Université Euro Méditerranéenne Fès
Euro Mediterranean University of Fez

Ecole d'Ingénierie Digitale et d'Intelligence Artificielle
(EIDIA)

Filière : 2ème année classes préparatoires Integrées

Semestre : 4

Module : Electronique embarquée

Thème :

Conception et Réalisation d'une Maison Intelligente Équipée d'une Station Météorologique

(Rapport de la station météorologique)

Soutenu le .. /…/24,

Encadré par :

Pr. A. SLIMANI

Préparé par l'étudiant :

- M. Gebli Achraf

Année Universitaire : 2023-2024

Année Universitaire : 2023-2024

1. Objectif du projet:

Notre projet consiste à créer une station météorologique intelligente contrôlée par un Arduino UNO, permettant une gestion optimisée d'une maison intelligente en fonction des conditions météorologiques en temps réel. La station collecte des données précises sur la température, l'humidité, la pression atmosphérique, la vitesse et la direction du vent, ainsi que les précipitations, grâce à des capteurs de haute précision. Ces informations sont transmises sans fil au système de gestion de la maison, qui ajuste automatiquement divers dispositifs domestiques tels que le chauffage, la climatisation. Cette intégration assure non seulement un confort accru pour les occupants, mais aussi une efficacité énergétique optimale. La station est également équipée d'un panneau de contrôle avec affichage en temps réel et d'une interface utilisateur conviviale pour une surveillance facile des conditions météorologiques et des ajustements manuels si nécessaire.

2. Introduction:

Dans le cadre de notre projet de fin de module en électronique embarquée, nous avons choisi de développer une station météorologique automatique connectée à une maison intelligente. Ce projet innovant combine les domaines de la météorologie, de l'Internet des objets (IoT) et de l'électronique embarquée pour améliorer le confort et l'efficacité énergétique des habitations modernes.

3. Simulation:

3.1- Composants :

Arduino UNO :

L'**Arduino UNO** est le cerveau du système, chargé de contrôler tous les composants électroniques de la station, pour afficher toutes les données nécessaires

LCD :

L'**écran LCD** est utilisé pour afficher les données à l'utilisateur, simplement et facilement pour lire et comprendre.

Pour communiquer avec l'afficheur **LCD**, nous avons utilisé les broches **I2C**, en connectant la broche **SCL** à la broche analogique A4 de l'Arduino et la broche **SDA** à la broche analogique A5 Nous avons employé la bibliothèque **"LiquidCrystal_I2C.h"** pour faciliter cette communication.

capteurs à effet hall :

•A3144:

est un interrupteur qui s'allume / éteint en présence d'un champ magnétique. Si aucun champ magnétique n'est présent, la ligne de signal du capteur est élevée (3,5 V). Si un champ magnétique est présenté au capteur, la ligne de signal est faible

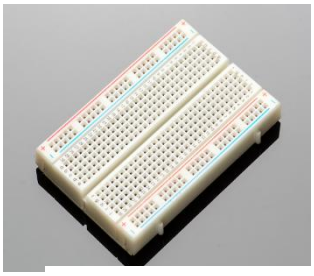
KY-024:

est un composant électronique utilisé pour mesurer les champs magnétiques. Il se compose d'un matériau semi-conducteur entouré d'un composant électronique est utilisé pour mesurer les champs magnétiques.

Breadboard :

Le **breadboard** permet de créer rapidement des prototypes en connectant les composants électroniques sans nécessiter de soudure. Cela permet aux concepteurs de tester et de valider rapidement leur concept avant de procéder à une implémentation plus permanente.Le breadboard permet d'organiser proprement les

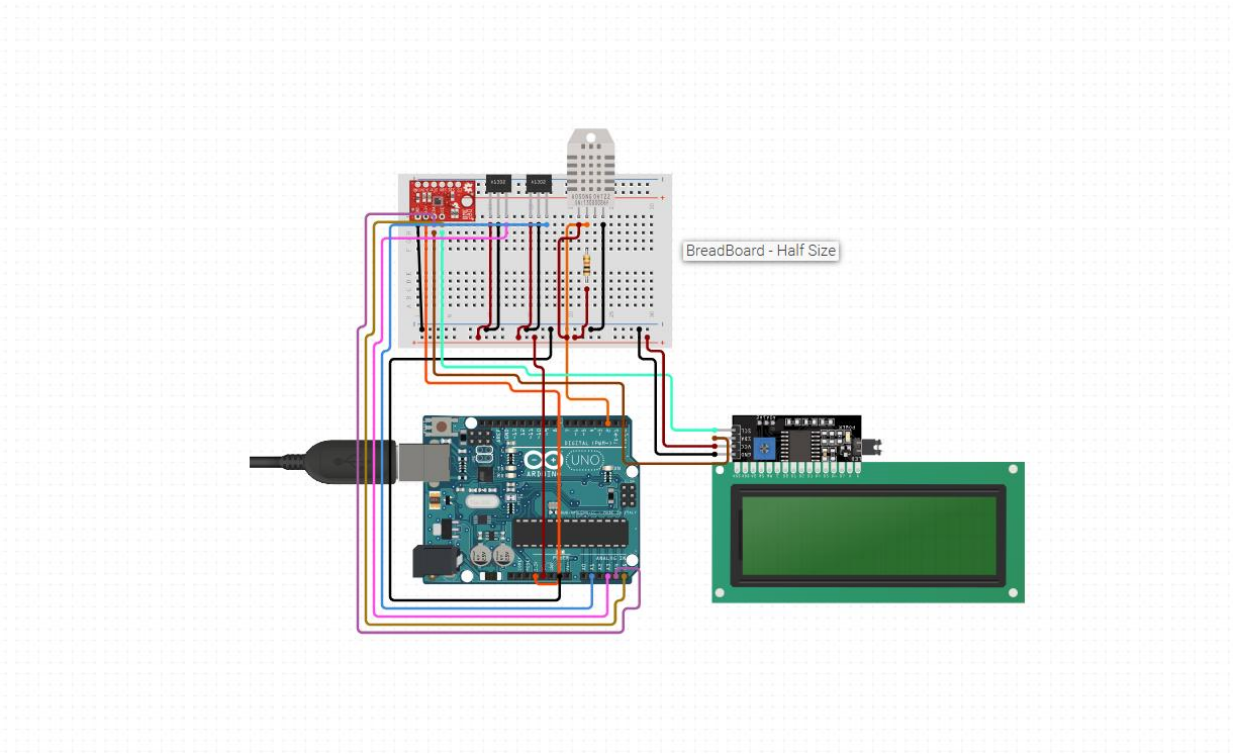
connexions entre les composants, ce qui facilite le suivi et le débogage du circuit. Les lignes de connexion du breadboard sont généralement disposées de manière logique, ce qui rend le câblage plus ordonné.



Resistance:

Les résistances sont utilisées en série avec les LEDs pour limiter le courant qui circule à travers elles. Cela garantit que le courant traversant la LED reste dans sa plage de fonctionnement sûre, prolongeant ainsi sa durée de vie et évitant tout dommage.

3.2- Montage:



3.3- Code:

code_final_meteo_withlcd.ino

```ino
#include <Wire.h>
#include <LiquidCrystal_I2C.h>
#include "DHT.h"

// Constants for the hall effect sensor (wind speed)
const int hallPin = 2; // Hall effect sensor connected to digital pin 2
const float magnetDistance = 0.10; // Distance per revolution in meters (adjust as needed)
volatile unsigned long startTime = 0; // Time when a magnet is detected
volatile unsigned long endTime = 0; // Time when the next magnet is detected
volatile boolean newWindData = false; // Flag to indicate new wind data
volatile int magnetCounter = 0; // Counter for magnet detections

// Constants for the rain gauge
const float mmPerPulse = 0.173; // Value of rain in mm for each movement of the bucket
float mmTotali = 0;
int sensore = 0;
int statoPrecedente = 0;

// Constants for the DHT sensor
#define DHTPIN 3 // Changed to avoid conflict with hallPin
#define DHTTYPE DHT22
DHT dht(DHTPIN, DHTTYPE);

// LCD initialization
LiquidCrystal_I2C lcd(0x27, 20, 4);
```

code_final_meteo_withlcd.ino

```ino

// Function declarations
void detectMagnet();
float calculateWindSpeed();
void displayWindSpeed(float windSpeed);
void displayRainfall(float rainfall);
void displayDHTData(float humidity, float tempC, float tempF, float heatIndexC, float heatIndexF);

void setup() {
  // Setup for hall effect sensor (wind speed)
  Serial.begin(9600); // Start serial communication at 9600 baud
  pinMode(hallPin, INPUT_PULLUP); // Set hallPin as input with pull-up resistor
  attachInterrupt(digitalPinToInterrupt(hallPin), detectMagnet, RISING); // Attach interrupt to hallPin
  Serial.println("Setup completed. Waiting for magnet detection...");

  // Setup for rain gauge
  pinMode(9, INPUT);
  lcd.init();
  lcd.backlight();
  lcd.setCursor(6, 0);
  lcd.print("HRAF");
  lcd.setCursor(3, 2);
  lcd.print("Weather Station");
  delay(1000);
  lcd.clear();
```

code_final_meteo_withlcd.ino

```ino
  // Setup for DHT sensor
  dht.begin();
}

void loop() {
  // Loop for hall effect sensor (wind speed)
  if (newWindData) { // If new wind data is available
    float windSpeed = calculateWindSpeed(); // Calculate wind speed
    Serial.print("Wind Speed: ");
    Serial.print(windSpeed);
    Serial.println(" m/s");
    newWindData = false; // Reset the flag
    Serial.println("Waiting for next magnet detection...");
    displayWindSpeed(windSpeed); // Display wind speed on LCD
  }

  // Loop for rain gauge
  sensore = digitalRead(9);
  if (sensore != statoPrecedente) {
    mmTotali = mmTotali + mmPerPulse;
  }
  statoPrecedente = sensore;
  displayRainfall(mmTotali); // Display rainfall on LCD

  // Loop for DHT sensor
```

Output

code_final_meteo_withlcd.ino

```
  delay(2000);
  float h = dht.readHumidity();
  float t = dht.readTemperature();
  float f = dht.readTemperature(true);

  if (isnan(h) || isnan(t) || isnan(f)) {
    Serial.println(F("Failed to read from DHT sensor!"));
    return;
  }

  float hif = dht.computeHeatIndex(f, h);
  float hic = dht.computeHeatIndex(t, h, false);

  Serial.print(F("Humidity: "));
  Serial.print(h);
  Serial.print(F("%  Temperature: "));
  Serial.print(t);
  Serial.print(F("°C "));
  Serial.print(f);
  Serial.print(F("°F  Heat index: "));
  Serial.print(hic);
  Serial.print(F("°C "));
  Serial.print(hif);
  Serial.println(F("°F"));

```

code_final_meteo_withlcd.ino

```
  displayDHTData(h, t, f, hic, hif); // Display DHT data on LCD
}

void detectMagnet() {
  magnetCounter++; // Increment magnet counter

  if (magnetCounter % 2 == 1) { // First detection in a pair
    startTime = millis();
    Serial.println("First magnet detected.");
  } else { // Second detection in a pair
    endTime = millis();
    Serial.println("Second magnet detected.");
    newWindData = true;
  }
}

float calculateWindSpeed() {
  float elapsedTime = (endTime - startTime) / 1000.0; // Convert milliseconds to seconds
  if (elapsedTime == 0) {
    Serial.println("Elapsed time is zero, avoiding division by zero.");
    return 0; // Avoid division by zero
  }
  float windSpeed = magnetDistance / elapsedTime; // Speed = Distance / Time
  startTime = endTime; // Reset the start time for the next calculation
  endTime = 0; // Reset the end time for the next calculation
```

code_final_meteo_withlcd.ino

```
  return windSpeed;
}

void displayWindSpeed(float windSpeed) {
  lcd.clear();
  lcd.setCursor(0, 0);
  lcd.print("Wind Speed:");
  lcd.setCursor(0, 1);
  lcd.print(windSpeed);
  lcd.print(" m/s");
  delay(5000); // Wait for 5 seconds
}

void displayRainfall(float rainfall) {
  lcd.clear();
  lcd.setCursor(0, 0);
  lcd.print("Pluviometer");
  lcd.setCursor(0, 1);
  lcd.print("Total precipitation:");
  lcd.setCursor(0, 2);
  lcd.print(rainfall);
  lcd.print(" mm");
  delay(5000); // Wait for 5 seconds
}

```

code_final_meteo_withlcd.ino

```
151
152 void displayDHTData(float humidity, float tempC, float tempF, float heatIndexC, float heatIndexF) {
153   lcd.clear();
154   lcd.setCursor(0, 0);
155   lcd.print("Humidity:");
156   lcd.setCursor(0, 1);
157   lcd.print(humidity);
158   lcd.print("%");
159
160   lcd.setCursor(0, 2);
161   lcd.print("Temp:");
162   lcd.setCursor(0, 3);
163   lcd.print(tempC);
164   lcd.print("C ");
165   lcd.print(tempF);
166   lcd.print("F");
167   delay(5000); // Wait for 5 seconds
168
169   lcd.clear();
170   lcd.setCursor(0, 0);
171   lcd.print("Heat Index:");
172   lcd.setCursor(0, 1);
173   lcd.print(heatIndexC);
174   lcd.print("C ");
175   lcd.print(heatIndexF);
```

code_final_meteo_withlcd.ino

```
159
160   lcd.setCursor(0, 2);
161   lcd.print("Temp:");
162   lcd.setCursor(0, 3);
163   lcd.print(tempC);
164   lcd.print("C ");
165   lcd.print(tempF);
166   lcd.print("F");
167   delay(5000); // Wait for 5 seconds
168
169   lcd.clear();
170   lcd.setCursor(0, 0);
171   lcd.print("Heat Index:");
172   lcd.setCursor(0, 1);
173   lcd.print(heatIndexC);
174   lcd.print("C ");
175   lcd.print(heatIndexF);
176   lcd.print("F");
177   lcd.setCursor(0, 2);
178   lcd.print("Pressure :");
179   lcd.setCursor(0, 3);
180   lcd.print("1020hPa");
181   delay(5000); // Wait for 5 seconds
182 }
183
```

---

4. Difficultés:

Lors de mon travail sur un projet Arduino consistant en une station météorologique, j'ai rencontré plusieurs difficultés. L'une des principales était la faible qualité des composants disponibles, souvent accompagnée de prix élevés. Cela a rendu le projet non seulement coûteux, mais aussi moins fiable, avec des composants qui ne fonctionnaient pas toujours comme prévu, nécessitant des remplacements fréquents et des ajustements constants.

---

5. Conclusion:

En somme, ce projet de station météorologique intelligente contrôlée par un Arduino UNO nous a permis d'explorer et de combiner divers domaines tels que la météorologie, l'Internet des objets (IoT) et l'électronique embarquée. Malgré les difficultés rencontrées, notamment la faible qualité des composants et les coûts élevés, nous avons réussi à mettre en place un système fonctionnel capable de collecter et de transmettre des données météorologiques en temps réel. Ces données permettent une gestion optimisée de la maison intelligente, améliorant ainsi le confort des occupants et l'efficacité énergétique. Ce projet a non seulement renforcé nos compétences techniques, mais a également souligné l'importance de la qualité des composants et de la planification budgétaire dans le développement de solutions technologiques complexes.