**דו"ח בקרה PID**

**מטרת המערכת ותיאור משימת הבקרה**

מטרת המערכת:

מטרת המערכת היא לדמות מטוטלת הפוכה על ידי מנוע שבקצה ציר הסיבוב שלו מחובר מוט. כיוון סיבוב המנוע המשפיע על כיוון תנועת המוט יקבע לפי כיוון הזרם הנכנס על ידי רכיב ה – H-Bridge.

משימת הבקרה:

על המערכת מיושם קוד בקרה מסוג P/PD אשר יאפשר לקלוט את זווית המוט על ידי החיישן ויתקן חזרה למצב הקלט הראשוני (מצב 0) לפי ההפרש של המצב הרצוי למצב המצוי (השגיאה).

**תיאור התכן**

|  |  |  |  |
| --- | --- | --- | --- |
| מספר הפין | שם המשתנה | תצורה | הפעלה |
| 10 | motorPin1 | OUTPUT | חיישני מנוע הDC. דרך החיווי המתקבל מהH-bridge החיישנים מעבירים אותו למנוע על כיוון התנועה. |
| 11 | motorPin2 | OUTPUT |
| 9 | enA | OUTPUT | חיישן הH-bridge. מעביר מידע על כיוון ומהירות המנוע. בעזרתו ניתן לשלוט במנוע בכל כיוון רצוי. |
| 2 | encoderPin1 | Encoder | חיישני Encoder. מעבירים מידע לבקר בנוגע למיקום ותנועת המוט. |
| 3 | encoderPin2 | Encoder |

**תיאור תהליך כיול פרמטרי הבקרים**

בכדי להביא את הכיול למצב האידיאלי, ראשית מדלנו את המערכת המבוקרת כמערכת מסדר שני בתרגיל התיאורטי. לאחר מכן, תכננו בקר PID ומצאנו פרמטרי בקרה עבור בקר P ובקר PD. מצאנו כי בקר P אינו עומד בדרישות המערכת מאחר וזמן ההתייצבות אינו קטן מ0.1 שניות. עבור בקר PD מצאנו את פרמטרי הבקרה המתאימים (Kp, Kd) ואותם הצבנו בקוד הבקר. (כלל החישובים בתרגיל התיאורטי התבצעו עבור 2 משקולות ולא 3 מפאת חוסר במשקולת בערכה).

בפועל, כאשר הצבנו את הערכים שהתקבלו בתרגיל התיאורטי הבקר אינו פעל כשורה ולכן היינו צריכים להתאים את ערכי הפרמטרים בהתאם להתנהגות הבקר. על מנת להביא ליציבות מקסימלית של המוט עבדנו בשיטת ניסוי וטעיה, תוך התחשבות במשמעות פרמטרי הבקרה kp וkd. התאמת ערכי kp ו-kd יכולה להשפיע על היציבות, זמן התגובה ועל הovershoot של מערכת הבקרה, כאשר הגדלת ערך הkd תוביל למתן משקל גבוה יותר לשינוי בקצב השגיאה ובעוד הגדלת ערך הkp תוביל למתן משקל גבוה יותר לשינוי בפרופורציית השגיאה.

**גרפים**

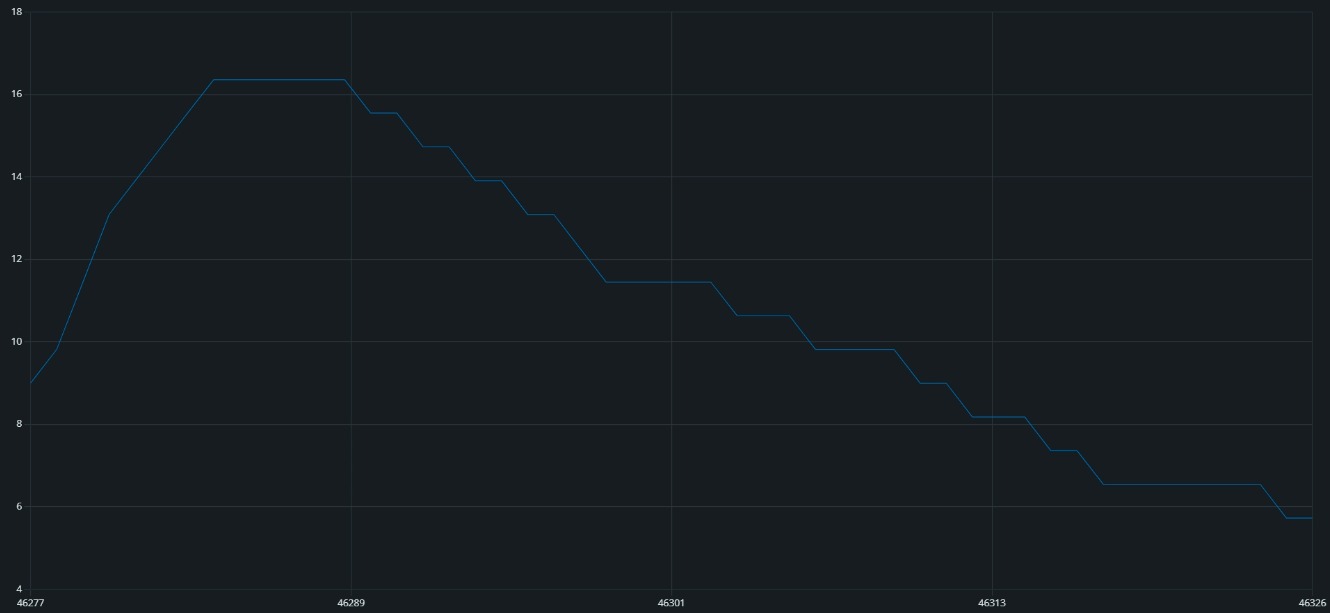
**בקר P**

מאחר ו- ts שמתקבל אינו עומד בדרישות המערכת (0.01 שניות), לא ניתן לייצב את המערכת ולבצע בקרה בעזרת בקר P בלבד.

**בקר PID**

מאחר ולא הצלחנו להביא ליצוב המערת על ידי PD נאלצו לייצב את המערכת לפי PID.

בגרף שהוצאנו מהserial plotter בIDE של הארדואינו ניתן לראות את החלק האחרון של ייצוב המערכת עבור 9600 baud rate



**דיון על הבדלי התוצאות בין התיאוריה למעשי.**

ניתן לראות שהערכים שקיבלנו עבור בקר pd בתרגיל 5 התיאורטי לא הצליחו לייצב את המערכת. נאלצנו לשחק עם הערכים כאשר הצבנו בkp =2 ובkd =1.2 כדי להגיע להתייצבות חלקית של המערכת. אנו סוברים שיכולות להיות מספר בעיות שיכולות לגרום לפער גדול בין התוצאות התיאורטיות לביצוע המעשי.

סיבות אפשרויות לחוסר יציבות הבקר הן:

1. **חוסר בקבוע אינטגרלי.** הקבוע האינטגרלי הוא פרופורציונלי לסכום כל שגיאות העבר ומשמש לביטול שגיאות במצב יציב במערכת בקרה. אם למערכת הבקרה יש שגיאה קבועה, החוסר בקבוע האינטגרלי יכול להניב מערכת בקרה שאינה יציבה.
2. **קבועים שגויים.** אם ערכי הקבועים (kp, kd) אינם מדויקים מספיק למערכת הבקרה, הבקר לא יהיה מסוגל לייצב את המערכת כראוי. לאור הקושי הרב שהיה בביצוע תרגיל בית 5 התיאורטי, יכול להיות שביצענו שגיאת חישוב
3. **מידול שגוי של המערכת.** במידה והמידול הראשוני של המערכת שגוי ואינו מייצג את הערכים הפיזיקליים באופן מדויק (יכול להיגרם גם כתוצאה משונות במיקום משקולות המוט, חוסר התחשבות במרכז המסה של המוט בתוספת המשקולות).

מאחר ולא הצלחנו לייצב את המערכת לפי בקרה PD בחרנו להשתמש בבקרת PID

(kp = 1, kd = 0.13, ki = 0.17)לאפשר יצב מדויק יותר של המערכת ובכך להקטין את השגיאה לערכים הרצויים.

**מס****קנות העבודה והמלצות להמשך**

ראינו שקיים פער גדול בהפיכת התיאוריה לפעולות מעשיות. פער זה יכול להגרם ממספר סיבות שציינו בסעיף הקודם. לכן אנו ממליצים ראשית לראות שבסיס ההבנה התיאורטית איכותית, ובמידה ועדיין לא מגיעים לפתרון המעשי הרצוי ניתן להשתמש בכלים שונים כגון ניסוי וטעיה, הוספה והורדה של משתנים כדי לעמוד במשימה המעשית.

**קובץ README**

הוראות הפעלה – Arduino2\_PIDController:

1. יש לחבר את הבקר לH-Bridge ולמנוע בהתאם להוראות מהתרגול.
2. יש להדק את הקרש אליו מחובר המנוע לשולחן כך שהמטוטלת תוכל להסתובב ב360 מעלות ללא מפריע.
3. לחבר לחשמל את המתג
4. יש לתת מכה לא חזקה מידי על המטוטלת לאחד הצדדים ולהמתין להתייצבותה
5. ניתן לחזור חלילה על התהליך.

**נספחים**

קוד עם שימוש בKi:

#include <Encoder.h>

#include <PIDController.h>

// Motor pins

const int motorPin1 = 10;

const int motorPin2 = 11;

const int enA = 9;

// Encoder pins

const int encoderPin1 = 2;

const int encoderPin2 = 3;

// PID constants

const float kp = 1;

const float ki = 0.17;

const float kd = 0.13;

const float goal\_deg = 0.0;

// Global variables

Encoder encoder(encoderPin1,encoderPin2);

PIDController pidController;

double angle = 0.0; // current angle of the pendulum

void setup() {

  // Initialize motor pins as output

  Serial.begin(9600);

  pinMode(enA, OUTPUT);

  pinMode(motorPin1, OUTPUT);

  pinMode(motorPin2, OUTPUT);

  pidController.begin(); // Initialize the PID controller

  pidController.limit(-180, 180); // The PID output can be limited to values between -255 to 255

  pidController.tune(kp, ki, kd); // Set PID parameters

  pidController.setpoint(goal\_deg); // the goal angle

}

void loop() {

  // // Read encoder and calculate current angle of pendulum

  long encoderCount = encoder.read();

  angle = tick\_to\_deg(encoderCount);

  // // Calculate control signal

   double control = pidController.compute(angle);

   Serial.print("angle: ");

   Serial.println(angle);

  if (control > 0.0) {

    forward();

  } else {

    reverse();

  }

  //Send pwm power to motor

  analogWrite(enA, abs(255\*control\*4/100));

}

//Convert ticks to degrees

float tick\_to\_deg(long tick){

    return tick\*360.0/440.0;

}

void forward(){

  digitalWrite(motorPin1, HIGH);

  digitalWrite(motorPin2, LOW);

  }

void reverse(){

  digitalWrite(motorPin1, LOW);

  digitalWrite(motorPin2, HIGH);

  }