Sistem de climatizare

Introducere

Sistemul de climatizare implementat este un sistem care menține temperatura sub o anumită valoare. Acesta este asemănător unui termostat, care, în funcție de temperatura înregistrată, știe când să pornească răcirea pentru a aduce temperatura la valoarea setată.

Ideea acestui proiect a pornit de la faptul că fiecare dintre noi avem în casele noastre câte un termostat pentru centrală, util în iarnă pentru a ști când să pornim centrala, dar foarte puțini dintre noi au un sistem de climatizare complet și pentru vară. În general, noi doar dăm drumul la aerul condiționat să meargă când ne este cald și când nu ne mai este il oprim. Însă toate acestea se fac manual. Astfel, am considerat că este util un astfel de sistem automat pentru răcirea încăperii. De asemenea, acest sistem poate fi implementat nu doar pentru încăperile noastre, ci și pentru locuințele animalelor noastre de companie și menținerea temperaturii acestora.

Descriere generală

Utilizatorul poate seta cu ajutorul unui potențiometru temperatura dorită în cameră, iar aceasta se afișează pe ecranul LCD. Folosind senzorul de temperatură, dacă se detectează că temperatura este peste valoarea setată, ventilatorul va porni până când temperatura ajunge din nou la valoarea corectă. În funcție de diferența de temperatură între cea aflată în încăpere și cea stabilită, ventilatorul va funcționa la o turată mai mare sau mai mică pentru a ajuta la răcirea eficientă a încăperii. În plus, cu ajutorul unei fotodiode, sistemul știe dacă este zi sau noapte, astfel că, dacă este noapte, ventilatorul nu va porni pentru ca utilizatorul să nu răcească în timp ce doarme.

Hardware Design

Lista pieselor utilizate

- plăcuța Arduino UNO ATmega328p
- breadboard
- LCD cu modul I2C
- senzor de temperatură și umiditate DHT11
- ventilator 12V
- potențiometru
- MOSFET

- fotodiodă

Schema electrică (realizată în Autodesk Fusion)

Conexiune intre piese

LCD + Modul I2C: acestea au venit deja lipite și astfel am folosit doar pinii modulului pentru a-i conecta la plăcuța Arduino în pinii analogici A4 și A5 și de asemenea VCC-ul și GND-ul pe breadboard în locul special pentru aceste conexiuni. Acesta are rolul de a afișa temperatura curentă și cea setată de utilizator.

Potentiometrul: este legat la VCC și GND și la pinul A1 de pe plăcuță. Are rolul de a seta temperatura de prag dorită.

Senzorul de temperatură: acesta are 4 pini dintre care unul nu se folosește. 2 dintre ei sunt pentru alimentare și masă iar al 3-lea pentru conectarea la pinul digital 7 de pe plăcuță. S-a folosit de asemenea o rezistență de 10KOhm. Are rolul de a măsura temperatura din încăpere.

Fotodiodă: este conectată la alimentare și la masă și de asemenea pe plăcuța Arduino pe pinul analogic A1. Rolul ei este de a detecta dacă afară este întuneric sau lumină pentru a se hotărî dacă este zi sau noapte și a ști dacă aerul condiționat va merge sau nu.

Ventilator: ventilatorul folosit este unul cu 3 pini (unul și pentru senzorul de viteză). Pentru conectarea ventilatorului și posibilitatea de a controla turațiile lui s-a folosit un MOSFET. Ventilatorul este conectat cu ajutorul său la întreg circuitul și astfel și la plăcuța Arduino prin pinul digital 10. Deși ventilatorul are un senzor special pentru controlul turațiilor (firul galben), eu nu l-am folosit preferând să folosesc proprietățile MOSFET-ului pentru a se putea folosi orice fel de ventilator nu doar unul mai scump. Are rolul de a răci încăperea până temperatura actuală să ajungă mai mică sau egală cu cea setată.

Software Design

Mediul de dezvoltare folosit este Arduino IDE versiunea 1.8.19.

Motivația alegerii bibliotecilor folosite

Bibliotecile folosite:

- Wire.h - este folosită pentru comunicația I2C. Este necesară pentru funcționarea corectă a ecranelor LCD care utilizează interfața I2C, astfel gestionează detaliile de nivel scăzut ale comunicării I2C, cum ar fi inițializarea și transferul de date între dispozitive.
- LiquidCrystal_I2C.h - o bibliotecă specializată pentru controlul ecranelor LCD care folosesc interfața I2C. Biblioteca simplifică utilizarea LCD-urilor, oferind funcții pentru inițializare, scriere de text, controlul iluminării de fundal și poziționarea cursorului.
- dht.h - Aceasta este biblioteca pentru senzorul DHT (Digital Humidity and Temperature), care este folosit pentru a citi temperatura și umiditatea din mediul înconjurător. Biblioteca conține funcții specializate pentru inițializarea senzorului și pentru citirea datelor de temperatură și umiditate.

Justificarea utilizării funcționalităților din laborator

I2C (Inter-Integrated Circuit)

I2C a fost utilizat pentru a simplifica conexiunea între Arduino și LCD, reducând numărul de fire necesare și facilitând adăugarea altor dispozitive I2C în viitor. Acest protocol asigură o comunicare eficientă și fiabilă, oferind un control precis asupra afișării informațiilor pe LCD și permițând o gestionare flexibilă a componentelor conectate.

ADC (Analog-to-Digital Converter)

ADC este un convertor care transformă un semnal analogic continuu într-un semnal digital discret. În cazul Arduino, ADC-ul integrat poate citi tensiuni de la 0V la 5V și le convertește într-o valoare digitală între 0 și 1023. Acesta a fost folosit pentru 2 componente:

- fotorezistor - Citirea valorii analogice de la fotodiodă se face folosind funcția analogRead(PHOTODIODE_PIN), care convertește nivelul de lumină într-o valoare digitală între 0 și 1023. Această valoare este folosită pentru a determina nivelul de lumină ambientală. Valoarea de lumină este utilizată pentru a ajusta viteza ventilatorului, permițând un control adaptativ în funcție de condițiile de iluminare.
- potențiometru - Citirea valorii analogice de la potențiometru se face folosind funcția analogRead(POTENTIOMETER_PIN), care convertește poziția potențiometrului într-o valoare digitală între 0 și 1023. Această valoare este mapată la un interval de temperaturi folosind funcția map(). Valoarea obținută de la potențiometru este utilizată pentru a seta temperatura limită, care este comparată cu temperatura curentă pentru a controla viteza ventilatorului.

PVM (Pulse Width Modulation)

PWM a fost utilizat pentru a ajusta turațiile motorului ventilatorului. Aceasta permite controlul precis al vitezei motorului în funcție de temperatura curentă și nivelul de lumină, asigurând un răspuns adaptativ al sistemului la condițiile de mediu. Utilizarea PWM este o metodă eficientă și flexibilă pentru controlul motoarelor în proiectele cu Arduino. În acest proiect s-a folosit pentru 2 lucruri:

- Controlul Vitezei Motorului - funcția analogWrite() generează un semnal PWM pe pinul specificat. În cazul proiectului meu, analogWrite(GATE, ventilator_speed) ajustează viteza motorului prin modificarea ciclicității semnalului PWM aplicat pe pinul GATE. Valoarea ventilator_speed determină ciclicitatea semnalului PWM, variind între 0 (0% ciclicitate, motor oprit) și 1000(100% ciclicitate, motor la viteză maximă).
- Calculul Vitezei Ventilatorului - Viteza ventilatorului este determinată pe baza diferenței dintre

temperatura curentă și temperatura limită setată. Viteza ventilatorului este ajustată și în funcție de valoarea luminii citită de la fotodiodă (light_value > 100). Funcția constrain() asigură că valoarea vitezei ventilatorului se încadrează între MIN_SPEED_THRESHOLD și 255.

Explicația scheletului proiectului

În această secțiune o să explic scheletul, cum am realizat calibrarea elementelor de senzorică, elementele de noutate ale proiectului și alte explicații suplimentare.

- Definierea pinilor pentru diferite componente sau senzori

```
#define TEMPERATURE_SENSOR_PIN 7
#define GATE 10
#define POTENTIOMETER_PIN A1
#define PHOTODIODE_PIN A2
#define DELAY 500 // Delay in milliseconds
```

- Declararea variabilelor globale pe care urmează să le folosesc în întreg codul

```
// Initialize the LCD with I2C address 0x27 and dimensions 16x2
LiquidCrystal_I2C lcd(0x27, 16, 2);
dht DHT;

// Declare variables to store sensor values and states
int potentiometer_value;
int limit_temperature = -1;
int current_temperature = -1;
int previous_limit_temperature = -1;
int previous_current_temperature = -1;
int light_value = 0;

// Define constants for the minimum and maximum speed threshold of the fan
const int MIN_SPEED_THRESHOLD = 70;
const int MAX_SPEED_THRESHOLD = 1000;
```

- Funcția setup()

```
void setup() {
  // Set the GATE pin as an output and initialize it to LOW
  pinMode(GATE, OUTPUT);
  digitalWrite(GATE, LOW);

  // Initialize the LCD and clear any previous display
  lcd.init();
  lcd.backlight();
  lcd.clear();
}
```

În această secțiune se realizează set-up-ul și starea inițială pentru ventilator și LCD. Ne asigurăm că

ventilatorul, la început, este oprit prin `digitalWrite(GATE, LOW)` și se setează pinul GATE ca fiind de tip OUTPUT. LCD-ului i se dă lumina de fundal și se șterge tot ce ar fi putut fi scris înainte pe el pentru a ne asigura că, ulterior, se afișează doar ce noi dorim.

- Funcția loop()

```
// Set cursor to the beginning of the first line and display "Set Temp: "
lcd.setCursor(0, 0);
lcd.print("Set Temp: ");

// Set cursor to the beginning of the second line and display "Current Temp:
"
lcd.setCursor(0, 1);
lcd.print("Current Temp: ");
```

Se setează cursorul astfel încât pe prima linie a LCD-ului să se afișeze “Set Temp: ”, iar pe a doua “Current Temp: ”. Acestea se fac la început pentru că ele sunt afișate mereu pe ecran și doar valorile care urmează a fi scrise se modifică.

```
// Read the light value from the photodiode
light_value = analogRead(PHOTODIODE_PIN);

// Read the potentiometer value to determine the limit temperature
potentiometer_value = analogRead(POTENTIOMETER_PIN);
limit_temperature = map(potentiometer_value, 1023, 0, 50, 0);
```

Folosind ADC pentru a transforma semnalul analog in unul digital, se citesc si se memoreaza valorile date de fotorezisotr si potentiometru. Funcția map() este folosită pentru a mapea o valoare dintr-un interval în altul. În acest caz, valoarea citită de la potențiometru care este în intervalul 0-1023 (datorită rezoluției ADC a plăcii Arduino) este mapeată într-un interval de temperaturi între 0 și 50 grade Celsius. Astfel, potențiometrul poate fi rotit pentru a selecta temperatura dorită, iar această valoare este utilizată ca temperatură limită pentru activarea ventilatorului în funcție de diferența de temperatură măsurată.

```
// Read the temperature from the DHT11 sensor
int input_temp = DHT.read11(TEMPERATURE_SENSOR_PIN);
current_temperature = DHT.temperature;

// Calculate the temperature difference between the current and limit
temperatures
int temperature_difference = current_temperature - limit_temperature;

// Calculate the fan speed using a non-linear mapping based on temperature
difference
int ventilator_speed = 0;
if (temperature_difference > 0 && light_value > 100) {
  ventilator_speed = MIN_SPEED_THRESHOLD + pow(temperature_difference, 1.5)
* (MAX_SPEED_THRESHOLD - MIN_SPEED_THRESHOLD) / pow(50, 1.5);
  ventilator_speed = constrain(ventilator_speed, MIN_SPEED_THRESHOLD,
MAX_SPEED_THRESHOLD);
}
```

```
// Set the fan speed using PWM on the GATE pin
analogWrite(GATE, ventilator_speed);
```

Folosind senzorul de temperatură și funcțiile bibliotecii dht.h, se înregistrează temperatura curentă din încăpere. Întrucât ventilatorul are o turație diferită în funcție de diferența de temperatură între cea curentă și cea setată, se calculează această diferență. În cazul în care diferența este > 0, iar valoarea fotodiodei este mai mare de 100 (această valoare de comparație a fost aleasă ca prag pentru a se stabili dacă este zi sau noapte în funcție de lumina înregistrată), se calculează turațiile cu care va funcționa ventilatorul.

Explicații pentru calcularea vitezei ventilatorului:

- MIN_SPEED_THRESHOLD - Reprezintă valoarea minimă a vitezei ventilatorului. Indică viteza minimă la care ventilatorul ar trebui să funcționeze.
- pow(temperature_difference, 1.5) - Această parte a formulei calculează o putere a diferenței de temperatură (înălțimea). Ridicând diferența de temperatură la puterea 1.5, asigurăm o relație neliniară între diferența de temperatură și viteza ventilatorului. Acest lucru înseamnă că ventilatorul va reacționa mai rapid la variațiile mari de temperatură și mai lent la variații mici.
- (1000 - MIN_SPEED_THRESHOLD) - Aceasta reprezintă intervalul de viteză maximă pe care îl poate avea ventilatorul, minus viteza minimă. Practic, este diferența dintre viteza maximă și viteza minimă a ventilatorului.
- pow(50, 1.5) - Această parte a formulei reprezintă un factor de scalare. 50 este diferența de temperatură la care vrem ca formula să ofere o viteză maximă a ventilatorului.

În concluzie, rezultatul acestei expresii este viteza pe care o vom atribui ventilatorului în funcție de diferența de temperatură. Cu cât diferența de temperatură este mai mare, cu atât viteza ventilatorului va fi mai mare. De asemenea, utilizarea funcției pow pentru ridicarea la putere a diferenței de temperatură permite o reglare neliniară a vitezei ventilatorului, astfel încât să fie mai reactiv la schimbările de temperatură.

Apoi, folosind funcția constrain(), ne asigurăm că viteza ventilatorului (ventilator_speed) rămâne între o valoare minimă (MIN_SPEED_THRESHOLD) și o valoare maximă (1000). Aceasta este importantă pentru a evita situațiile extreme, cum ar fi o viteză prea mică, care ar opri ventilatorul, sau o viteză prea mare, care ar fi inutilă sau ar crea zgomot excesiv.

```
// Update the LCD if the limit temperature has changed
if (limit_temperature != previous_limit_temperature) {
  lcd.setCursor(11, 0); // Set cursor after "Set Temp: "
  lcd.print("    "); // Clear old value
  lcd.setCursor(11, 0); // Reset cursor position
  lcd.print(limit_temperature); // Display new limit temperature
  previous_limit_temperature = limit_temperature; // Update previous limit
temperature
}

// Update the LCD if the current temperature has changed
if (current_temperature != previous_current_temperature) {
  lcd.setCursor(14, 1); // Set cursor after "Current Temp: "
  lcd.print("    "); // Clear old value
```

```
  lcd.setCursor(14, 1); // Reset cursor position
  lcd.print(current_temperature); // Display new current temperature
  previous_current_temperature = current_temperature; // Update previous
current temperature
}

// Wait for the specified delay before the next loop iteration
delay(DELAY);
```

Acest cod se ocupă de actualizarea afișajului LCD în funcție de schimbările în temperatură. În primul bloc de cod, se verifică dacă temperatura limită setată a fost schimbată față de valoarea precedentă. Dacă da, se șterge vechea valoare de pe afișaj și se afișează noua valoare a temperaturii limită.

În al doilea bloc de cod, se verifică dacă temperatura curentă a fost schimbată față de valoarea precedentă. Dacă da, se șterge vechea valoare de pe afișaj și se afișează noua valoare a temperaturii curente.

În ambele cazuri, se actualizează variabilele previous_limit_temperature și previous_current_temperature pentru a reflecta valorile curente ale temperaturilor, astfel încât să se poată verifica schimbările ulterioare.

În final, după ce afișajul LCD a fost actualizat sau nu, programul așteaptă un anumit timp (DELAY) înainte de a itera din nou prin bucla principală (loop()). Aceasta asigură o pauză între actualizările afișajului pentru a nu supraîncărca sistemul cu actualizări inutile.

Rezultate Obţinute

Concluzii

Acest proiect a fost o ocazie foarte buna de a aplica cunoștințele acumulate la PM și ED. In plus am utilizat Fusion pentru a crea schema electrică aplicand inca o data cunostintele de la TSC. A fost o provocare inginerească ce mi-a oferit o înțelegere mai profundă a interacțiunii dintre hardware și software. Am experimentat cu ADC și PWM, am lucrat cu senzori, am afișat date pe LCD folosind I2C și, cel mai important, am înțeles cu adevărat cum se utilizează un Arduino. Proiectul a ieșit exact cum mi-am dorit, iar in urma sa chiar mi-am descoperit o noua pasiune

Download

Arhiva cu implementarea și schema electrică se poate descărca de aici.

Bibliografie/Resurse

Resurse hardware:

- Simbol si foorprint placuta Arduino
- Simbol si foorprint senzor de temperatura
- LCD + I2C
- Simulari online pe Tinkercad

Resurse software:

- Cum sa controlezi turatiile ventilatorului
- Cum sa folosesti senzorul de temperatura DHT11