Sistem de Monitorizare al Confortului Termic

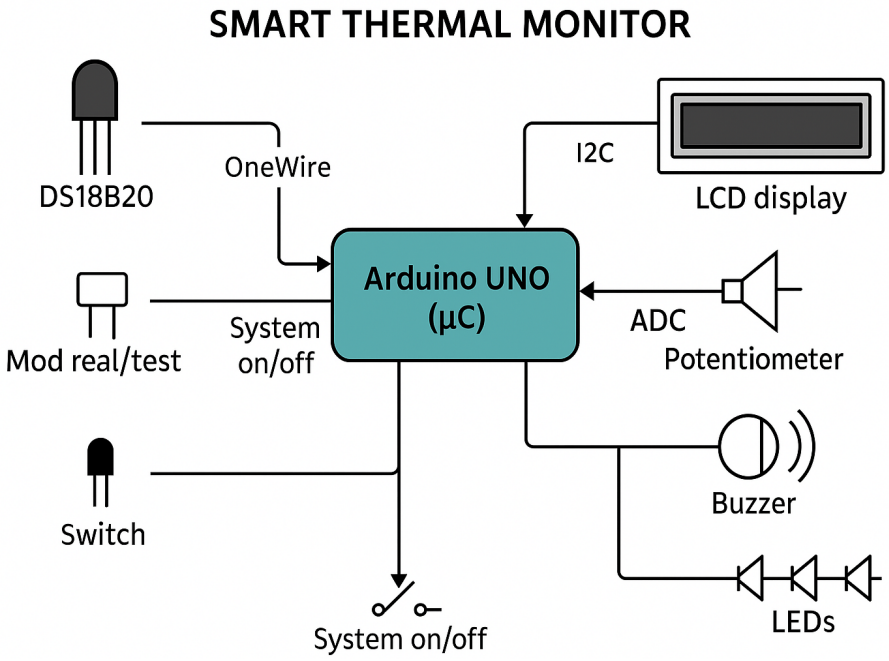
Introducere

Proiectul Smart Thermal Monitor este un sistem embedded de monitorizare și alertare termică, controlat cu Arduino UNO, realizat pentru a demonstra integrarea mai multor periferice externe și funcționalități software într-un produs fizic coerent. Dispozitivul funcționează în mod real, prin citirea temperaturii de la senzorul digital DS18B20, dar include și un mod de testare, activat cu ajutorul unui comutator, în care temperatura este simulată printr-un potențiometru (ADC).

Utilizatorul poate observa în timp real temperatura și starea sistemului prin intermediul unui afișaj LCD I2C, iar o temperatură ridicată este semnalizată vizual și auditiv printr-un buzzer și un ansamblu de 4 LED-uri controlate direct din registre (DDRx și PORTx). Sistemul este controlat manual printr-un buton ON/OFF implementat cu fire și detectare software, fără debounce hardware.

Proiectul se axează pe respectarea cerințelor academice, înglobând concepte studiate în cadrul laboratorului de PM, precum utilizarea GPIO, ADC, I2C, UART și control direct al perifericelor prin accesarea registrelor microcontrolerului. De asemenea, este evidențiată separarea logicii de test față de logica de funcționare reală, aspect esențial în dezvoltarea sistemelor embedded robuste.

Descriere generala

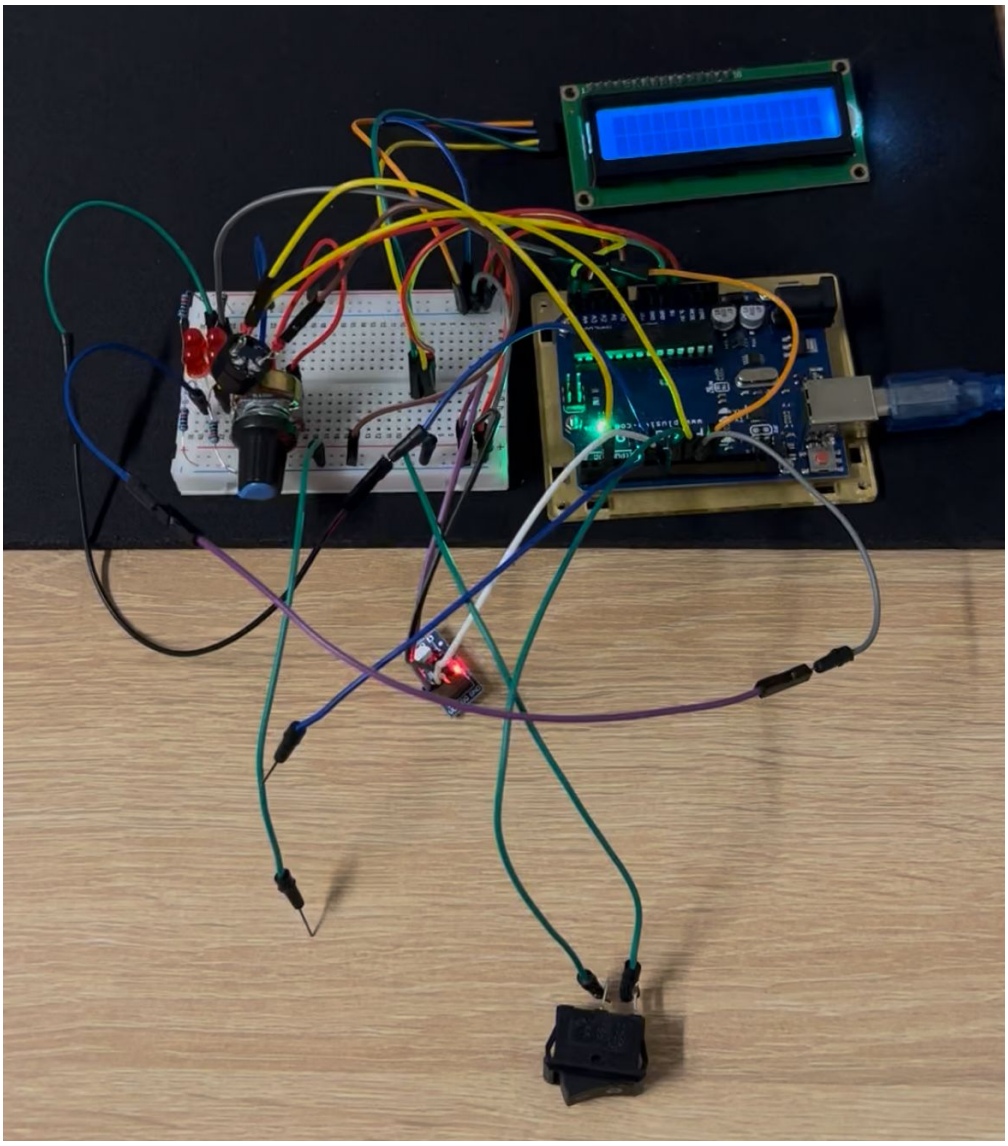


Sistemul este centrat pe un microcontroller Arduino UNO, care gestionează toate funcțiile dispozitivului. Senzorul de temperatură DS18B20 furnizează date în modul real, iar potențiometrul este utilizat în modul de test, permițând simularea valorii temperaturii. Comutarea între cele două moduri se realizează cu ajutorul unui switch dedicat.

Temperatura este citită, procesată și afișată în timp real pe un LCD I2C. În funcție de valoarea detectată, sistemul activează o alertă acustică printr-un buzzer și o alertă vizuală prin intermediul a patru LED-uri controlate direct prin registre (DDRx, PORTx). Un buton fizic (ON/OFF) permite activarea și dezactivarea sistemului în mod manual, fără resetarea plăcii.

Astfel, proiectul combină citirea de date, afișarea în timp real, controlul logic pe praguri și utilizarea eficientă a perifericelor externe, acoperind toate cerințele hardware și software prevăzute pentru proiect. Implementarea controlului LED-urilor prin registre oferă un plus de eficiență și demonstrează înțelegerea nivelului hardware al microcontrolerului.

Hardware Design



 Microcontroller

- Arduino Uno R3 (compatibil)

 Afișaj

- LCD 16×2 cu interfață I2C (adresă 0x27)

 Senzori și comenzi

- Senzor de temperatură DS18B20 digital
- Potențiometru 10kΩ linear (pentru setare temperatură simulată)
- Switch 2 poziții ON-OFF (pentru comutare mod Real / Test)
- Buton tactil 12mm (pentru ON/OFF general) (DEFECT!!!)

 Semnalizare

- Buzzer activ 5V
- LED roșu 5mm (pentru indicator stare) X4
- Rezistor limitare curent LED (220Ω) X4

 Conexiuni și structură

- Breadboard 400 puncte
- Set fire jumper male-male, male-female, female-female

Software Design

1. Inițializarea pinilor (registre)

```
#define BUZZER_PIN 2
#define SWITCH_PIN 8
#define BUTON_ONOFF 5

// LED-uri: D6 (PD6), D9 (PB1), D10 (PB2), D11 (PB3)
const byte LED_MASK_D = (1 << PD6);
const byte LED_MASK_B = (1 << PB1) | (1 << PB2) | (1 << PB3);

void setup() {
  // Configurare registre pentru LED-uri
  DDRD |= LED_MASK_D;   // setează D6 ca output
  DDRB |= LED_MASK_B;   // setează D9, D10, D11 ca output

  // Inițial LED-urile sunt stinse
  PORTD &= ~LED_MASK_D;
  PORTB &= ~LED_MASK_B;
```

```
}
```

2. Comutarea sistemului ON/OFF cu buton

```
bool sistemPornit = false;
static bool lastButon = HIGH;

void loop() {
  bool currentButon = digitalRead(BUTON_ONOFF);
  if (lastButon == HIGH && currentButon == LOW) {
    sistemPornit = !sistemPornit;
    lcd.clear();
    lcd.setCursor(0, 0);
    lcd.print(sistemPornit ? "Sistem pornit" : "Sistem oprit");
    delay(300); // debounce simplu
  }
  lastButon = currentButon;
```

3. Selectarea modului Real / Test

```
bool modTest = digitalRead(SWITCH_PIN) == LOW;
float tempC;

if (modTest) {
  int potValue = analogRead(POT_PIN);
  tempC = map(potValue, 0, 1023, 150, 400) / 10.0;
} else {
  sensors.requestTemperatures();
  tempC = sensors.getTempCByIndex(0);
}
```

4. Control alertă: LED-uri + buzzer

```
// ALERTĂ de temperatură
if (tempC >= TEMP_PRAG) {
  static unsigned long lastStep = 0;
  static int ledIndex = 0;

  if (millis() - lastStep >= 150) {
    lastStep = millis();
    for (int i = 0; i < 4; i++) {
      if (i == ledIndex) {
        if (i == 0) PORTD |= (1 << PD6);
        else PORTB |= (1 << (PB1 + i - 1));
      } else {
        if (i == 0) PORTD &= ~(1 << PD6);
        else PORTB &= ~(1 << (PB1 + i - 1));
      }
    }
    ledIndex = (ledIndex + 1) % 4;
```

```
  }
  digitalWrite(BUZZER_PIN, HIGH);
} else {
  // LED-uri aprinse continuu, buzzer oprit
  PORTD |= LED_MASK_D;
  PORTB |= LED_MASK_B;
  digitalWrite(BUZZER_PIN, LOW);
}
```

Rezultate Obţinute

Proiectul a fost realizat cu succes, respectând toate cerințele tehnice și funcționale impuse. Sistemul a fost testat atât în mod real, folosind senzorul DS18B20 pentru citirea temperaturii ambientale, cât și în mod de test, utilizând un potențiometru pentru simularea valorilor.

Funcționalitățile implementate și verificate:

- Afișarea temperaturii în timp real pe ecranul LCD 1602 (I2C)
- Comutare între mod real și mod test cu ajutorul unui switch
- Aprinderea LED-urilor și activarea buzzer-ului la atingerea unui prag critic (28°C)
- Controlul LED-urilor direct prin registre (DDRx și PORTx)
- Pornirea/oprirea întregului sistem cu ajutorul unui buton fizic
- Trimiterea datelor și mesajelor de stare prin Serial Monitor (UART)

Testele efectuate au confirmat comportamentul așteptat în toate scenariile simulate: încălzire activă, standby și alertă (overheating). Sistemul a răspuns stabil la toate comenzile și a menținut funcționalitatea corectă pe durata rulărilor extinse.

Proiectul demonstrează integrarea eficientă a perifericelor externe și a conceptelor de programare embedded studiate în cadrul laboratorului: GPIO, ADC, UART, I2C, registre directe și timere software.

În urma implementării acestui proiect, am înțeles mai bine atât interacțiunea dintre componentele hardware, cât și modul de control eficient al acestora la nivel de microcontroler.

Concluzii

Proiectul Smart Thermal Monitor a demonstrat cu succes capacitatea de a integra multiple componente periferice într-un sistem embedded funcțional, utilizând microcontrolerul Arduino UNO. Am reușit să implementăm un sistem complet care măsoară, afișează și reacționează la variațiile de temperatură în timp real, atât în mod real cât și în mod de testare.

Prin utilizarea directă a registrelor de control (DDRx și PORTx) pentru LED-uri, proiectul evidențiază o înțelegere avansată a modului în care funcționează perifericele digitale la nivel de microcontroler, oferind o alternativă optimizată față de funcțiile de nivel înalt precum `digitalWrite()`.

De asemenea, utilizarea I2C, UART și ADC confirmă aplicarea practică a noțiunilor de laborator și creează o bază solidă pentru dezvoltarea de sisteme embedded mai complexe.

Proiectul este stabil, fiabil și ușor de extins (de exemplu, prin adăugarea unui sistem real de încălzire, integrarea cu un ecran OLED sau comunicare wireless). Această experiență a oferit o înțelegere aprofundată a fluxului complet de dezvoltare embedded – de la conectarea fizică până la scrierea și optimizarea codului.

În concluzie, proiectul reflectă o execuție atent planificată, care îmbină eficient hardware și software într-o soluție practică și educațională.

Download

Codul sursă, commit-urile și istoricul complet al proiectului pot fi consultate la linkul de mai jos:

 Proiectul complet pe GitHub

Jurnal

Săptămâna	Interval	Activitate
10	28 Apr – 2 Mai	Confirmare finală temă proiect și început implementare hardware: test senzor DS18B20 și LCD I2C
11	5 – 9 Mai	Citirea temperaturii, afișaj pe LCD, integrare comutator mod real/test, testare potențiometru (ADC)
12	12 – 16 Mai	Adăugare buton ON/OFF, integrare buzzer, LED-uri controlate cu `digitalWrite()`
13	19 – 23 Mai	Refactorizare LED-uri pe registre (PORTx/DDR), testare completă, integrare serială, cod final și documentație
14	26 – 30 Mai	Validare funcționalități și încărcare proiect pe OCW și GitHub

Bibliografie/Resurse

Export to PDF