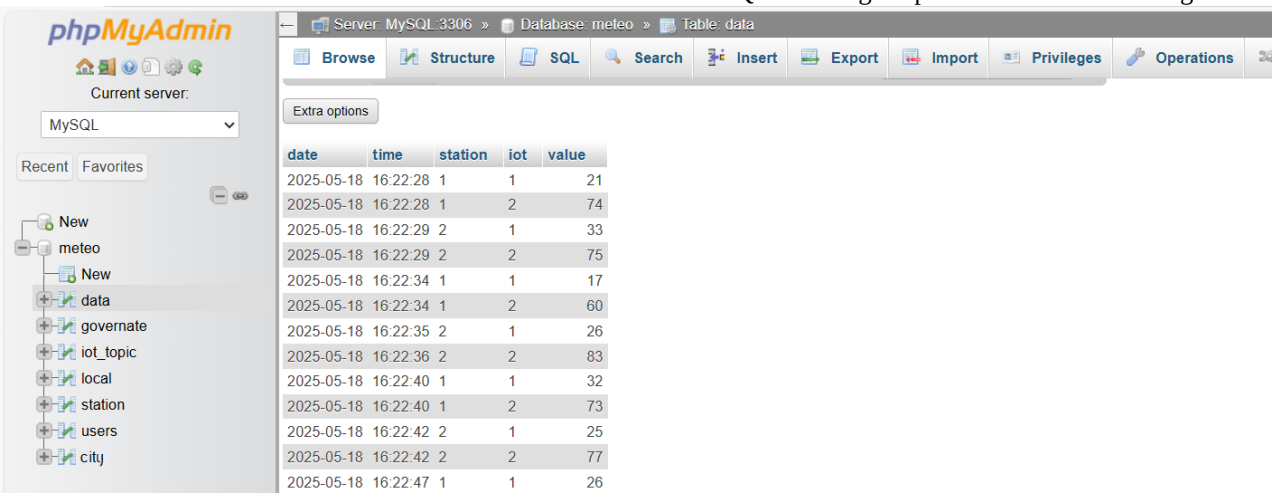


Fig. 24. Receiving automatically generated random values

C. ***Description of an MQTT message***

Let's take an example of a received message and try to explain its content:

('Raw data:' ,
'{ "cmd":"PUBACK","topic":"Humidity","client":"2 (3e659ff3262730L)","payload" :"{ \\"Station\\ ":\ \"2\\ ",\ \"value\\": 67}" ,"qos":1,"packetId":10446,"dup":0,"retain": 0}' )

JSON format

{
" cmd ": "PUBACK",
"topic": " Humidity ",
"client ": "2 (3e659ff3262730L)",
" payload ": "{\"Station\ ":\ "2\ ",\ "value\ ": 67}",
" qos ": 1,
" packetId ": 10446,
"dup": 0,
"retain": 0
}

- cmd ":"PUBACK"
  - Publish type message Acknowledgement .
  - This happens because the initial message uses QoS 1 → the broker confirms to the client that it has received the message.
  - topic": " Humidity "
  - This is the topic (communication channel) on which the message is published.
- customer": "2 (3e659ff3262730L)"
  - This is the ID of the MQTT client that published the message.
  - Here the sensor is identified as Station 2.
- payload ": "{\"Station\ ":\ "2\ ",\ "value\ ": 67}"
  - The useful content of the message ( payload ).
  - In JSON, it contains:
  - "Station ": "2" → weather station number 2,
  - "value ": 67 → the measured humidity is 67%.
- qos": 1
  - Quality of Service Level = 1.
  - Means: "at least once" → the message is acknowledged by the broker (hence PUBACK).
- packetId": 10446
  - Unique package identifier to track the exchange and manage acknowledgments.
- dup": 0
  - dup = 0 → this is not a duplicate message.
  - If it was 1, it would mean that the broker is returning a message already sent (useful in case of reconnection).
- retain": 0
  - retain = 0 → this message is not retained by the broker.

Weather station #2 published a humidity measurement = 67% on the Humidity topic , with a QoS delivery guarantee of 1.

The broker responded with an acknowledgment (PUBACK) to confirm receipt of the message.

In MQTT, the cmd field corresponds to the type of command or control packet exchanged between a client and a broker. Each MQTT packet begins with a ' Fixed Header' whose first field is the Control Packet Type (often represented as cmd):

TABLE I : CMD DESCRIPTION

| cmd ( Package Type ) | Role |
|---|---|
| **CONNECT** | Establishing a connection between a client and the broker. |
| **CONNACK** | Broker acknowledgment after a connection request. |
| **PUBLISH** | Sending a message (data) by a client on a *topic* . |
| **PUBACK** | Acknowledgement of a message published with QoS 1. |
| **PUBREC** | First response for QoS 2 (indicates that the message is received). |
| **PUBREL** | Sender confirmation for QoS 2 (message release). |
| **PUBCOMP** | Last step of QoS 2: Confirms that the message is delivered exactly once. |
| **SUBSCRIBE** | Request for a client to subscribe to one or more *topics* . |
| **SUBACK** | Broker response confirming a subscription. |
| **UNSUBSCRIBED** | Request to unsubscribe from a *topic* . |
| **UNSUBACK** | Broker's response confirming unsubscription. |
| **PINGREQ** | Request from a client to check if the broker is active ( keep -alive). |
| **PINGRESP** | Broker response to a PINGREQ. |
| **DISCONNECT** | Clean disconnection of a customer. |
| ***(MQTT 5.0 only )*** **AUTH** | Enables advanced challenge/response based authentication. |

D. ***Database manipulation***

Below we present some ' Screenshots ' of the database manipulation screens in Frontend.

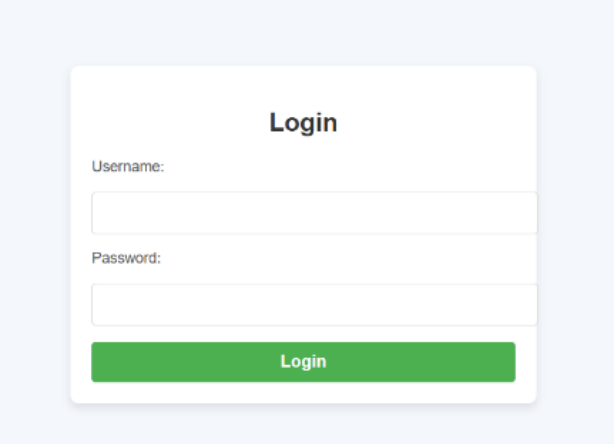


Fig. 25. System Registration

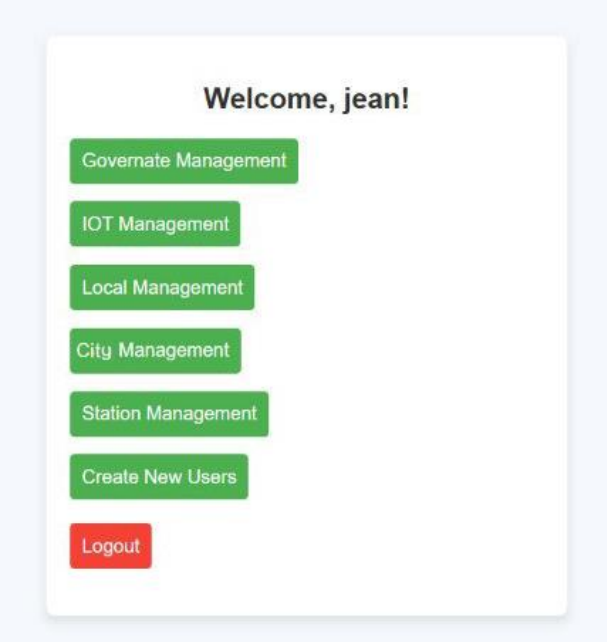


Fig. 26. Main Menu

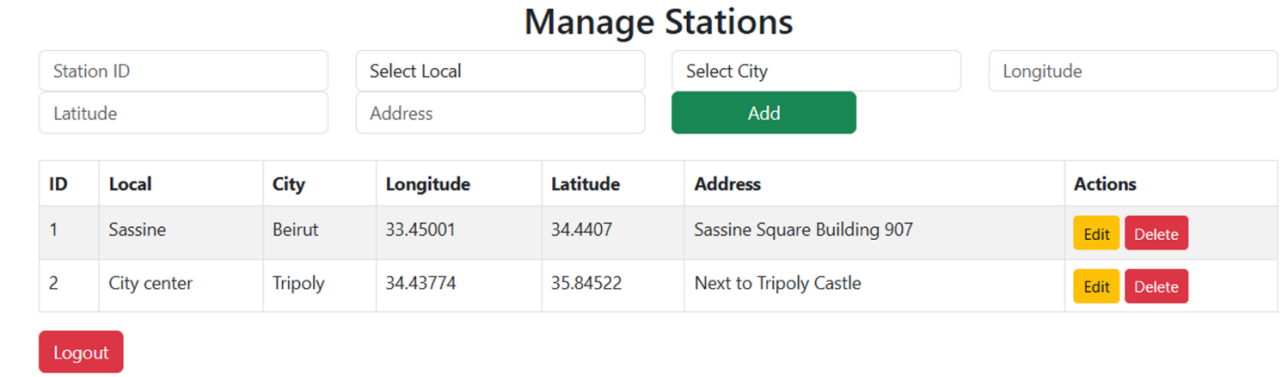


Fig. 27. Manipulating the 'Stations' Table

E. ***The module: display.php***

This is the only publicly accessible module: a web page displaying real-time weather data from various stations. This module includes an SQL query that allows you to select, for each station, the most recent temperature and humidity values, sorted appropriately (Figure 28).

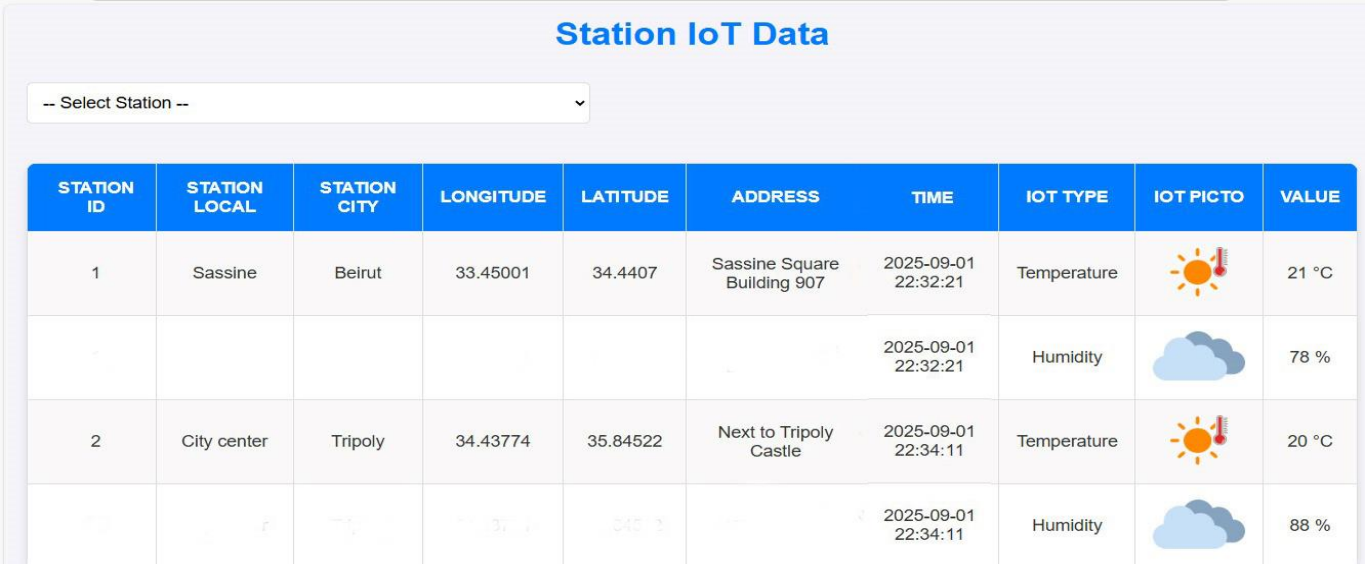


Fig. 28. Web Page – Displaying Weather Status

F. ***Discussions***

***1) MQTTS protocol option***

MQTT can guarantee a level of security suitable for sensitive applications, provided it is correctly configured, considering:

- Customer authentication: the MQTT protocol allows basic authentication by sending a username and password when a client connects to the broker. This method, while simple, can have weaknesses if the data is transmitted unencrypted. To strengthen this step, the use of digital certificates and encryption, as part of a TLS (MQTTS) connection, is recommended.
- Access control and authorizations: MQTT brokers typically incorporate access control mechanisms to limit client operations. This allows you to define which entities are allowed to publish or subscribe to specific topics, reducing the risk of misuse or unauthorized access to data.
- Reliability and integrity of messages: finally, MQTT offers three levels of Quality of Service (QoS) that guarantee a varying degree of reliability in message delivery:
  - QoS 0: at most once (no guarantee),
  - QoS 1: at least once (with acknowledgment),
  - QoS 2: exactly once (most secure).

These mechanisms ensure that published information, for example weather data, is transmitted correctly without loss or duplication.

Note : The current version of Cisco Packet Tracer does not support MQTTS (MQTT over TLS )!

***2) Choosing between IPv4 and IPv6***

This project uses IPv4, but it is preferable to migrate to IPv6. The latter offers better addressing management, which is essential in the context of connected objects.

IPv4, which is widely used and compatible with most existing devices, remains a solid option, while IPv6 offers a nearly unlimited address space, which is a major advantage in the IoT world, where each station requires a unique IP address. In addition, IPv6 incorporates security and auto-configuration features that facilitate large-scale device deployment and management.

***3) Choosing between Cisco Packet Tracer [9] and GNS3 [10]***

Cisco Packet Tracer was used for network simulation, and it achieved the goal, linking the simulation to the external real server with the Python class RealHTTPClient , but the simulation environment could be optimized with GNS3 .

GNS3 allows emulation rather than simulation through the integration of virtual machines and IOS ( Internetworking Operating System) systems.

GNS3 allows you to run real Cisco IOS images and other systems like Linux, Juniper or MikroTik . It offers a realistic and professional simulation environment, ideal for engineers. It is open source and very powerful, but requires more system resources.

## V. Conclusion

This project shows a concrete illustration of the implementation of technologies specific to smart cities while presenting a developed system of a modern and efficient solution for the collection, processing, dissemination and analysis of meteorological data . Thanks to this smart architecture, local authorities, researchers and citizens have access to reliable environmental information, updated in real time, thus facilitating data-driven decision-making. The Internet of Things ( IoT ) plays a central role here, enabling the automation of surveys, instant transmission of data and their use for monitoring and alerting purposes.

In addition to the context already presented, the project can be further improved to:

- Allow import of existing data (in national meteorological establishments) over the past years into its new database.
- Be extended and connected with other neighboring countries for monitoring air masses.
- Access satellite images for observing atmospheric layers.
- Integrate mathematical and physical formulas to predict the weather for the next few days, or even with artificial intelligence.
- Send alert SMS messages to citizens where dangers arise.

## REFERENCES


[1] Chaib M., & Daoulhadj A., Study and construction of a meteorological station , [Master's thesis, Ahmed Draïa University - Adrar, Faculty of Science and Technology], (2021), (p2).

[2] OLDANI J., Meteorology – Knowing and predicting the weather , EDITIONS DE VECCHI, (2000), (p. 58).

[3] Chourabi H., Nam T., Walker S., et al. Understanding Smart Cities: An Integrative Framework. 2012 45th Hawaii International Conference on System Sciences.(2012)

[4] Cirani S., Ferrari G., Picone M., Veltri L., Internet of Things – Architectures, Protocols and Standards , (2019). (pp . 1, 3).

[5] WAMPSERVER, Download and Installation . (2025) [Online]. Available at: https://www.wampserver.com/

[6] Naik, N. Choice of effective messaging protocols for IoT systems: MQTT, CoAP, AMQP and HTTP. 2017 IEEE International Systems Engineering Symposium (ISSE). (2017).

[7] Codd E. F, A relational model of data for large shared data banks. Communications of the ACM , (1970). ( pp. 377–387). [Online]. Available at: https://doi.org/10.1145/362384.362685

[8] BAPTISTE J., MERISE – Practical guide , EDITIONS ENI, (2009), (p. 24).
[9] Gubbi J., Buyya R., Marusic S., & Palaniswami M. Internet of Things (IoT): A vision, architectural elements, and future directions. Future Generation Computer Systems, 29 (7), (2013), 1645-1660.
[10] Silva B., Khan M., & Han K. Towards sustainable smart cities: A review of trends, architectures, components and open challenges in smart cities. Sustainable Cities and Society, 38, (2018). 697–713.
[11] Basha E., & Rus D. Design of early warning flood detection systems for developing countries. Proceedings of the 6th international conference on Information processing in sensor networks. (2007).
[12] Banks A., & Gupta R. MQTT Version 3.1.1. OASIS Standard. (2014)
[13] Batty M., Axhausen K., Giannotti F., et al. Smart cities of the future. The European Physical Journal Special Topics, 214(1),(2012). 481–518.
[14] ESCWA – United Nations Economic and Social Commission for Western Asia. Climate Change and Disaster Risk Reduction in Lebanon. (2020)
[15] Cisco Networking Academy, Getting Started with Cisco Packet Tracer . (2025) [Online]. Available at: https://www.netacad.com/fr/cisco-packet-tracer
[16] GNS3 Documentation, Getting Started with GNS3 . (2025) [Online]. Available at: https://docs.gns3.com/docs/
[17] IBM Solutions, Gomstyn A., & Jonker A., What is a Smart City? (2023). [Online]. Available at: https://www.ibm.com/fr-fr/topics/smart-city
[18] MQTT, The Standard for IoT Messaging . (2024) [Online]. Available at: https://mqtt.org/