



الجمهورية الجزائرية الديمقراطية الشعبية
République Algérienne Démocratique et Populaire
وزارة التعليم العالي والبحث العلمي
Ministère de l'Enseignement Supérieur et de la Recherche Scientifique
المدرسة الوطنية للتكنولوجيات المتقدمة
Ecole Nationale Supérieure des Technologies Avancées



Rapport de mini-projet

Station Météo Intelligente



fait par :

- OTMANI Mohamed Amine
- LERHLERH Soundous
- LADJOUZI Adam
- DJABALLAH Akram

Le 09/01/2025

Table des matières

Table des matières	2
Introduction	4
Objectifs du projet.....	4
Fonctionnement Global du Système	5
Introduction.....	6
Objectifs du projet	7
Fonctionnement Global du Système	8
1. Partie Électronique	9
1.1. Composants Utilisés.....	9
1.1.1 Microcontrôleur	9
1.1.2 Alimentation.....	9
1.1.3 Capteurs	10
1.2. Conception et Réalisation	11
1.2.1. Fonctionnement des Capteurs	11
❖ DHT11 : Température et Humidité.....	11
❖ LDR : Intensité Lumineuse.....	11
❖ Capteur de Pluie.....	11
1.2.2. Conception de l'Anémomètre personnalisée DIY	12
❖ Étude Théorique	12
❖ Conception et Réalisation	14
Conclusion de la Conception	15
1.3. Schéma Électronique	16
2. Partie IoT	18
2.1. Introduction	18
2.2. Architecture du système IoT	18
2.3 Gestion de données	19
2.3.1 Collecte des données	19
2.3.2 Transmission des données.....	19
2.3.3 Stockage des données	19
2.3.4 Utilisation des données.....	20
2.3.5 Flux de données IoT	20
4.4. Correction des données	21
2.5. Avantages de cette architecture	21
3. Partie Prédiction (IA).....	22
3.1. Préparation des Données.....	22
1. Importation des bibliothèques	22
Explication :	22
3.2. Modèle IA et Algorithmes	26
4. Partie Interface Utilisateur	35

4.1. Design du Site Web	35
4.2. Fonctionnalités Offertes	35
4.2.1. Carte Interactive.....	35
4.2.2. Graphiques et Tableaux	36
4.2.3. Prévisions IA	37
4.2.4. Responsive Design	38
4.2.5. Intégration API	39
4.3. Expérience Utilisateur.....	39
4.4. Sections Détaillées	39
4.4.1. Carte Interactive.....	39
4.4.2. Tableau de Bord.....	39
4.4.3. FAQ	39
4.4.4. Notre Équipe	40
4.5. Partie Commerciale	40
Conclusion et perspectives	41

Liste de figure :

1Figure 1 : Exemple d'une station Météo intelligente	4
2Figure2: la carte ESP32	5
3Figure 3: chéma de liaison entre la batterie et la protection	6
4figure 4 : schéma illustre les caracteristique de TP4056	6
5Figure 5 : DHT11	7
6Figure 6 : LDR	7
7figure 7 : Anemometre DIY	7
8 figure 8:capteur de pluie.....	7
9fgure 9 : capteur de pression	7
10figure10:capteur effet hall	7
11 figure 11 : code pour lire les donnees du capteur DHT11	8
12 figure 12 : illustration de principe de l'effet Hall.....	9
13 figure 13 : photo reel de projet.....	12
14 figure 14 : schema de montage des composants electronique	13
15 figure 15 : la boitier qui couvre les composant electronique	13

Introduction

Avec l'évolution rapide des technologies, l'Internet des Objets (IoT) et l'Intelligence Artificielle (IA) jouent un rôle crucial dans la transformation de divers secteurs, tels que l'agriculture, la domotique, la santé et l'industrie. Ces technologies permettent une collecte, une analyse et une utilisation intelligente des données en temps réel, favorisant ainsi des solutions plus efficaces et adaptatives.



1Figure 1 : Exemple d'une station Météo intelligente

Dans ce projet, nous nous intéressons à la création d'une station météo intelligente qui intègre l'ESP32, un microcontrôleur performant et polyvalent, en combinaison avec des capteurs IoT et des modèles d'IA. La problématique principale est de développer un système capable de collecter des données environnementales, de les traiter, et de fournir des informations ou des prévisions pertinentes aux utilisateurs de manière autonome.

Objectifs du projet

L'objectif principal de ce projet est de concevoir une station météo intelligente capable de surveiller et d'analyser en temps réel plusieurs paramètres environnementaux, en exploitant les technologies de l'IoT et de l'IA. Les objectifs spécifiques sont les suivants

- **Collecte et traitement des données :**

Développer un système robuste pour la collecte, le stockage et le traitement des données environnementales (température, humidité, luminosité, vitesse et direction du vent) à l'aide de capteurs connectés à un microcontrôleur ESP32.

- **Conception d'un anémomètre personnalisé :**

Concevoir et réaliser un anémomètre DIY (Do It Yourself) pour mesurer la vitesse et la direction du vent, en utilisant des capteurs à effet Hall et des composants accessibles.

- **Intégration de l'IA :**

Implémenter un modèle d'intelligence artificielle pour interpréter les données, détecter des tendances et fournir des prédictions météorologiques précises, notamment la température du jour suivant.

- **Transmission des données :**

Assurer la transmission des données via une infrastructure IoT, permettant un accès en temps réel à une plateforme cloud (Google Sheets) et une visualisation via une interface web interactive.

- **Flexibilité et évolutivité :**

Concevoir une solution modulaire et évolutive, adaptable à différents scénarios d'utilisation, que ce soit pour des applications domestiques, agricoles ou industrielles.

- **Interface utilisateur intuitive :**

Mettre en place une interface web conviviale pour visualiser les données en temps réel, les tendances historiques et les prévisions météorologiques, offrant ainsi une expérience utilisateur complète et accessible.

Fonctionnement Global du Système

La station météo intelligente fonctionne selon le processus suivant :

- **Collecte des données** : Les capteurs (DHT11, LDR, anémomètre DIY, capteur de pluie) mesurent en temps réel les paramètres environnementaux.
- **Transmission des données** : Les données sont envoyées via Wi-Fi à une base de données en ligne (Google Sheets) pour stockage et analyse.
- **Analyse et prédiction** : Un modèle d'IA traite les données pour fournir des prévisions météorologiques, notamment la température du jour suivant.
- **Visualisation** : Une interface web interactive permet aux utilisateurs de consulter les données historiques et les prévisions en temps réel.

1. Partie Électronique

1.1. Composants Utilisés

1.1.1 Microcontrôleur

Le microcontrôleur utilisé pour ce projet est l'ESP32. Ce composant est choisi pour ses capacités avancées, notamment l'intégration du WiFi, qui facilite le transfert des données vers des plateformes cloud comme Google Sheets et Firebase.



2Figure2: la carte ESP32

Caractéristiques principales :

- Connectivité intégrée : Wi-Fi 2.4 GHz et Bluetooth (4.2 + BLE).
- Processeur puissant : Double cœur 32 bits jusqu'à 240 MHz.
- Mémoire : jusqu'à 520 Ko de RAM et mémoire flash extensible.
- Interfaces variées : GPIO, SPI, I2C, UART, PWM, ADC, DAC.
- Faible consommation d'énergie : Optimisé pour les projets sur batterie.

Avantages :

- Polyvalence pour des applications variées (domotique, robotique, capteurs connectés).
- Facilité d'utilisation avec Arduino IDE, MicroPython et ESP-IDF.
- Coût accessible et documentation abondante.

1.1.2 Alimentation

Le système est alimenté par une batterie lithium-ion de 2500 mAh (5V) couplée à un module de charge et de protection TP4056. Cette configuration permet une autonomie prolongée tout en assurant la sécurité du circuit grâce aux fonctions de protection contre les surcharges et les courts-circuits.

A. Batterie Li-ion 2500 mAh 5V

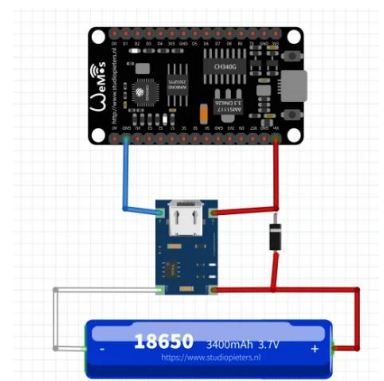
Caractéristiques principales :

- **Capacité** : 2500 mAh.
- **Tension nominale** : 3,7 V (standard pour les batteries Li-ion).
- **Tension de charge complète** : 4,2 V.
- **Courant maximal de décharge** : Généralement de 1C à 2C (2,5 A à 5 A).
- **Tension de coupure** : 2,5 V à 3 V (dépend de la batterie).

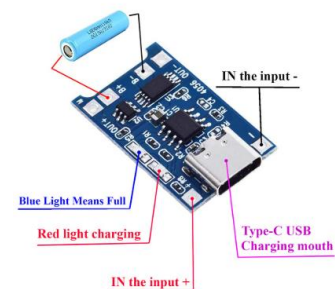
B. module TP4056

Fonctionnalités :

- **Recharge** :
 - Charge les batteries Li-ion (5 V).
 - Courant de charge typique : 1 A
- **Protection intégrée** :
 - Empêche la tension de dépasser 4,2V.
 - Coupe l'alimentation si la tension descend en dessous de 2,5 V à 3 V.
 - Déconnecte automatiquement la batterie en cas de court-circuit.









3Figure 3: schéma de liaison entre la batterie et la protection



4figure 4 : schéma illustre les caractéristique de TP4056

1.1.3 Capteurs

	Nom	Rôle
 <p>5Figure 5 : DHT11</p>	DHT11	pour mesurer la température et l'humidité
 <p>6Figure 6 : LDR</p>	LDR	mesure la luminosité ambiante
 <p>7figure 7 : Anemometre DIY</p>	Anémomètre DIY Personnalisée	Vitesse et direction du vent
 <p>8 figure 8:capteur de pluie</p>	Module de présence d'eau	Capteur de détection de pluie
 <p>9figure 9 : capteur de pression</p>	BMP280	Un capteur numérique précis conçu pour mesurer la pression atmosphérique
 <p>10figure10:capteur effet hall</p>	capteur effet hall	mesurer une variation de champ magnétique

1.2. Conception et Réalisation

1.2.1. Fonctionnement des Capteurs

❖ DHT11 : Température et Humidité

Le capteur DHT11 fournit des mesures numériques de la température et de l'humidité. Ces données sont utilisées pour surveiller l'environnement ambiant. Le fonctionnement repose sur la lecture des signaux numériques émis par le capteur via l'ESP32.

Voici un extrait de [code.ion](#) pour lire les données du capteur DHT11 :

```
#include "DHT.h"
#define DHTTYPE DHT11
const int DHTPin = 5;
DHT dht(DHTPin, DHTTYPE);
```



```
void loop() {
  float t = dht.readTemperature();
  float h = dht.readHumidity();

  if (isnan(t) || isnan(h)) {
    Serial.println("Échec de lecture du DHT11 !");
    return;
  }

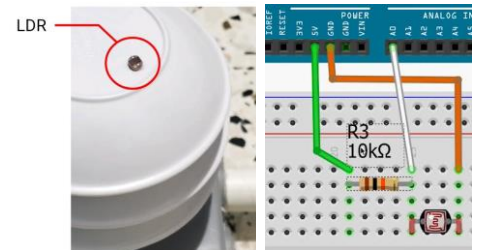
  Serial.print("Température: ");
  Serial.print(t);
  Serial.println(" °C");
}
```

11 figure 11 : code pour lire les données du capteur DHT11

❖ LDR : Intensité Lumineuse

La LDR (Light Dependent Resistor) donne une sortie analogique variant de 0 à 4096 en fonction de l'intensité lumineuse détectée. Cette valeur est convertie en pourcentage de luminosité pour une meilleure visualisation. Voici un extrait de code utilisé pour cette conversion : [Lien code.ion LDR](#)

```
int sensorValue = analogRead(ldrPin);
float luminosity = 100.0 - ((float)sensorValue /
4095.0) * 100.0;
luminosity = constrain(luminosity, 0.0, 100.0);
```



Les capteurs de température, d'humidité et de pression sont intégrés dans un boîtier spécial semi-ouvert. Ce boîtier protège les capteurs des conditions extérieures tout en permettant une circulation d'air suffisante pour mesurer avec précision les variations de température et d'humidité.

❖ Capteur de Pluie

Le capteur de pluie détecte les précipitations légères. Pour mesurer la quantité de pluie, leur sortie est un booléen : si une valeur seuil (dans notre cas 100) est dépassée, le capteur renvoie "true", indiquant la présence de pluie.

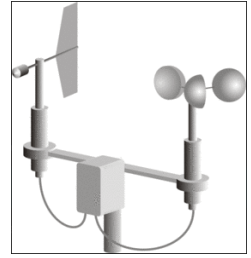
Ce capteur fournit également des résultats analogiques variant de 0 à 1080, mais cette information n'était pas directement exploitable dans notre cas.



1.2.2. Conception de l'Anémomètre personnalisée DIY

❖ Étude Théorique

L'anémomètre DIY a été conçu pour mesurer la vitesse et la direction du vent en utilisant des **capteurs à effet Hall**. Voici les principes théoriques et les choix techniques qui ont guidé sa conception :

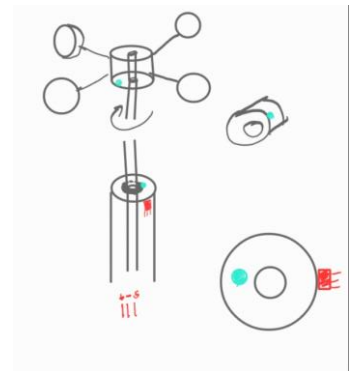


1. Principe de Fonctionnement :

L'anémomètre est composé d'un **axe de rotation** équipé d'un **aimant permanent**. Cet axe est fixé à un **ventilateur à 4 hémisphères** (ou coupelles) espacés de **29 cm**. Lorsque le vent souffle, il fait tourner le ventilateur, entraînant la rotation de l'axe et de l'aimant.

2. Détection de la Rotation :

Un **capteur à effet Hall** est placé à proximité de l'axe de rotation. Ce capteur détecte le passage de l'aimant à chaque rotation complète du ventilateur. Chaque détection génère une impulsion électrique, qui est comptabilisée par le microcontrôleur ESP32.



3. Pourquoi utiliser un capteur à effet Hall ?

- Les capteurs à effet Hall sont robustes, précis et adaptés aux environnements extérieurs.
- Ils n'ont pas de pièces mobiles, ce qui réduit l'usure mécanique.
- Ils permettent une mesure sans contact grâce à leur sensibilité aux champs magnétiques, idéale pour détecter les rotations d'un axe équipé d'un aimant.

4. Principe de fonctionnement de l'effet Hall :

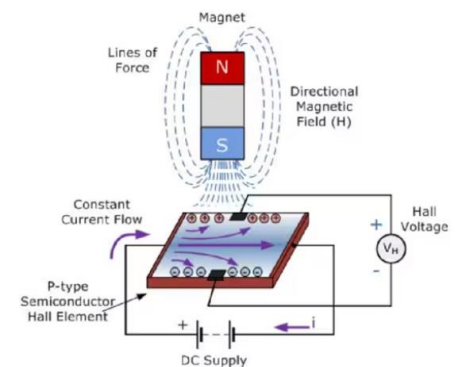
L'effet Hall repose sur le phénomène suivant : lorsqu'un courant électrique traverse un conducteur ou un semi-conducteur soumis à un champ magnétique perpendiculaire, une différence de potentiel (tension Hall) est générée perpendiculairement au courant et au champ magnétique.

○ Expression mathématique :

La tension Hall : V_H est donnée par :

$$V_H = B \cdot I \cdot q \cdot n$$

- **Avantages pour l'anémomètre** : Grâce à leur précision et leur rapidité, les capteurs à effet Hall permettent de détecter les rotations avec une excellente fiabilité.



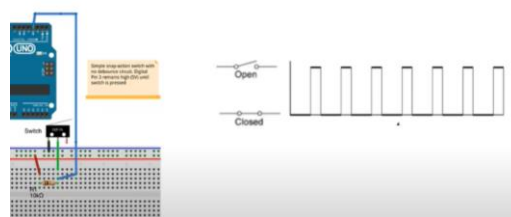
12 figure 12 : illustration de principe de l'effet Hall

5. Calcul de la Vitesse du Vent :

La vitesse du vent est déduite de la vitesse de rotation du ventilateur. Voici les étapes du calcul :

○ Circonférence du Ventilateur :

La distance parcourue par une coupelle en une rotation est donnée par la circonférence du cercle décrit par les coupelles :



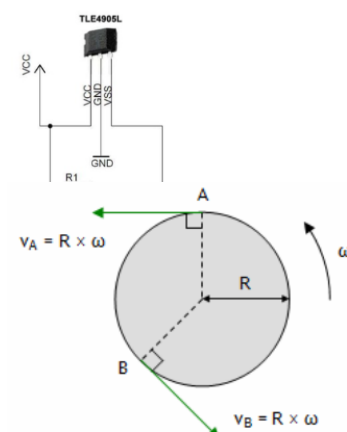
$$\text{Circonférence} = \pi \times \text{Diamètre} = 3.1415 \times 0.29 \text{ m} = 0.91 \text{ m}$$

○ Rotations par Seconde :

Le nombre de rotations par seconde est calculé en divisant le nombre total de rotations détectées par la durée de mesure (10 secondes dans ce cas).

○ Vitesse Linéaire du Vent :

La vitesse du vent est obtenue en multipliant les rotations par seconde par la circonférence et un facteur d'efficacité (2.5 dans ce projet) qui tient compte des pertes dues à la friction et à la résistance de l'air :

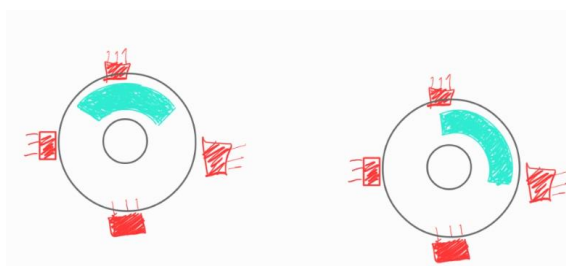
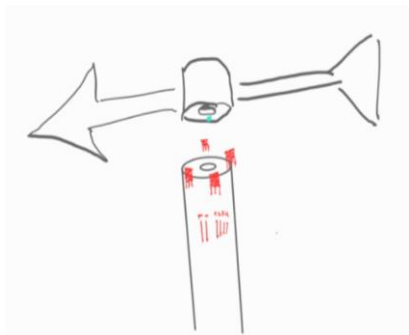


$$\text{Vitesse du vent (m/s)} = \text{Rotations par seconde} \times \text{Circonférence} \times \text{Facteur d'efficacité}$$

6. Direction du Vent :

Pour mesurer la direction du vent, un système à **4 capteurs à effet Hall** a été mis en place. Ces capteurs sont disposés en croix (Nord, Sud, Est, Ouest). Un aimant rotatif, monté sur le même axe que le ventilateur, active deux capteurs simultanément lorsqu'il passe entre deux directions cardinales (par exemple, entre Nord et Est). Cela permet de déterminer les 8 directions principales

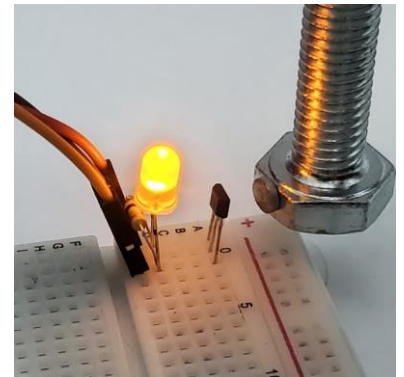
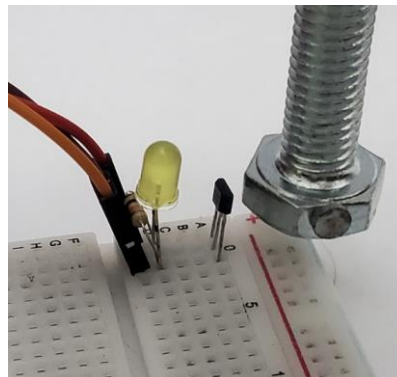
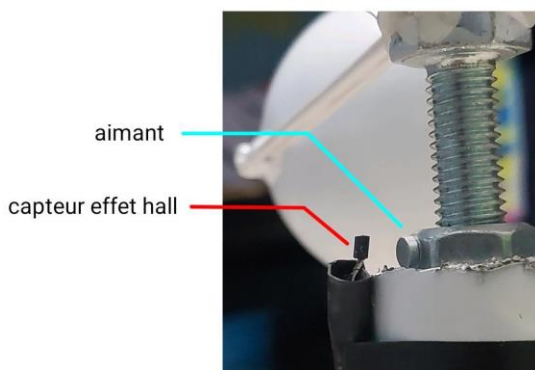
(N, NE, E, SE, S, SW, W, NW) avec seulement 4 capteurs.



❖ Conception et Réalisation

Conception Mécanique :

- **Axe de Rotation** : Un axe en métal léger a été choisi pour minimiser la friction et permettre une rotation fluide.
- **Ventilateur à 4 Coupelles** : Les coupelles ont été conçues pour offrir une résistance optimale au vent tout en étant légères.
- **Aimant et Capteur à Effet Hall** : Un aimant néodyme a été fixé sur l'axe, et le capteur à effet Hall a été positionné à une distance précise pour garantir une détection fiable.



Algorithme de Calcul :

Le microcontrôleur ESP32 compte le nombre de rotations détectées par le capteur à effet Hall sur une période de 10 secondes. Ce nombre est ensuite utilisé pour calculer la vitesse du vent. Voici l'extrait de code correspondant :

```
float calculateWindSpeed() {
    unsigned long currentTime = millis();
    if (currentTime - startTime >= 10000) { // Mesure sur 10 secondes
        float rotationsPerSecond = rotationCount / 10.0; // Rotations par seconde
        float windSpeed = rotationsPerSecond * vaneCircumference * anemometerFactor;
        // Calcul de la vitesse
        rotationCount = 0; // Réinitialisation du compteur
        startTime = currentTime; // Réinitialisation du temps
        return windSpeed;
    }
    return 0.0; // Retourne 0 si la période de mesure n'est pas écoulée
}
```

Le [code.ino](#) (GITHUB) pour tester la vitesse du vent pour notre anémomètre

Gestion de la Direction du Vent :

Les 4 capteurs à effet Hall sont lus en permanence pour déterminer la direction du

vent. voici un extrait de code.ino simplifié pour déterminer la direction du vent :

```
String readWindDirection() {  
  int eastState = !digitalRead(hallPinEast);  
  int northState = !digitalRead(hallPinNorth);  
  int westState = !digitalRead(hallPinWest);  
  int southState = !digitalRead(hallPinSouth);  
  
  if (eastState && northState) return "N-E";  
  if (eastState && southState) return "S-E";  
  if (southState && westState) return "S-W";  
  if (westState && northState) return "N-W";  
  if (eastState) return "East";  
  if (westState) return "West";  
  if (southState) return "South";  
  if (northState) return "North";  
  return "None";  
}
```



Tests et Résultats :



- **Tests de Vitesse** : L'anémomètre a été testé avec un ventilateur à vitesse contrôlée pour valider la précision des mesures. Les résultats ont montré une erreur maximale de $\pm 10\%$ par rapport à un anémomètre commercial.
- **Tests de Direction** : Le système de détection de direction a été validé en simulant les 8 directions principales. Les résultats ont confirmé la fiabilité du système.

Conclusion de la Conception

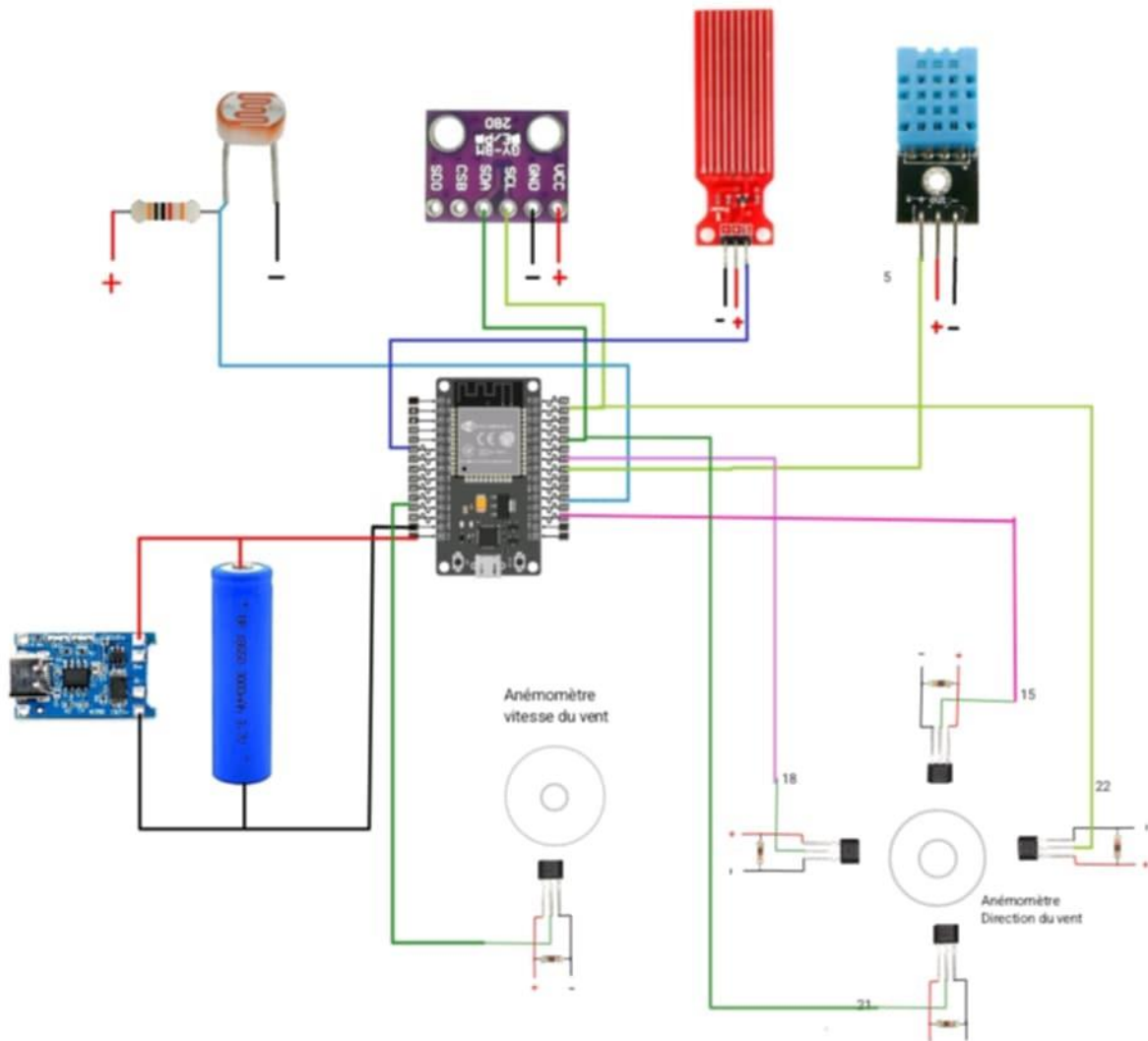
L'anémomètre DIY, bien que simple et économique, offre des performances comparables à celles des anémomètres commerciaux. Sa conception modulaire permet une maintenance facile et des améliorations futures, comme l'ajout de capteurs supplémentaires pour une précision accrue.



13 figure 13 : photo reel de project

1.3. Schéma Électronique

Le schéma électronique global, incluant tous les capteurs et leur connexion à l'ESP32, est présenté dans la Figure 13 ([lien GitHub](#)). Un aperçu des connexions principales :



14 figure 14 : schema de montage des composants électronique

- DHT11 connecté au GPIO5.
- LDR connecté à un port analogique (GPIO4).
- Capteurs à effet Hall pour le vent connecté aux GPIO13, GPIO15, GPIO18, GPIO21, et GPIO22.

Tous les composants électroniques, y compris l'ESP32, la batterie, le module TP4056 et les câbles de connexion, sont intégrés dans **un boîtier** en PVC **imperméable**. Ce boîtier assure une protection optimale contre les conditions extérieures (humidité, poussière, etc.), tout en maintenant une organisation propre et sécurisée du système électronique. Cette conception garantit la durabilité et la fiabilité de la station météo dans des environnements variés.



15 figure 15 : la boîtier qui couvre les composant électronique

❖ Tests et résultats



	A	B	C	D	E	F	G	H
1	Date	Time	Temperature (°C	Humidity (%)	luminosity	wind direction	wind speed (m/s)	rainy
500	10/01/2025	23:26:21	19.00	72,00	100.00	South	0.00	NO
501	10/01/2025	23:26:34	19.00	72,00	100.00	N-E	0.00	NO
502	10/01/2025	23:26:47	18.60	73,00	100.00	East	0.00	NO
503	10/01/2025	23:27:00	18.60	73,00	100.00	N-E	0.00	NO
504	10/01/2025	23:27:12	18.60	73,00	100.00	N-E	0.00	NO
505	10/01/2025	23:27:26	18.60	73,00	100.00	N-E	0.00	NO
506	10/01/2025	23:27:40	18.60	74,00	100.00	N-E	0.00	NO
507	10/01/2025	23:27:53	18.60	74,00	100.00	N-E	0.00	NO
508	10/01/2025	23:28:06	18.70	74,00	100.00	S-W	0.00	NO
509	10/01/2025	23:28:18	19.00	74,00	100.00	S-W	0.46	NO
510	10/01/2025	23:28:32	19.00	74,00	100.00	S-W	0.00	NO
511	10/01/2025	23:28:45	19.00	74,00	100.00	S-W	0.00	NO
512	10/01/2025	23:29:13	19.00	74,00	100.00	S-W	0.00	NO
513	10/01/2025	23:29:26	19.00	74,00	100.00	S-W	0.00	NO
514	10/01/2025	23:29:38	19.00	74,00	100.00	S-W	0.00	NO
515	10/01/2025	23:29:52	19.00	74,00	100.00	S-W	0.00	NO
516	10/01/2025	23:30:06	19.00	73,00	100.00	S-W	0.00	NO

2. Partie IoT

2.1. Introduction

La partie IoT (Internet des Objets) de ce projet joue un rôle central dans la collecte, la transmission et le stockage des données météorologiques. Elle repose sur l'utilisation d'un microcontrôleur ESP32 pour collecter les données des capteurs, les envoyer à un Google Sheet via une connexion WiFi, et les rendre accessibles pour l'interface utilisateur.

2.2. Architecture du système IoT

Le système IoT est composé des éléments suivants :

1. Capteurs

2. Microcontrôleur ESP32 :

- Collecte les données des capteurs.
- Se connecter au WiFi pour envoyer les données à Google Sheets.

```
Serial.print("Connecting to WiFi");
while (WiFi.status() != WL_CONNECTED) {
    Serial.print(".");
    digitalWrite(ON_Board_LED, LOW); // Flashing LED while connecting
    delay(250);
    digitalWrite(ON_Board_LED, HIGH);
    delay(250);
}
digitalWrite(ON_Board_LED, HIGH); // Turn off LED when connected
Serial.println("\nWiFi connected");
Serial.print("IP Address: ");
Serial.println(WiFi.localIP());
}
```

```
//-----Host & Script ID
const char* host = "script.google.com";
String GAS_ID = "AKfycbzUciUEvTM0dgRmWjCLryqXRYgH3yKKxMQofiyi8Un1Lx8isFdLwFiY2z_Ut_76Xd7X0g"; // Spreadsheet Script ID
```

3. Google Apps Script :

- Reçoit les données de l'ESP32 via une requête **HTTP GET**.
- Stocke les données dans un Google Sheet.

4. Google Sheets :

- Stocke les données météorologiques pour une utilisation ultérieure par l'interface utilisateur.

2.3 Gestion de données

2.3.1 Collecte des données

L'ESP32 collecte les données des capteurs à intervalles réguliers (toutes les 10 secondes) :

- **Température et humidité** : Lues via le capteur DHT11.
- **Direction du vent** : Détectée via quatre capteurs à effet Hall (Est, Nord, Ouest, Sud).
- **Vitesse du vent** : Calculée en comptant les rotations de l'anémomètre sur une période de 10 secondes.
- **Luminosité** : Mesurée via un capteur LDR.

2.3.2 Transmission des données

Les données collectées sont envoyées à Google Sheets via une requête HTTP GET.

```
66 // Subroutine for sending data to Google Sheets
67 void sendData(float tem, int hum) {
68   if (WiFi.status() == WL_CONNECTED) { // Ensure WiFi is connected
69     HTTPClient http;
70
71     String string_temperature = String(tem);
72     String string_humidity = String(hum);
73     String url = "https://" + String(host) + "/macros/s/" + GAS_ID + "/exec?temperature=" + string_temperature + "&humidity=" + string_humidity;
74   }
75 }
```

L'URL de la requête contient les données sous forme de paramètres, par exemple :

“https://script.google.com/macros/s/<GAS_ID>/exec?temperature=25&humidity=60&luminosity=50&wind_direction=N-E&wind_speed=3.5”

Cette URL est générée par l'ESP32 dans la fonction `sendData()`.

2.3.3 Stockage des données



Un [script Google Apps](#) Script (`doGet`) est utilisé pour recevoir les données et les stocker dans un Google Sheet. Voici comment cela fonctionne :

1. Réception des données :

- Le script récupère les paramètres de la requête HTTP GET (température, humidité, luminosité, direction du vent, vitesse du vent).

2. Formatage des données :

- La date et l'heure actuelles sont ajoutées dans les colonnes A et B.
- Les données des capteurs sont ajoutées dans les colonnes C à H.

```
var sheet_id = '1wNvRSV0WmJ008cy5nZzK2qd1DNV6SKMGpwVbCKH68FU'; // Spreadsheet ID
var sheet = SpreadsheetApp.openById(sheet_id).getActiveSheet();
var newRow = sheet.getLastRow() + 1;
var rowData = [];
var Curr_Date = new Date();
rowData[0] = Curr_Date; // Date in column A
var Curr_Time = Utilities.formatDate(Curr_Date, "Africa/Algeria", 'HH:mm:ss');
rowData[1] = Curr_Time; // Time in column B
```


3. Ajout des données :

- Une nouvelle ligne est ajoutée à la fin de la feuille de calcul.
- Les données sont insérées dans cette ligne à l'aide de la méthode `setValues()`.

2.3.4 Utilisation des données

Les données stockées dans Google Sheets sont ensuite utilisées par l'interface utilisateur :

- Les données sont affichées sous forme de graphiques et de tableaux.
- Les utilisateurs peuvent visualiser les conditions météorologiques en temps réel et suivre les tendances historiques.

4.4. Correction des données

Les données collectées peuvent parfois contenir des erreurs dues à plusieurs facteurs :

1. Erreurs de capteurs :

- Les capteurs peuvent ne pas détecter correctement les valeurs (par exemple, une température de **45000°C**).
- Des valeurs aberrantes peuvent être générées en raison de problèmes matériels ou environnementaux.

2. Erreurs de transmission :

- Les données peuvent être corrompues pendant la transmission, par exemple si le WiFi est instable.

3. Erreurs de format :

- Les données peuvent être envoyées dans un format incorrect (par exemple, une chaîne de caractères au lieu d'un nombre).

1	Date	Time	Temperature (°C)	Humidity (%)	luminosity	wind direction	wind speed (m/s)
382	30/12/2024	21:08:28	28.90	39,00	50,00	East	1.393
383	30/12/2024	21:08:42	28.90	39,00	50,00	East	1.394
384	30/12/2024	21:08:55	28.90	39,00	50,00	East	1.395
385	30/12/2024	21:09:10	28.90	39,00	50,00	East	1.396
386	30/12/2024	21:09:24	28.90	39,00	50,00	East	1.397
387	30/12/2024	21:09:38	28.90	40,00	50,00	East	1.398
388	30/12/2024	21:09:52	28.90	40,00	50,00	East	1.399
389	30/12/2024	21:10:06	29.00	39,00	50,00	East	1.400
390	30/12/2024	21:10:20	45 594.00	40,00	50,00	East	1.401
391	30/12/2024	21:10:33	29.30	39,00	50,00	East	1.402
392	30/12/2024	21:10:47	29.30	39,00	50,00	East	1.403
393	30/12/2024	21:11:01	29.30	39,00	50,00	East	1.404
394	30/12/2024	21:11:15	29.30	39,00	50,00	East	1.405
395	30/12/2024	21:11:28	29.30	39,00	50,00	East	1.406

1	Date	Time	Temperature (°C)	Humidity (%)	luminosity	wind direction	wind speed (m/s)
710	30/12/2024	22:28:49	28.00	40,00	50,00	East	1.721
711	30/12/2024	22:29:03	28.00	40,00	50,00	East	1.722
712	30/12/2024	22:29:16	28.50	40,00	50,00	East	1.723
713	09/01/2025	21:24:43	45.00	62,00	50,00	East	1.724
714	09/01/2025	21:24:57	23.80	62,00	50,00	East	1.725
715	09/01/2025	21:25:10	23.80	62,00	50,00	East	1.726
716	09/01/2025	21:25:23	23.80	62,00	50,00	East	1.727

Pour **résoudre ces problèmes**, des mécanismes de validation et de correction ont été mis en place dans le script Google sheet:

1. Validation des données :

- Le script vérifie que les données reçues sont dans des plages réalistes (par exemple, une température entre -50°C et 60°C).
- Si une valeur est aberrante, elle est remplacée par une valeur par défaut (par exemple, 0 ou une valeur précédente valide).

2. Correction des formats :

- Le script s'assure que les données sont dans le bon format (nombres pour la température, l'humidité, etc.).

- Si une valeur est incorrecte, elle est corrigée ou ignorée.
- 3. **Journalisation des erreurs :**
 - Toutes les erreurs détectées sont enregistrées dans un journal pour permettre une analyse ultérieure et une amélioration du système.

2.5. Avantages de cette architecture

- **Simplicité** : L'utilisation de Google Sheets comme base de données simplifie le stockage et la gestion des données.
- **Accessibilité** : Les données sont accessibles en temps réel via l'interface utilisateur.
- **Évolutivité** : L'architecture peut être facilement étendue pour inclure d'autres capteurs ou fonctionnalités.

3. Partie Prédiction (IA)

3.1. Préparation des Données

1. Importation des bibliothèques

```
[ ] import tensorflow as tf
import numpy as np
import pandas as pd
from tensorflow.keras.models import Sequential, load_model
from tensorflow.keras.layers import LSTM, Dense, Dropout
from sklearn.preprocessing import MinMaxScaler
from sklearn.model_selection import train_test_split
from matplotlib import pyplot as plt
from sklearn.metrics import mean_absolute_error, mean_squared_error, r2_score
```

Explication :

- **tensorflow** : Utilisé pour construire et entraîner le modèle de réseau neuronal.
- **numpy** : Pour les opérations numériques, notamment la manipulation de tableaux.
- **pandas** : Pour la gestion et le prétraitement des données tabulaires (par exemple, lecture de fichiers CSV).
- **keras** :
 - **Sequential** : Permet de définir un réseau neuronal couche par couche.
 - **LSTM** : Un type spécifique de réseau de neurones récurrents (RNN) conçu pour les tâches de prévision de séquences.
 - **Dense** : Une couche de réseau neuronal complètement connectée.

- **Dropout** : Technique de régularisation pour prévenir le surapprentissage en mettant à zéro aléatoirement certaines sorties de neurones pendant l'entraînement.
- **MinMaxScaler** : Normalise les données dans une plage fixe (généralement entre 0 et 1), ce qui aide les réseaux neuronaux à converger plus rapidement.
- **train_test_split** : Divise le jeu de données en ensembles d'entraînement et de test.
- **pyplot (matplotlib)** : Pour tracer des graphiques (par exemple, courbes de perte, valeurs prédites vs réelles).
- **mean_absolute_error, mean_squared_error, r2_score** : Métriques courantes pour évaluer les modèles de régression.

2. Lecture des données:

```
data = pd.read_csv('/content/jena_climate_2009_2016.csv') # Charger un fichier CSV
print(data)
```

- **pd.read_csv()** lit le jeu de données à partir d'un fichier CSV.
- **print(data)** affiche l'ensemble des données.

À propos du jeu de données :

Vous utilisez le jeu de données **Jena Climate 2009–2016**, qui contient des données météorologiques collectées à Jena, en Allemagne, sur plusieurs années. Ce jeu de données inclut généralement les colonnes suivantes :

- Température
- Pression
- Humidité
- Vitesse du vent
- Rayonnement solaire

3. Fonction de segmentation des signaux:

```
def signal_segmentation(dt, steps, ahead):
    x, y = [], []
    for i in range(steps, dt.shape[0] - ahead):
        x.append(dt[i-steps:i, :])
        y.append(dt[i:i+ahead, -1])
    return np.array(x), np.array(y)

# Load and prepare data
sf = data['T (degC)'].values.reshape(-1, 1)
train_size = int(len(sf) * 0.75)
train, test = sf[:train_size, :], sf[train_size:, :]

sequence_length = 144 # I defined the sequence length
ahead = 1 # Forecasting one step ahead (un pas en avant)
# Segmenting data
X_train, y_train = signal_segmentation(train, sequence_length, ahead)
X_test, y_test = signal_segmentation(test, sequence_length, ahead)
print(X_train.shape)
print(y_train.shape)
print(X_test.shape)
print(y_test.shape)
```

Explication :

- **steps** : Le nombre d'étapes temporelles précédentes utilisées comme entrée (longueur de la séquence).
- **ahead** : Le nombre d'étapes à prévoir (dans ce cas, 1 étape).

La fonction crée des fenêtres glissantes de taille **steps** à partir du jeu de données.

- Chaque **x** contient une séquence de caractéristiques allant de **i - steps** à **i**.
- Chaque **y** contient la valeur suivante (cible de la prévision) pour **ahead** étapes.

Résultats :

- **x** : Un tableau de séquences d'entrée de forme (num_samples, steps, num_features).
- **y** : Un tableau de cibles de sortie de forme (num_samples, ahead).
- Extrait la colonne de température '**T (degC)**' et la remet en forme en un tableau 2D de taille (num_samples, 1).
- Divise les données en **75 % pour l'entraînement** et **25 % pour le test**.
- Définit la longueur de la séquence (**steps**) à 144, ce qui signifie que chaque séquence d'entrée utilisera les données des **144 étapes temporelles précédentes** (par exemple, 6 jours si les données sont horaires).
- **ahead = 1** signifie prévoir une étape à l'avance.
- Appelle la fonction **signal_segmentation** pour créer les paires entrée/sortie pour les ensembles d'entraînement et de test.

Formes résultantes :

- **X_train** : La forme sera (num_samples, 144, 1), où :
 - **num_samples** : Nombre d'échantillons d'entraînement.
 - **144** : Longueur de la séquence (steps).
 - **1** : Nombre de caractéristiques (uniquement la température).
- **y_train** : La forme sera (num_samples, 1), où :
 - **num_samples** : Nombre d'échantillons d'entraînement.
 - **1** : Nombre d'étapes à prévoir.

4. Restructuration des données pour la mise à l'échelle

```

X_train = np.reshape(X_train, (X_train.shape[0], X_train.shape[1] ))
X_test = np.reshape(X_test, (X_test.shape[0], X_test.shape[1]))
scaler = MinMaxScaler().fit(X_train)
X_train = scaler.transform(X_train)
X_test = scaler.transform(X_test)
# Reshape input
X_train = np.reshape(X_train, (X_train.shape[0], sequence_length, 1))
X_test = np.reshape(X_test, (X_test.shape[0], sequence_length, 1))
# Reshape output
y_train = np.reshape(y_train, (y_train.shape[0], ahead))
y_test = np.reshape(y_test, (y_test.shape[0], ahead))
print(X_train.shape)
print(y_train.shape)
print(X_test.shape)
print(y_test.shape)

```

Explication :

Initialement, **X_train** et **X_test** sont des tableaux 3D de formes (**num_samples**, **sequence_length**, **num_features**). Ce code aplatit la deuxième dimension (**sequence_length**) pour préparer les données à la mise à l'échelle. Après cette réorganisation :

- La forme de **X_train** devient (**num_samples**, **sequence_length**).
- La forme de **X_test** devient (**num_samples**, **sequence_length**).
- Le **MinMaxScaler** normalise les valeurs d'entrée dans une plage de [0, 1]. Cela aide à améliorer la convergence du modèle LSTM pendant l'entraînement.
- Le scaler est ajusté sur **X_train** pour apprendre les valeurs minimales et maximales, puis les deux ensembles **X_train** et **X_test** sont transformés à l'aide du même scaler.
- Après la mise à l'échelle, les données sont réorganisées pour retrouver leur forme originale en 3D :

(**num_samples**, **sequence_length**, **num_features**).

Ici, **num_features = 1** car seules les données de température sont utilisées.

Les sorties (**y_train** et **y_test**) sont réorganisées pour garantir qu'elles ont le bon format pour l'entraînement du modèle.

Puisque **ahead = 1**, la forme devient (**num_samples**, **1**), ce qui signifie que chaque échantillon de sortie correspond à une étape temporelle prédite.

Cela affiche les formes finales des données d'entrée et de sortie :

- **X_train** : (**num_samples**, **sequence_length**, 1)
- **y_train** : (**num_samples**, 1)
- **X_test** : (**num_samples**, **sequence_length**, 1)

- `y_test : (num_samples, 1)`

3.2. Modèle IA et Algorithmes

1. Définition du modèle

```
model = Sequential([
    LSTM(64, return_sequences=False, input_shape=(144, 1)), # Increased units
    Dense(8, activation='relu'),
    Dense(1, activation='linear'),
])
model.summary()
```

1.1. Décomposition couche par couche

Couche LSTM : 64 Nombre d'unités LSTM (neurones). L'augmentation du nombre d'unités permet au modèle d'apprendre des motifs plus complexes à partir des séquences d'entrée.

return_sequences=False : Comme nous avons seulement besoin de la sortie finale de la couche LSTM (et non de toute la séquence), cette option est définie à **False**.

input_shape=(144, 1) : Spécifie la forme de l'entrée :

- **144** : Longueur de la séquence (nombre d'étapes temporelles précédentes utilisées pour la prévision).
- **1** : Nombre de caractéristiques (puisque nous utilisons uniquement la température comme entrée).

Couche Dense :

Cette couche complètement connectée contient **8 neurones** et utilise la fonction d'activation **ReLU**, qui introduit une non-linéarité dans le modèle.

L'objectif est d'apprendre des relations complexes supplémentaires entre les caractéristiques extraites par la couche LSTM.

Couche de sortie :

- Une couche complètement connectée avec **1 neurone** et une activation linéaire est utilisée, car il s'agit d'une tâche de régression (prévoir une valeur continue, c'est-à-dire la température).
- **activation='linear'** signifie qu'aucune fonction d'activation n'est appliquée et que la sortie est une valeur numérique brute.

1.2. Récapitulatif du modèle

Lorsque vous appelez **model.summary()**, il affiche un récapitulatif du modèle, y compris :

Le nombre de couches.

La forme de sortie de chaque couche.

Le nombre de paramètres entraînables.

1.3. Interprétation des paramètres entraînables

- **Couche LSTM :**

Le nombre de paramètres dans la couche LSTM est calculé comme suit :

$$\text{Params} = 4 \times (\text{units} \times (\text{units} + \text{input_features} + 1))$$

En substituant les valeurs :

$$\text{Params} = 4 \times (64 \times (64 + 1 + 1)) = 16,896$$

Le facteur 4 tient compte des poids et des biais associés aux portes d'entrée, d'oubli, de cellule et de sortie dans la couche LSTM.

- **Couche Dense :**

La couche dense avec **8 neurones** et **64 entrées** a :

$$\text{Params} = 64 \times 8 + 8 = 520$$

Les 8 paramètres supplémentaires sont les biais pour chaque neurone.

- **Couche de sortie :**

La couche dense finale avec 1 neurone et 8 entrées a :

$$\text{Params} = 8 \times 1 + 1 = 9$$

2. Compilation du modèle

```
model.compile(optimizer='adam', loss='mse', metrics=['mae'])
# Increase epochs during training
model.fit(X_train, y_train, epochs=50, batch_size=64, validation_split=0.2)
```

- **optimizer='adam'** : L'optimiseur Adam est largement utilisé pour l'entraînement des modèles de deep learning, car il adapte le taux d'apprentissage pendant l'entraînement, le rendant efficace et rapide.
- **loss='mse'** :

- L'erreur quadratique moyenne (**Mean Squared Error**, MSE) est utilisée comme fonction de perte, car il s'agit d'un problème de régression (prédiction de valeurs continues comme la température).
- Le MSE pénalise davantage les grandes erreurs, ce qui est utile lorsque les valeurs aberrantes peuvent avoir un impact significatif.
- **metrics=['mae']** :
 - L'erreur absolue moyenne (**Mean Absolute Error**, MAE) est utilisée comme métrique pour évaluer les performances du modèle. Elle fournit une mesure interprétable de l'erreur moyenne des prédictions dans les mêmes unités que la variable cible (température en degrés Celsius).

Entraînement du modèle

- **epochs=50** : Le modèle sera entraîné pendant 50 époques, ce qui signifie qu'il parcourra l'ensemble du jeu de données d'entraînement 50 fois.
- **batch_size=64** : Le modèle traite les données par lots de 64 échantillons à la fois avant de mettre à jour les poids. Cela permet un entraînement plus rapide et plus stable par rapport à l'utilisation de l'ensemble du jeu de données d'un coup.
- **validation_split=0.2** :
 - Cela divise automatiquement 20 % des données d'entraînement pour la validation, afin de surveiller les performances du modèle sur des données non vues pendant l'entraînement.
 - La validation permet de détecter le surapprentissage : si la perte de validation commence à augmenter tandis que la perte d'entraînement continue de diminuer, cela indique un surapprentissage.

3. Prédictions sur l'ensemble de test

```
test_predictions = model.predict(X_test)
test_predictions = test_predictions.flatten()
y_test = y_test.flatten()
train_results = pd.DataFrame(data={'Train Predictions':test_predictions, 'Actuals':y_test})
train_results
```

Cette ligne utilise le modèle LSTM entraîné pour effectuer des prédictions sur l'ensemble de test (**X_test**).

La sortie, **test_predictions**, est un tableau 2D de forme (**num_samples**, 1) car le modèle a été entraîné pour prédire une seule valeur (température) pour chaque séquence d'entrée.

Aplatir les prédictions et les valeurs réelles

La méthode **flatten()** convertit les tableaux 2D en tableaux 1D, ce qui les rend plus faciles à comparer et à visualiser.

Après aplatissage :

- **test_predictions** : Un tableau 1D des températures prédites.
- **y_test** : Un tableau 1D des températures réelles.

Création d'un DataFrame pour comparaison

La fonction **pd.DataFrame()** crée un DataFrame pandas pour afficher proprement les valeurs prédites et réelles côte à côte.

Le DataFrame résultant contient deux colonnes :

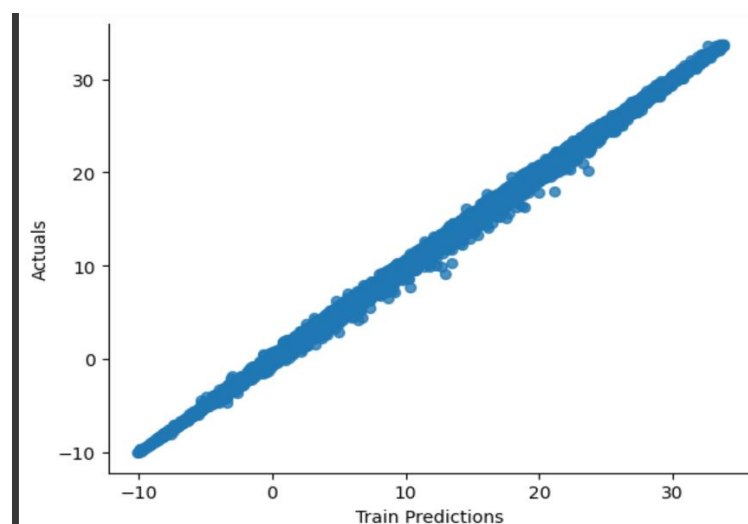
- **Prédictions** : Les températures prédites par le modèle.
- **Valeurs Réelles** : Les températures réelles de l'ensemble de test.

Visualisation des prédictions

Ce que fait ce code :

- **plt.plot(train_results['Prédictions'][50:10000])** :
 - Trace les valeurs de température prédites à partir de l'indice 50 jusqu'à 9999.
- **plt.plot(train_results['Valeurs Réelles'][50:10000])** :
 - Trace les valeurs de température réelles pour la même plage.

Cela permet de comparer les valeurs prédites et réelles de la température sur une plage spécifiée.



4. Calcul des métriques d'évaluation

```
#####  
# Calculate MAE  
mae = mean_absolute_error(y_test, test_predictions)  
print(f"Mean Absolute Error (MAE): {mae:.2f}")  
  
# Calculate MSE  
mse = mean_squared_error(y_test, test_predictions)  
print(f"Mean Squared Error (MSE): {mse:.2f}")  
  
# Calculate RMSE  
rmse = np.sqrt(mse)  
print(f"Root Mean Squared Error (RMSE): {rmse:.2f}")  
  
# Calculate R2 score  
r2 = r2_score(y_test, test_predictions)  
print(f"R2 Score: {r2:.2f}")  
#####
```

1. Erreur absolue moyenne (MAE)

La fonction **mean_absolute_error()** calcule la magnitude moyenne des différences absolues entre les valeurs prédites et réelles.

Une MAE plus faible indique de meilleures performances.

2. Erreur quadratique moyenne (MSE)

La fonction **mean_squared_error()** calcule la moyenne des carrés des différences entre les valeurs prédites et réelles.

Le carré des erreurs pénalise davantage les grandes erreurs que les petites.

3. Racine de l'erreur quadratique moyenne (RMSE)

Le **RMSE** est la racine carrée du MSE et fournit une estimation de l'erreur dans les mêmes unités que la variable cible (par exemple, degrés Celsius pour la température).

Il est souvent plus facile à interpréter que le MSE, car il est sur la même échelle que les données.

4. Score R²

La fonction **r2_score()** mesure dans quelle mesure les prédictions correspondent à la variance des données réelles.

Elle varie de 0 à 1 :

- **$R^2 = 1$** : Ajustement parfait (le modèle explique toute la variance).
- **$R^2 = 0$** : Le modèle ne fait pas mieux que prédire la moyenne.
- **Valeurs négatives** : Le modèle fait pire que prédire la moyenne.

Un score R^2 plus élevé indique de meilleures performances.

Interprétation des métriques

- **MAE** : Une MAE faible signifie qu'en moyenne, les prédictions du modèle sont très proches des valeurs réelles.
- **MSE et RMSE** : Les deux sont utiles pour comprendre l'erreur globale, mais le RMSE est souvent préféré car il est dans la même unité que la variable cible.
- **Score R^2** : Un score R^2 proche de 1 indique que le modèle explique la plupart de la variance des données.
 - Si le R^2 est proche de 0 ou négatif, cela suggère que le modèle sous-apprend ou ne parvient pas à capturer les motifs des données.

5. Extraire le data de station :

```
import gspread
from google.oauth2.service_account import Credentials

scopes = [
    'https://www.googleapis.com/auth/spreadsheets',
    'https://www.googleapis.com/auth/drive'
]

creds = Credentials.from_service_account_file('credentials.json', scopes=scopes)
client = gspread.authorize(creds)
sheet_id = "15aYsguqDwRKMUshvN8VqOC5XT-FEIwNMwwlNgYYgXo"
workbook = client.open_by_key(sheet_id)
sheet = workbook.worksheet("Sheet2")
temperature = sheet.col_values(3) # 3 represents column temperature
temperature = temperature[2:]
```

```
# Convert temperature values to floats
temperature = [t.replace('\u202f', '').replace(',', '.') for t in temperature] # Remove narrow no-break space and replace comma with period
temperature = np.array(temperature, dtype=np.float64) # Convert to float64

# Only use the last 'sequence_length' values if you have more than 'sequence_length'
if len(temperature) > sequence_length:
    temperature = temperature[-sequence_length:]

# Reshape to match the model's input shape (samples, sequence_length, features)
temperature = temperature.reshape(1, len(temperature), 1) # Use len(temperature)

# Scale the data using the same scaler used for training
temperature = scaler.transform(temperature.reshape(1, -1)) # Reshape, then scale
temperature = temperature.reshape(1, sequence_length, 1) # Reshape back to original shape

temp_predict = model.predict(temperature)
print(temp_predict)
```

```

import gspread
from google.oauth2.service_account import Credentials
scopes = [
    'https://www.googleapis.com/auth/spreadsheets',
    'https://www.googleapis.com/auth/drive'
]

creds = Credentials.from_service_account_file('credentials.json', scopes=scopes)
client = gspread.authorize(creds)
sheet_id = "1_3DtsUNf7wvkqfnMUjXgEwuCuM6L3NnsUgYfoQUVLDs"
workbook = client.open_by_key(sheet_id)
sheet = workbook.worksheet("sheet1")
sheet.update_cell(1, 1, "temperature")
num_predictions = temp_predict.shape[0]

# Prepare the data for updating multiple cells
cell_list = sheet.range(f'A2:A{num_predictions + 1}') # Start from A2 to avoid overwriting "temperature"
for i, cell in enumerate(cell_list):
    # Convert the NumPy float32 to a regular Python float
    cell.value = float(temp_predict[i][0])

# Update the cells in one batch
sheet.update_cells(cell_list)

```

Ce code utilise la bibliothèque **gspread** pour s'authentifier auprès de Google Sheets et récupérer les données de température depuis la troisième colonne d'une feuille nommée "**Sheet2**". Il suppose que le fichier d'identifiants (**credentials.json**) est correctement configuré pour votre projet Google Cloud.

Nettoyage des données :

Le code supprime les espaces insécables étroits (\u202f) et remplace les virgules par des points afin de standardiser le format des nombres.

Conversion en type float64 :

Les données de température nettoyées sont converties en un tableau numpy de type **float64** pour permettre les opérations numériques.

Utilisation d'une fenêtre glissante pour la longueur de séquence :

Si les données de température contiennent plus de valeurs que la longueur de séquence spécifiée (**sequence_length**), seules les dernières **sequence_length** valeurs sont conservées, afin de fournir au modèle les données les plus récentes.

Remodelage des données :

Les données de température sont remodelées en un tableau 3D avec des dimensions (**échantillons, longueur de séquence, caractéristiques**), conformément à la forme d'entrée attendue par le modèle LSTM.

Mise à l'échelle des données :

Les données sont mises à l'échelle à l'aide du même scaler utilisé lors de l'entraînement. Cette étape nécessite de remodeler temporairement les données en un tableau 1D pour la mise à l'échelle, avant de les remettre en forme.

Prédiction du modèle :

Les données mises à l'échelle sont passées au modèle entraîné pour générer une prédiction de température.

Authentification auprès de Google Sheets :

Le code s'authentifie auprès de l'API Google Sheets à l'aide d'un compte de service (**credentials.json**) et accède à la feuille via **gsread**.

Ouverture de la feuille :

Le **sheet_id** correspond à un Google Sheet spécifique, et le code accède à une feuille nommée "**Sheet1**".

Mise à jour de l'en-tête :

L'en-tête de la feuille (cellule A1) est mis à jour avec la valeur "**temperature**", indiquant que les données de la colonne A représentent des valeurs de température.

Préparation pour les mises à jour multiples des cellules :

Le nombre de prédictions (**num_predictions**) est déterminé en fonction de la forme de **temp_predict**. La méthode **sheet.range()** crée une liste de cellules à partir de **A2** pour éviter d'écraser l'en-tête.

Définition des valeurs des cellules :

La boucle itère sur la liste des cellules (**cell_list**), et la valeur de chaque cellule est définie avec la prédiction correspondante dans **temp_predict**. La fonction **float()** garantit que les valeurs sont au format flottant avant de les attribuer aux cellules.

Mise à jour par lot :

La méthode **sheet.update_cells()** envoie toutes les mises à jour en une seule fois, ce qui améliore l'efficacité.

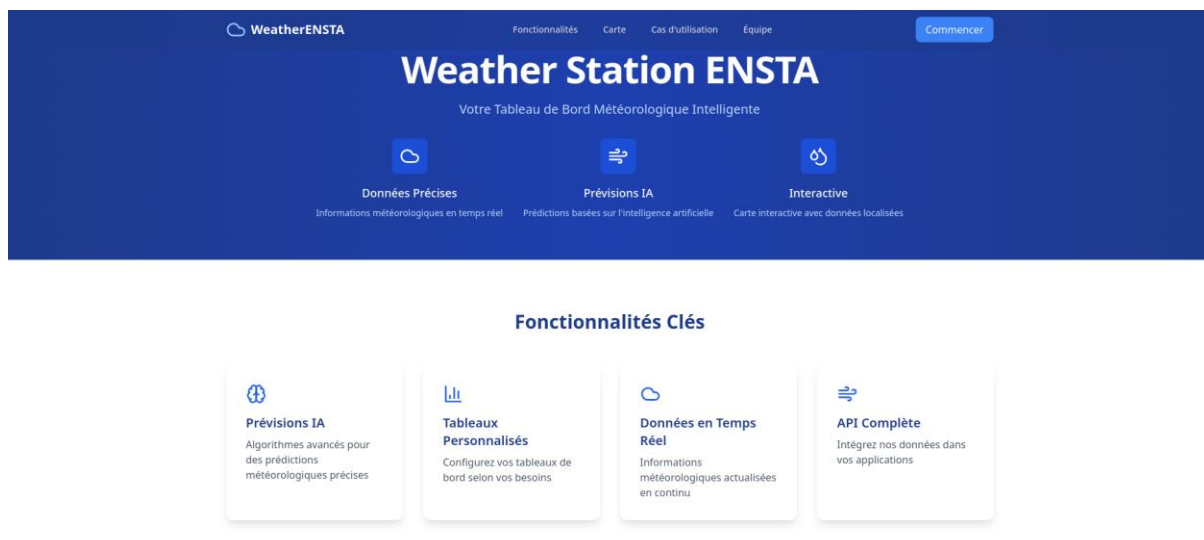
4. Partie Interface Utilisateur

4.1. Design du Site Web

L'interface utilisateur a été conçue pour offrir une expérience intuitive et conviviale, permettant aux utilisateurs d'accéder facilement aux données météorologiques en temps réel et aux prévisions basées sur l'intelligence artificielle. Le design a été réalisé en utilisant les technologies suivantes :

- React : Pour la structure du site et la gestion des composants dynamiques.
- TypeScript : Pour la gestion des types de données et une meilleure robustesse du code.
- JavaScript : Pour les fonctionnalités interactives et dynamiques.
- Tailwind CSS : Pour le style et la mise en page responsive, permettant une personnalisation rapide et efficace.
- Firebase : Pour l'authentification des utilisateurs et le stockage des données en temps réel.

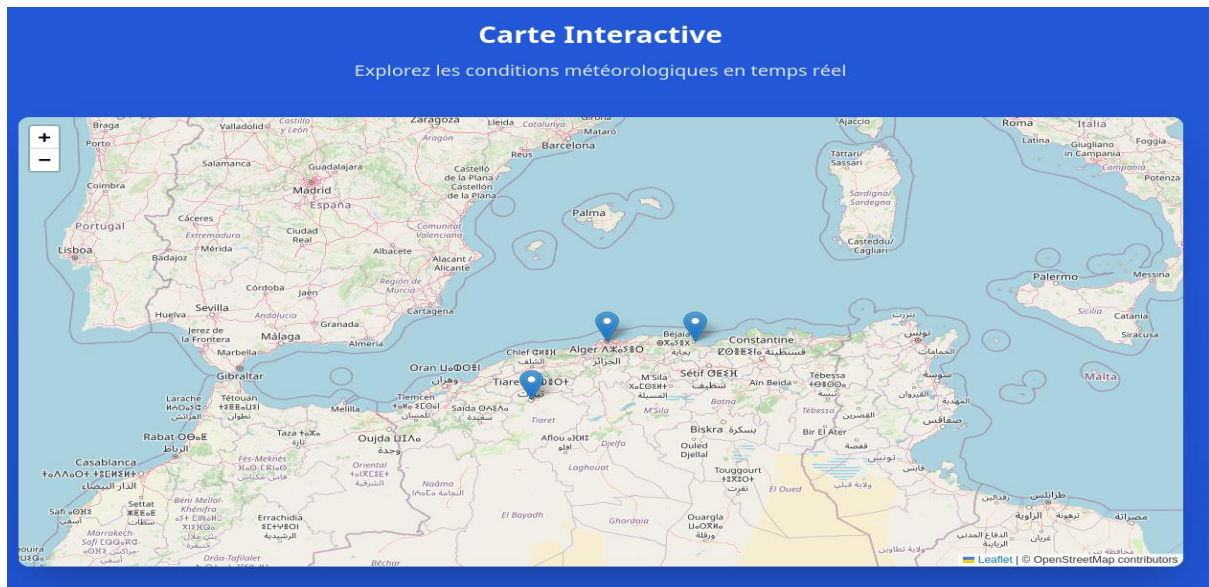
Le site web est structuré autour de plusieurs sections clés, chacune conçue pour offrir une expérience utilisateur optimale.



4.2. Fonctionnalités Offertes

4.2.1. Carte Interactive

La carte interactive permet aux utilisateurs d'explorer les conditions météorologiques en temps réel. En cliquant sur un emplacement spécifique, comme Alger, les utilisateurs peuvent accéder à des données détaillées telles que la température, l'humidité, la vitesse du vent, et la luminosité. La carte est intégrée à l'aide de bibliothèques JavaScript telles que Leaflet ou Google Maps API, permettant une visualisation fluide et interactive.



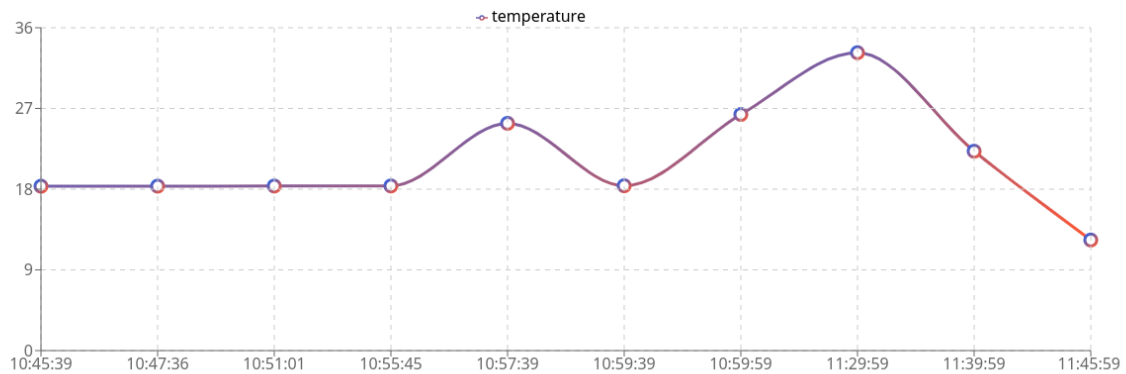
4.2.2. Graphiques et Tableaux

Les données météorologiques sont récupérées à partir d'une Google Sheet via l'API Google Sheets. Voici comment cela fonctionne :

- 1. Récupération des données :** L'application utilise l'API Google Sheets pour accéder aux données stockées dans une feuille de calcul Google. Ces données incluent des mesures de température, d'humidité, de vitesse du vent, et de luminosité.
- 2. Affichage des données :** Les données sont ensuite affichées sous forme de graphiques en temps réel et de tableaux pour les valeurs journalières. Les graphiques sont générés à l'aide de bibliothèques comme Chart.js ou D3.js, offrant une visualisation claire et interactive des tendances météorologiques.
- 3. Mise à jour en temps réel :** Les données sont mises à jour automatiquement à intervalles réguliers, garantissant que les utilisateurs ont toujours accès aux informations les plus récentes.

Météo à Alger					
30/12/2024 Lieu : Alger		28.76°C			
Jour	Température	Humidité	Vitesse Vent	Luminosité	Pluie
30/12/2024	28.76°C	41.78%	0.54 m/s	100.1050	Oui
09/01/2025	21.34°C	68.55%	0.35 m/s	100.339	Oui
10/01/2025	19.81°C	70.35%	0.19 m/s	100.00	Oui
11/01/2025	18.21°C	72.00%	0.01 m/s	100.251	Oui
12/01/2025	18.23°C	72.00%	0.03 m/s	100.252	Oui
13/01/2025	18.24°C	72.00%	0.04 m/s	100.254	Oui
14/01/2025	18.25°C	72.00%	0.05 m/s	100.255	Oui
15/01/2025	18.26°C	72.00%	0.06 m/s	100.256	Oui
16/01/2025	19.24°C	72.00%	0.14 m/s	100.257	Oui

Température au fil des 10 Derniers Résultats



4.2.3. Prévisions IA

Les prévisions météorologiques sont générées à l'aide d'algorithmes d'intelligence artificielle. Ces prévisions sont affichées dans une section dédiée, offrant des informations précises sur les conditions météorologiques futures. Les utilisateurs peuvent ainsi planifier leurs activités en fonction des prévisions.

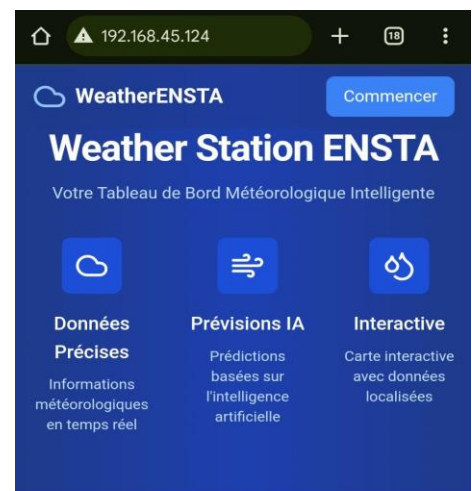
4.2.4. Responsive Design

L'interface est conçue pour être responsive, s'adaptant à différents appareils (ordinateurs de bureau, tablettes, smartphones) pour une expérience utilisateur optimale.

Tailwind CSS facilite cette adaptabilité grâce à son système de grille et ses utilitaires de mise en page.

4.2.5. Intégration API

Les utilisateurs peuvent intégrer les données météorologiques dans leurs systèmes existants grâce à une API RESTful. Cette API permet une intégration facile et personnalisée des données dans d'autres applications ou services.



Fonctionnalités Clés



Prévisions IA

Algorithmes avancés pour des prédictions météorologiques précises

4.3. Expérience Utilisateur

L'interface a été conçue avec un accent particulier sur l'expérience utilisateur. Les retours des clients, tels que ceux de Sofiane Bouzbara (Directeur Logistique, TransCorp) et Faouzi Assous (Agriculteur), soulignent l'utilité et la précision des prévisions IA, ainsi que l'intuitivité de l'interface. Les utilisateurs apprécient particulièrement la facilité d'accès aux données en temps réel et la possibilité de visualiser les tendances météorologiques via des graphiques et des tableaux.

4.4. Sections Détaillées

4.4.1. Carte Interactive

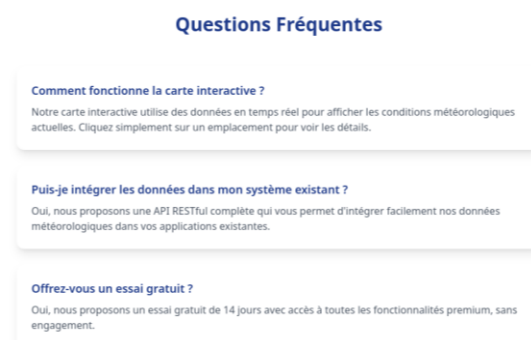
La carte interactive permet aux utilisateurs de visualiser les conditions météorologiques en temps réel. En cliquant sur un emplacement, les utilisateurs peuvent accéder à des données détaillées, telles que la température, l'humidité, et la vitesse du vent. Cette fonctionnalité est particulièrement utile pour les secteurs comme la logistique et l'agriculture, où les conditions météorologiques jouent un rôle crucial.

4.4.2. Tableau de Bord

Le tableau de bord affiche les données météorologiques sous forme de graphiques et de tableaux. Les graphiques montrent les valeurs instantanées, tandis que les tableaux présentent les données journalières des jours précédents. Cette section est essentielle pour suivre les tendances et prendre des décisions informées.

4.4.3. FAQ

La section FAQ répond aux questions fréquentes des utilisateurs, telles que le fonctionnement de la carte interactive, l'intégration des données via API, et les options d'essai gratuit. Cette section est conçue pour être claire et concise, offrant des réponses rapides aux questions courantes.

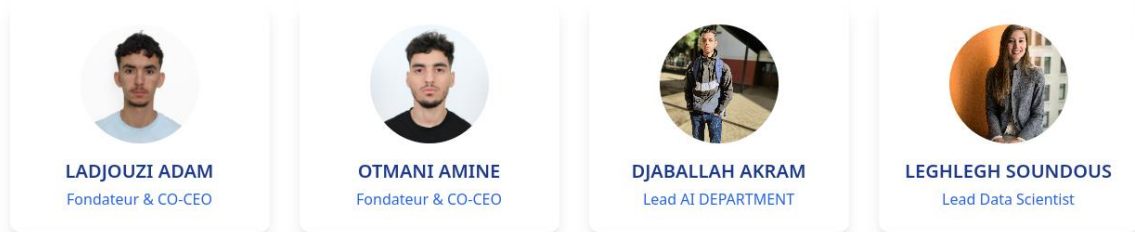


4.4.4. Notre Équipe

Cette section présente les membres de l'équipe, mettant en avant leur expertise en météorologie et en technologie. Elle renforce la confiance des utilisateurs en montrant les compétences et l'expérience de l'équipe derrière le projet.

Notre Équipe

Des experts passionnés par la météorologie et la technologie



4.5. Partie Commerciale

Notre solution météorologique intelligente cible plusieurs secteurs clés, notamment le transport et la logistique, l'agriculture, l'organisation d'événements. Ces secteurs bénéficient de prévisions météorologiques précises et de données en temps réel pour optimiser leurs opérations.

Nous proposons trois plans tarifaires pour répondre aux besoins de nos clients :

1. Starter (000 DZD/mois) : Accès aux données de base, carte interactive limitée, et support par email.
2. Pro (2900 DZD/mois) : Données en temps réel, carte interactive complète, prévisions IA avancées, et accès à l'API.
3. Enterprise (Sur mesure) : Intégration sur mesure, soutien personnalisé, et support client 24/7.

Un essai gratuit de 14 jours est également disponible, permettant aux utilisateurs de tester toutes les fonctionnalités premium sans engagement. Cette approche commerciale vise à établir des partenariats durables en offrant des solutions adaptées et un service client de qualité.

Tarifs Simples et Transparents

Choisissez le plan qui correspond à vos besoins

Starter	Pro	Enterprise
0DZD/mois	2900DZD/mois	Sur mesure/mois
<ul style="list-style-type: none">✓ Accès aux données de base✓ Carte interactive limitée✓ Mises à jour quotidiennes✓ Support par email	<ul style="list-style-type: none">✓ Données en temps réel✓ Carte interactive complète✓ Prévisions IA avancées✓ API Access✓ Support prioritaire	<ul style="list-style-type: none">✓ Solutions personnalisées✓ Intégration sur mesure✓ SLA garanti✓ Support dédié 24/7✓ Formation incluse
Commencer	Commencer	Commencer

Conclusion et perspectives

Ce projet démontre le potentiel des nouvelles technologies dans la transformation des outils de mesure traditionnels en solutions intelligentes, ouvrant la voie à des applications encore plus sophistiquées dans le futur.

En termes de perspectives, plusieurs axes d'amélioration et d'évolution peuvent être envisagés. Il s'agirait notamment d'améliorer la précision et la fiabilité des mesures en intégrant des capteurs plus avancés et en développant des algorithmes de correction pour les conditions extrêmes.

Le projet pourrait également élargir ses fonctionnalités en ajoutant de nouveaux capteurs, tels que ceux dédiés à la qualité de l'air ou aux niveaux de pollution, ou en mettant en place des alertes configurables. Par ailleurs, l'intelligence artificielle utilisée pourrait être optimisée en entraînant les modèles avec des ensembles de données plus larges et en explorant des algorithmes plus avancés.

Enfin, des efforts pourraient être faits pour renforcer l'autonomie énergétique du système grâce à des solutions comme des panneaux solaires et pour développer des interfaces encore plus accessibles, notamment sous forme d'applications mobiles