Mist Wise

- Nume: Bădîrcea Alexandru-Mădălin
- Grupa: 331CB

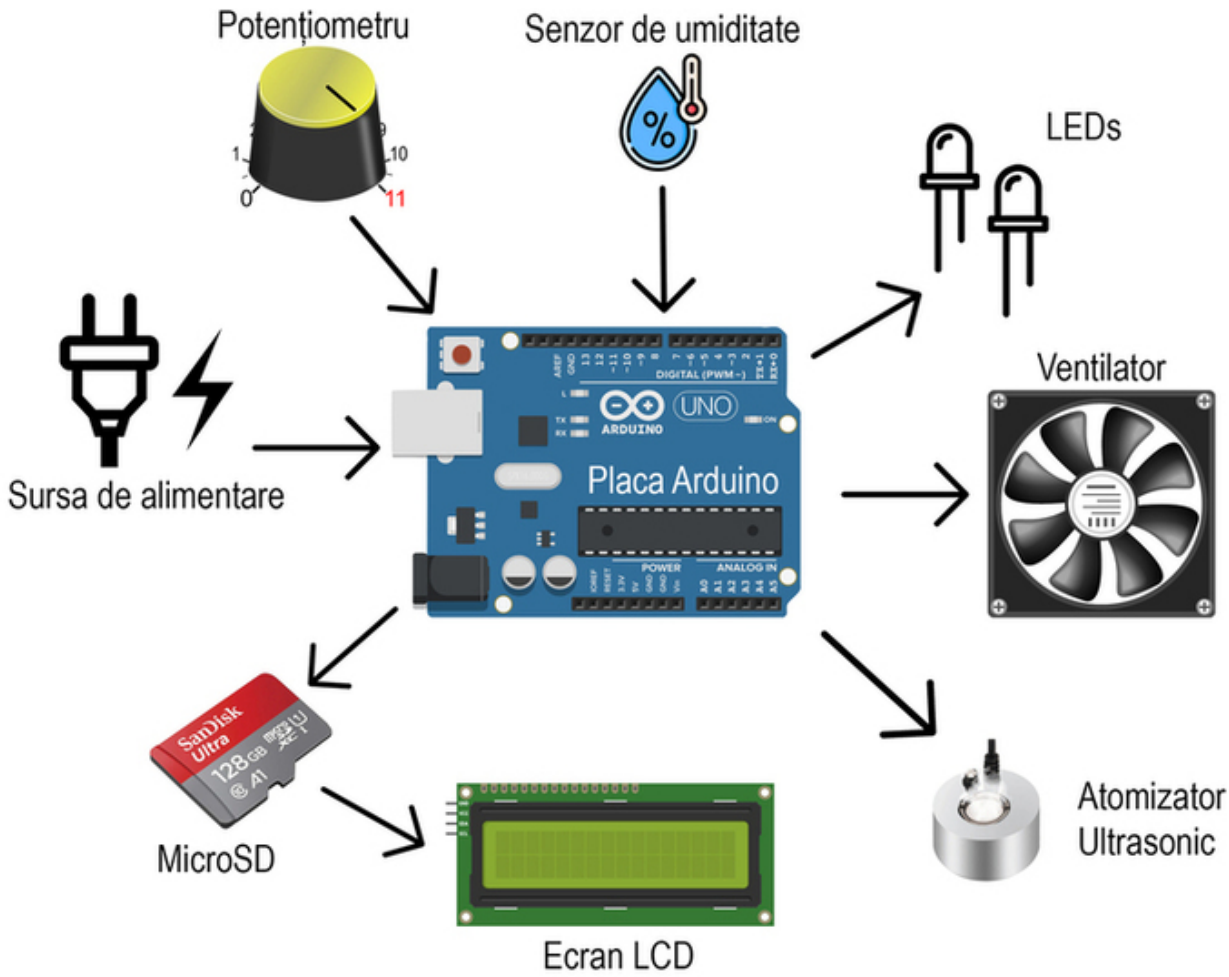
Introducere

Mist Wise este definiția confortului îmbinat cu tehnologia. Proiectul constă într-un umidificator automat care transformă atmosfera casei tale într-o oază de prospețime și relaxare, un dispozitiv inteligent care îți monitorizează și reglează automat nivelul de umiditate din cameră, creând un mediu optim pentru sănătate și bunăstare.

Ideea a luat naștere de la observația că nivelul de umiditate din aer poate avea un impact semnificativ asupra concentrării. Am câștigat recent la un concurs un umidifcator și l-am folosit în camera de cămin. După câteva zile, mi-am dat seama de beneficiile importante ale acestuia. Oricine și-ar dori un mediu cat mai confortabil pentru lucru și tot odata mai sanătos.

Descriere generală

Schema bloc



Placuța de dezvoltare va fi alimentată de la o sursă de alimentare de 24V, astfel că va fi necesar un modul de coborâre a tensiunii pentru a furniza 5V către componentele sensibile. Modulul de coborâre a tensiunii (LM2596) va transforma tensiunea de 24V în 5V, asigurând o alimentare stabilă și protejată pentru Arduino UNO și alte componente electronice.

Sistemul va colecta date de la doi senzori principali: un senzor de umiditate DHT22 și un potențiometru. Senzorul de umiditate va monitoriza constant nivelul de umiditate din cameră și va trimite aceste date către Arduino. Potențiometrul va permite utilizatorului să seteze manual nivelul de umiditate dorit. Arduino va interpreta valoarea ajustată de potențiometru și o va compara cu datele de la senzorul DHT22.

Valorile de umiditate măsurate vor fi stocate pe un card microSD, permițând astfel păstrarea unui istoric al nivelurilor de umiditate. Modulul microSD comunică cu Arduino prin protocolul SPI, asigurând transferul eficient și fiabil al datelor.

Când nivelul de umiditate din cameră scade sub pragul setat de utilizator, Arduino va activa atomizatorul ultrasonic prin intermediul unui MOSFET. Atomizatorul ultrasonic va transforma apa în vapori, umidificând aerul din cameră. Pentru a asigura o distribuție uniformă a vaporilor, un ventilator de 12V va fi activat simultan, controlat de Arduino prin semnale PWM.

Starea sistemului va fi semnalizată vizual prin intermediul a trei LED-uri:

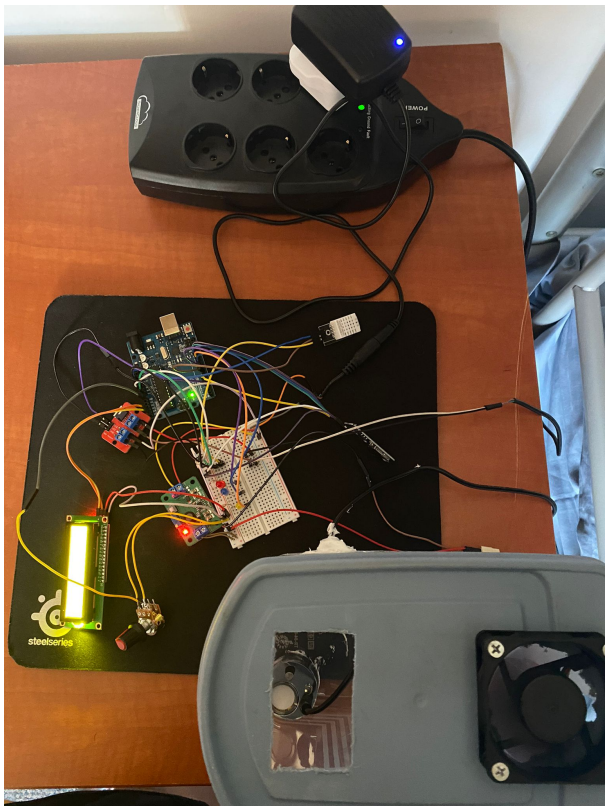
- LED albastru: indică faptul că nivelul de umiditate a depășit pragul dorit de utilizator, iar sistemul nu are nevoie să funcționeze pentru a adăuga mai multă umiditate.
- LED galben: utilizatorul poate vizualiza datele istorice despre nivelurile de umiditate din cameră pe ecranul LCD, fără ca atomizatorul sau ventilatorul să fie active.

- LED roșu: sistemul funcționează pentru a adăuga umiditate în cameră, activând atomizatorul ultrasonic și ventilatorul pentru a dispersa vapori de apă.

Pe ecranul LCD conectat la Arduino vor fi afișate diferite mesaje în funcție de starea sistemului. În timpul funcționării normale, ecranul va afișa nivelul curent de umiditate și un mesaj care indică îmbunătățirea umidității în cameră. Dacă sistemul nu este activ, ecranul va afișa istoricul nivelurilor de umiditate stocate pe cardul microSD, oferind utilizatorului o imagine de ansamblu asupra fluctuațiilor de umiditate din timp.

Hardware Design

- Arduino UNO (ATmega328p)
- Sursă de alimentare 24V
- Atomizator Ultrasonic
- Senzor de umiditate DHT22
- Card de memorie - microSD
- Ecran LCD
- Potențiometru 10kOhm
- Ventilator 12V
- Modul coborâre tensiune
- LEDs
- Rezistentori
- Modul slot microSD
- modul MOSFET



Pini Conectati

- **Pinul Analogic A0**: Conectat la potențiometru, permițând utilizatorului să seteze manual nivelul de umiditate dorit. Acesta traduce valoarea variabilă a potențiometrului într-un semnal pe care Arduino îl poate folosi pentru a ajusta funcționarea sistemului
- **Pinii Analogici A4 și A5**: Utilizați pentru comunicația I2C cu ecranul LCD. Pinul A4 (SDA) și pinul A5 (SCL) permit afișarea informațiilor în timp real despre starea sistemului și nivelurile de umiditate.
- **Pinul Digital 2**: Controlează MOSFET-ul care la rândul său controlează alimentarea atomizatorului ultrasonic (Fogger). Acest pin permite pornirea și oprirea atomizatorului pentru generarea vaporilor de apă.
- **Pinul Digital 4**: Folosit pentru modulul microSD, asigurând stocarea datelor de umiditate pe cardul de memorie. Acesta este esențial pentru păstrarea unui istoric al valorilor măsurate.
- **Pinul Digital 7**: Conectat la senzorul de umiditate și temperatură DHT22 pentru a citi datele de mediu. Acest pin este crucial pentru detectarea condițiilor de umiditate și temperatură din cameră.
- **Pinii Digitali 8, 9 și 10**: Conectați la LED-uri, care semnalizează starea sistemului (de exemplu, funcționare normală, erori). Aceste LED-uri oferă feedback vizual utilizatorului despre starea curentă a umidificatorului.
- **Pinul Digital 11 (MOSI)**: trimite date către cardul microSD.
- **Pinul Digital 12 (MISO)**: primește date de la cardul microSD.
- **Pinul Digital 13 (SCK)**: Serial Clock, folosit pentru sincronizarea transferului de date între Arduino și cardul microSD.

Alte conexiuni

- **Modul de coborâre a tensiunii (LM2596)**: este un regulator de tensiune step-down care transformă tensiunea de intrare de 24V la o tensiune de ieșire mai mică și stabilă de 5V. Acest lucru este esențial pentru a alimenta corect și în siguranță Arduino și alte componente electronice care funcționează la 5V.
- **Atomizator**: Atomizatorul ultrasonic folosește vibrații de înaltă frecvență pentru a transforma apa în vapori. Acest proces creează o ceață fină care contribuie la umidificarea aerului din cameră. Alimentat printr-un MOSFET și conectat la o sursă de alimentare de 24V.
- **Ventilator**: Ventilatorul ajută la dispersarea uniformă a vaporilor de apă generați de atomizatorul ultrasonic. Conectat la 5V.

Software Design

Pentru dezvoltarea proiectului am ales folosirea mediului de dezvoltare Arduino IDE.

Biblioteci folosite:

- DHT: Biblioteca DHT este folosită pentru citirea datelor de la senzorul de umiditate și temperatură DHT22.

- Wire: Biblioteca Wire este utilizată pentru comunicarea I2C, necesară pentru afișajul LCD.
- SPI și SD: Aceste biblioteci sunt esențiale pentru comunicarea cu cardul SD, permițând stocarea datelor de la senzor pentru analiza ulterioară.
- LiquidCrystal_I2C: Biblioteca LiquidCrystal_I2C simplifică interacțiunea cu afișajele LCD bazate pe I2C

Elementul de noutate al proiectului Mist Wise constă în integrarea armonioasă a mai multor funcționalități, precum monitorizarea și controlul automat al umidității, stocarea datelor istorice pe un card SD și afișarea informațiilor pe un LCD. Utilizând un senzor DHT22, un potențiometru și un atomizator ultrasonic controlat prin MOSFET, sistemul ajustează automat nivelul de umiditate pentru a menține un mediu optim. LED-urile de stare și interfața user-friendly oferă feedback vizual și acces facil la datele de umiditate. Acest proiect inovativ aduce un confort sporit și un mediu sănătos în orice cameră.

Implementare software:

- **global:** În acest cod, se definesc pinii la care sunt conectate diverse componente ale proiectului și se inițializează obiectele necesare pentru interacțiunea cu acestea. Se definește pinul 7 pentru senzorul de umiditate și temperatură DHT22 și se creează un obiect DHT pentru a facilita citirea datelor de la senzor. Pinii 8, 9 și 10 sunt alocați pentru controlul LED-urilor roșu, albastru și galben, iar pinul 2 este folosit pentru controlul MOSFET-ului, care activează atomizatorul ultrasonic. Se inițializează un obiect LiquidCrystal_I2C pentru controlul unui ecran LCD cu adresa 0x27, care are 16 coloane și 2 rânduri. Pinul 4 este setat pentru cardul SD, iar variabilele pot și dataFile sunt declarate pentru a stoca valoarea citită de la potențiometru și pentru gestionarea fișierului de date pe cardul SD.

```
#define DHTPIN 7      // Pinul la care este conectat senzorul
#define DHTTYPE DHT22 // AM2302 este echivalent cu DHT22
DHT dht(DHTPIN, DHTTYPE); // Inițializarea senzorului DHT

#define LED_RED 8 // Pinul la care este conectat LED-ul roșu
#define LED_BLUE 9 // Pinul la care este conectat LED-ul albastru
#define LED_YELLOW 10 // Pinul la care este conectat LED-ul galben

#define MOSFET_PIN 2 // Pinul la care este conectat MOSFET-ul

LiquidCrystal_I2C lcd(0x27, 16, 2); // Inițializarea LCD-ului

const int chipSelect = 4; // Pinul la care este conectat cardul SD
int pot; // Valoarea citită de la potențiometru
File dataFile; // Fișierul pentru scrierea datelor
```

- **setup:** În funcția `setup()`, se configurează și inițializează toate componentele necesare pentru funcționarea proiectului Mist Wise. Se inițializează comunicarea serială și senzorul DHT22, se configurează pinii pentru MOSFET și LED-uri ca ieșiri, și se inițializează cardul SD pentru stocarea datelor, verificând prezența acestuia și creând un fișier CSV cu antetul corespunzător. Se inițializează comunicația I2C pentru afișajul LCD și se configurează LCD-ul pentru a afișa datele, iar fișierul pe cardul SD este curățat pentru a începe cu date proaspete. Astfel, toate componentele hardware și software sunt pregătite pentru monitorizarea și controlul umidității.

```
void setup() {
  Serial.begin(9600);
  dht.begin();
  pinMode(MOSFET_PIN, OUTPUT);
  pinMode(LED_RED, OUTPUT);
  pinMode(LED_YELLOW, OUTPUT);
  pinMode(LED_BLUE, OUTPUT);
  digitalWrite(MOSFET_PIN, 0);

  // Initializarea cardului SD
  if(!SD.begin(chipSelect)) {
    Serial.println("Card failed, or not present");
    return;
  }
  Serial.println("Card initialized.");

  // Deschiderea fișierului pentru scriere
  dataFile = SD.open("datalog.csv", FILE_WRITE);
  if (dataFile) {
    dataFile.println("Potentiometer, Temperature, Humidity");
    dataFile.close();
  } else {
    Serial.println("Error opening datalog.csv");
  }

  // Inițializarea comunicării I2C
  Wire.begin();

  // Inițializarea LCD-ului
  lcd.init();
  lcd.begin(16, 2);
  lcd.backlight();

  // Curățarea conținutului fișierului
  clearFileContent("datalog.csv");
}
```

- **loop:** În funcția `loop()`, se citesc valorile de umiditate și temperatură de la senzorul DHT22, verificând validitatea acestora. Valorile sunt afișate pe monitorul serial și se citește valoarea de la potențiometru, care este rotunjită și folosită pentru a determina pragul de umiditate. Datele sunt scrise pe cardul SD. În funcție de umiditatea citită și pragul setat, se controlează MOSFET-ul și LED-urile pentru a activa sau dezactiva atomizatorul și a semnaliza starea sistemului. Dacă LED-ul galben este aprins, se calculează și afișează media valorilor istorice de umiditate și temperatură pe ecranul LCD.

```
void loop() {
  // Citirea umidității și a temperaturii
  float h = dht.readHumidity();
  float t = dht.readTemperature();

  // Verificarea dacă citirile sunt valide
```

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

  // Afișarea rezultatelor
  Serial.print("Umiditate: ");
  Serial.print(h);
  Serial.print(" %\t");
  Serial.print("Temperatura: ");
  Serial.print(t);
  Serial.println(" *C");

  pot = analogRead(A0);
  pot = round(pot / 200.0) * 200;
  int partNumber = pot / 200;
  Serial.print("Potentiometer: ");
  Serial.println(pot);
  Serial.println(partNumber);

  writeDataToFile("datalog.csv", partNumber, t, h);


  int humidityThreshold = 0;
  switch (partNumber) {
    case 1: humidityThreshold = 30; break;
    case 2: humidityThreshold = 40; break;
    case 3: humidityThreshold = 50; break;
    case 4: humidityThreshold = 60; break;
    case 5: humidityThreshold = 70; break;
    default: humidityThreshold = 0; break;
  }

   // Controlul MOSFET-ului și LED-urilor
  if (h < humidityThreshold) {
    digitalWrite(MOSFET_PIN, HIGH);  // Pornește MOSFET-ul
    digitalWrite(LED_RED, HIGH);     // Aprinde LED-ul roșu
    digitalWrite(LED_BLUE, LOW);     // Stinge LED-ul albastru
    digitalWrite(LED_YELLOW, LOW);   // Stinge LED-ul galben
  } else {
    digitalWrite(MOSFET_PIN, LOW);   // Oprește MOSFET-ul
    digitalWrite(LED_RED, LOW);      // Stinge LED-ul roșu
    digitalWrite(LED_BLUE, HIGH);    // Aprinde LED-ul albastru
    digitalWrite(LED_YELLOW, HIGH);  // Aprinde LED-ul galben
  }

  if (digitalRead(LED_YELLOW) == HIGH) {
    lcd.clear();
    float avgTemp = 0.0;
    float avgHum = 0.0;
    int numEntries = 0;
```

```
    calculateAverageFromFile("datalog.csv", avgTemp, avgHum, numEntries);

    lcd.setBacklight(HIGH);
    lcd.setCursor(0, 0);
    lcd.print("Humidity & Temp");
    lcd.setCursor(0, 1);
    lcd.print("HISTORY");
    delay(3000);
    lcd.setCursor(0, 0);
    lcd.print("Avg Temp:");
    lcd.print(avgTemp);
    lcd.print("*C");
    lcd.setCursor(0, 1);
    lcd.print("Avg Hum:");
    lcd.print(avgHum);
    lcd.print("%");
    delay(5000);  // Afișează media timp de 5 secunde
  } else {
    lcd.setBacklight(LOW);
    lcd.clear();
  }
}
```

Functii auxiliare:

- **clearFileContent** Șterge datele stocate pe cardul SD.

```
void clearFileContent(const char *fileName) {
  // Sterge fișierul dacă există
  if (SD.exists(fileName)) {
    SD.remove(fileName);
    Serial.print("File removed: ");
    Serial.println(fileName);
  } else {
    Serial.print("File does not exist: ");
    Serial.println(fileName);
  }

  // Creează un fișier nou
  File file = SD.open(fileName, FILE_WRITE);
  if (file) {
    file.println("Potentiometer, Temperature, Humidity"); // Adaugă antetul
    Serial.print("New file created: ");
    Serial.println(fileName);
    file.close();
  } else {
    Serial.print("Error creating file: ");
    Serial.println(fileName);
  }
}
```

- **writeDataToFile** Scrie datele obținute de la senzori pe cardul SD.

```
void writeDataToFile(const char *fileName, int pot, float temp, float hum) {
  // Deschide fișierul pentru scriere
  File file = SD.open(fileName, FILE_WRITE);
  if (file) {
    // Scrie datele în fișier
    file.print(pot);
    file.print(", ");
    file.print(temp);
    file.print(", ");
    file.println(hum);
    file.close();
  } else {
    Serial.print("Error opening file: ");
    Serial.println(fileName);
  }
}
```

- **calculateAverageFromFile** Deschide fișierul de pe cardul SD și calculează media valorilor de temperatură și umiditate stocate în acesta. Inițial, se verifică dacă fișierul poate fi deschis. Dacă reușește, se initializează variabilele pentru suma temperaturilor (tempSum), suma umidităților (humSum) și numărul de înregistrări (count). Se parcurg liniile fișierului, ignorând linia de antet, și se extrag valorile de temperatură și umiditate din fiecare linie, pe baza poziției virgulelor. Aceste valori sunt convertite la float și adăugate la sumele corespunzătoare, incrementând contorul de înregistrări. După ce toate liniile au fost procesate, dacă există cel puțin o înregistrare, se calculează media pentru temperatură și umiditate. În caz contrar, mediile sunt setate la zero. Dacă fișierul nu poate fi deschis, se afișează un mesaj de eroare

```
void calculateAverageFromFile(const char *fileName, float &avgTemp, float &
avgHum, int &numEntries) {
  File file = SD.open(fileName, FILE_READ);
  if (file) {
    float tempSum = 0.0; // Suma temperaturilor
    float humSum = 0.0; // Suma umidităților
    int count = 0; // Numărul de înregistrări
    bool isHeader = true;

    while (file.available()) {
      String line = file.readStringUntil('\n');
      if (isHeader) {
        isHeader = false; // Ignoră antetul
        continue;
      }

      int firstComma = line.indexOf(','); // Găsește prima virgulă
      int secondComma = line.indexOf(',', firstComma + 1);
      int thirdComma = line.indexOf(',', secondComma + 1);

      if (firstComma > 0 && secondComma > 0) { // Verifică dacă sunt date
valide
```

```
        float temp = line.substring(firstComma + 2, secondComma).toFloat();
        float hum = line.substring(secondComma + 2, line.length()).toFloat();

        tempSum += temp;
        humSum += hum;
        count++;
      }
    }

    file.close();

    if (count > 0) {
      avgTemp = tempSum / count;
      avgHum = humSum / count;
      numEntries = count;
    } else {
      avgTemp = 0.0;
      avgHum = 0.0;
      numEntries = 0;
    }
  } else {
    Serial.print("Error opening file for reading: ");
    Serial.println(fileName);
  }
}
```

Proiectul Mist Wise integrează trei noțiuni esențiale învățate în laboratoare: ADC, SPI și I2C. Citirea valorii de la potențiometru se realizează prin funcția analogRead(A0) utilizând ADC, care convertește semnalul analogic într-o valoare digitală pentru setarea pragurilor de umiditate. Protocolul SPI este folosit pentru comunicarea cu cardul SD, permițând stocarea și citirea datelor de umiditate și temperatură prin funcții precum SD.begin(chipSelect) și SD.open(), asigurând un istoric al valorilor măsurate. I2C este utilizat pentru comunicarea cu afișajul LCD, iar funcțiile Wire.begin(), lcd.init() și lcd.print() permit afișarea în timp real a stării sistemului și a valorilor de umiditate și temperatură. Aceste noțiuni fundamentale demonstrează aplicarea practică a cunoștințelor dobândite în laboratoare.

Rezultate Obţinute

https://www.youtube.com/watch?v=UZt1q6e2dPk

Concluzii

În final, proiectul a atins un stadiu care poate fi îmbunătățit cu mai multe funcționalități, însă aspectul general a fost realizat. Sunt foarte mulțumit de proiect, punctul forte al acestuia fiind independența

față de laptop și faptul că îl voi putea folosi și pe parcursul verii ce urmează. Am întâmpinat câteva probleme la lucrul cu 24V, reușind să ard o placuță, modulul de coborâre nefiind setat corespunzător. O altă problema ar fi, integritatea circuitului ar trebui îmbunătățită, jumperele nefiind suficient de sigure, fiind necesară o atenție sporită la transport. Ca și îmbunătățiri imediate ce pot fi aduse proiectului, ar fi un ventilator de 24V, pentru un randament mult mai bun al aparatului.

Lăsând la o parte micile probleme și provocări, experiența de lucru la proiectul Mist Wise a fost extrem de plăcută și unică. După ce am depășit dificultățile inițiale și am înțeles cum funcționează toate componentele, am abordat fiecare etapă cu entuziasm. Satisfacția de a vedea proiectul final funcționând și de a ști că am creat ceva util pentru viața de zi cu zi este îmbucurătoare. Acest proiect nu doar că mi-a îmbunătățit considerabil abilitățile tehnice, dar mi-a oferit și o profundă satisfacție personală și o experiență valoroasă de învățare.

Download

mist_wise.zip

Jurnal

02.05.2024 - Comandare piese
05.05.2024 - Creare pagină și adăugare introducere, descriere generală, componente folosite
08.05.2024 - Verificarea funcționalității pieselor
17.05.2024 - Completarea milestone-ului HARDWARE
19.05.2024 - Realizarea codului
24.05.2024 - Completarea milestone-ului SOFTWARE

Bibliografie/Resurse

- https://www.waveshare.com/wiki/DHT22_Temperature-Humidity_Sensor
- https://docs.arduino.cc/learn/electronics/lcd-displays/
- https://www.instructables.com/Micro-SD-Card-Tutorial/
- https://electropeak.com/learn/interfacing-irf520-mosfet-driver-module-switch-button-hcmodu0083-with-arduino/
- https://www.instructables.com/How-to-Use-DC-to-DC-Buck-Converter-LM2596/
- https://github.com/adafruit/DHT-sensor-library
- https://github.com/arduino-libraries/LiquidCrystal
- https://fritzing.org