

ARCHITECTURE DES SYSTÈMES EMBARQUÉS



AI-DRIVEN EMBEDDED SYSTEM FOR PRECISE SOLAR TRACKING

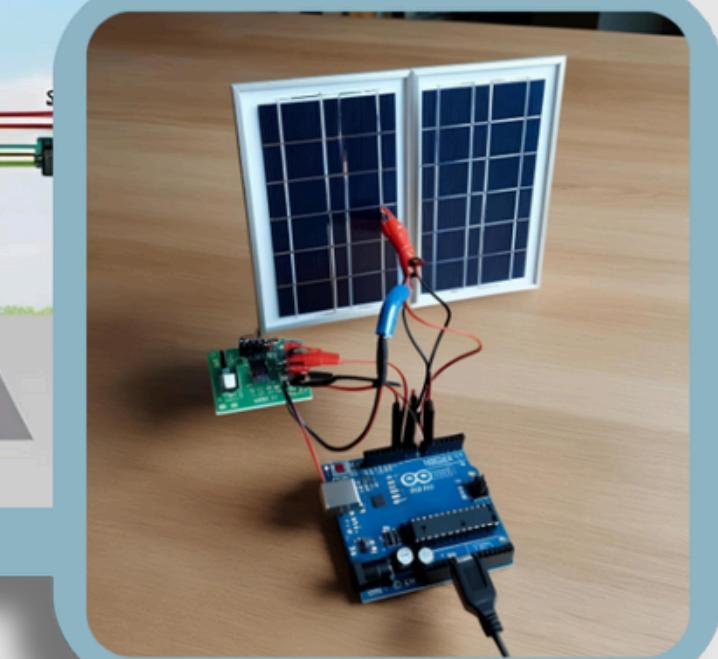
Presented by :

**KUNAKA DANIEL
SIMPORE TAOBATA
DIALLO Abdoul-Moumouni**

Professor : YOUNES WADIAI

SUN TRACKER ARDUINO SOLAR PANELS

Arduino-based system
for sun trackers



Sommaire



INTRODUCTION

02

PARTIE HARDWARE

03

PARTIE APPLICATION

11

POURQUOI L'IA ?

19

CONCLUSION

20

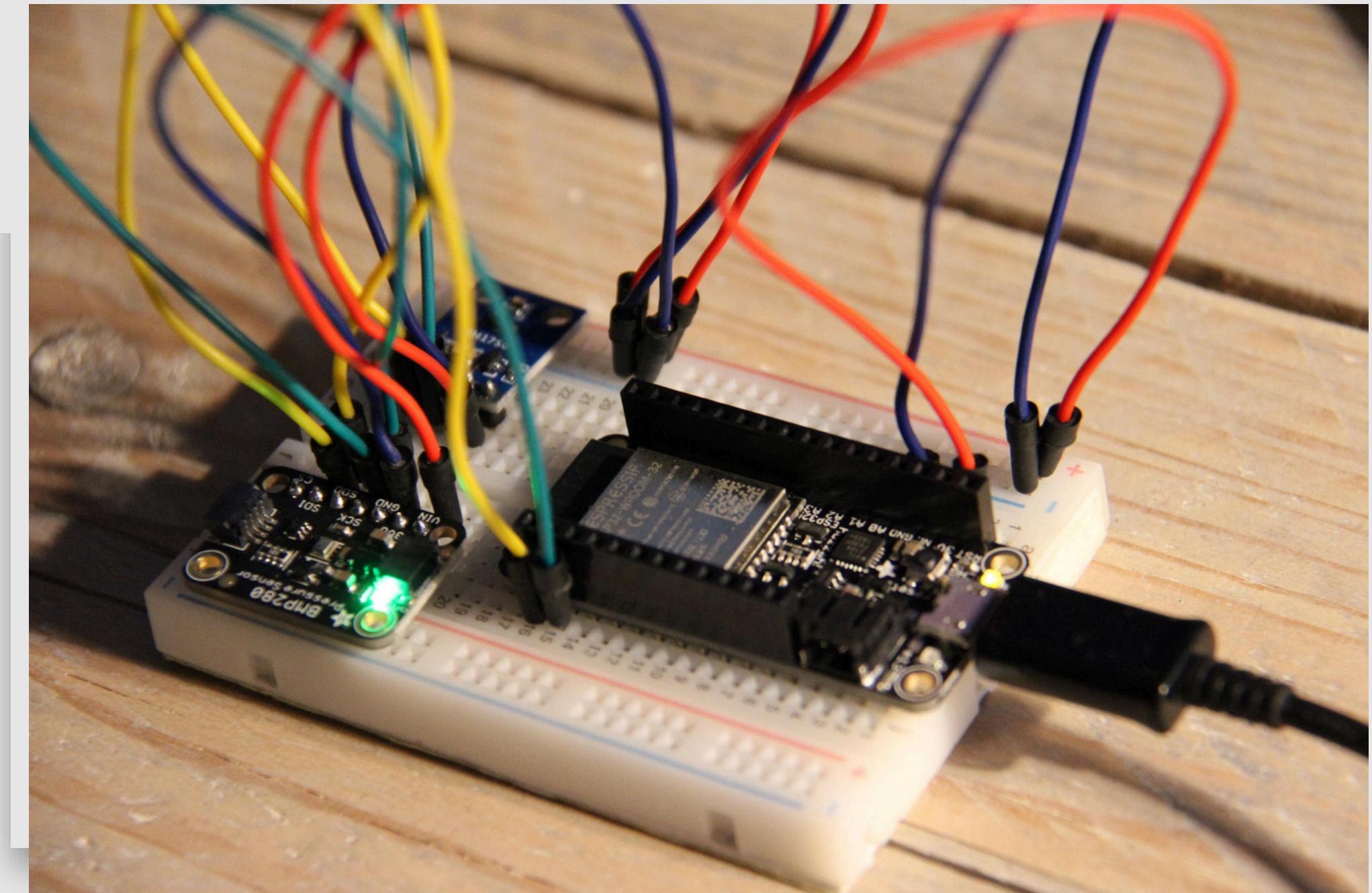
01

Introduction

Le système de suivi solaire intelligent optimise la production d'énergie photovoltaïque en combinant le suivi du soleil en temps réel, via des capteurs LDR et des servomoteurs, avec des prévisions météorologiques basées sur l'IA. Une application mobile permet de suivre en direct les valeurs des capteurs, les prévisions et la charge des panneaux. Ce système prédictif et connecté assure un rendement optimal, même en cas de météo variable.

Partie Hardware

Conception du système



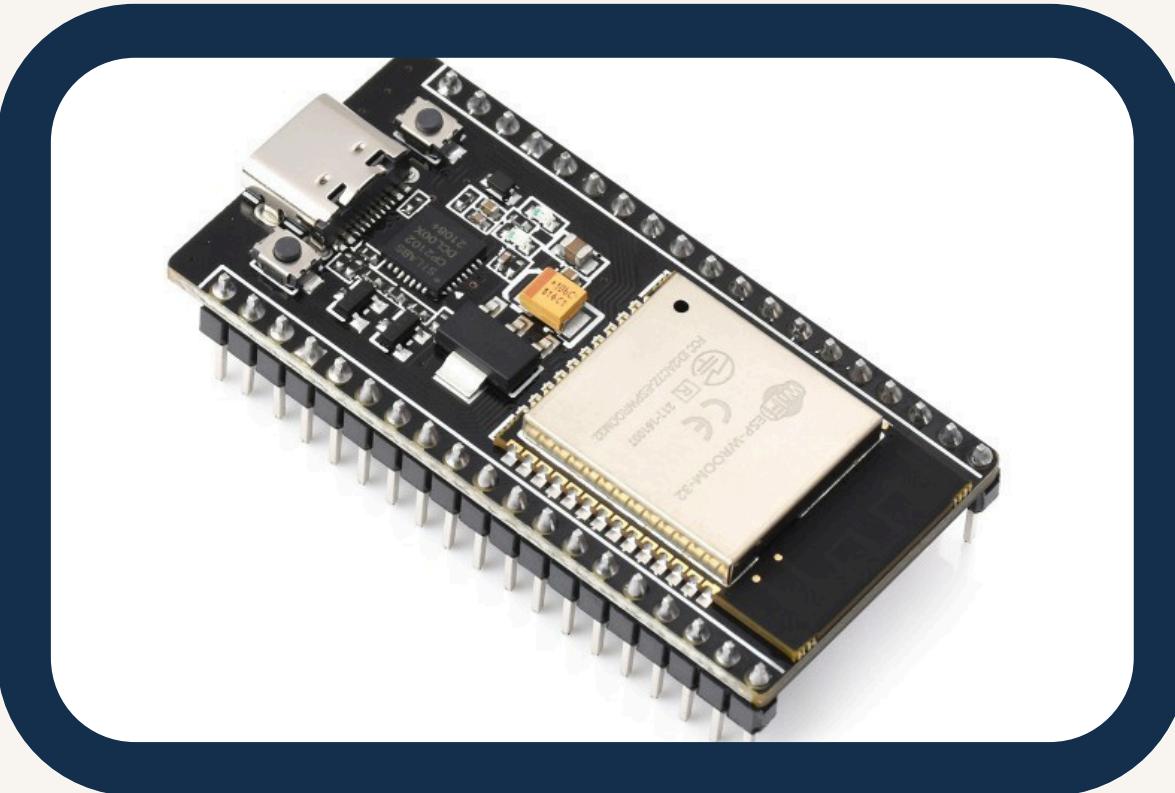
PRÉSENTATION DU SYSTÈME

Ce projet vise à concevoir un système intégré de suivi solaire optimisé, en utilisant des capteurs météorologiques et un modèle d'intelligence artificielle (IA) pour améliorer la gestion des panneaux solaires.

Le système repose sur un ESP32 pour collecter les données de capteurs ainsi que des valeurs liées à la charge et communiquer les informations à une plateforme cloud via Firebase.

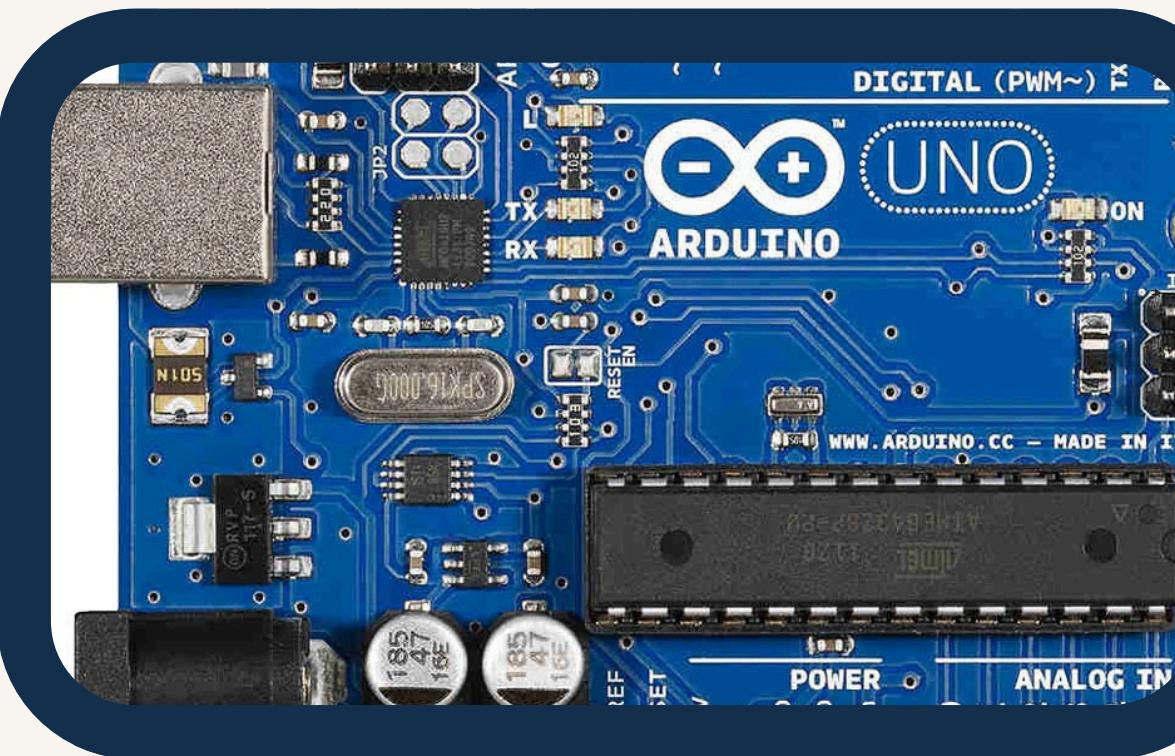
Le contrôle de l'orientation des panneaux solaires est effectué par une carte Arduino uno qui reçoit les données par communication série (RX-TX) à partir de l'ESP32.

Présentation des composants



ESP32 : NodeMCU 32S

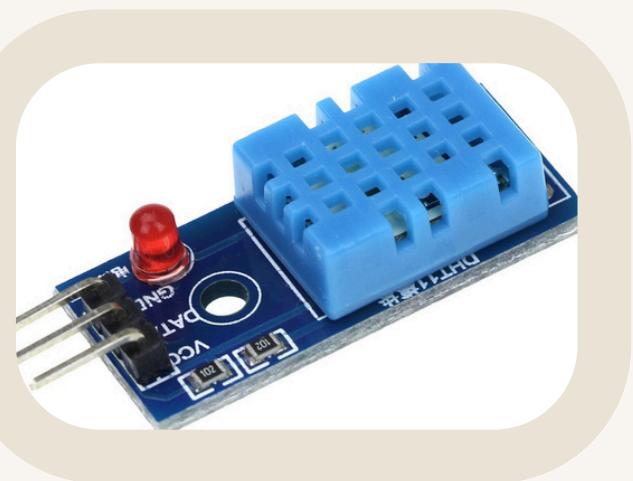
Carte de développement intégrant module WiFi et Bluetooth



Arduino Uno

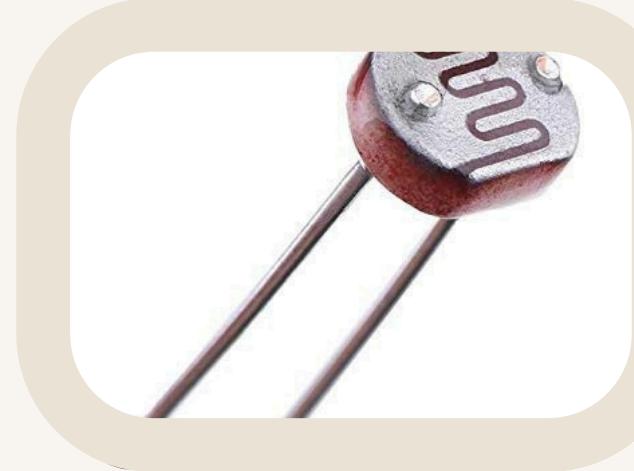
Carte de développement à microcontrôleur ATMega328

Présentation des composants



DHT11

Capteur de température et humidité



LDR

Photorésistance utilisée comme capteur de lumière



Potentiomètre

Résistance variable utilisée ici pour simuler la mesure de la puissance reçue.



Servomoteur

Composant à engrenage capable de mouvement de rotation utilisé pour contrôler le panneau

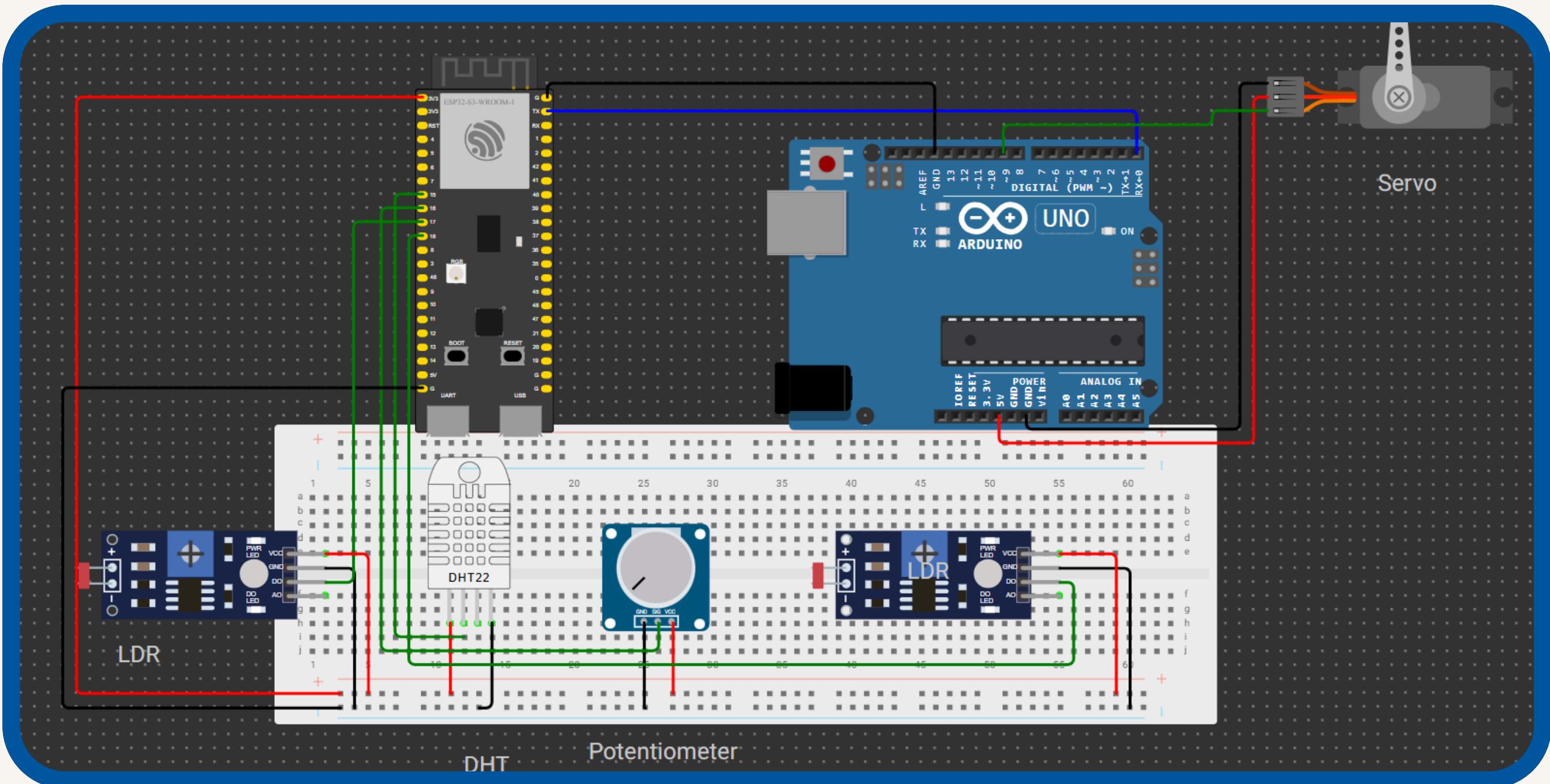


Module d'alimentation

Module connecté à une batterie ou au secteur pour alimenter le système de façon autonome

Conception du système

Circuit simplifié



✓ L'ESP lit les valeurs des capteurs

✓ L'ESP transfère les données vers Firebase

✓ L'ESP transfère par communication série les valeurs des LDRs à Arduino

✓ Arduino récupère les données du port série et contrôle le servo

Programmation de la carte ESP32

Importation des bibliothèques et fichiers nécessaires

Définition des identifiants de connexion, broches utilisées et des variables globales

```
#include <Arduino.h>
#include <WiFi.h>
#include <Firebase_ESP_Client.h>
#include <DHT.h>
#include "addons	TokenNameHelper.h"
#include "addons/RTDBHelper.h"

// === Wi-Fi credentials ===
#define WIFI_SSID "Orange_wifi_9CB8"
#define WIFI_PASSWORD "*****"
// === Firebase credentials ===
#define API_KEY
"*****"
#define DATABASE_URL
"....4-default.firebaseio.com"

FirebaseData fbdo;
FirebaseAuth auth;
FirebaseConfig config;
// === DHT sensor configuration ===
#define DHTPIN 26
#define DHTTYPE DHT11
DHT dht(DHTPIN, DHTTYPE);
// === Capteurs ===
#define LDR1 39
#define LDR2 36
const int potPin = 34;
// === Variables partagées ===
int r1 = 0, r2 = 0, potValue = 0;
float temperature = 0.0, humidity =
0.0, lux_moyenne = 0.0;

unsigned long startTime;
SemaphoreHandle_t dataMutex

void TaskLectureCapteurs(void *
parameter) {
    for(;;) {
        int ldr1 = analogRead(LDR1);
```

```
        float hum = dht.readHumidity();

        xSemaphoreTake(dataMutex,
        portMAX_DELAY);
        r1 = ldr1;
        r2 = ldr2;
        potValue = pot;
        lux_moyenne = lux;
        temperature = temp;
        humidity = hum;
        xSemaphoreGive(dataMutex);

        vTaskDelay(pdMS_TO_TICKS(500));
    }
}

void TaskUART(void * parameter) {
    for(;;) {
        xSemaphoreTake(dataMutex,
        portMAX_DELAY);
        int ldr1 = r1, ldr2 = r2;
        xSemaphoreGive(dataMutex);

        vTaskDelay(pdMS_TO_TICKS(500));
    }
}

void TaskFirebase(void * parameter) {
    for(;;) {
        unsigned long duration =
        (millis() - startTime) /
        1000;

        xSemaphoreTake(dataMutex,
        portMAX_DELAY);
        int pot = potValue;
        float lux = lux_moyenne, temp =
        temperature, hum =
        humidity;
        xSemaphoreGive(dataMutex);
```

```
Firebase RTDB.setFloat(&fbdo,
"Luminosite/Moyenne", lux);
Firebase RTDB.setInt(&fbdo,
"Charge/Puissance", pot);

Firebase RTDB.setFloat(&fbdo,
"DHT11/Temperature", temp);
Firebase RTDB.setFloat(&fbdo,
"DHT11/Humidite", hum);
vTaskDelay(pdMS_TO_TICKS(5000));
}

void setup() {
    serialZ.begin(9600, SERIAL_8N1, -1,
    17);
    dht.begin();
    WiFi.begin(WIFI_SSID,
    WIFI_PASSWORD);
    WiFi.setAutoReconnect(true);
    config.api_key = API_KEY;
    config.database_url = DATABASE_URL;

    config.token_status_callback =
    tokenStatusCallback;
    Firebase.begin(&config, &auth);
    ...
    startTime = millis();
    dataMutex =
    xSemaphoreCreateMutex();

    xTaskCreatePinnedToCore(TaskLectureCapteurs,
    "LectureCapteurs", 4096, NULL,
    1, NULL, 1);
    xTaskCreatePinnedToCore(TaskUART,
    "UART", 2048, NULL, 1, NULL,
    0);
    xTaskCreatePinnedToCore(TaskFirebase,
    "Firebase", 8192, NULL, 1,
```

Ecriture sur le port série des valeurs des LDRs

Initialisation des différents éléments et connexion au Wifi et àFirebase ensuite

Appel des tâches créées précédemment

Création de tâches FreeRTOS

Programmation de la carte Arduino

Importation des bibliothèques nécessaires

Définition broches utilisées et des variables globales

Contrôle du servomoteur en fonction des valeurs des capteurs lues sur le port série

```
#include <Servo.h>
#include <SoftwareSerial.h>

SoftwareSerial Serial1(2, 3); // RX, TX
Servo sg90;
int servoPin = 9;
int position = 115;
int seuil = 150;
int zoneMorte = 10;

void setup() {
  Serial.begin(9600);
  Serial1.begin(9600);
  sg90.attach(servoPin);
  sg90.write(position); // Position initiale
}

void loop() {
  if (Serial1.available()) {
    String data = Serial1.readStringUntil('\n');
    data.trim();

    int sep = data.indexOf(',');
    if (sep > 0 && sep < data.length() - 1) {
      int r1 = data.substring(0, sep).toInt();
      int r2 = data.substring(sep + 1).toInt();

      if (r1 == 0 && r2 == 0) return;
      int diff = abs(r1 - r2);

      if (diff > seuil + zoneMorte) {
        if (r1 > r2) position -= 2;
        else position += 2;

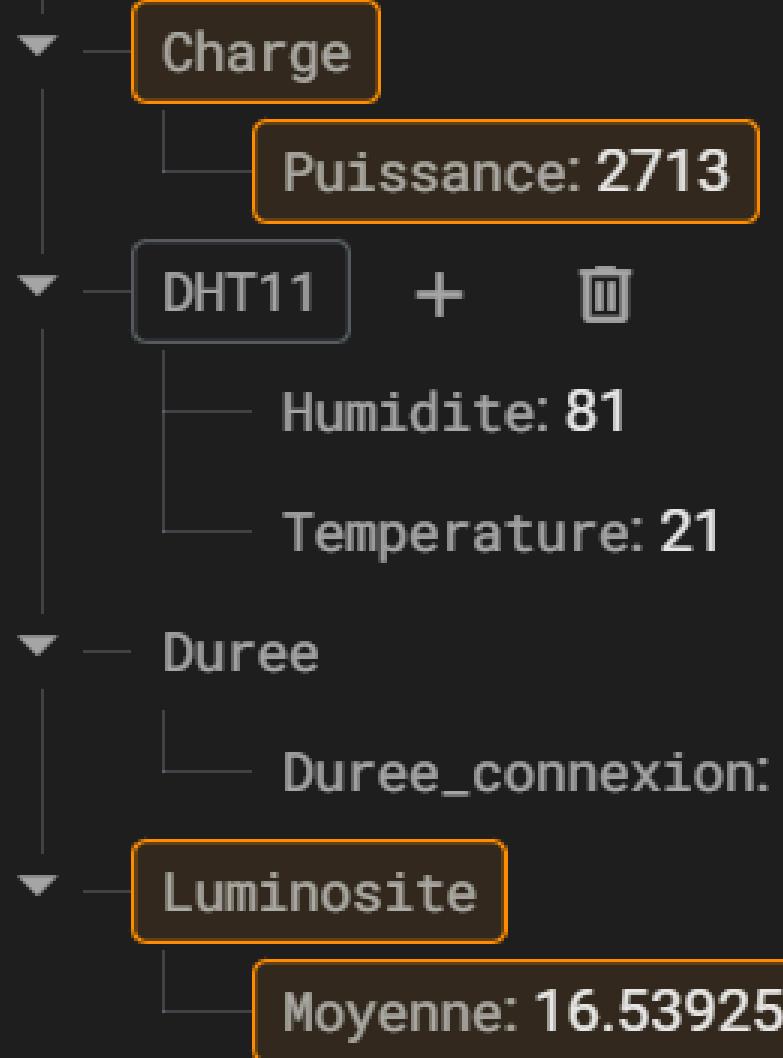
        position = constrain(position, 75, 145);
        sg90.write(position);
      } else { }
    }
  }
}
```

Conception du système

Aperçu dans Firebase

🔗 <https://esp32-c4304-default.firebaseio.com>

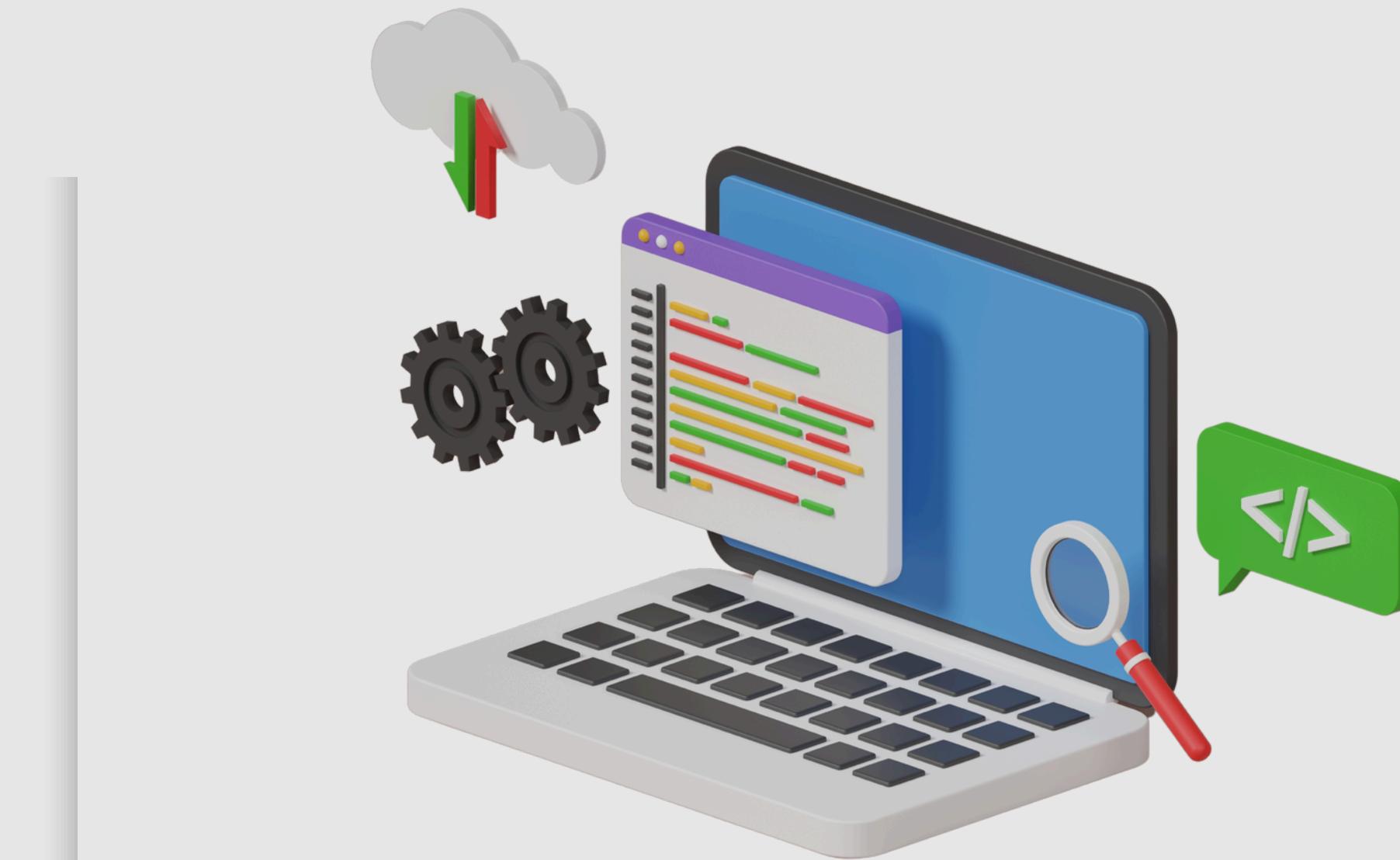
https://esp32-c4304-default-rtdb.firebaseio.com/



Partie Application

FLUTTER

SUN TRACK APP

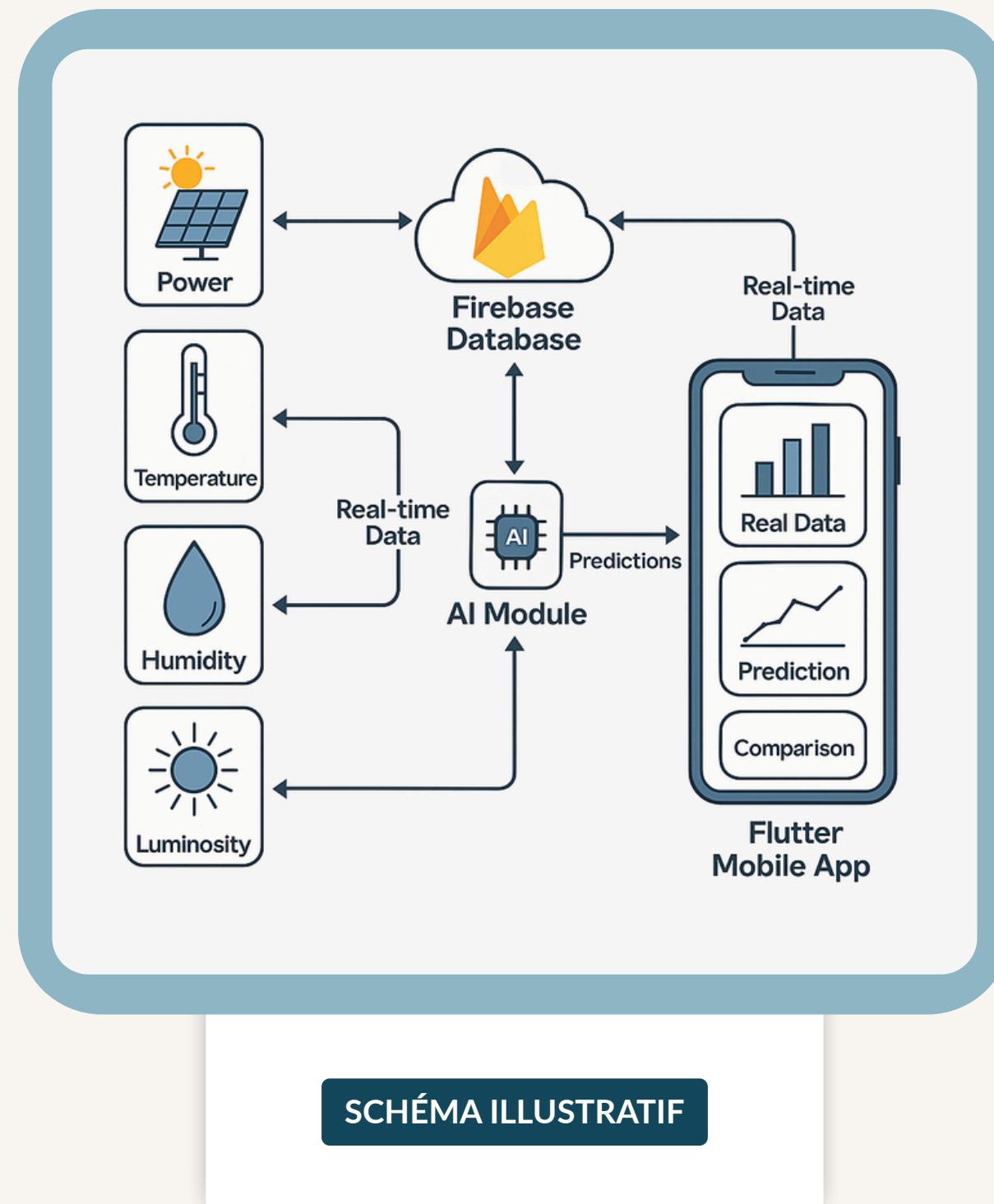


ARCHITECTURE ET TECHNOLOGIES

Application Connectée et Dynamique

TECHNOLOGIES UTILISÉES

- ✓ **Flutter & Dart**
Développement multi-plateforme.
- ✓ **Firebase Database**
Stockage et récupération des données.
- ✓ **Machine Learning**
Prédictions basées sur les tendances solaires.



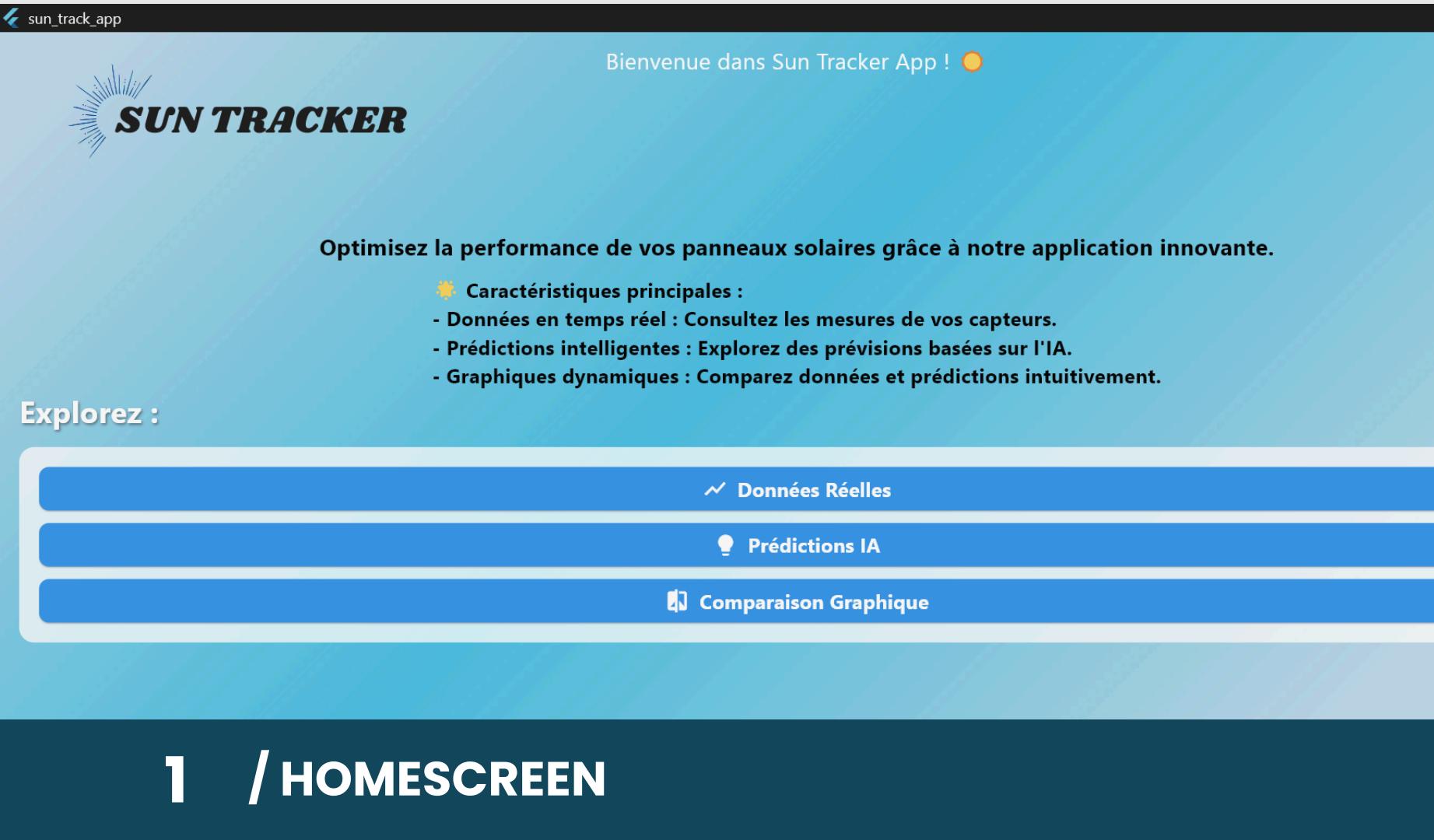
ARCHITECTURE BACKEND

- ✓ Firebase récupère les données en temps réel.
- ✓ L'application affiche et traite ces données pour analyse et visualisation.

Connexion et Navigation entre les Écrans

HomeScreen

- Écran d'accueil avec présentation du projet et accès aux différentes fonctionnalités.
- Permet la navigation vers les autres écrans via des boutons interactifs.



1 / HOMESCREEN

The RealDataScreen displays the following data:

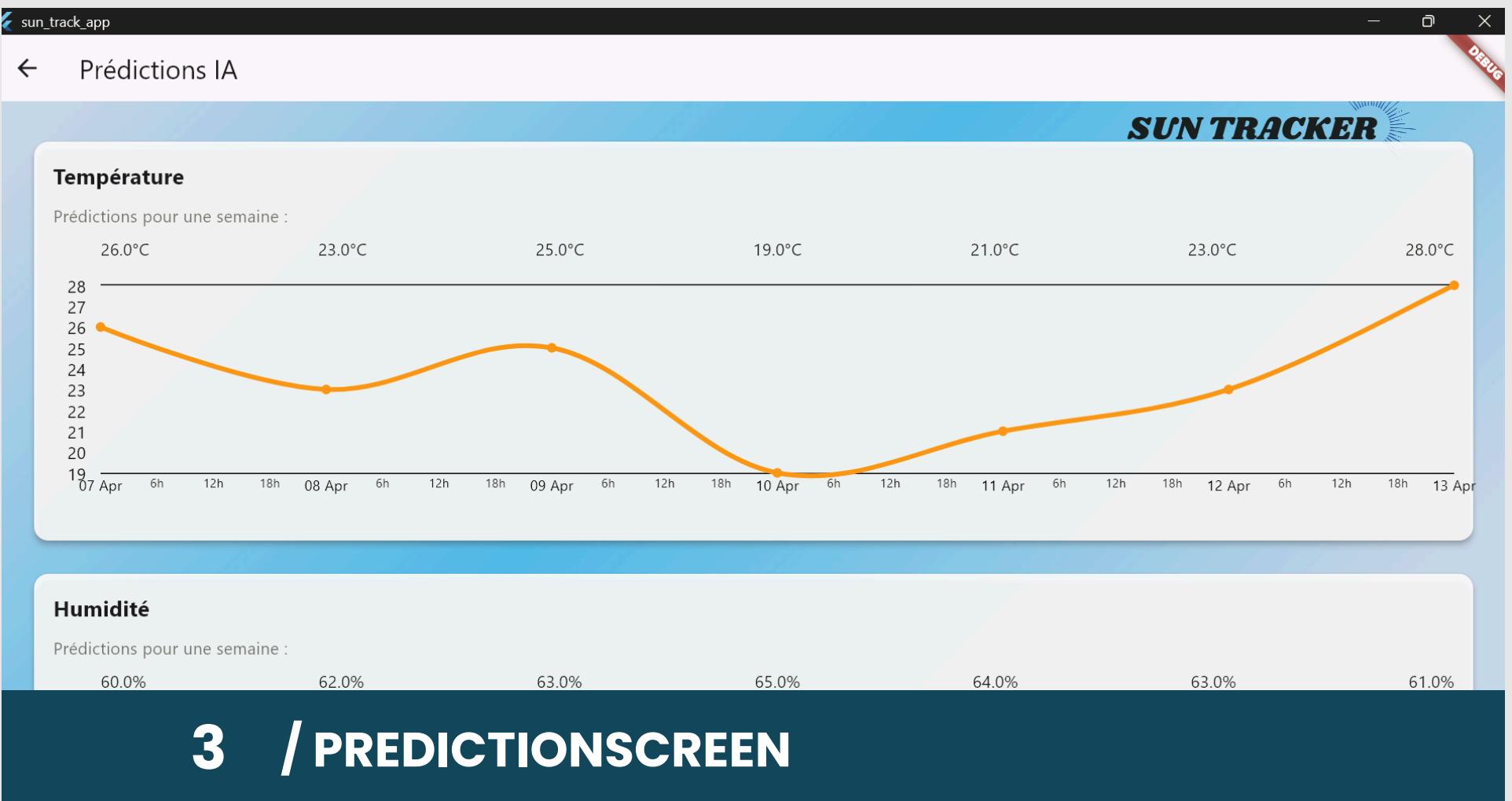
Donnée	Valeur
Température	23.9°C
Humidité	60%
Luminosité	596 Lux
Puissance Solaire	323 W
Temps de Branchement	75 heures
Énergie Récoltée	556 kWh

Visual elements include a battery icon, a sun icon, and two images of solar panels.

2 / REALDATASCREEN

RealDataScreen

- Récupère les données en temps réel depuis Firebase.
- Affiche la puissance, l'énergie récoltée, le temps de charge, la luminosité, température, humidité.



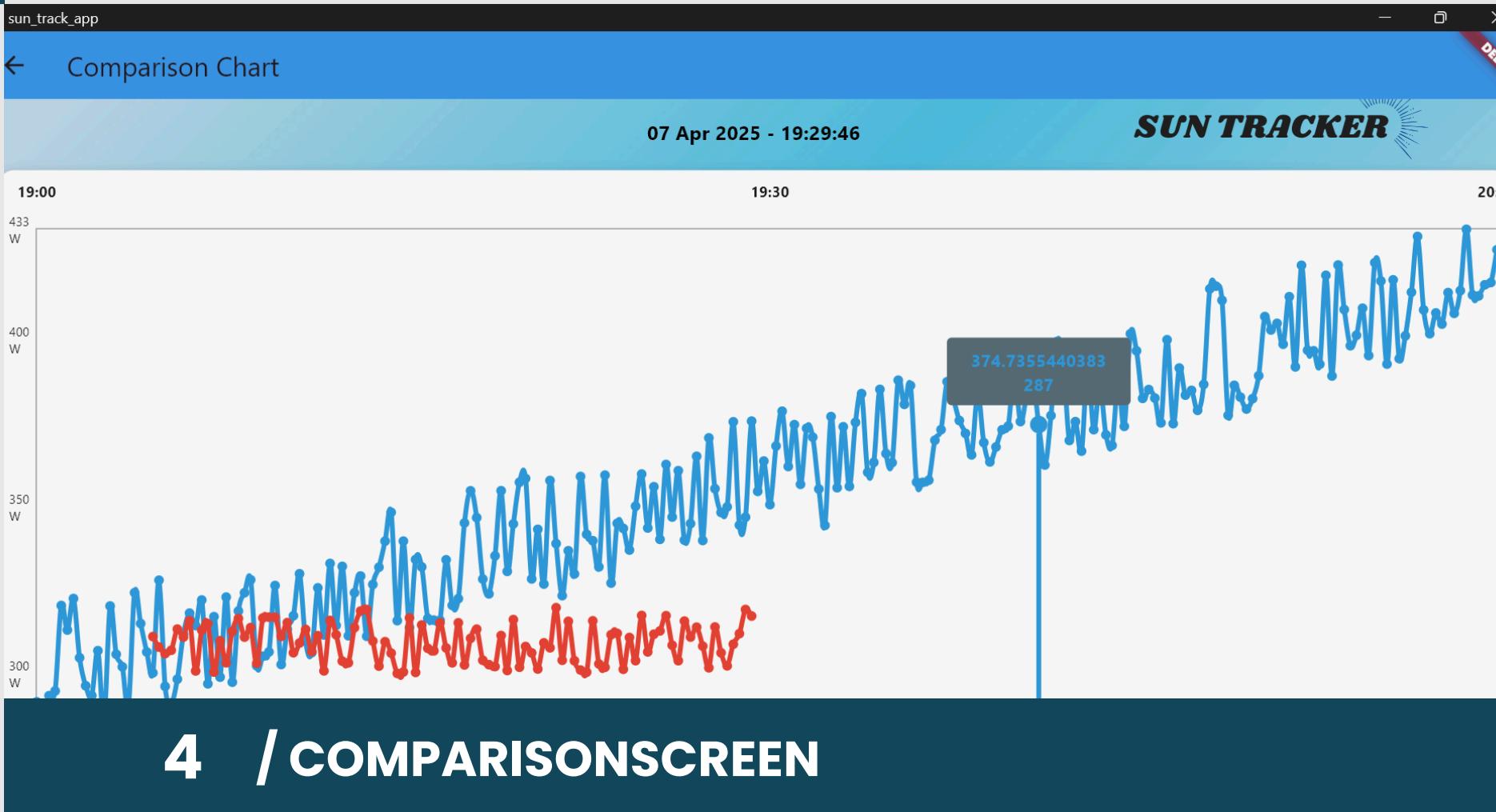
3 / PREDICTIONSCREEN

PredictionScreen

- Affiche les prédictions IA basées sur l'historique et les tendances.
- Modèle Machine Learning utilisé pour calculer les estimations futures.
- Connecté aux données stockées sur Firebase et au moteur de prédiction.

ComparisonScreen

- Compare les mesures réelles et les prévisions en affichant des graphiques dynamiques.
- Aide à ajuster le modèle IA en fonction des écarts constatés.



4 / COMPARISONSCREEN

Écran des Données Réelles

Données affichées :

- ✓ Puissance actuelle (W)
- ✓ Temps de chargement (s)
- ✓ Énergie récoltée (Wh)
- ✓ Température (°C)
- ✓ Luminosité (Lux)
- ✓ Humidité (%)



Écran des Prédictions IA

Visualisation des tendances :



Prédictions pour les prochaines heures/jours.



Graphiques dynamiques pour comprendre l'évolution.



Modèle IA utilisé :
Algorithme basé sur l'historique des données.

Écran Comparatif

Comparaison entre :

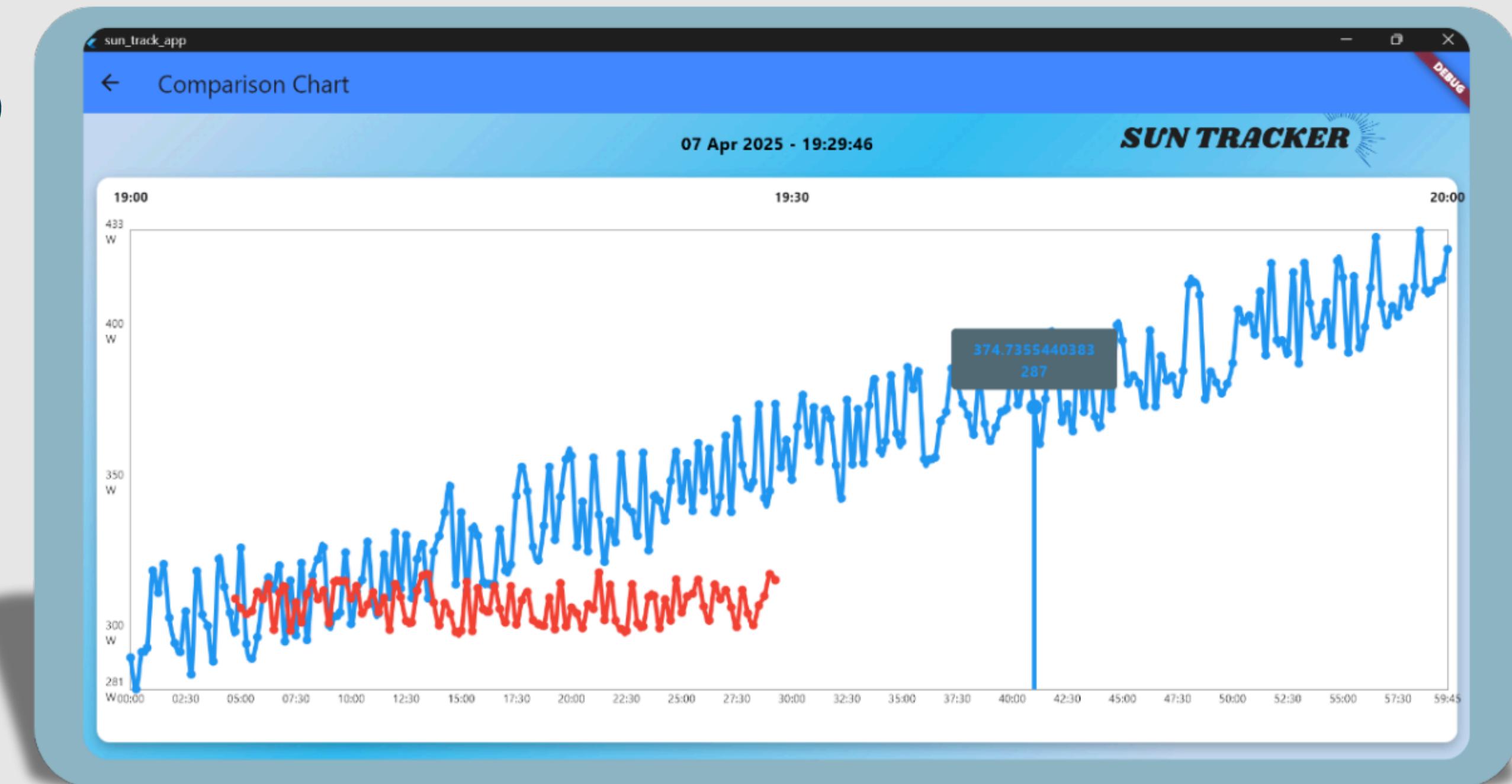
*Données réelles (capteurs)
vs Prédiction IA.*

*Évolution de la production
solaire.*

Objectif :

*Ajuster les paramètres en
fonction des erreurs de
prédiction.*

Affiner le modèle IA



Interface Optimisée pour l'Utilisateur

ASPECTS UX/UI :

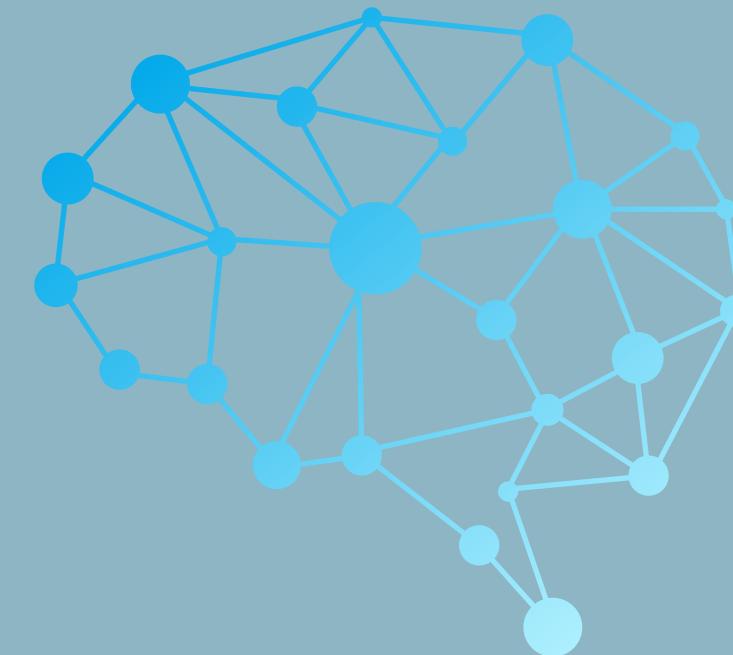
- Design minimaliste & intuitif → Navigation fluide.
- Mises à jour instantanées avec Firebase.
- Graphiques colorés pour visualiser facilement les données.

Explorez :

↗ Données Réelles

💡 Prédictions IA

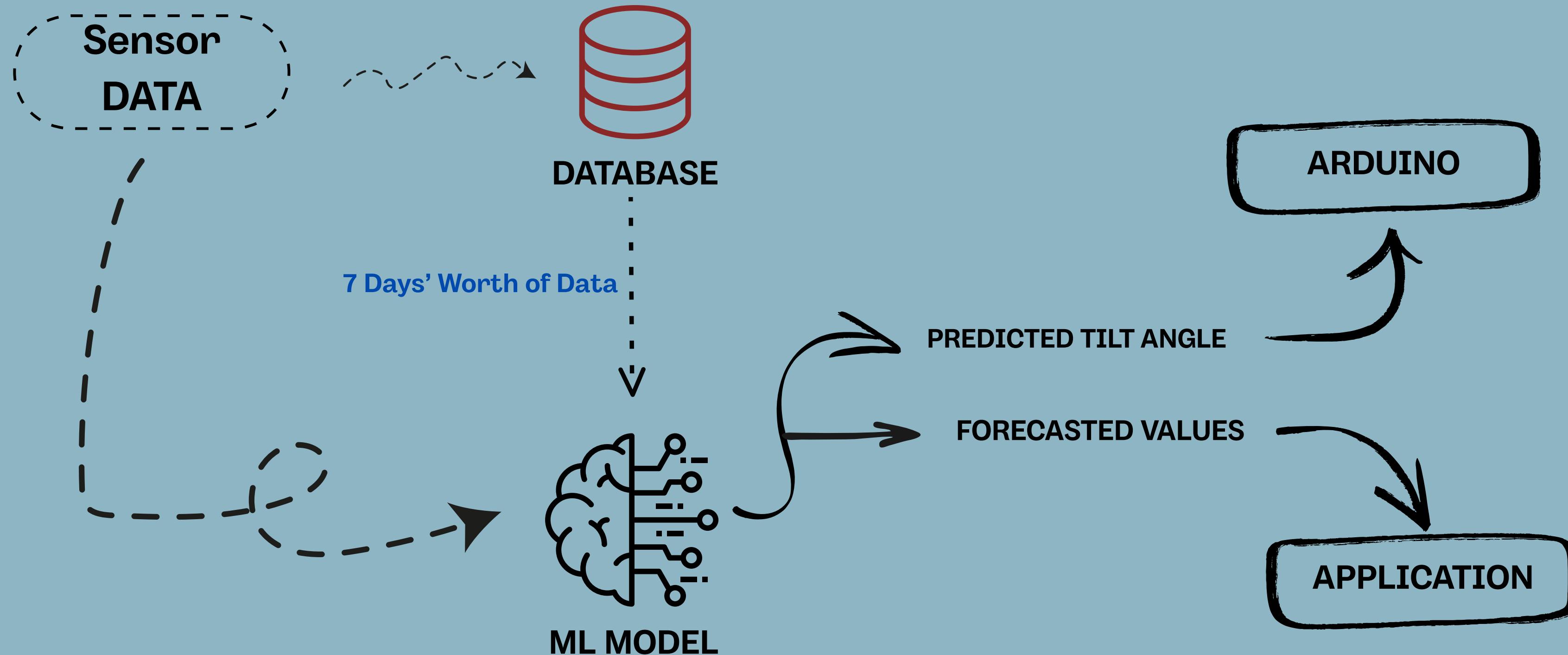
📊 Comparaison Graphique



WHY AI ?

“Smart Sun, Smarter Panel”

System Flowchart



S I M U L A T I O N

