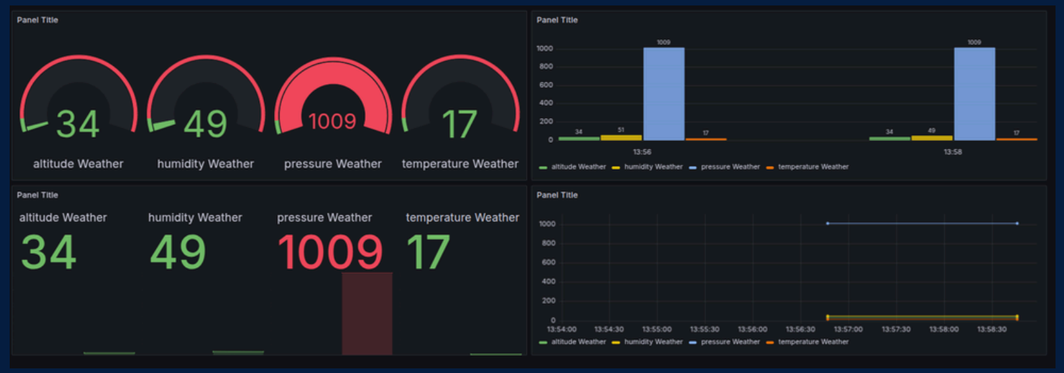
# PRÉSENTATION PROJET DISCIPLINAIRE :

## TABLEAU DE BORD MÉTÉO EN TEMPS RÉEL




*Présenter par :*

*Fraj Jasser*

*Ben Sassi Ahmed Khalil*

# TABLE OF CONTENTS

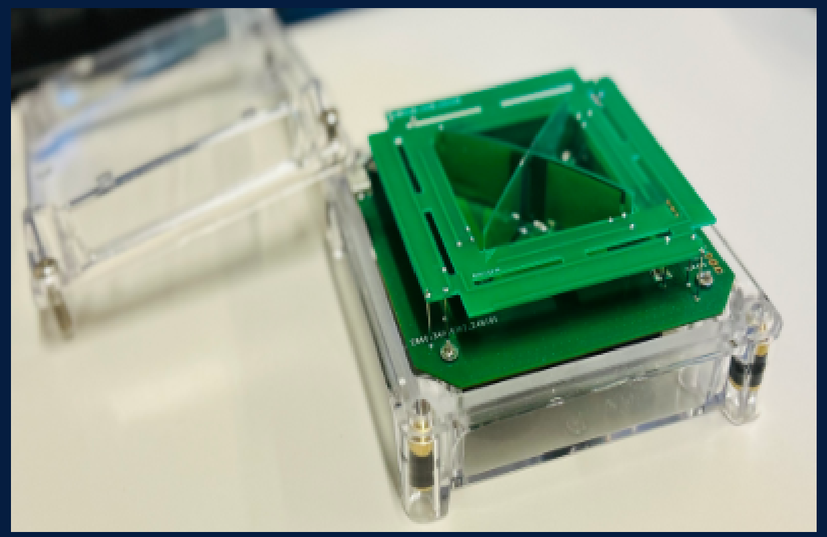
# Chapter I : Introduction

- Surveillance météorologique autonome par satellite
- Collecte des données (température, humidité, pression, altitude)
- Visualisation des tendances climatiques en temps réel
- Envoie un mail de rain scoring chaque 24h

# Chapter II :
# Architecture Matérielle
# &
# Logicielle

## Matériel

- STM32U585CIT6 : microcontrôleur
- EchoStar EM2050 : transmission satellite
- BME280 : capteur météo

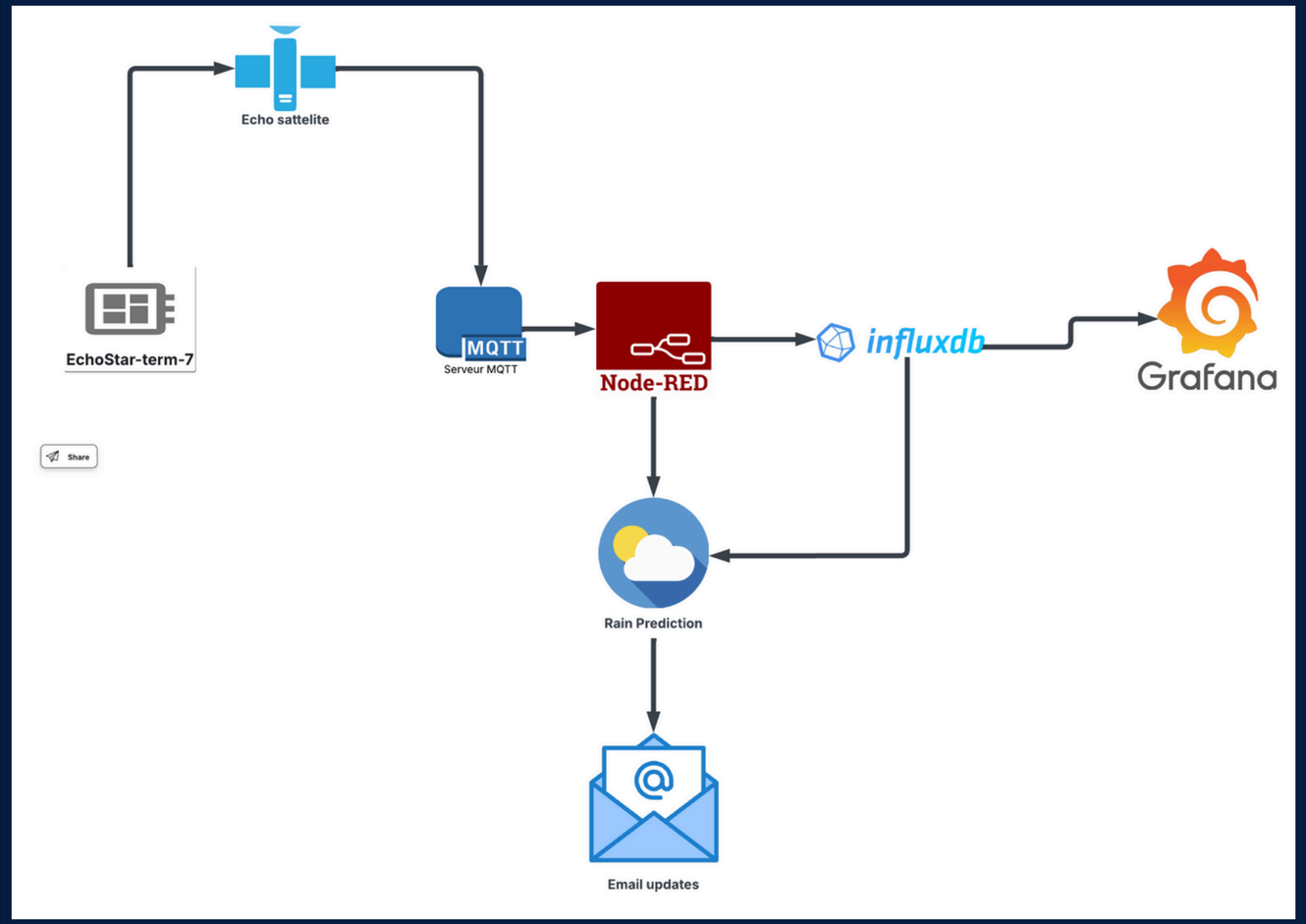


## Logiciel

- Arduino (Firmware STM32) : gestion de la lecture et de la transmission des données
- Node-RED : gestion et conversion des données
- InfluxDB : base de données pour stockage historique
- Grafana : affichage et analyse en temps réel

# Chapter III :
# Architecture du Système

## Chapter III :
## Architecture du Système




- Capteur BME280 : mesure des paramètres environnementaux
- Transmission via EchoStar-Term-7
- Serveur MQTT pour la réception des données
- Node-RED pour le traitement des données
- InfluxDB pour le stockage
- Grafana pour la visualisation
- Rain scoring pour le calcul et l'envoie d'un mail chaque 24h

# Chapter IV :
# Explication du Code

## Initialisation et Connexion au Satellite

```
SATELLITE_SERIAL.begin(115200);
delay(1000);
digitalWrite(LED_BUILTIN, LOW);
SATELLITE_SERIAL.println("AT+JOIN");
Serial.println("Connexion au satellite...");
```

## Lecture des Données du Capteur

```
void read_sensor_data() {
    temp = sensor.readTemperature();
    press = sensor.readPressure() / 100.0F;
    alt = sensor.readAltitude(SEA_LEVEL_PRESSURE);
    hum = sensor.readHumidity();
    delay(50);
}
```

## Construction et Envoi des Données au Satellite

```
void send_data() {
    read_sensor_data();
    char command[50];
    sprintf(command, "AT+SEND=1,0,8,1,%d,%d,%d,%d\r\n", temp, press, alt, hum);
    SATELLITE_SERIAL.print(command);
    Serial.println("Données envoyées au satellite.");
}
```

## Vérification de la Connexion et de l'Acquittement

```
if (SATELLITE_SERIAL.available()) {
    String response = SATELLITE_SERIAL.readStringUntil('\n');
    Serial.print("Réponse du satellite: ");
    Serial.println(response);

    if (!satellite_connected && response.indexOf("Successfully joined network") != -1) {
        satellite_connected = true;
        Serial.println("Connexion au satellite réussie");
    }

    if (response.indexOf("QUEUED:1") != -1) {
        Serial.println("Données confirmées par le satellite.");
        value_confirmed = true;
    }
}
```

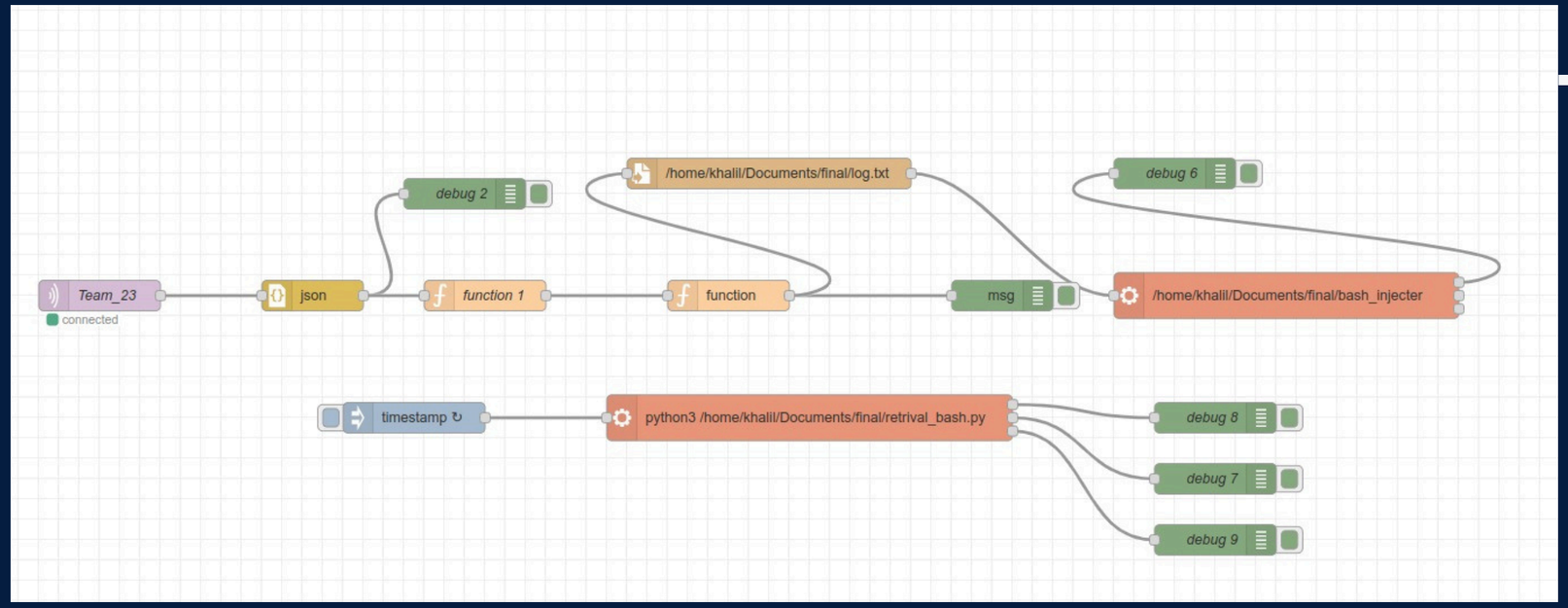
## Envoie et Ré-essai d'Envoi en Cas d'Échec

```
if (satellite_connected) {
    unsigned long current_time = millis();

    if (value_confirmed && current_time - last_send_time >= 120000) {
        value_confirmed = false;
        send_data();
        last_send_time = current_time;
    } else if (!value_confirmed && current_time - last_send_time >= 20000) {
        send_data();
        last_send_time = current_time;
    }
}
```

# Résultats sur le Serial Monitor

```
16:43:30.511 -> Attempting to join satellite network...
16:43:32.793 -> Satellite response: INFO: Echostar Mobile OEM LoRa Module FW1.7 started, cause: [Power ON][Low Voltage][External Reset] @ 01/01/1970 00:00
16:43:32.826 -> Satellite response: INFO: LoRaWAN stack started with region MSS-S, joining network. @ 01/01/1970 00:00:03
16:43:32.826 -> Satellite response: INFO: Join procedure started, retry n. 1 @ 01/01/1970 00:00:03
16:43:40.982 -> Satellite response:
16:43:40.982 -> Satellite response: Successfully joined network
16:43:40.982 -> Satellite network joined successfully.
16:43:40.982 -> Satellite response: OK
16:43:40.982 -> Satellite response: INFO: Successfully joined network @ 01/01/1970 00:00:11
16:43:48.241 -> Sensor data resent to satellite.
16:43:48.241 -> Satellite response: AT+SEND=1,0,8,1,16,1021,-68,77
16:43:48.241 -> Satellite response:
16:43:48.241 -> Satellite response: QUEUED:1
16:43:48.241 -> Value acknowledged by satellite.
16:43:48.241 -> Satellite response: OK
```

# Chapter IV : Node-RED




```bash
#!/bin/bash

data=$(grep -o '\(-\?[0-9]\+\),\(-\?[0-9]\+\),\(-\?[0-9]\+\),\(-\?[0-9]\+\)' ~/Documents/final/log.txt)

IFS=',' read -r temperature pressure altitude humidity <<< "$data"

curl -X POST "http://localhost:8086/api/v2/write?bucket=khalil&org=khalil" \
  -H "Authorization: Token Ms0Kij_URidvjZKgrYkLQHplYNAj1Ty70g7s7L7hXx1uFZusC1p-oaYff6w3HQ2XEbIrE6s0L5fIMx0X7jgvqA==" \
  -H "Content-Type: text/plain; charset=utf-8" \
  --data-raw "SmartWeather,location=Weather temperature=$temperature,pressure=$pressure,altitude=$altitude,humidity=$humidity"
```

```javascript
if (msg.payload.fPort === 8 && msg.payload.frmPayload)
    return msg;
}
```

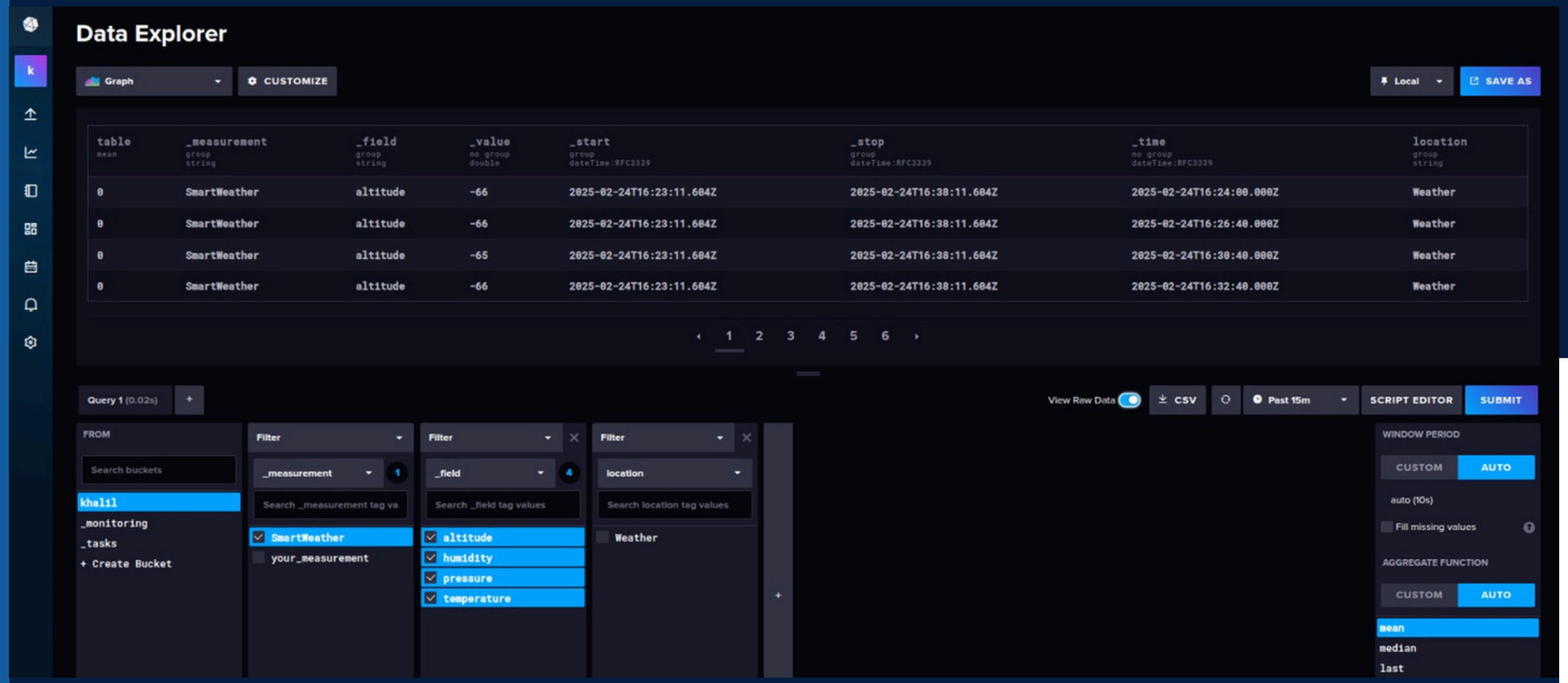
```javascript
var frmPayload = msg.payload.frmPayload;

var decodedPayload = Buffer.from(frmPayload, 'base64').

return {
    payload: {
        frmPayload: decodedPayload
    }
};
```

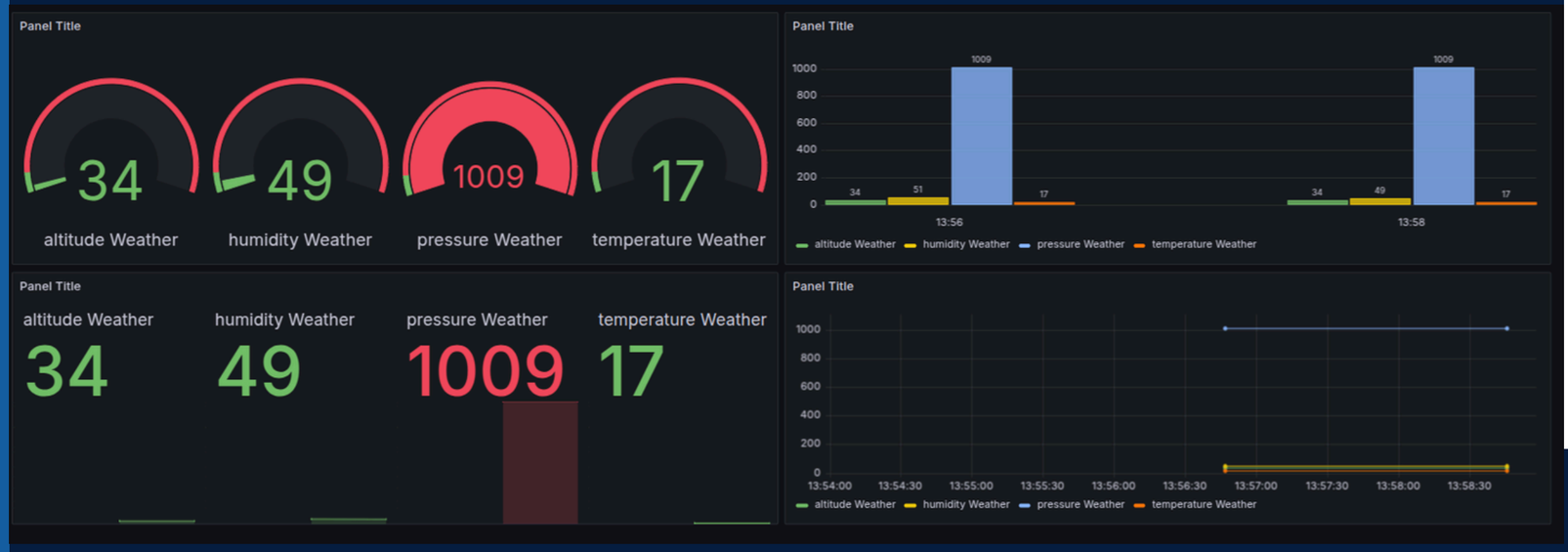
# Résultas:

## debug2

# Chapter V : InfluxDB & Grafana & Rain Scoring

# Chapter V : InfluxDB & Grafana & Rain Scoring




## Rain Prediction: Low chance of rain. ☀️

Here's your daily weather update:

### Last Raw Value in Dataset:

- **Temperature:** 15.0°C
- **Humidity:** 81.0%
- **Pressure:** 1021.0 hPa
- **Altitude:** -65.0 m

### Rain Score Calculation:

- **Humidity Contribution:** 0.4 * 79.67 = 31.87
- **Pressure Change Contribution:** 0.3 * -37.21 = -11.16
- **Temperature Contribution:** 0.1 * (30 - 15.25) = 1.48
- **Humidity Change Contribution:** 0.2 * 15.67 = 3.13
- **Altitude Contribution:** 0.1 * (0.00 / 1000) = 0.00

**Total Rain Score:** 25.31

Stay prepared and have a great day!

# Chapter VI : Conclusion

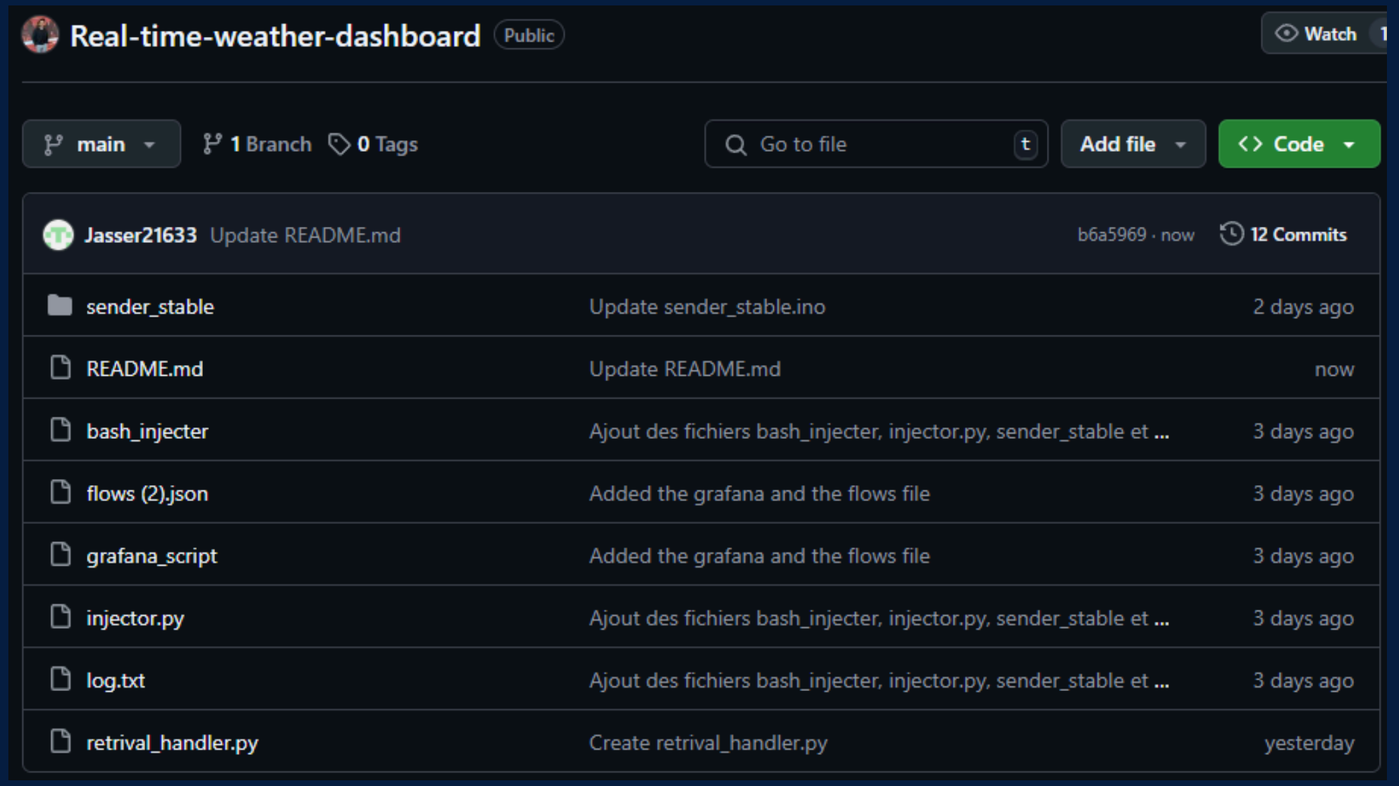
# CONCLUSION

- Le système permet une surveillance météo fiable et autonome.
- Transmission efficace via satellite, adaptée aux zones isolées.
- Optimisation du stockage et de la visualisation des données.

# PERSPECTIVES

L'objectif futur est de développer un algorithme de machine learning capable de prédire avec plus de précision la probabilité de pluie en analysant les motifs récurrents. De plus, une application pour smartwatch sera conçue afin d'afficher ces prévisions directement sur l’appareil, offrant ainsi un accès rapide et pratique aux données météorologiques en temps réel.

# Github :




https://github.com/ahmed-Khalil404/Real-time-weather-dashboard

# MERCI POUR VOTE ATTENTION !