Ministerul Educației și Cercetării
Universitatea Tehnică a Moldovei
Facultatea Calculatoare, Informatică și Microelectronică
Departamentul Ingineria Software și Automatică

Raport

Curs: Internetul lucrurilor

Tema: Sensori - Achizitia de informatii.

A elaborat: st.gr.SI-211 Chirița Stanislav
A verificat: Asist. Univ.Astafi Valentina

Chișinău 2024

Definirea problemei 1

Realizarea unei aplicații pentru MCU care va citi datele de la un senzor de umiditate.

Aplicația va efectua

1. Citirea de datelor de pe senzor;
2. Prelucrarea datelor în valori reale;
3. Medierea datelor prin 2 tehnici;
4. Afișarea datelor pe LCD și în serial.

Obiective

- configurarea aplicației pentru citirea datelor de pe senzor;
- crearea schemei și codului conform sarcinii propuse;
- înțelegerea sistemului creat și procesele ce se întâmplă.

Introducere

Un senzor este un dispozitiv izolat structural care conține unul sau mai mulți traductori de măsurare primari. Senzorul este proiectat pentru a genera un semnal de informație de măsurare într-o formă convenabilă pentru transmisie, conversie ulterioară, procesare și (sau) stocare, dar nu direct perceptibilă de către un observator.

Senzorul poate conține în plus traductoare de măsurare intermediare, precum și o măsură. Senzorul poate fi amplasat la o distanță considerabilă de dispozitivul care primește semnalele acestuia. Cu un raport normalizat al valorii cantității la ieșirea senzorului cu valoarea corespunzătoare a cantității de intrare, senzorul este un instrument de măsură.

Metode și materiale

Materiale Necesare:

- Arduino Board (de exemplu, Arduino Uno);
- breadboard și fire de conexiune;
- senzor de umiditate;
- editor Arduino IDE instalat pe un calculator.

Metoda de Implementare:

- conexiuni hardware:
    - conectarea senzorului de umiditate la un pin analogic al Arduino;
    - conectare LCD la pinii digitali ai Arduino.
- scrierea codului în Arduino IDE:
    - definirea variabilelor și pinilor corespunzători pentru senzorul de umiditate și LCD;

- implementare funcții pentru citirea temperaturii de la senzor și aplicarea tehnicilor de "sare și piper" și medie ponderată;
- afișarea valorilor pe LCD și a umidității curente în serial.

– testarea și debugging:

- încărcare cod pe Arduino folosind Arduino IDE;
- monitorizare comportament senzorului;
- identificare și rezolvare eventualelor erori în cod.

Rezultate

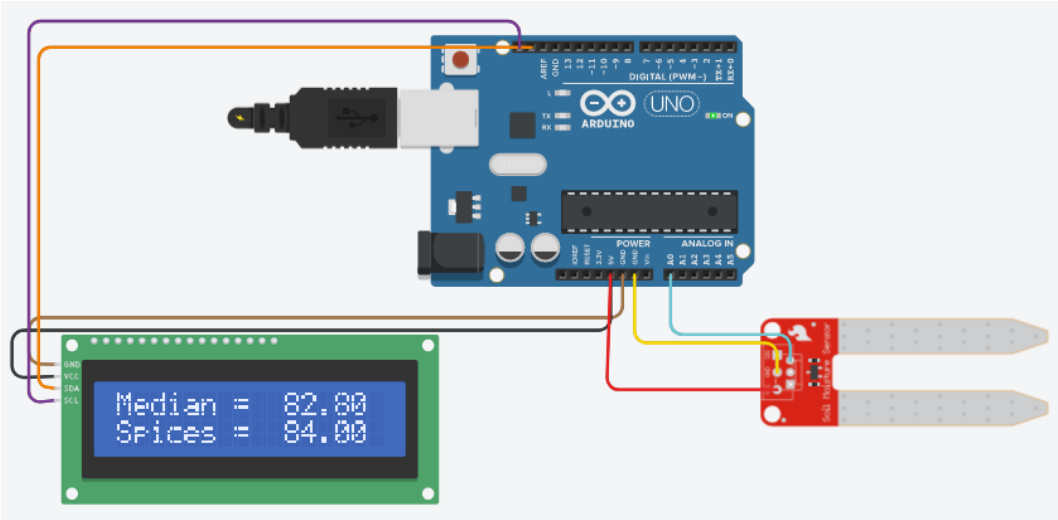

Figura 1 – Asamblarea circuitului pentru senzorul de umiditate

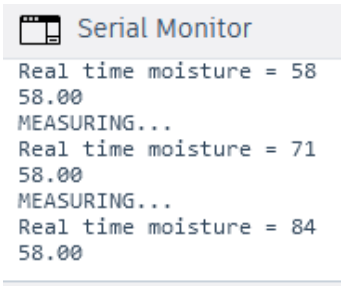

Figura 2 – Datele senzorului de umiditate

Concluzii

În timpul laboratorului, am studiat procesul de achiziție și prelucrare a datelor furnizate de un senzor de temperatură și umiditate. Am învățat cum să conectăm senzori la o placă Arduino și cum să citim valorile analogice pe care le furnizează, transformându-le în valori reale. De asemenea, am aplicat două tehnici de prelucrare a datelor: calculul mediei ponderate și utilizarea unui filtru de tip "sare și piper". Pentru a face proiectul mai practic și util, am integrat un afișaj LCD pentru a vizualiza în timp real datele prelucrate. Această adăugire a adus o dimensiune practică proiectului, permițându-ne să monitorizăm și să interpretăm evoluția datelor în timp real.

Anexa 1 – Codul sursă

```
#include <Adafruit_LiquidCrystal.h>


// Definirea pinului pentru senzor
#define sensor A0
// Inițializarea LCD-ului
Adafruit_LiquidCrystal lcd(0);
// Funcția pentru "sliding window" a datelor
void sw(float a[]) {
  for (int i = 0; i < 5; i++) {
    a[i] = a[i + 1];
  }
}
// Funcția pentru afișarea datelor pe LCD
void disp(float t1, float t2) {
  lcd.setCursor(10, 0);
  lcd.print(t1);
  lcd.setCursor(10, 1);
  lcd.print(t2);
}
```

```
// Funcţia pentru obţinerea umidităţii
float getMoisture() {
  //conversia
  float voltage = (analogRead(sensor) * (5.0 / 1023.0));
  int moisture = map(voltage * 100, 0, 428, 0, 100);
  Serial.print("Real time moisture = ");
  Serial.println(moisture);
  return moisture;
}
// Funcţia pentru calculul medianei ponderate
float weighted_median(float moists[]) {
  int weight = 2;
  float moisture = ((weight * moists[0] + weight * moists[1] +
                     weight * moists[2] + weight * moists[3] +
                     weight * moists[4]) / (weight * 5));
  return moisture;
}
// Funcţia pentru interschimbarea a două valori (utilizată pentru filtrul sare si piper)
void swap(float *p, float *q) {
  float t;
  t = *p;
  *p = *q;
  *q = t;
}
// Funcţia pentru filtrul "sare şi piper"
float salt_and_pepper(float moists[]) {
  int i, j, n = 5;

  float a[5];
```

```
  // Copierea valorilor într-un alt array
  for (i = 0; i < 5; i++) {
    a[i] = moists[i];
  }
  // Sortarea array-ului
  for (i = 0; i < n - 1; i++) {
    for (j = 0; j < n - i - 1; j++) {
      if (a[j] > a[j + 1])
        swap(&a[j], &a[j + 1]);
    }
  }
  int med_n = (n + 1) / 2 - 1; // Alegerea valorii din mijloc
  float moisture = a[med_n];
  Serial.println(moisture);
  return moisture;
}
void setup() {
  pinMode(sensor, INPUT);
  Serial.begin(9600);
  lcd.begin(16, 2);
  lcd.setCursor(0, 0);
  lcd.print("Median = ");
  lcd.setCursor(0, 1);
  lcd.print("Spices = ");
}
int i = 0;
void loop() {
  float moists[5];
  Serial.println("MEASURING...");
```

```
  // Adăugarea datelor la buffer
  if (i < 4) {
    moists[i] = getMoisture();
  }
  if (i >= 4) {
    sw(moists);
    moists[4] = getMoisture();
  }
  // Calcularea mediilor și afișarea lor
  float val1 = weighted_median(moists);
  float val2 = salt_and_pepper(moists);
  disp(val1, val2);
  delay(1000);
  i++;
}
```