1. 🛰️ LPO Meteorological Project – Connected station and visualization of environmental data

1.1 🔍 Origin and Context

This project is part of an educational approach combining science, technology, and environmental issues. It was born out of a concrete need: to enable the organization for the bird protection league to monitor local weather conditions independently, in real time, and with an interface accessible to all (students, teachers, staff, etc.).

The project aims to raise awareness of climate issues, assist people while enabling the mobilization of technical skills in a concrete context: electronics, embedded development, networks, data processing and visualization.

1.2 🎯 Project objectives

- **Measure -** automatically essential weather parameters: temperature, humidity, brightness, luminous flux, etc.
- **To transmit -** real-time data via a network connection to a central platform.
- **Store, process and visualize -** this data in the form of dynamic graphs that can be viewed via a web interface.
- **Involve -** students in a stimulating multidisciplinary project, linked to current events and sustainable development issues.

1.3 🧰 Techniques and solutions implemented

1.3.1. Embedded hardware (ESP32):

- **Sensors:**
  - SHT31: Measurement of temperature and humidity.
  - BH1750: Brightness measurement.
  - SEN0575: Measurement of rainfall.
  - ENS160: Measurement of CO2 level.
  - MAX9814: Measurement of sound rate.
- **Server and Client:**
  - Raspberry Pi 5: Reception, Storage and Visualization.
  - ESP 8266: Sensor data recovery and retransmission to server.
- **Camera:**
  - ESP32-CAM: Capture the video - stream.

1.3.2. Processing, distribution and display via PC:

- Data sent via JSON request in HTTP POST to the Server.
- Python script: Receiving and transmitting to MQTT to link Node-Red and InfluxDB.
- Broker MQTT direction topic LPO.

1.4 🌍 Educational and environmental impacts

- **Educational Scope:** mobilization of cross-disciplinary skills (programming, network, data processing, electronics).
- **Environmental Scope:** It raises awareness among users of the importance of weather observation and the evolution of local climatic conditions.

Table of Contents

SSH Banner Configuration :

- Edit the file: /home/USERNAME/.bashrc
- Add this at the end file:

```
# Start banner
if [ -n "$SSH_CONNECTION" ]; then
    clear
    figlet "Weather-Station by Sim & TinTin" | lolcat
    printf " \n \n"
fi
```

Requirements:

```
sudo apt install -y figlet lolcat
```

2. 💻 Raspberry Pi 5 Installation Steps

2.1: Main configuration

[Raspberry OS Installation]

2.1.1: IP Address Location: ifconfig

2.1.2: Major Updates:

```
sudo apt update && sudo apt full-upgrade
```

2.2: Mosquitto installation and Mosquitto-Clients

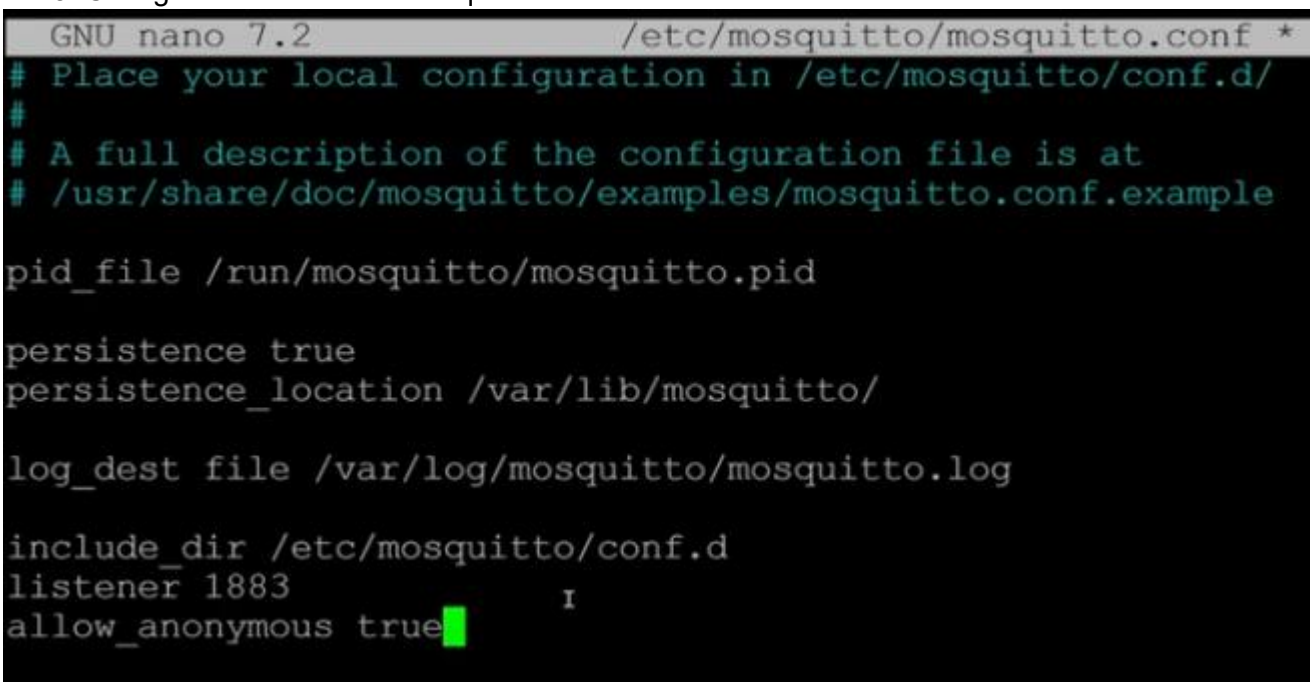
2.2.1: Mosquitto installation:

```
sudo apt install -y mosquitto mosquitto-clients
```

2.2.2: Enabling systemctl:

```
sudo systemctl enable mosquitto.service
```

2.2.3: Configuration of the file: mosquitto.conf:



2.2.4: Restart Mosquitto:

```
sudo systemctl restart mosquitto
```

2.3: Node-Red Installation

2.3.1: Node-Red installation:https://nodered.org/docs/getting-started/raspberrypi

```
bash <(curl -sL https://raw.githubusercontent.com/node-red/linux-installers/master/deb/update-nodejs-and-nodered)
```

2.3.2: Start Node-Red:

```
node-red-pi --max-old-space-size=256
```

2.3.3: Config systemctl:

```
sudo systemctl enable nodered.service
```

2.4: Influx-DB Installation

2.4.1: Installing InfluxData repository:

```
sim@sim:~ $ curl https://repos.influxdata.com/influxdata-archive.key | gpg --d
earmor | sudo tee /usr/share/keyrings/influxdb-archive-keyring.gpg >/dev/null
```

2.4.2: Installation Signature:

```
sim@sim:~ $ echo "deb [signed-by=/usr/share/keyrings/influxdb-archive-keyring.
gpg] https://repos.influxdata.com/debian stable main" | sudo tee /etc/apt/sour
ces.list.d/influxdb.list
```

2.4.3: UPDA

```
sudo apt update && sudo apt full-upgrade
```

2.4.4: InfluxDB-2 installation:

```
sudo apt install influxdb2
```

2.4.5: Unmask & systemctl:

```
sudo systemctl unmask influxdb && sudo systemctl enable influxdb
sudo systemctl start influxdb
```

2.5: Grafana Installation

2.5.1: Key installation:

```
sim@sim:~ $ curl https://apt.grafana.com/gpg.key | gpg --dearmor | sudo tee /usr
/share/keyrings/grafana-archive-keyrings.gpg >/dev/null
```

2.5.2: Signature:

```
sim@sim:~ $ echo "deb [signed-by=/usr/share/keyrings/grafana-archive-keyrings.
gpg] https://apt.grafana.com stable main" | sudo tee /etc/apt/sources.list.d/g
rafana.list
```

2.5.3: UPDAT

```
sudo apt update && sudo apt full-upgrade
```

2.5.4: Grafana Installation:

```
sudo apt install grafana
```

2.5.5: Systemctl:

```
sudo systemctl enable grafana-server
sudo systemctl start grafana-server
```

Default ID: admin / admin.

https://www.youtube.com/watch?v=0mluq0oJk7o

3. Initial Configuration: Local test via Wi-Fi (ESP32 -> Raspberry: Mosquitto -> Node Red)

3.1: Python & venv installation:

```
sudo apt install python3 python3-pip python3-venv
```

3.1.1: Creation of virtual environment:

```
mkdir ~/dht11
e directory '/home/pi/dht11': File exists
cd ~/dht11
t11 $ python3 -m venv env
t11 $ source env/bin/activate
```

https://gist.github.com/Hariharnath-Paduchuru/2785d23ca6303a13b2eadae3ea0a923d

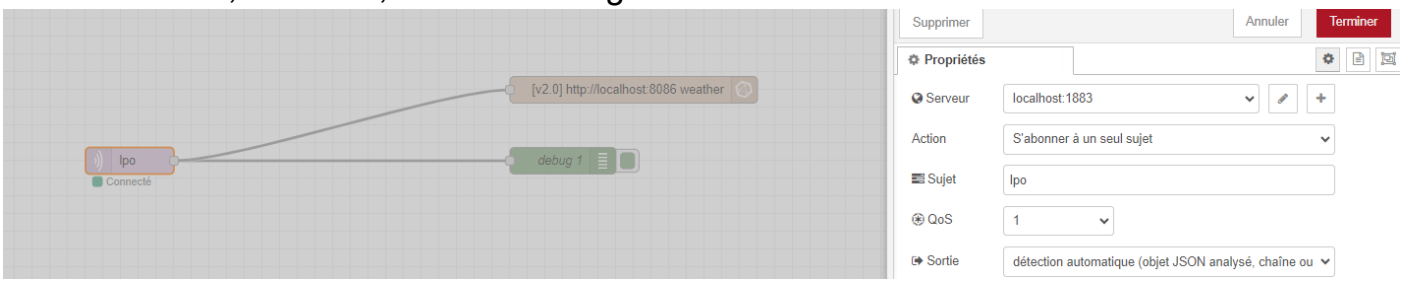
3.2: Library installation: MQTT – **DHT 11**: `pip3 install paho-mqtt`

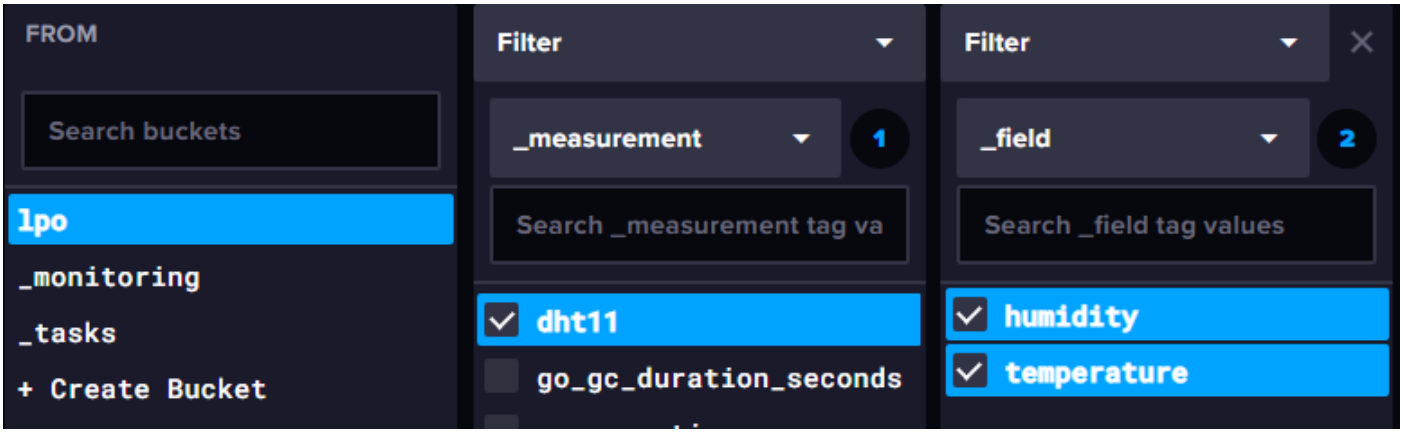
3.3: Client -> server link scripts (Python + Ino) – **All sensors test**:

Customer : ESP32 Code/ Libraries: ESP8266 – ArduinoJson – ESP8266WIFI

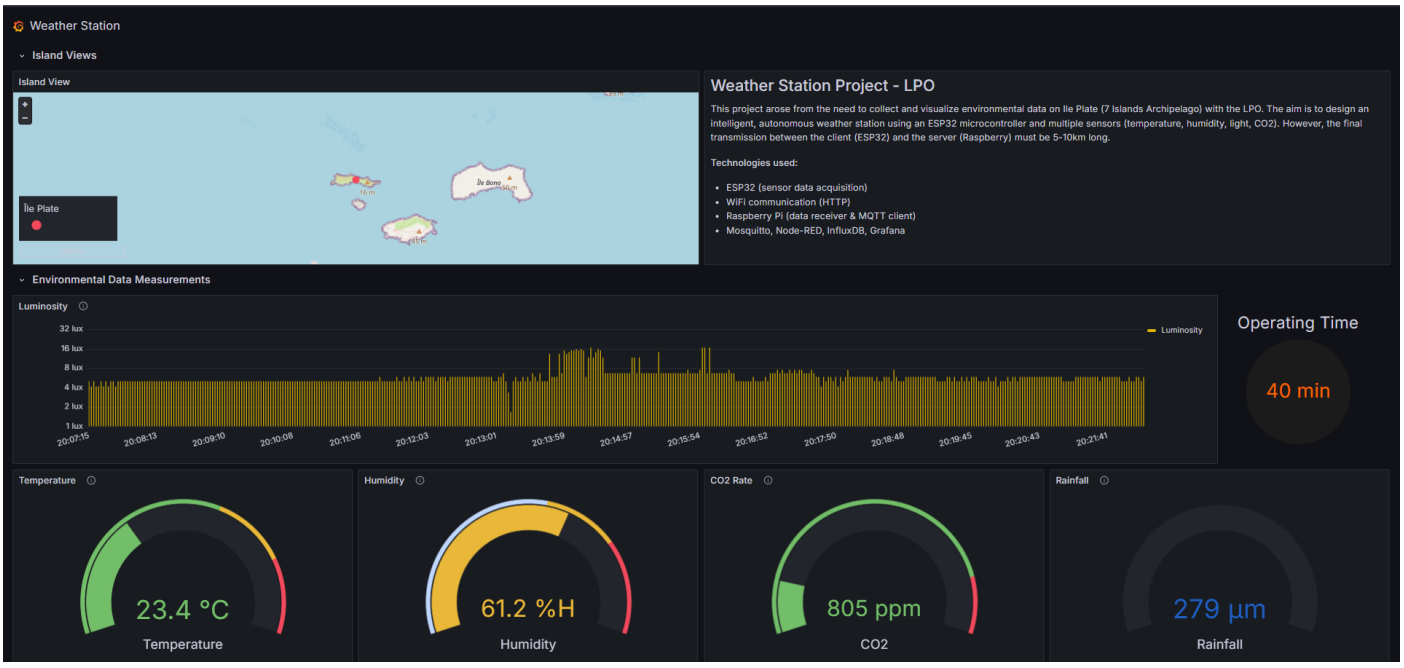
Server : Raspberry Code/ Libraries: paho.mqtt.client - Flask

3.4: Node-Red, InfluxDB, Grafana configuration:





DB Management: Flow.



4. 📽 Add Camera and Sound system (ESP32-CAM)

4.1 Familiarization with ESP32-CAM documentation

Attached files: libraries/ESP32-CAM/Tutorial(Read Me First).pdf and Tuto-ESP32Cam-PriseenMain

Installation Driver pl2303_Serial attached: libraries/ESP32-CAM/"pl2303_Serial to USB_Driver for Windows"/ PL2303_Prolific_DriverInstaller_v110.exe

4.2 Connection, Setup and Scripting

Set up a port opening on your box to ensure access to the station from outside the network: open port 81 (esp32cam), 3000 (grafana), 443 (https port), SSH_PORT (if needed from outside) via TCP protocol + set these devices to static ip (RaspberryPi and ESP32CAM) via the box interface.

Arduino IDE script:ESP32-CAM Github Script

Retrieving the video stream and retransmitting it locally on the network port 81.

4.3 Creating a reverse proxy

HTTPS SSL certificate protection via Letsencrypt (proxy server, grafana, domain).

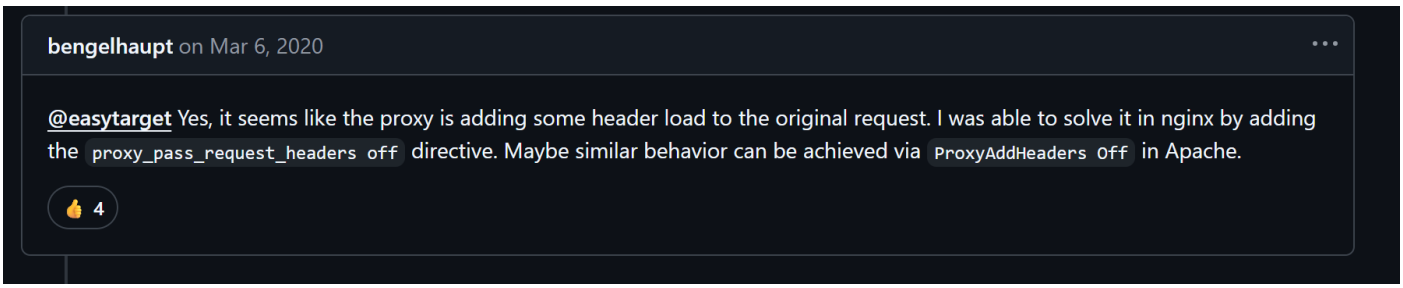
In order to display the video stream on Grafana (protected with SSL certificate in HTTPS), it is necessary to secure the display of the stream externally (http -> https). To do this, you must create a reverse proxy via NGINX to display the video stream from the local IP to the linked domain while being secure.

See Script and configuration in testCam.

After creating and launching the reverse proxy, you must verify that it is displayed securely on the link: https://<domain_name>/stream.

Possible Error:"431 Request Header Fields Too Large"

Solution :



Add the line "proxy_pass_request_headers off" in the nginx config.

4.4 Incorporation into Grafana

Create a new visualization "Text" and in the HTML ta b: "<iframe src='https://<ip_server>/stream' width='..' height='..'></iframe>"

4.5 Sound System

For the sound system, we didn't have time to set it up - now it's your job!

5. Final Panel Display