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

Raport

Curs: Internetul lucrurilor

Tema: **Control - ON-OFF cu histereza**

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

Chișinău 2024

Definirea problemei 2

Realizarea unei aplicații in baza de MCU care va implementa sisteme de control pentru control turatii motor cu aplicarea metodei PID cu un encoder in calitate de sensor, si driver L298 pentru aplicarea puterii la motor.

Obiective

- configurarea aplicației pentru controlul motorului cu aplicarea metodei PID;
- crearea schemei și codului conform sarcinii propuse;
- înțelegerea sistemului creat și procesele ce se întâmplă.

Introducere

Controlul cu PID (Proportional-Integral-Derivative) este o metodă de control utilizată extensiv în domeniul ingineriei și automatizării. Acest sistem de control integrat utilizează trei componente principale pentru a asigura o reglare eficientă și stabilă a unui sistem dat. Acronimul PID derivă din funcțiile acestor componente distincte, și anume Proportional (P), Integral (I) și Derivative (D).

Componenta proporțională (P) acționează în funcție de diferența dintre valoarea măsurată a procesului și setpoint-ul dorit. Cu cât această diferență este mai mare, cu atât acțiunea proporțională devine mai puternică, asigurând un răspuns rapid la schimbările bruște ale sistemului.

Componenta integrală (I) intervine în funcție de integrala erorii în timp. Aceasta contribuie la eliminarea erorilor persistente dintre setpoint și valoarea măsurată, asigurând o reglare stabilă pe termen lung.

Componenta derivațională (D) atenuează acțiunea de control în funcție de rata de schimbare a erorii. Prin aceasta, se împiedică supra-reglarea și se contribuie la stabilizarea sistemului, asigurând că acesta nu reacționează excesiv la schimbările bruște.

Controlul cu PID este o abordare echilibrată și versatilă, adaptabilă la o gamă largă de aplicații, inclusiv sistemele de încălzire și climatizare sau procesele industriale complexe. Prin combinarea armonioasă a acestor trei componente, un controlador PID ajustează continuu acțiunea de control pentru a menține sistemul la setpoint și pentru a minimiza erorile.

În concluzie, controlul cu PID rămâne un instrument esențial în dezvoltarea și optimizarea sistemelor automate și în ingineria controlului, oferind un echilibru între rapiditatea de răspuns, precizie și stabilitate pe termen lung.

Metode și materiale

Materiale Necesare:

– Arduino Board (de exemplu, Arduino Uno);
– breadboard și fire de conexiune;
– un motor în current continuu;
– editor Arduino IDE instalat pe un calculator;
– un LCD display;
– un potentiometru.

Metoda de Implementare:

– conexiuni hardware:
    - conectarea motrolului la pini Arduino;
    - conectarea LCD;
    - conectarea potentiometru.
– scrierea codului în Arduino IDE:
    - definirea variabilelor și pinilor corespunzători pentru motorul;
    - implementare funcții pentru citirea schimbarea vitezei, direcției a motorului.
– testarea și debugging:
    - încărcare cod pe Arduino folosind Arduino IDE;
    - monitorizare comportament motor;
    - identificare și rezolvare eventualelor erori în cod.

Rezultate

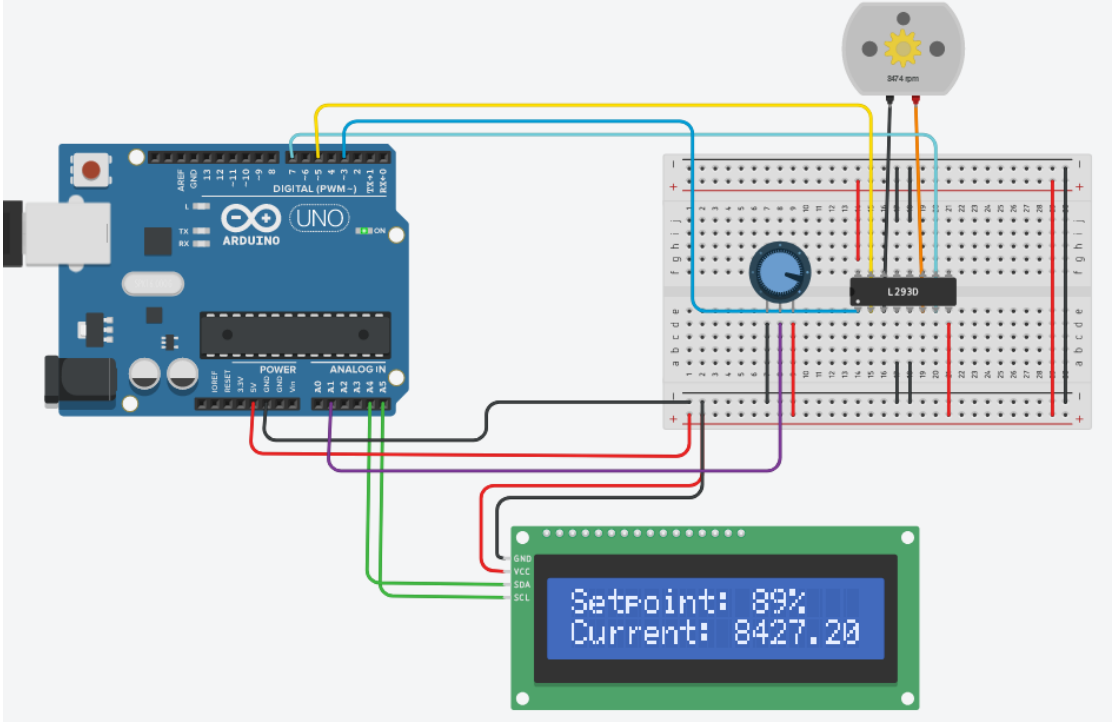

Figura 2 – Asamblarea circuitului pentru sarcina 2

Concluzii

În concluzie, dezvoltarea aplicațiilor în baza de microcontroler (MCU) pentru controlul temperaturii, respectiv pentru controlul rotației unui motor, a reprezentat o incursiune semnificativă în domeniul sistemelor de control. Abordarea metodei de control ON-OFF cu histeresis pentru gestionarea temperaturii sau umidității, cu utilizarea unui releu pentru acționare, a oferit o soluție eficientă și robustă pentru stabilizarea sistemului. Implementarea acestei metode se dovedește utilă în aplicații unde fluctuațiile mici în jurul setpoint-ului nu constituie o problemă majoră, iar adăugarea histerezei contribuie la prevenirea comutărilor frecvente ale dispozitivului de acționare, consolidând astfel stabilitatea sistemului În ceea ce privește controlul rotației motorului, aplicarea metodei PID (Proportional-Integral-Derivative) cu un encoder ca senzor și un driver pentru gestionarea puterii a adus precizie și control fin asupra sistemului. Integrarea acestor componente într-o aplicație bazată pe MCU a permis ajustarea continuă a turatării motorului pentru a atinge și menține setpoint-ul dorit. Acest tip de control este deosebit de util în aplicații unde este necesară o reglare precisă și rapidă a turatării, cum ar fi în sistemele de control al motoarelor.

Anexa 2 – Codul sursă sarcina 2

```
#include <LiquidCrystal_I2C.h>

#define SENSOR_PIN A1

#define FAN_ON true

#define FAN_OFF false

#define FAN_ON_AT 500

#define FAN_OFF_AT 20

#define FAN_INITIAL_STATE FAN_OFF

#define EN1 3

#define M1A 5

#define M1B 7

#define REFRESH_INTERVAL 1000

LiquidCrystal_I2C lcd(32, 16, 2);

int pos, veloc, oldpos = -1;

bool fan_on = FAN_INITIAL_STATE;

float PID_error = 0;

float previous_error = 0;

float elapsedTime, time, timePrev;

int PID_value;

//PID constants

int kp = 0.85; int ki = 0.3; int kd = 1.8;

int PID_p = 0; int PID_i = 0; int PID_d = 0;

void setup()

{

  lcd.init();

  lcd.begin(16, 2);

  lcd.backlight();

 Serial.begin(9600);

 pinMode(M1A, OUTPUT);
```

```
 pinMode(M1B, OUTPUT);
 time = millis();
}
void print_status(void) {
  lcd.setCursor(0, 0);
  lcd.print("Setpoint: ");
  lcd.print(map(veloc, 0, 255, 0, 100));
  lcd.print("%   ");

  lcd.setCursor(0, 1);
  lcd.print("Current: ");
  int current = analogRead(SENSOR_PIN);
  lcd.print(current);
}
void set_fan_state(const bool new_state) {
 fan_on = new_state;
}
void decide_fan_state(void) {
 if (fan_on) { // fan was on
 if (pos <= FAN_OFF_AT)
 set_fan_state(FAN_OFF);
 } else { // fan was off
 if (pos >= FAN_ON_AT)
 set_fan_state(FAN_ON);
 }
// print_status();
}
double input, output;
void set_motor_speed(void) {
```

```
  pos = analogRead(SENSOR_PIN);
  veloc = map(pos, 0, 1023, 0, 255);
  output = computePID(veloc);
  lcd.setCursor(0, 0);
  lcd.print("Setpoint: ");
  lcd.print(map(veloc, 0, 255, 0, 100));
  lcd.print("%   ");
 // int current = analogRead(SENSOR_PIN);
  lcd.setCursor(0, 1);
  lcd.print("Current: ");
  lcd.print(veloc * 36.8);
  lcd.print("   ");
  analogWrite(EN1, output);
  if (fan_on) {
    digitalWrite(M1A, LOW);
    digitalWrite(M1B, HIGH);
  } else {
    digitalWrite(M1A, HIGH);
    digitalWrite(M1B, LOW);
  }
}
double computePID(double inp) {
 PID_error = 255 - inp;
 PID_p = kp * PID_error;
 if(-3 < PID_error && PID_error <3)
 {
 PID_i = PID_i + (ki * PID_error);
 }
 //For derivative we need real time to calculate speed change rate
```

```
 timePrev = time; // the previous time is stored before the actual time read
 time = millis(); // actual time read
 elapsedTime = (time - timePrev) / 1000;
 PID_d = kd*((PID_error - previous_error)/elapsedTime); //Now we can calculate the D value
 PID_value = PID_p + PID_i + PID_d;


 if(PID_value < 0)
 { PID_value = 0; }
 if(PID_value > 255)
 { PID_value = 255; }
 return 255-PID_value;
}
void loop()
{
 set_motor_speed();
 decide_fan_state();
 delay(REFRESH_INTERVAL);
}
```