Projet Semestriel 4

Développer un système IOT basé sur ESP32 Pour récupérer et afficher les données météorologiques

Réalisé par :

Oussama Khlif & Yasmine Mustapha

Classe : 4TR

Année Universitaire 2023/2024

Introduction générale :

Notre projet consiste à mettre en place un système IoT permettant la collecte et l'affichage en temps réel des données météorologiques à l'aide de divers capteurs, tels qu'un module de détection de pluie, un module de détection de lumière, un capteur de température et d'humidité DHT11, ainsi que des dispositifs de signalisation tels qu'un buzzer et une LED, tous intégrés à un microcontrôleur ESP32.

L'objectif principal est de fournir une solution efficace et conviviale pour surveiller les conditions météorologiques locales. Pour cela, nous utiliserons Firebase pour stocker et partager les données collectées, et nous développerons une application mobile conviviale à l'aide de MIT App Inventor, permettant aux utilisateurs d'accéder facilement aux données météorologiques depuis leurs appareils mobiles.

Ce projet combine l'électronique, la programmation et le développement d'applications mobiles pour créer une solution intégrée et innovante, offrant un accès facile aux données météorologiques en temps réel pour une variété d'applications telles que l'agriculture, la planification des activités de plein air et la surveillance environnementale. Notre rapport détaillera le processus de conception, de développement et de mise en œuvre de ce système IoT, mettant en lumière les défis rencontrés et les solutions apportées à chaque étape du projet.

Architecture :

Le projet de services météo est conçu pour offrir une expérience météorologique complète et interactive. En combinant des capteurs environnementaux tels que le DHT11, un capteur de pluie et un capteur de lumière, avec des dispositifs de signalisation comme des LEDs et un buzzer, nous avons créé un système robuste pour détecter et réagir aux conditions météorologiques en temps réel. En transférant ces données vers Firebase, vous avez établi une connexion fluide entre votre dispositif et une application mobile développée avec MIT App Inventor. Cette application permet aux utilisateurs de visualiser facilement les informations météorologiques pertinentes, telles que la température, l'humidité, la présence de pluie et l'intensité lumineuse, offrant ainsi une solution pratique et conviviale pour suivre les changements météorologiques.

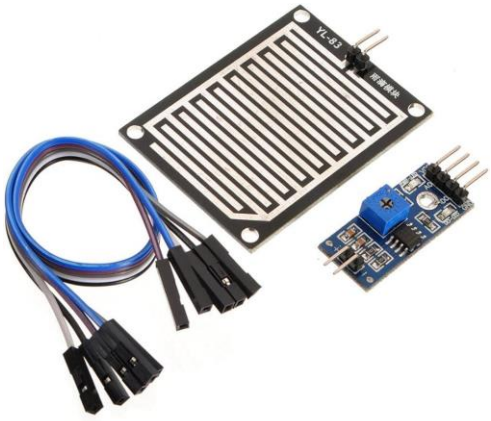
Réalisation :

La réalisation de notre projet de système IoT pour la collecte et l'affichage en temps réel des données météorologiques s'est déroulée en plusieurs étapes clés.

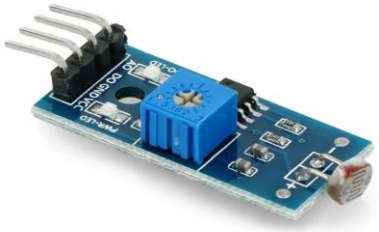
1. Sélection et Intégration des Capteurs

Nous avons sélectionné plusieurs capteurs pour mesurer les différentes variables météorologiques :

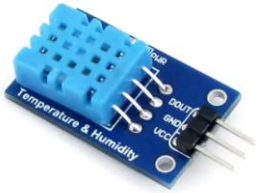
- **Module de détection de pluie :** Un module de capteur de gouttes de pluie analogique à 4 broches est utilisé pour détecter la présence et l'intensité des gouttes de pluie. Ce type de module est souvent utilisé dans des projets électroniques comme les systèmes d'irrigation automatique, les dispositifs de détection de pluie pour les essuie-glaces de voiture, et d'autres applications sensibles à l'humidité.



- **Module de détection de lumière :** Un module de capteur de lumière analogique à 4 broches est un composant électronique courant utilisé pour détecter l'intensité lumineuse. Ces modules incluent généralement un composant sensible à la lumière (tel qu'un photoresistor ou une photodiode) et sont utilisés dans diverses applications comme le contrôle automatique de l'éclairage, l'éclairage extérieur et les projets électroniques DIY.



- **Capteur de température et d'humidité DHT11 :** Le DHT11 est un capteur numérique utilisé pour mesurer la température et l'humidité. Il est largement utilisé dans divers projets électroniques et de bricolage en raison de sa simplicité et de son faible coût.



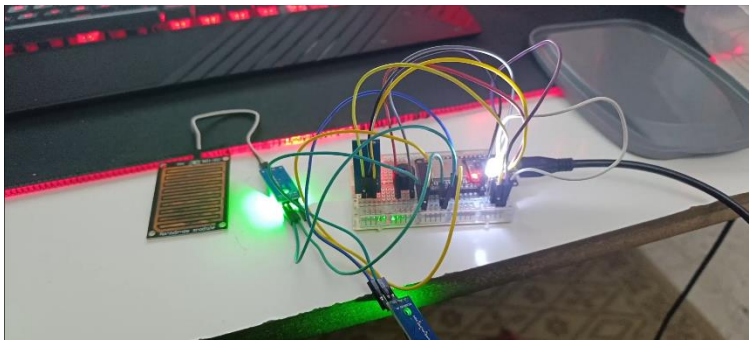
Ces capteurs ont été intégrés à un microcontrôleur ESP32, choisi pour sa connectivité Wi-Fi, facilitant l'envoi de données vers Firebase.

2. Assemblage du Matériel

L'assemblage du matériel a impliqué le câblage des capteurs au microcontrôleur ESP32 et la mise en place des dispositifs de signalisation :

- Buzzer : Activé pour alerter en cas de conditions météorologiques extrêmes.
- LED : Indique l'état de fonctionnement du système.

Le schéma de câblage a été soigneusement élaboré pour assurer une connexion correcte et fiable de tous les composants.



3. Programmation du Microcontrôleur

Nous avons programmé le microcontrôleur ESP32 en utilisant l'IDE Arduino. Le code a été écrit pour collecter les données des capteurs, les traiter et les envoyer en temps réel à Firebase. Les principales fonctionnalités du code incluent :

Lecture des données des capteurs à intervalles réguliers.

Envoi des données collectées à Firebase.

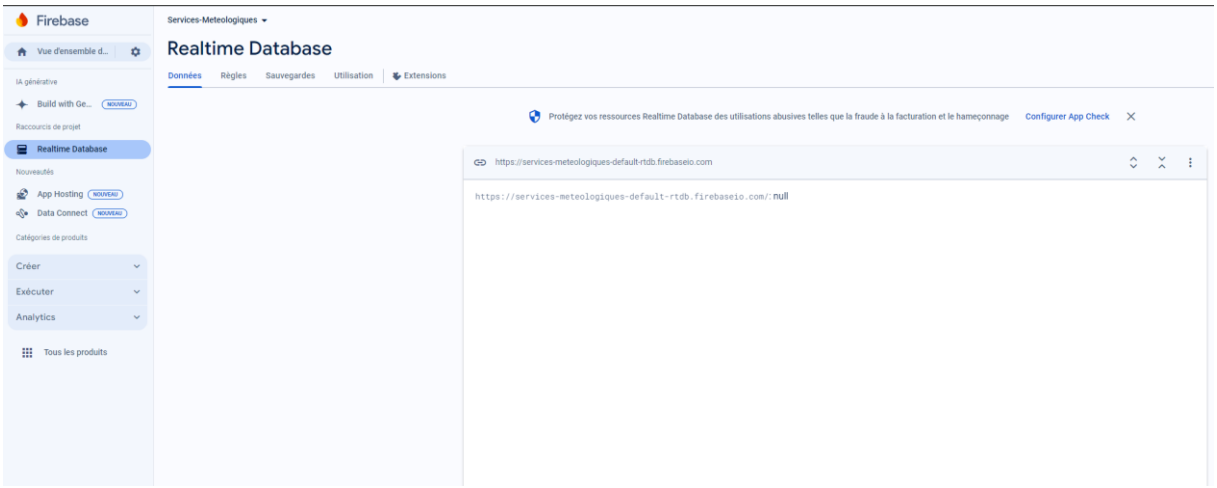
Activation du buzzer et de la LED en fonction des conditions météorologiques détectées.

```
#include <WiFi.h>
#include <DHT.h>
#include <IOXhop_FirebaseESP32.h>

const char* WIFI_SSID = "IOT";
const char* WIFI_PASSWORD = "iot12345";

#define FIREBASE_HOST "https://services-meteologiques-default-rtdb.firebaseio.com/"
#define FIREBASE_AUTH "3Zg8ehZjxaGoUlrhE343tvTdgXZhli04KwlaPUrQ"

#define DHT_PIN 14
#define LIGHT_SENSOR_PIN 35
#define RAIN_SENSOR_PIN 32
#define LED_PIN 12
#define BUZZER_PIN 13

#define DHT_TYPE DHT11
#define RAIN_THRESHOLD 500
#define COLD_THRESHOLD 15
#define HOT_THRESHOLD 30

DHT dht(DHT_PIN, DHT_TYPE);

void setup() {
  Serial.begin(115200);

  WiFi.begin(WIFI_SSID, WIFI_PASSWORD);
  while (WiFi.status() != WL_CONNECTED) {
    delay(1000);
    Serial.println("Connecting to WiFi...");
  }
  Serial.println("Connected to WiFi");

  dht.begin();

  pinMode(LIGHT_SENSOR_PIN, INPUT);
  pinMode(RAIN_SENSOR_PIN, INPUT);
  pinMode(LED_PIN, OUTPUT);
  pinMode(BUZZER_PIN, OUTPUT);

  Firebase.begin(FIREBASE_HOST, FIREBASE_AUTH);
}

void loop() {
  float temperature = dht.readTemperature();
  float humidity = dht.readHumidity();
  int lightIntensity = analogRead(LIGHT_SENSOR_PIN);
  int rainValue = analogRead(RAIN_SENSOR_PIN);

  bool isDaytime = (lightIntensity > 500);
  bool isRaining = (rainValue > RAIN_THRESHOLD);
  bool isCold = (temperature < COLD_THRESHOLD);
  bool isHot = (temperature > HOT_THRESHOLD);
```

```
digitalWrite(LED_PIN, isDaytime ? HIGH : LOW);

if (isRaining) {
  for (int i = 0; i < 3; i++) {
    digitalWrite(BUZZER_PIN, HIGH);
    delay(500);
    digitalWrite(BUZZER_PIN, LOW);
    delay(500);
  }
} else if (isCold) {
  digitalWrite(BUZZER_PIN, HIGH);
  delay(1000);
  digitalWrite(BUZZER_PIN, LOW);
  delay(500);
} else if (isHot) {
  for (int i = 0; i < 2; i++) {
    digitalWrite(BUZZER_PIN, HIGH);
    delay(500);
    digitalWrite(BUZZER_PIN, LOW);
    delay(500);
  }
} else {
  digitalWrite(BUZZER_PIN, LOW);
}


  Serial.print("Temperature: ");
  Serial.print(temperature);
  Serial.print("°C, Humidity: ");
  Serial.print(humidity);
  Serial.print("%, Light Intensity: ");
  Serial.print(lightIntensity);
  Serial.print(", Rain Value: ");
  Serial.println(rainValue);

  Firebase.setFloat("oussama/temperature", temperature);
  Firebase.setFloat("oussama/humidity", humidity);
  Firebase.setInt("oussama/light_intensity", lightIntensity);
  Firebase.setInt("oussama/rain_value", rainValue);

  delay(2000);
}
```

4. Configuration de Firebase

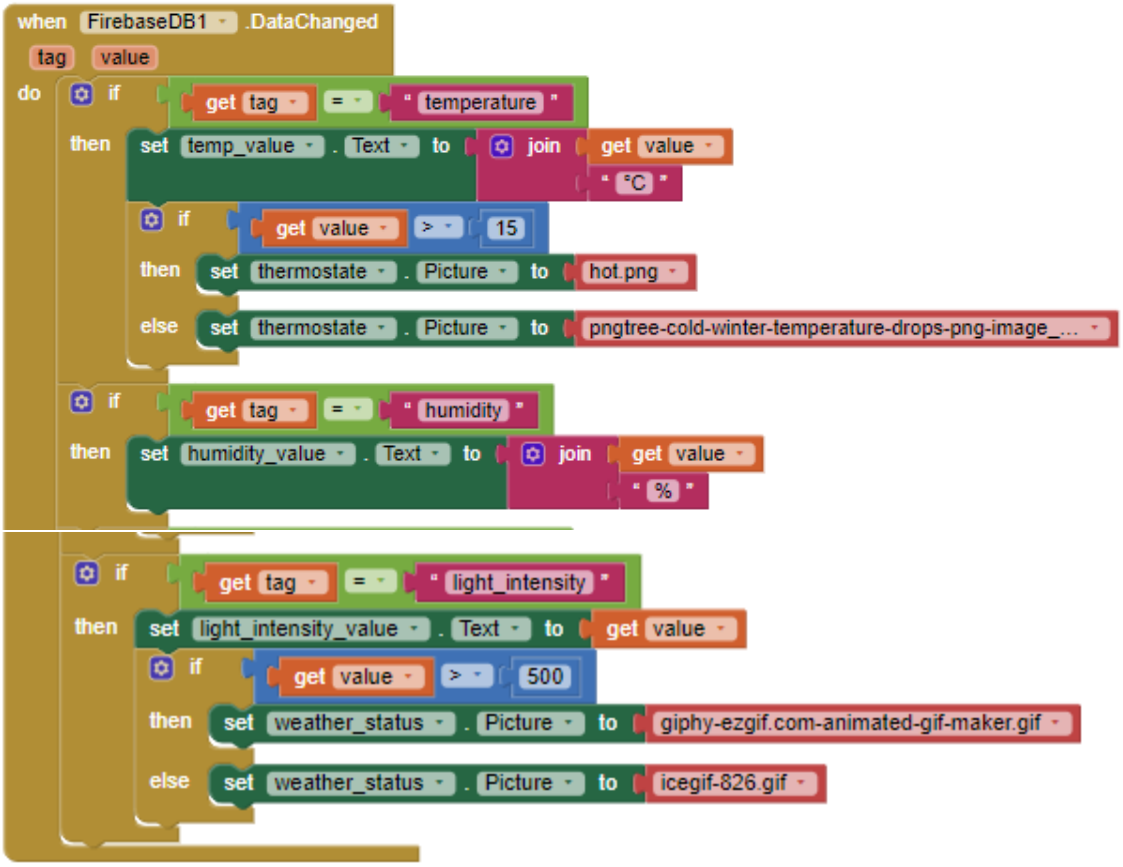
Firebase a été configuré pour stocker et partager les données météorologiques. Nous avons créé une base de données en temps réel permettant de stocker les valeurs des capteurs. Des règles de sécurité ont été mises en place pour contrôler l'accès aux données.



5. Développement de l'Application Mobile

Nous avons utilisé MIT App Inventor pour développer une application mobile conviviale. L'application permet aux utilisateurs d’afficher les données météorologiques en temps réel.

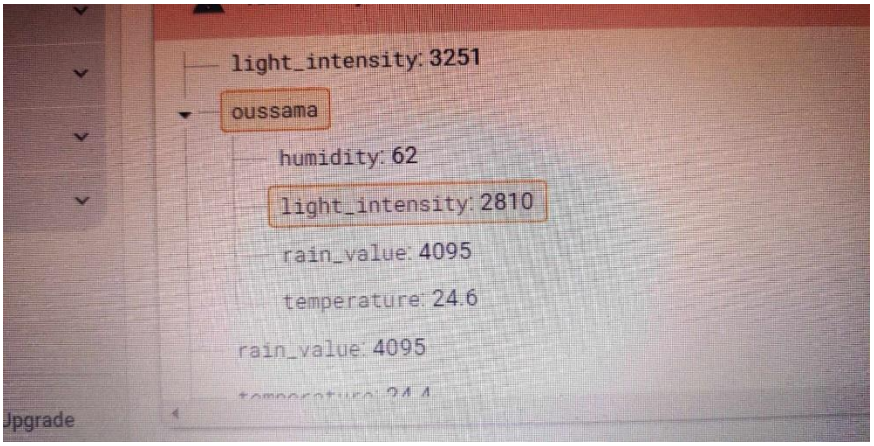

Display hidden components in Viewer
Phone size (505,320)
Android 5+ Devices (Android Material)
12:30
Météo
Prévisions Météorologiques
Emplacement:
Sousse
Activer les services de localisation
23°C
Dégagé 23°/16°
QIA 45
Humidité: 80%
Non-visible components
FirebaseDB1


when FirebaseDB1 .DataChanged
tag value
do if get tag = " temperature "
then set temp_value . Text to join get value " °C "
if get value > 15
then set thermostate . Picture to hot.png
else set thermostate . Picture to pngtree-cold-winter-temperature-drops-png-image_...
if get tag = " humidity "
then set humidity_value . Text to join get value " % "
if get tag = " light_intensity "
then set light_intensity_value . Text to get value
if get value > 500
then set weather_status . Picture to giphy-ezgif.com-animated-gif-maker.gif
else set weather_status . Picture to icegif-826.gif

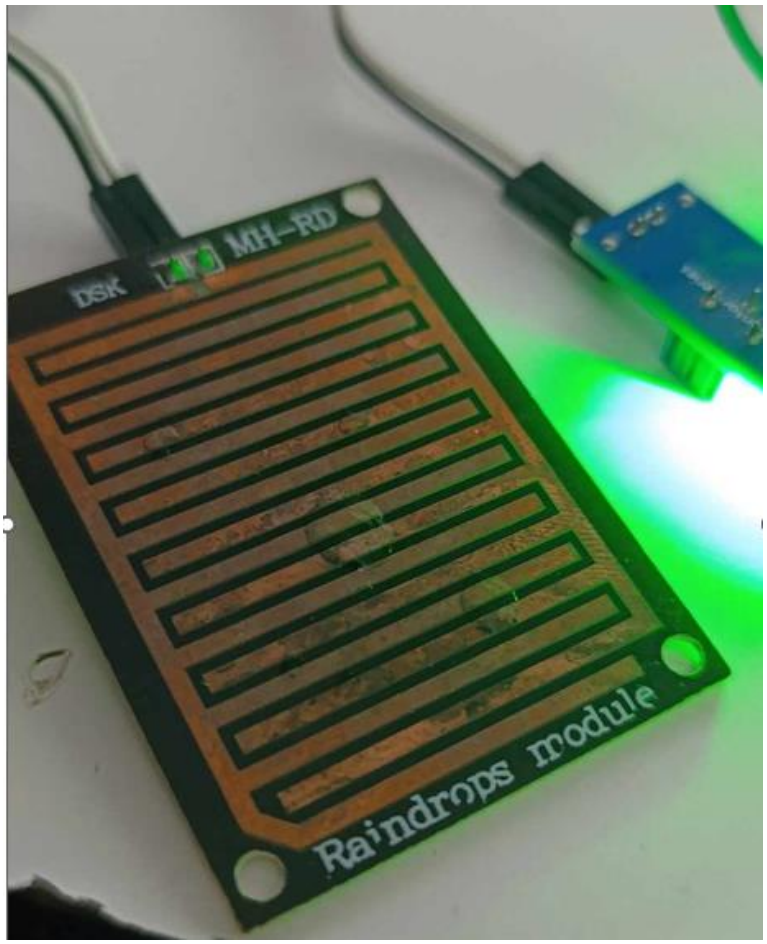
6. Tests et Validation

Des tests rigoureux ont été effectués pour assurer le bon fonctionnement du système. Les tests ont inclus :

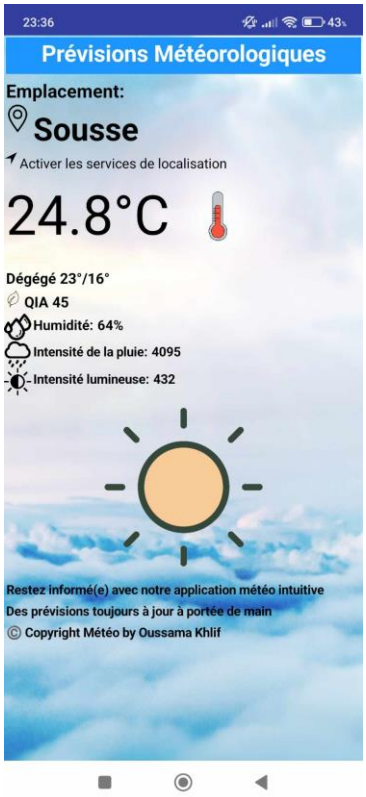
Vérification de l'exactitude des données collectées par les capteurs et test de la connectivité entre l'ESP32 et Firebase.



Simulation de conditions météorologiques pour tester les réponses du buzzer et de la LED.



Validation des fonctionnalités de l'application mobile.



Conclusion

La réalisation de ce projet a nécessité une collaboration efficace entre les membres de l'équipe et une gestion rigoureuse des tâches et des ressources. Le système IoT développé est capable de collecter et d'afficher en temps réel les données météorologiques, offrant une solution innovante et pratique pour diverses applications. Nous avons réussi à surmonter les défis techniques et à fournir un produit final fonctionnel, prêt à être utilisé par les utilisateurs finaux.