

# Conception d'une station météo connectée et intelligente avec ESP32 et MQTT

June 1, 2023

**Antonin Lefevre**

antoninlefevre45@icloud.com

## Introduction

Cette article propose de détailler la conception et la réalisation d'une station météorologique connectée, économique et intelligente, destinée à la collecte et à la transmission de données à un serveur local. Pour ce faire, l'architecture du système repose sur l'utilisation d'une multitude de capteurs environnementaux, de la plateforme ESP32 pour le traitement et la transmission des données, ainsi que du protocole MQTT pour la coordination entre notre station météorologique et le serveur hôte. Par ailleurs, est prévu la création d'une interface utilisateur qui non seulement affiche ces données en temps réel, mais les conserve également pour une analyse ultérieure. Enfin, afin d'optimiser l'utilité et la précision de notre système, un modèle d'apprentissage automatique a été intégré pour permettre la prévision des conditions météorologiques à court terme à partir des données recueillies.

**Le code de ce projet est disponible sur [Github](#).**

## Prérequis

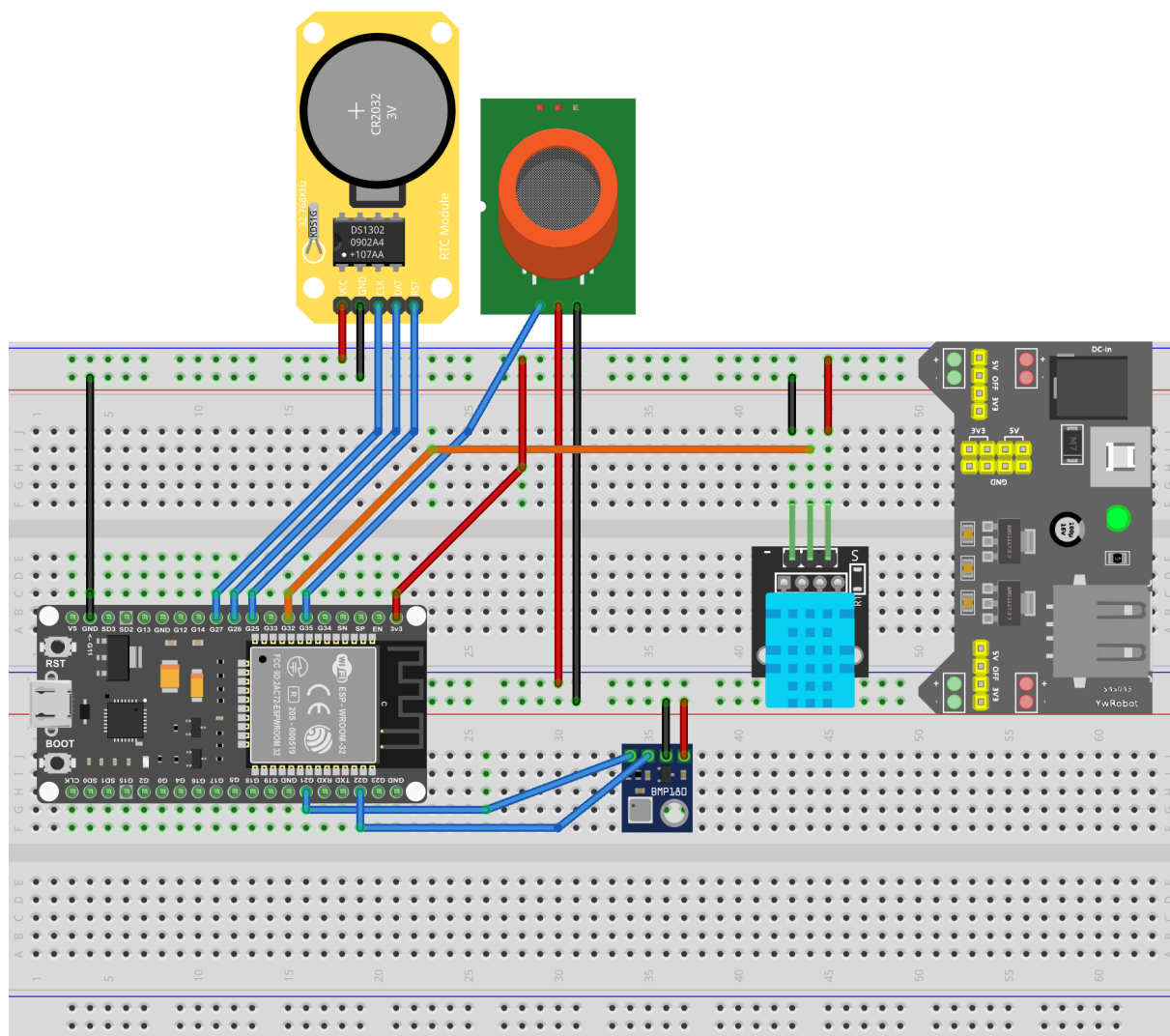
Avant de commencer, voici le matériel et les logiciels requis pour ce projet. Au niveau du matériel, notre montage nécessitera l'utilisation d'un **ESP32**, qui est un ordinateur à microcontrôleur chargé de la lecture des données des capteurs et de leur transmission à notre serveur. Nous utiliserons également divers capteurs, dont le capteur de qualité de l'air **Grove** pour mesurer la qualité de l'air, un capteur **BMP180** pour évaluer la température et la pression atmosphérique, ainsi qu'un capteur **HTU21** destiné à mesurer l'humidité. Pour horodater les données nous utiliserons un module **RTC** avec puce DS1302. Enfin, nous aurons besoin d'un ordinateur, dans notre cas un Mac, qui servira à exécuter notre serveur MQTT et notre interface utilisateur.

Du côté des logiciels, l'IDE Arduino sera notre outil de choix pour programmer l'ESP32. Python, pour sa part, sera employé pour implémenter notre serveur MQTT, notre interface utilisateur et notre modèle d'apprentissage automatique. Enfin, pour le broker MQTT, nous utiliserons Mosquitto, qui est une solution open source.

Pour vous procurer les capteurs ou l'ESP32 vous pouvez regarder sur le site [AZ-Delivery](#)

## Configuration du matériel

Sur la Figure 1, nous avons illustré l'agencement précis de notre circuit. Chaque connexion est clairement identifiée, vous permettant de visualiser comment les différents capteurs sont connectés à l'ESP32.



fritzing

Figure 1: Schéma du câblage.

Un point crucial à ne pas négliger dans la réalisation de ce projet concerne la tension de fonctionnement des composants utilisés. En effet, l'ESP32 ainsi que les capteurs environnementaux employés ici sont conçus pour fonctionner sous une tension de 3,3 volts. Il est impératif de respecter cette exigence pour préserver l'intégrité de ces composants. Ainsi, il faudra faire attention à comment sont installées les broches au niveau de l'adaptateur en haut à droite sur le schéma pour délivrer 3.3V.

La prise en compte de ce paramètre technique est une part fondamentale de la conception de tout système électronique. Elle garantit la durabilité du système, sa fiabilité et la validité des données recueillies.

La Figure 2 est une photo de notre station météo une fois entièrement assemblée. Comme vous pouvez le constater, toutes les connexions correspondent au schéma de câblage fourni précédemment. Vous pouvez observer comment chaque capteur est connecté à l'ESP32, formant ainsi un système cohérent et compact.

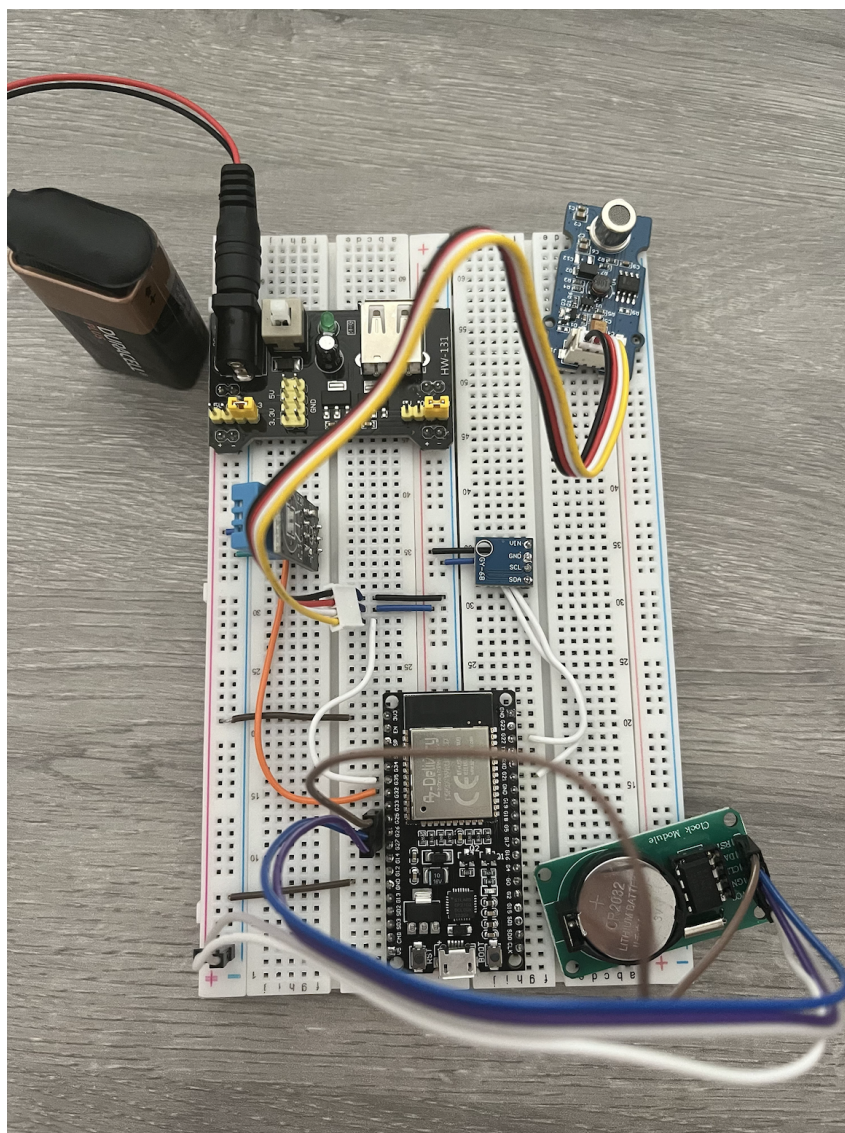


Figure 2: Photo du montage.

La réalisation d'un système tel que notre station météorologique ne serait pas complète sans une attention particulière portée à son intégration physique. En effet, le montage de tous les composants doit être soigneusement effectué afin de garantir non seulement le bon fonctionnement de l'ensemble, mais également sa durabilité et son esthétique.

Dans cette optique, nous avons prévu d'intégrer l'ensemble du montage dans un boîtier personnalisé, conçu spécifiquement pour notre station météorologique. Ce boîtier sera fabriqué grâce à la technologie d'impression 3D. La conception du boîtier a été réalisée avec Autodesk Fusion 360, un outil de modélisation 3D complet et puissant.

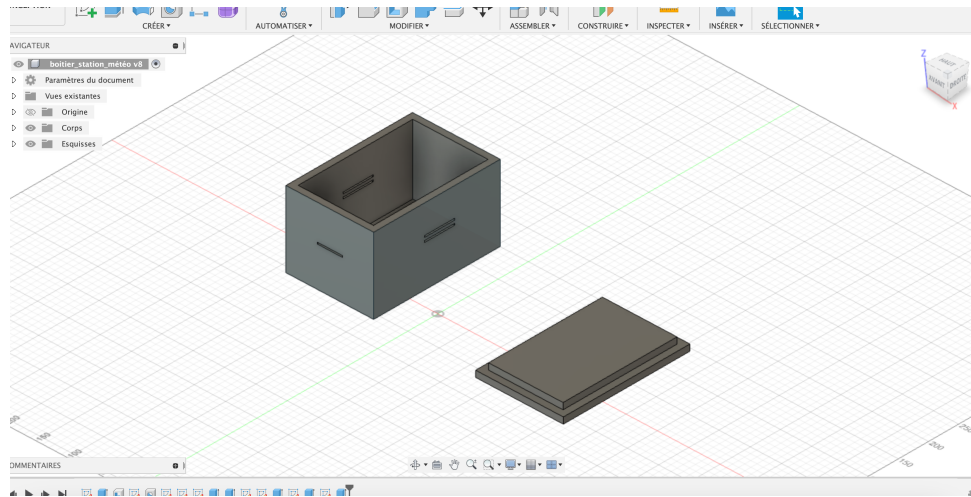


Figure 3: Conception du boîtier pour la station météo sous Fusion 360.

## MQTT

MQTT, pour Message Queuing Telemetry Transport, est un protocole de messagerie léger et efficace qui a été conçu pour les systèmes de communication Machine-à-Machine (M2M) et Internet des Objets (IoT). L'un des atouts de MQTT est son modèle de publication/abonnement qui permet aux messages de parvenir à plusieurs destinataires en même temps, ce qui est idéal pour la surveillance en temps réel des données de capteurs.

Le protocole MQTT fonctionne via un système de « broker » (serveur) et de « clients ». Les clients peuvent être des éditeurs (publishers), qui envoient des messages sur des sujets (topics) spécifiques, et/ou des abonnés (subscribers), qui reçoivent des messages sur les sujets auxquels ils sont abonnés.

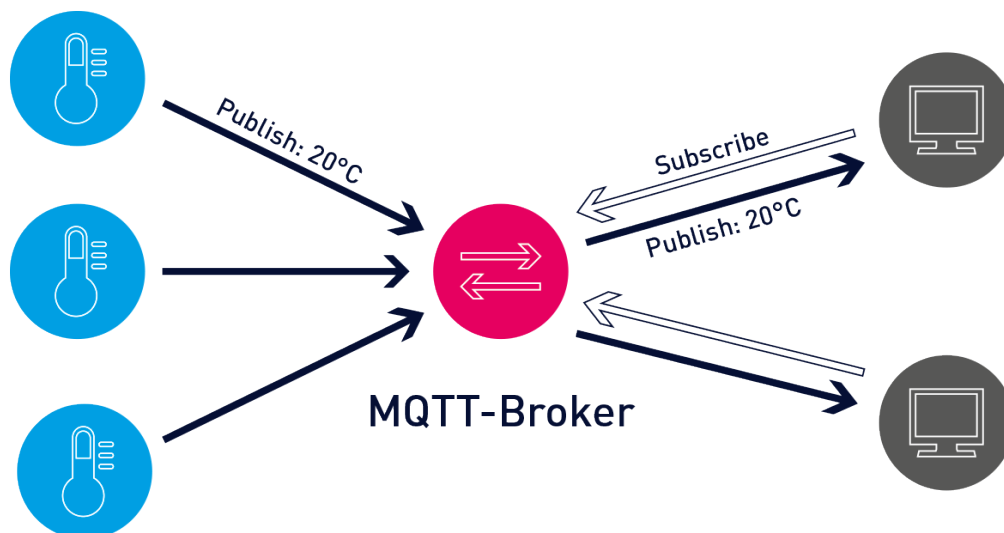


Figure 4: Architecture MQTT.

### Créer un serveur MQTT

Pour ce projet, nous utiliserons Mosquitto [4], un broker MQTT open-source et facile à utiliser. Une fois Mosquitto installé, vous pouvez démarrer le serveur MQTT.

## S'abonner à un topic MQTT

Pour s'abonner à un topic MQTT, nous utiliserons la bibliothèque `paho-mqtt` de Python. Pour installer `paho-mqtt` via le gestionnaire de package python PIP:

```
pip install paho-mqtt
```

Voici la portion de script Python qui se connecte à notre broker MQTT et s'abonne au topic « weather station »:

```
import paho.mqtt.client as mqtt

ssid = config['ssid']
password = config['password']
mqtt_server = config['mqtt_server_ip']

def on_connect(client, userdata, flags, rc):
    print("Connecté avec le code résultat " + str(rc))
    client.subscribe("weather station")

def on_message(client, userdata, msg):
    data = json.loads(msg.payload.decode('utf-8'))
    [...]
    conn.commit()

client = mqtt.Client()
client.on_connect = on_connect
client.on_message = on_message

client.connect(host=mqtt_server, port=1883, keepalive=60)

client.loop_forever()
```

## Publier sur un topic MQTT avec l'ESP32

Sur l'ESP32, nous utiliserons la bibliothèque `PubSubClient` pour publier sur un topic MQTT. Voici un exemple de code qui envoie un message sur le topic « weather station » :

```
#include <WiFi.h>
#include <PubSubClient.h>

const char* ssid = "your_SSID";
const char* password = "your_PASSWORD";
const char* mqtt_server = "your_MQTT_SERVER_IP_ADDRESS";

WiFiClient espClient;
PubSubClient client(espClient);
```

```

void setup() {
  Serial.begin(115200);
  setup_wifi();
  client.setServer(mqtt_server, 1883);
}

void setup_wifi() {
  WiFi.begin(ssid, password);
  while (WiFi.status() != WL_CONNECTED) { delay(500); }
  Serial.println("WiFi connected");
}

void loop() {
  if (!client.connected()) {
    reconnect();
  }
  client.loop();

  String payload = "Hello MQTT!";
  client.publish("weather station", payload.c_str());
  delay(10000); // Publish every 10 seconds
}

void reconnect() {
  // Loop until we're reconnected
  while (!client.connected()) {
    if (client.connect("ESP32Client")) {
      client.publish("weather station", "ESP32 connected");
    } else {
      delay(5000);
    }
  }
}
}

```

Ainsi, à chaque fois que vous exécutez ce script sur l'ESP32, il se connectera à votre réseau WiFi, puis se connectera à votre serveur MQTT et publiera un message sur le topic « weather station » toutes les 10 secondes.



## Récupération et stockage des données

Le cœur de notre système de station météo réside dans la capacité à récupérer et à stocker efficacement les données générées par divers capteurs environnementaux. Ces informations cruciales sont captées par notre microcontrôleur ESP32 et envoyées à notre serveur via le protocole MQTT. Pour gérer ces flux de données, nous avons écrit un script en Python qui s'abonne à notre sujet MQTT et récupère en continu les données météorologiques transmises.

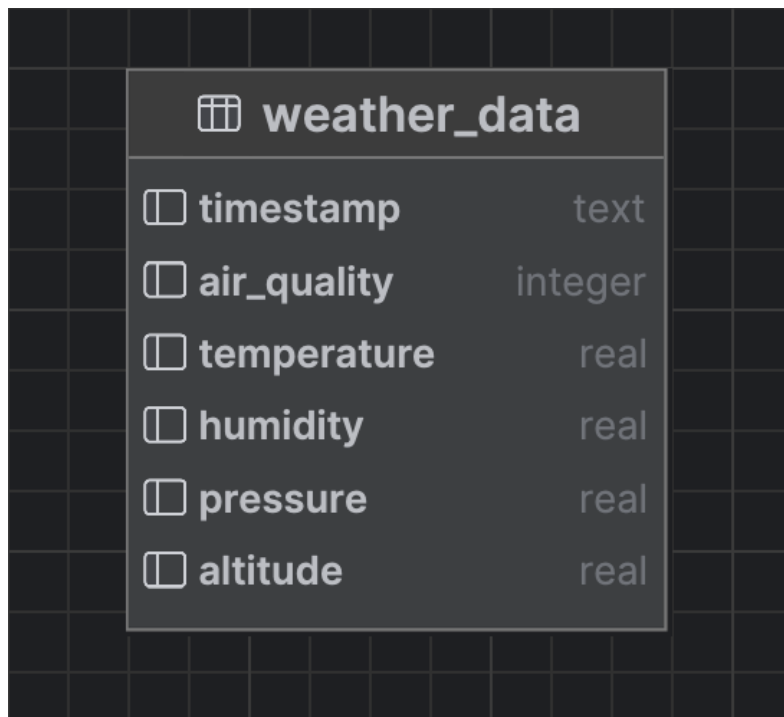
Pour garantir la pérennité et l'accès à ces données, notre script stocke chaque nouveau lot de données dans une base de données SQLite locale. Cette base de données, stocke chaque relevé météorologique avec son horodatage correspondant, permettant ainsi une analyse historique détaillée. L'utilisation de SQLite offre une solution de stockage robuste, tout en conservant la flexibilité nécessaire pour travailler avec notre base de données directement depuis Python.

L'ajout à la base se fait lorsque les données sont reçues depuis le broker MQTT de la manière suivant:

```
def on_message(client, userdata, msg):
    data = json.loads(msg.payload.decode('utf-8'))

    c.execute("INSERT INTO weather_data VALUES (?, ?, ?, ?, ?, ?)",
              (data['time'], data['air_quality'], data['temperature'],
               data['humidity'], data['pressure'], data['altitude'])
              )
    conn.commit()
```

Voici le détail de la table weather\_data:



The image shows a screenshot of a database table definition for 'weather\_data'. The table has six columns: 'timestamp' (text), 'air\_quality' (integer), 'temperature' (real), 'humidity' (real), 'pressure' (real), and 'altitude' (real). Each column name is preceded by a small square icon.

weather_data	
timestamp	text
air_quality	integer
temperature	real
humidity	real
pressure	real
altitude	real

Figure 5: Table weather\_data.

## Création d'un tableau de bord

Dans cette section de l'article, nous nous intéressons à la création d'un tableau de bord pour notre station météo. C'est un élément clé de notre projet qui permet de visualiser les informations météorologiques en temps réel et d'examiner l'historique des données.

Pour ce faire, nous utilisons Streamlit, une bibliothèque Python qui permet de créer des applications web rapidement. Nous avons mis en place une interface utilisateur intuitive, dotée d'une barre latérale personnalisée qui affiche les informations météorologiques actuelles. Cette barre latérale offre un aperçu immédiat des conditions météorologiques, notamment la température, la pression atmosphérique, l'humidité et la qualité de l'air.

Au-delà de cette vue instantanée, notre tableau de bord offre une analyse plus détaillée grâce à une série de widgets interactifs. Pour la température, nous avons deux graphiques: l'un montrant l'évolution de la température au cours des dernières 24 heures, l'autre montrant la tendance sur les trois derniers jours. Cela nous permet de percevoir rapidement les variations de température et d'identifier les tendances à plus long terme.

En ce qui concerne la pression atmosphérique et l'humidité, nous avons adopté une approche similaire. Chaque mesure a son propre widget de jauge qui indique le niveau actuel, ainsi qu'un graphique détaillé qui trace les valeurs sur les trois derniers jours. La qualité de l'air est également suivie de près grâce à un graphique dédié qui montre les fluctuations de cette mesure au fil du temps.

Pour assurer l'actualité de ces informations, notre tableau de bord est conçu pour se mettre à jour toutes les 10 minutes, garantissant ainsi que vous disposez toujours des informations les plus récentes. Grâce à cette approche, notre tableau de bord offre une vue complète et actualisée des conditions météorologiques locales, facilitant la compréhension et l'interprétation des données de notre station météo.

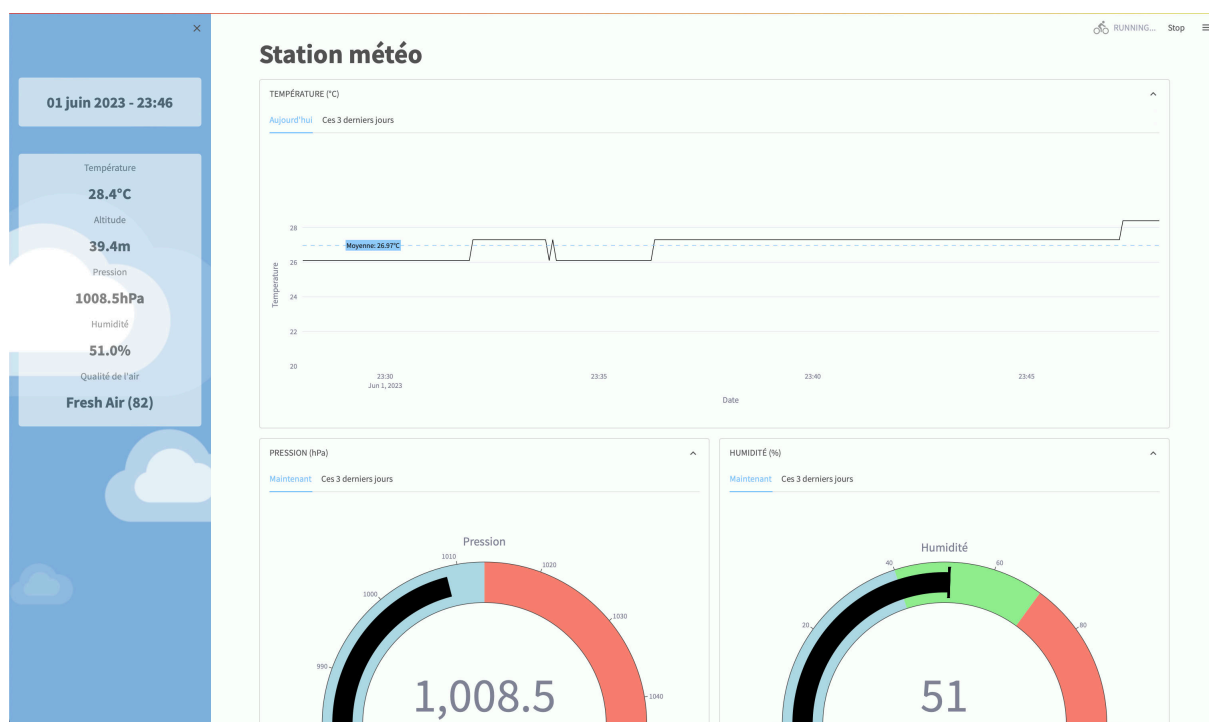


Figure 6: Dashboard.



## Création du modèle pour la prévision météo

Dans le cadre de ce projet, une approche de régression polynomiale a été spécifiquement adoptée pour la prévision de la température. Cette méthode a été privilégiée pour plusieurs raisons. Premièrement, étant donné que seules les données des trois derniers jours sont retenues, la quantité de données à traiter reste gérable et ne nécessite pas de techniques de modélisation plus complexes ou lourdes en termes de calcul. Deuxièmement, il est essentiel de maintenir un modèle à la fois léger et rapide à entraîner, afin de garantir une efficacité optimale et une réactivité en temps réel. La régression polynomiale répond parfaitement à ces critères.

La valeur de la température prédite est ensuite intégrée de manière visuelle au tableau de bord par une courbe en pointillés.

## Conclusion

Dans cet article, nous avons détaillé la mise en œuvre d'une station météorologique connectée, conçue pour collecter, analyser et afficher en temps réel les données météorologiques. Le système est structuré autour d'un ensemble de capteurs environnementaux, d'une plateforme de traitement et de transmission des données basée sur l'ESP32, et du protocole MQTT pour la synchronisation de la station avec le serveur hôte.

L'élément central de ce système est l'exploitation efficace des données recueillies. Pour cela, nous avons déployé un script en Python permettant de récupérer et de stocker ces données dans une base de données SQLite locale. De cette façon, nous pouvons assurer un accès continu et la conservation des données historiques pour des analyses futures.

Au-delà de la simple collecte de données, ce système comprend également un tableau de bord visuellement attrayant, capable de représenter les données météorologiques en temps réel et d'examiner l'historique des données. Grâce à l'utilisation de Streamlit, nous avons pu réaliser une interface utilisateur intuitive qui fournit un aperçu immédiat des conditions météorologiques actuelles, ainsi qu'une représentation détaillée des tendances sur une période plus longue.

Pour maximiser l'utilité de la station météorologique, nous avons intégré un modèle d'apprentissage automatique pour prédire les conditions météorologiques à court terme. En utilisant une régression polynomiale, nous avons créé un système à la fois efficace et réactif, capable de fournir des prévisions de température basées sur les données des trois derniers jours.

En résumé, grâce à une combinaison judicieuse de matériel, de logiciels et de techniques d'analyse de données, nous avons réussi à créer une station météorologique connectée efficace et économique. Ce système fournit non seulement des données météorologiques en temps réel, mais offre également des prévisions à court terme et une analyse historique détaillée, ce qui en fait un outil précieux pour la surveillance et la compréhension des conditions météorologiques locales.

## Ressources

[1] [Ajouter la prise en charge de la carte ESP32 sur l'IDE Arduino](#)

[2] [Plan des pins GPIO de l'ESP32](#)

[3] [Logiciel de création des schémas électriques : Fritzing](#)

[4] [Documentation du serveur MQTT Mosquitto](#)