Ministerul Educaţiei, Culturii și Cercetării al Republicii Moldova
Universitatea Tehnică a Moldovei
Departamentul Ingineria Software și Automatică

RAPORT

Lucrare de laborator Nr.5.2
Disciplina: IoT
Tema: Control PID

A efectuat: st.gr.TI-212,
Vlasitchi Stefan

A verificat : asist. univ.
Lupan Cristian

Chișinău 2024

Definirea problemei:

1. Sa se realizeze o aplicatie 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.
2. Set point (valoarea de referinta pentru control) se va seta de la una din surse, la alegere
   - un potentiometru
3. Valoarea de Setpoint si cea Curenta se vor afisa la LCD,

Obiective:

1. Realizarea unuei aplicatii care va permite controlul unui moror cu ajutorul metodei PID
2. Valoarea va fi schimbata de la un potentiometru
3. Valoare de set point si cea curenta se va afisa la LCD

INTRODUCERE

Controlul PID (Proporțional-Integral-Derivat) este o tehnică larg utilizată în domeniul automaticii pentru a menține o variabilă (de exemplu, temperatura, presiunea, nivelul) la o valoare dorită (valoare de referință). Acesta funcționează prin calcularea unei ieșiri (semnal de control) bazat pe diferența dintre valoarea dorită și valoarea măsurată a variabilei.

Controlul PID este format din trei termeni:

- Termenul Proporțional (P**):** Acesta este direct proporțional cu eroarea (diferența dintre valoarea dorită și cea măsurată). Un termen P mare duce la o corecție rapidă a erorii, însă poate genera oscilații.
- Termenul Integral (I)**:** Acesta ia în calcul suma erorilor în timp. Ajutorul termenului I este eliminarea erorilor de stare staționară (diferența constantă între valoarea dorită și cea măsurată).
- Termenul Derivativ (D): Acesta este proporțional cu rata de schimbare a erorii. Termenul D ajută la anticiparea erorilor și îmbunătățește stabilitatea sistemului.

Avantaje ale controlului PID

- Simplu de implementat
- Robust și funcționează eficient pentru o gamă largă de sisteme
- Poate fi reglat prin ajustarea coeficienților termenilor P, I și D

Dezavantaje ale controlului PID

– Necesită reglarea fină a coeficienților pentru performanțe optime
– Poate fi dificil de reglat pentru sisteme cu dinamică complexă

EFECTUAREA LUCRĂRII

Materiale necesare:

- **Microcontroler** (Arduino Mega) folosit ca microcontroler principal pentru controlul releului și afișarea stării pe LCD. Acesta va primi comenzi prin interfața serială și va controla becul;
- **Motor cu encoder** acesta va fi motorul pe care il vom controla;
- **L293D** vom folosi acest microcontroller pentru a controla motorul;
- **Display LCD I2C** pentru a afișa starea becului (aprins sau stins). Interfața I2C simplifică conexiunile necesare pentru LCD, reducând numărul de pini necesari pe Arduino Mega;
- **Potentiometru** vom utiliza pentru a incrementa si decrementa set pointul;

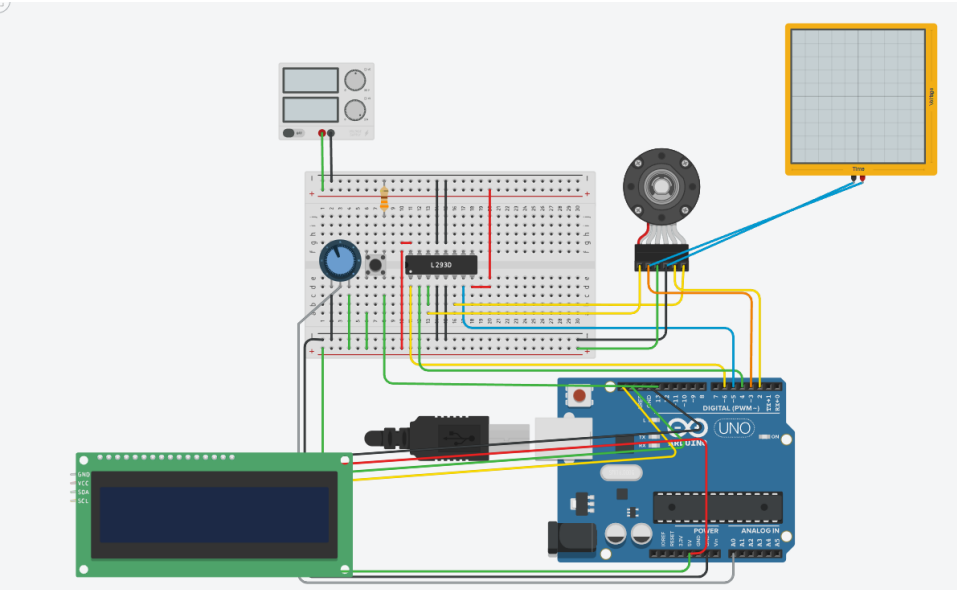

Figura 1 – Ansamblarea circuitului virtual

Modul de lucru

Utilizând această aplicație în bază de MCU, putem influenta starea motor, utilizând un controller de tip PID, cu ajutorul acestuia schimbam directia si puterera motorului pentru a ajunge la un setpoint anumit, acesta fiind schimbat de la un potentiometru.

La schimbarea valorii pe potentiometru se schimba si valoarea rotatiilor pe minut(rmp), aceasta se datoreaza controlului PID care foloseste parametrii Kp,Kp si Kd

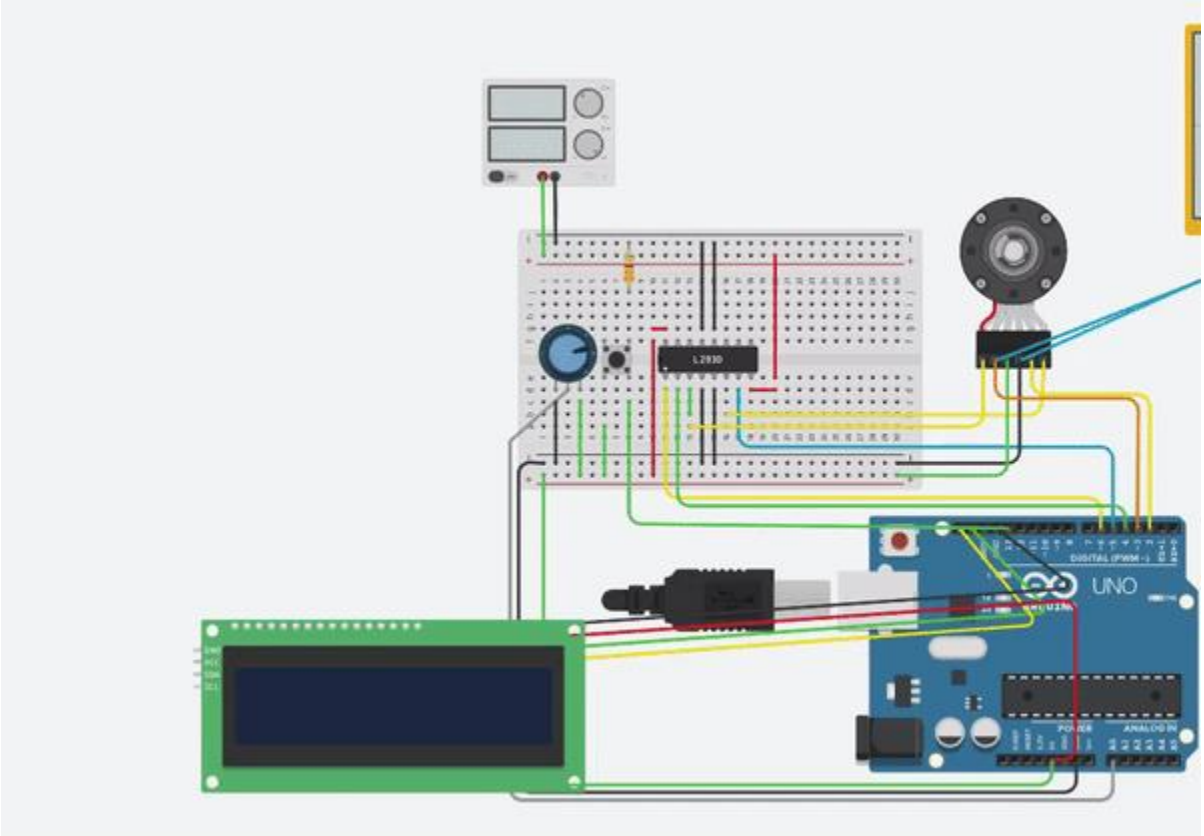

Figura 2 – Exemplu de control PID

CONCLUZIE

Lucrarea de laborator nr. 5.2, având ca temă "Control PID", a oferit oportunitatea de a explora și implementa unul dintre cele mai utilizate sisteme de control în domeniul automatizării. Utilizând un microcontroler (MCU) bazat pe Arduino și o serie de componente electronice, am realizat două aplicații practice: controlul temperaturii/umidității folosind metoda On-Off cu histeresis și acționare prin releu, și controlul turării unui motor utilizând metoda PID, un encoder ca senzor și un driver L298 pentru puterea motorului.

Pe parcursul lucrării, am învățat despre principiile fundamentale ale controlului PID și am experimentat procesul de implementare a acestuia în aplicații practice. Am identificat avantajele, precum și dezavantajele acestei metode de control, cum ar fi necesitatea unei ajustări fine a parametrilor pentru performanțe optime și dificultatea în gestionarea sistemelor cu dinamici complexe. Prin intermediul codului sursă și a simulărilor realizate în platforma Tinkercad, am demonstrat funcționarea și eficacitatea algoritmilor de control PID în cadrul aplicațiilor noastre. Am analizat interacțiunea dintre parametrii PID și comportamentul sistemului controlat, evidențiind importanța reglajului corespunzător pentru obținerea rezultatelor dorite.

Anexa 1

main.cpp

```
#include <LiquidCrystal.h>

LiquidCrystal lcd(12, 11, 10, 9, 8, 7);

const int pot = A0;
const int motorDirPin = 4, motorPWMPin = 5, EnablePin = 6;
const int encoderPinA = 2, encoderPinB = 3;
volatile long encoderPos = 0;
const float ratio = 360. / 188.611 / 48.;
float setpoint = 0, motorVelocity;
int control, direction = 1;
long pastEncoderPosition = 0, newEncoderPosition;
unsigned long newTime, oldTime = 0;
float Kp_vel = 2, Ki = 5, Kd = 0.00005, errorPrev = 0, errorAcum = 0;
double dt;

void setup() {
  Serial.begin(9600);
  pinMode(pot, INPUT);
  pinMode(encoderPinA, INPUT_PULLUP);
  pinMode(encoderPinB, INPUT_PULLUP);
  pinMode(motorDirPin, OUTPUT);
  pinMode(EnablePin, OUTPUT);
  lcd.begin(16, 2);
  attachInterrupt(digitalPinToInterrupt(encoderPinA), doEncoderA, CHANGE);
  attachInterrupt(digitalPinToInterrupt(encoderPinB), doEncoderB, CHANGE);
}

void loop() {
  setpoint = float(analogRead(pot)) / 1023 * 45;
  control += PID();
  doMotor(direction == 1, min(abs(control), 255));

  lcd.setCursor(0, 0);
  lcd.print("val set: ");
  lcd.print(setpoint * direction);
  lcd.print("  ");
  lcd.setCursor(0, 1);
  lcd.print("rpm: ");
  lcd.print(motorVelocity);
  lcd.print("  ");
  delay(100);
}
```

```
void doEncoderA() { encoderPos += (digitalRead(encoderPinA) == digitalRead(encoderPinB))
? 1 : -1; }
void doEncoderB() { encoderPos += (digitalRead(encoderPinA) == digitalRead(encoderPinB))
? -1 : 1; }

void doMotor(int dir, int vel) {
  digitalWrite(motorDirPin, dir);
  analogWrite(motorPWMPin, dir ? (255 - vel) : vel);
  digitalWrite(EnablePin, 1);
}

float PID() {
  newEncoderPosition = encoderPos;
  newTime = micros();
  dt = ((newTime - oldTime) / 1000.0) / 60.0;
  motorVelocity = 1000 * ((newEncoderPosition - pastEncoderPosition) * 125.0 / 1131666.0)
/ (((newTime - oldTime) / 1000.0) / 60.0);
  oldTime = newTime;
  pastEncoderPosition = newEncoderPosition;
  float error = abs(setpoint) - abs(motorVelocity);
  float errorPen = 1000 * (error - errorPrev) / (dt == 0 ? 0.0001 : dt);
  errorPrev = error;
  errorAcum += error * dt / 1000.0;
  return (Kp_vel * error + Ki * errorAcum + Kd * errorPen);
}
```