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 2

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

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

Introducere

În prezent, termenii „senzor” și „detector” sunt folosiți interschimbabil pentru a se referi la un traductor de măsurare care îndeplinește funcțiile de a detecta o cantitate de intrare și de a genera un semnal de măsurare, deși termenul de „detector” se concentrează pe percepția cantității de intrare, și termenul „senzor” privind formarea și emiterea unui semnal de măsurare.

Senzorii multifuncționali pot percepe și converti mai multe cantități de intrare și, în plus față de funcția principală (detectarea unei cantități și generarea unui semnal de măsurare), efectuează o serie de funcții suplimentare, cum ar fi funcții de filtrare, procesare a semnalului etc.

Metode și materiale

Materiale Necesare:

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

Metoda de Implementare:

- conexiuni hardware:

- conectarea senzorului de temperatură la un pin analogic al Arduino;
- conectare LCD la pinii digitali ai Arduino.

– scrierea codului în Arduino IDE:

- definire variabilele și pinii corespunzători pentru senzorul de temperatură 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 temperaturii curente în serial.

– testarea și debugging:

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

Rezultate

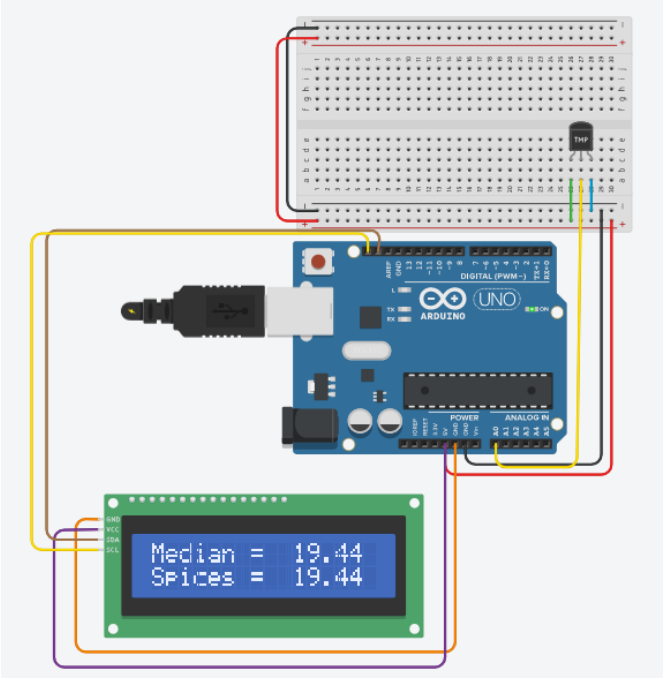

Figura 3 – Asamblarea circuitului pentru senzorul de temperatură

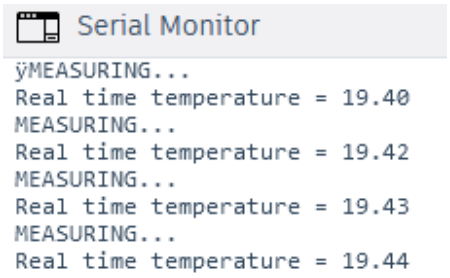

Figura 4 – Datele senzorului de temperatură

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 conectat la senzorul de temperatură

#define sensor A0

// Inițializarea bibliotecii pentru LCD

Adafruit_LiquidCrystal lcd(0);

// Funcția pentru deplasarea valorilor în array

void sw(float a[]) {

  for (int i = 0; i < 5; i++) {

    a[i] = a[i + 1];

  }

}

// Funcția pentru afișarea valorilor pe LCD

void disp(float t1, float t2) {
```

```
  lcd.setCursor(10, 0);
  lcd.print(t1);
  lcd.setCursor(10, 1);
  lcd.print(t2);
}
// Funcția pentru conversia de la voltaj la temperatură
float convertToTemperature(float voltage) {
  // Specificațiile senzorului
  float sensorSensitivity = 10.0;
  // Conversie de la voltaj la temperatură
  float temperature = (voltage - 500) / sensorSensitivity;
  return temperature;
}
// Funcția pentru achiziționarea temperaturii de la senzor
float getTemperature() {
  // Citirea valorii analogice de la senzor și conversia la voltaj
  int sensorValue = analogRead(sensor);
  float voltage = (sensorValue * 5.0) / 1023.0; // 5V este tensiunea de referință
  // Apelul funcției pentru conversia la temperatură
  float temperature = convertToTemperature(voltage);
  Serial.print("Real time temperature = ");
  Serial.println(temperature);
  return temperature;
}
// Funcția pentru calcularea mediei ponderate
float weighted_median(float temps[]) {
  int weight = 2;
  float temperature = ((weight * temps[0] + weight * temps[1] +
                        weight * temps[2] + weight * temps[3] +
```

```
                        weight * temps[4]) / (weight * 5));
  return temperature;
}
// Funcţia pentru interschimbarea a două valori
void swap(float *p, float *q) {
  float t;
  t = *p;
  *p = *q;
  *q = t;
}
// Funcţia pentru filtrul "sare și piper"
float salt_and_pepper(float temps[]) {
  int i, j, n = 5;
  float a[5];

  for (i = 0; i < 5; i++) {
    a[i] = temps[i];
  }
  for (i = 0; i < n - 1; i++) { // Sortarea array-ului
    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 temperature = a[med_n];
  return temperature;
}
void setup() {
```

```
  // Configurarea modului pinului conectat la senzor
  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 temps[5];

  Serial.println("MEASURING...");
  if (i < 4) {
    temps[i] = getTemperature();
  }
  if (i >= 4) {
    sw(temps);
    temps[4] = getTemperature();
  }
  float temp1 = weighted_median(temps);
  float temp2 = salt_and_pepper(temps);
  // Afișarea temperaturilor pe LCD
  disp(temp1, temp2);
  delay(2000);
  i++;
}
```