MINISTERUL EDUCAŢIEI ȘI CERCETĂRII AL REPUBLICII MOLDOVA
Universitatea Tehnică a Moldovei
Facultatea Calculatoare, Informatică și Microelectronică
Departamentul Ingineria Software și Automatică
Programul de studii: Tehnologia informației

RAPORT

Disciplina „IoT – Internetul Lucrurilor”
Tema: Condiționare Semnal

Student(ă): _____ **Vlașițchi Ștefan , TI-212**

Coordonator universitate: _____ **Lupan Cristian, asist.univ.**

Chișinău, 2024

Scopul: Sa se realizeze o aplicație in baza de MCU care va condiționa semnalul preluat de la sensor (vezi Lab 3.1) , si va afișa parametrul fizic la un terminal (LCD si/sau Serial).

Obiective:

1. Sa se achiziționeze semnalul de la senzor (vezi Lab 3.1);
2. Să se condiționeze semnalul implicând filtre digitale și alte metode;
3. Să se afișeze datele pe afișor LCD și / sau Serial.

Introducere

În contextul dezvoltării aplicațiilor IoT, condiționarea semnalelor preluate de la senzori reprezintă un pas esențial pentru obținerea unor date corecte și precise. Semnalele preluate de la senzori sunt supuse diferitelor surse de zgomot și fluctuații, iar utilizarea unor tehnici de filtrare digitală devine necesară pentru asigurarea unei analize corecte a datelor. În acest laborator, se va dezvolta o aplicație bazată pe un microcontroler (MCU) care va prelua semnalele de la un senzor ales și va aplica tehnici de filtrare, precum filtrul „Sare și piper" și filtrul de „Mediere ponderată", pentru a condiționa semnalul. Scopul principal este de a obține un parametru fizic valid, care va fi afișat pe un afișaj LCD sau pe monitorul serial.

Materiale utilizate

- Microcontroler (Arduino sau echivalent) cu suport pentru citirea senzorilor (digitali și/sau analogici).
- Senzori (de exemplu, senzorul LDR, DHT22 sau un senzor ultrasonic).
- Afișaj LCD 16x2 sau terminal serial pentru vizualizarea rezultatelor.
- Diverse componente electronice: rezistori, butoane, leduri pentru testare suplimentară.
- Biblioteca pentru senzorul ales (ex. DHT22, LDR).
- IDE de programare Arduino (sau un mediu similar de dezvoltare).

Metodologie

1. **Achiziția semnalului de la senzor:**

   Se va începe prin citirea semnalului de la senzorul selectat. Senzorii vor fi conectați la pinuri specifice ale microcontrolerului (de exemplu, senzorul LDR pe un pin analogic sau DHT22 pe un pin digital). Datele vor fi citite la intervale regulate și vor fi prelucrate de microcontroler.

2. **Aplicarea filtrării semnalului:**

   După achiziția semnalului, se vor aplica două tipuri de filtre:

   - **Filtrul „Sare și piper"** pentru eliminarea fluctuațiilor bruște și pentru netezirea semnalului.
   - **Filtru de mediere ponderată** pentru a oferi o valoare medie ponderată bazată pe mai multe citiri ale semnalului. Acest filtru ajută la reducerea erorilor datorate zgomotului.

3. **Conversia ADC la tensiune și parametru fizic:**

   Semnalele analogice vor fi citite folosind un ADC (convertor analog-digital), iar valorile vor fi convertite în unități fizice corespunzătoare (de exemplu, lux pentru un LDR, temperatură pentru un DHT22). Conversiile vor fi realizate folosind formule simple sau funcții predefinite.

4. **Afișarea datelor:**

   După prelucrarea semnalului, datele vor fi afișate fie pe un LCD 16x2, fie pe un terminal Serial Monitor. Valorile afișate vor include atât semnalele prelucrate (de exemplu, lux, temperatură și umiditate), cât și tensiunea citită de la senzor.

5. **Testare și validare:**

   În final, aplicația va fi testată folosind simulatoare (precum Proteus) pentru a valida funcționalitatea circuitului și a aplicației de cod. După validare, se vor efectua măsurători fizice pentru a demonstra că soluția funcționează în condiții reale. Datele vor fi analizate și comparate cu valorile așteptate.

Rezultate

Pentru acest laborator am creat o aplicație pentru MCU care primește informația de la careva senzori achiziția și afișarea datelor de la senzori într-un sistem IoT simplu. Utilizarea senzorilor LDR și DHT22 a permis monitorizarea parametrilor de mediu în timp real, iar implementarea interfeței seriale a oferit utilizatorului o metodă ușoară și rapidă de a vizualiza aceste date (Vezi Figura 1).

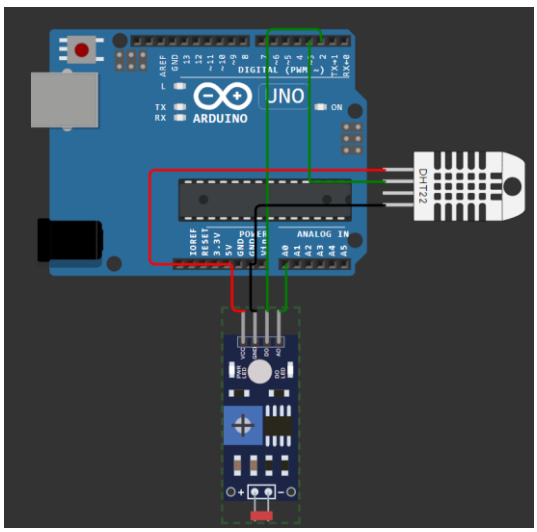

Figura 1 – Schema circuitului Arduino

Fragment cod sursa **main.ino**

```
#if ARDUINO >= 100
#include "Arduino.h"
#else
#include "WProgram.h"
#endif
#include <stdio.h>
#include "stdinout.h"

// Function that printf and related will use to print
static int serial_putchar(char c, FILE *f)
{
  if (c == '\n') {
    serial_putchar('\r', f);
  }

  return Serial.write(c) == 1 ? 0 : 1;
}
// Function that scanf and related will use to read
static int serial_getchar(FILE *)
{
  // Wait until character is avilable
  while (Serial.available() <= 0) { ; }

  return Serial.read();
}

static FILE serial_stdinout;

static void setup_stdin_stdout()
{
  // Set up stdout and stdin
  fdev_setup_stream(&serial_stdinout, serial_putchar, serial_getchar, _FDEV_SETUP_RW);
  stdout = &serial_stdinout;
  stdin  = &serial_stdinout;
  stderr = &serial_stdinout;
}

// Initialize the static variable to 0
size_t initializeSTDINOUT::initnum = 0;

// Constructor that calls the function to set up stdin and stdout
initializeSTDINOUT::initializeSTDINOUT()
{
  if (initnum++ == 0) {
    setup_stdin_stdout();
  }
}
```

În cadrul laboratorului 3.2, senzorii au fost integrați cu microcontrolerul pentru a prelucra și condiționa semnalele preluate și pentru a afișa parametrii fizici în timp real. Senzorul LDR a fost conectat la un pin analogic al microcontrolerului pentru a măsura intensitatea luminii ambientale, iar senzorul DHT22 a fost conectat la un pin digital, având rolul de a citi temperatura și umiditatea aerului. Semnalele analogice de la LDR au fost preluate prin intermediul convertorului analog-digital (ADC) al microcontrolerului, acestea fiind transformate inițial într-o valoare de tensiune. Ulterior, tensiunea a fost convertită în intensitatea luminii,

exprimată în LUX. În cazul senzorului DHT22, valorile de temperatură și umiditate au fost citite direct, utilizând o bibliotecă dedicată, fără a fi necesare conversii suplimentare.

Pentru a condiționa semnalele preluate, s-au implementat tehnici de filtrare digitală. Filtrul „Sare și piper" a fost aplicat pentru eliminarea fluctuațiilor bruște ale semnalului, iar filtrul de „Mediere ponderată" a fost utilizat pentru a oferi o valoare medie ponderată pe baza mai multor citiri ale semnalului. Aceste metode au fost aplicate pentru a asigura o interpretare corectă și precisă a datelor. După procesarea semnalului, datele obținute au fost afișate pe un afișaj LCD sau pe monitorul serial al Arduino, oferind utilizatorului o vizualizare completă a parametrilor monitorizați. Astfel, pe monitorul serial au fost afișate valorile de lumină în LUX, temperatura și umiditatea, împreună cu tensiunea măsurată de la senzorul LDR, demonstrând eficiența soluției implementate pentru condiționarea semnalului și vizualizarea acestuia (Vezi Fig. 2).

```
Lumina (LUX) LDR:  81.46       Tensiune LDR:   1.22 V
Temperatura:  25.00 °C  Umiditate:  50.00 %
Lumina (LUX) LDR: 108.61       Tensiune LDR:   1.22 V
Temperatura:  25.00 °C  Umiditate:  50.00 %
Lumina (LUX) LDR: 122.19       Tensiune LDR:   1.22 V
Temperatura:  25.00 °C  Umiditate:  50.00 %
```

Figura 2 – Rezultatul afișat pe Serial Monitor

Concluzie

În cadrul acestui laborator, s-a realizat o aplicație bazată pe microcontroler pentru citirea și condiționarea semnalului de la senzori, folosind filtre digitale pentru îmbunătățirea calității semnalului. Prin integrarea senzorilor analogici și digitali (LDR și DHT22), am reușit să măsurăm și să monitorizăm parametri fizici precum lumina (LUX), temperatura și umiditatea.

Senzorul LDR a fost folosit pentru a măsura intensitatea luminii ambientale, iar semnalele sale analogice au fost preluate prin ADC-ul microcontrolerului și transformate în valori de tensiune și lux. Senzorul DHT22 a furnizat datele de temperatură și umiditate, care au fost preluate direct cu ajutorul unei biblioteci specifice acestui tip de senzor.

Filtrele digitale aplicate în acest laborator (filtrul sare și piper și filtrul de mediere ponderată) au avut scopul de a îmbunătăți acuratețea citirilor, eliminând zgomotul și fluctuațiile necontrolate ale semnalelor. Aceste tehnici de filtrare sunt esențiale în prelucrarea semnalelor din surse de măsurare de mediu, în special când se lucrează cu senzori analogici.

În final, valorile preluate de la senzori au fost afișate în timp real pe Serial Monitor, oferind o vizualizare clară și accesibilă a parametrilor monitorizați. Implementarea funcțiilor de conversie și filtrare a semnalului a fost esențială pentru obținerea unor date fiabile și precise.

Acest laborator a demonstrat importanța prelucrării semnalelor și a condiționării acestora în aplicațiile de monitorizare și măsurare a parametrilor fizici, subliniind rolul esențial al filtrelor digitale în asigurarea unui semnal de calitate.

Anexa A

stdinout.h

```
#ifndef _STDINOUT_H
#define _STDINOUT_H

// no need to make an instance of this yourself
class initializeSTDINOUT
{
        static size_t initnum;
public:
        // Constructor
        initializeSTDINOUT();
};

// Call the constructor in each compiled file this header is included in
// static means the names won't collide
static initializeSTDINOUT initializeSTDINOUT_obj;

#endif
```

stdinout.cpp

```
#if ARDUINO >= 100
#include "Arduino.h"
#else
#include "WProgram.h"
#endif
#include <stdio.h>
#include "stdinout.h"

// Function that printf and related will use to print
static int serial_putchar(char c, FILE *f)
{
  if (c == '\n') {
    serial_putchar('\r', f);
  }

  return Serial.write(c) == 1 ? 0 : 1;
}
// Function that scanf and related will use to read
static int serial_getchar(FILE *)
{
  // Wait until character is avilable
  while (Serial.available() <= 0) { ; }

  return Serial.read();
}

static FILE serial_stdinout;

static void setup_stdin_stdout()
{
  // Set up stdout and stdin
  fdev_setup_stream(&serial_stdinout, serial_putchar, serial_getchar, _FDEV_SETUP_RW);
  stdout = &serial_stdinout;
  stdin  = &serial_stdinout;
  stderr = &serial_stdinout;
}

// Initialize the static variable to 0
size_t initializeSTDINOUT::initnum = 0;

// Constructor that calls the function to set up stdin and stdout
```

```
initializeSTDINOUT::initializeSTDINOUT()
{
  if (initnum++ == 0) {
    setup_stdin_stdout();
  }
}
```

Anexa B

Lab3.2.ino

```
#include <DHT.h>
#include "stdinout.h"  // Include pentru stdinout.h

#define DHTPIN 3
#define DHTTYPE DHT22

DHT dht(DHTPIN, DHTTYPE);

int analogPin = A0; // Pinul LDR
int analogValue = 0;
float tensiuneLDR = 0.0;
float lux = 0.0;
float temperatura = 0.0;
float umiditate = 0.0;

float readings[5]; // Fereastra de citiri pentru filtrare
float weights[5] = {0.1, 0.2, 0.3, 0.2, 0.1}; // Ponderi pentru mediere ponderată

char buffer[20];  // Buffer pentru stocarea valorilor convertite în string

// Funcția pentru filtrul Sare și Piper
float saltAndPepperFilter(float* readings, int size) {
  float minVal = readings[0];
  float maxVal = readings[0];

  // Căutăm valorile minime și maxime din setul de date
  for (int i = 1; i < size; i++) {
    if (readings[i] < minVal) minVal = readings[i];
    if (readings[i] > maxVal) maxVal = readings[i];
  }

  // Înlocuim valorile extreme cu media restului
  if (maxVal - minVal > 10) {  // Pragul de zgomot
    return (minVal + maxVal) / 2;
  } else {
    return readings[size / 2];  // Dacă nu există valori extreme, returnăm valoarea mediană
  }
}

// Funcția pentru filtrul de mediere ponderată
float weightedMovingAverage(float* readings, int size, float* weights) {
  float weightedSum = 0;
  float weightSum = 0;

  for (int i = 0; i < size; i++) {
    weightedSum += readings[i] * weights[i];
    weightSum += weights[i];
  }

  return weightedSum / weightSum;
}

void setup() {
  Serial.begin(9600);
  dht.begin();
  for (int i = 0; i < 5; i++) {
    readings[i] = 0;  // Inițializare citiri
  }
}
```

```
void loop() {
  // Citirea valorilor de la LDR
  analogValue = analogRead(analogPin);
  tensiuneLDR = adcToVoltage(analogValue);
  lux = voltageToLux(tensiuneLDR);

  // Citirea datelor de la DHT22
  umiditate = dht.readHumidity();
  temperatura = dht.readTemperature();

  // Filtrarea valorilor pentru LDR
  for (int i = 4; i > 0; i--) {
    readings[i] = readings[i - 1];
  }
  readings[0] = lux;  // Adăugăm noua valoare citită
  float filteredLux = weightedMovingAverage(readings, 5, weights);  // Aplicăm filtrul de mediere
ponderată

  // Aplicarea filtrului sare și piper
  int filteredReading = saltAndPepperFilter(readings, 5);

  // Convertim valorile în string și le afișăm folosind printf
  dtostrf(filteredLux, 6, 2, buffer);  // Conversie lux în string
  printf("Lumina (LUX) LDR: %s\t", buffer);

  dtostrf(tensiuneLDR, 6, 2, buffer);  // Conversie tensiune în string
  printf("Tensiune LDR: %s V\n", buffer);

  dtostrf(temperatura, 6, 2, buffer);  // Conversie temperatură în string
  printf("Temperatura: %s °C\t", buffer);

  dtostrf(umiditate, 6, 2, buffer);  // Conversie umiditate în string
  printf("Umiditate: %s %%\n", buffer);

  delay(2000);  // Pauză de 2 secunde între citiri
}

float adcToVoltage(int adcValue) {
  return adcValue * (5.0 / 1023.0);  // Conversia ADC la tensiune (presupunând 5V)
}

float voltageToLux(float voltage) {
  return voltage * 100;  // Conversia tensiunii la lux (simplificat)
}
```