MODUL 4 SISTEM KENDALI PID Kasus - P

Mata Kuliah : Sistem Kendali

Dosen : MHI

Kelas : D3TK-43-03

Kelompok GEMA BOBA :

1. Nanda Nur Rizqi 6702190018
2. Ihsan Darojatul U'la 6702194083

PROGRAM STUDI D3 TEKNOLOGI KOMPUTER
FAKULTAS ILMU TERAPAN
UNIVERSITAS TELKOM
BANDUNG
2021

MODUL 4 Sistem Kendali PID Kasus P

1. JUDUL PRAKTIKUM

Sistem Kendali PID Kasus P

2. MAKSUD DAN TUJUAN

Maksud dan tujuan dari praktikum ini adalah :

1. Mahasiswa dapat memahami fungsi dan cara kerja PID pada motor DC
2. Mahasiswa dapat membuat program sistem kendali berbasis PID dengan error yang dihubungkan dengan konstanta proporsional

3. PARAMETER PENILAIAN

No.	Parameter	Persentase (%)
1.	Lembar Penilaian Praktikum	40%
2.	Jurnal/Laporan Praktikum	60%

4. PERALATAN DAN BAHAN

Alat dan Bahan :

1. Robot Kit Line Follower
2. Baterai LiPo 2-Cell 1300 mAh
3. Kabel Mini-USB
4. Arduino Nano
5. Battery Checker
6. Battery Balancer

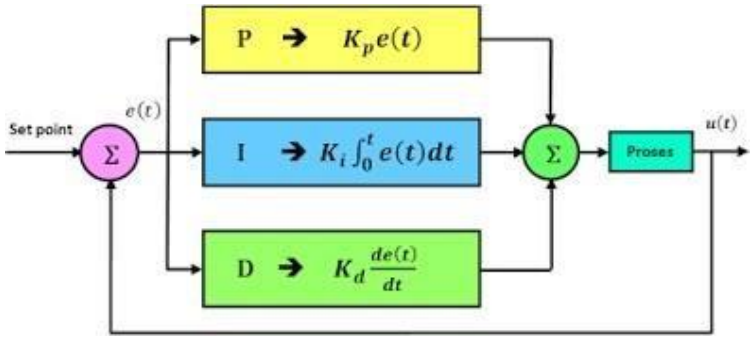
Perangkat Lunak :

1. Software IDE Arduino
2. Software Proteus (untuk simulasi)

5. TEORI DASAR

5.1. Sistem Kendali PID

Teknik kendali PID adalah pengendali yang merupakan gabungan antara aksi kendali proporsional ditambah aksi kendali integral ditambah aksi kendali derivatif/turunan (Ogata, 1996). PID merupakan kependekan dari *proportional integral derivative*. Kombinasi ketiga jenis aksi kendali ini bertujuan untuk saling melengkapi kekurangan-kekurangan dari masing- masing aksi kendali. Untuk memudahkan dalam memahami konsep teknik kendali PID silakan menyermati diagram blok pengendali PID pada gambar 1 di bawah ini.



Gambar 1. Diagram blok pengendali PID

Dalam aksi kendali PID, ada beberapa parameter variabel (dapat diubah/berubah) yang dapat dimanipulasi untuk tujuan menghasilkan aksi kendali terbaik dalam aplikasinya. Cara manipulasi parameter ini sering dinamakan dengan Manipulated Variable (MV). Dalam notasi matematikanya dapat ditulis dengan MV(t) atau u(t). Berikut persamaan matematik kendali PID.

$$u(t) = MV(t) = K_p e(t) + K_i \int_0^t e(\tau)d\tau + K_d \frac{d}{dt} e(t) \quad \text{…….} \quad (1)$$

$$K_i = \frac{K_p}{T_i} \quad \text{……...…...……} \quad (2)$$

$$K_d = K_p T_d \quad \text{……..…..……} \quad (3)$$

Persamaan (2) dan (3) disubtitusikan ke dalam persamaan (1) makan akan menjadi:

$$u(t) = MV(t) = K_p e(t) + \frac{K_p}{T_i} \int_0^t e(\tau)d\tau + K_p T_d \frac{d}{dt} e(t) \quad \text{…...…} \quad (4)$$

$$u(t) = MV(t) = K_p \left(e(t) + \frac{1}{T_i} \int_0^t e(\tau)d\tau + T_d \frac{d}{dt} e(t) \right) \quad \text{…...…} \quad (5)$$

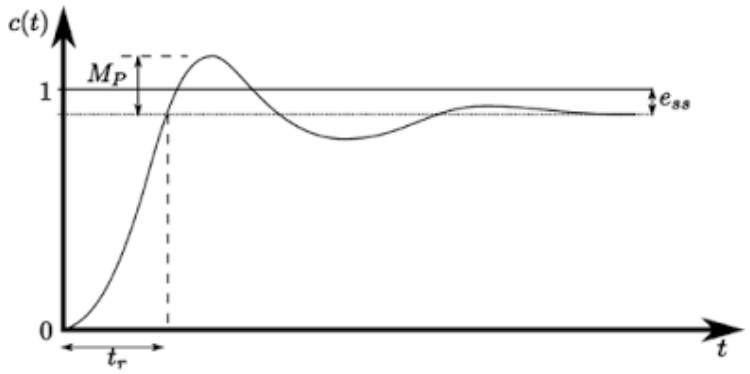
Apabila kita terapkan transformasi Laplace pada persamaan (4) di atas, maka penulisannya adalah sebagai berikut:

$$G(s) = K_p + \frac{K_i}{s} + K_d S \quad \text{.....} \quad (6)$$

$$G(s) = \frac{K_d s^2 + K_p s + K_i}{s} \quad \text{.....} \quad (7)$$

Respon Sistem Kendali PID

Gambar 2 di bawah ini merupakan ilustrasi grafik respon sistem kendali PID.

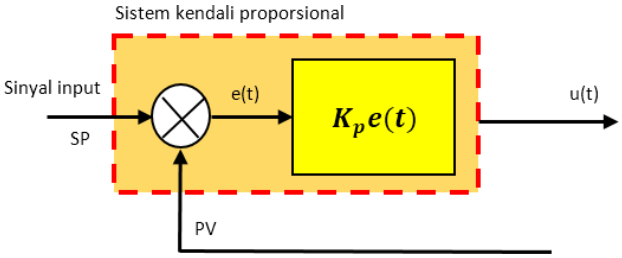


Gambar 2. Sinyal respon sistem kendali PID
MP = maksimum overshoot, ess = steady state error, tr = rise time

Aksi kendali PID memiliki karakter mampu mengurangi rise time (tr), mengurangi overshoot maksimum (MP), dan menghilangkan kesalahan keadaan tunak atau steady-state errors (ess).

5.2. Pengertian Sistem Kendali PID Kasus P (*Proportional*)

Aksi kendali proporsional (P) adalah aksi kendali yang memiliki karakter dapat mengurangi waktu naik (rise time), tetapi tidak menghilangkan kesalahan keadaan tunak (steady satate error).



Gambar 3. Diagram blok sistem kendali proporsional (P)

Persamaan hubungan antara keluaran sistem u(t) dengan sinyal *error* e(t) pada aksi kendali proporsional adalah sebagai berikut.

$$u(t) = K_p e(t)$$ ..... (1)

Sedangkan persamaan sinyal *error* -nya adalah:

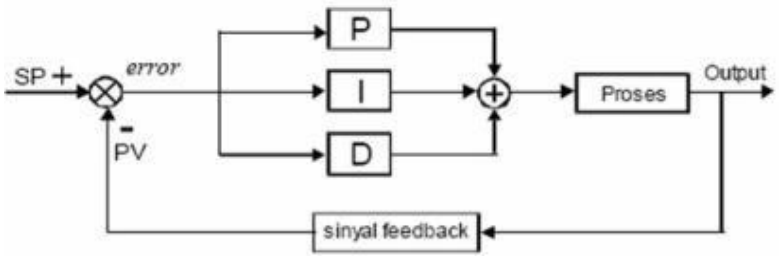
$$e(t) = SP - PV$$ ..... (2)

Pada praktikum ini nilai PV (*process value*) adalah error dengan setpoint (SP) sensor dianggap 0.

Dimana,
$u(t)$ = sinyal keluaran sistem kendali
K_p = Konstanta penguatan proporsional
$e(t)$ = sinyal *error*
SP = *Set Point*
PV = *Process Value* (nilai aktual)
t = waktu

5.3. Aplikasi PID pada Robot Line Follower

Sistem kendali PID ini bertujuan untuk menentukan paramater aksi kendali Proportional, Integratif, Derivatif pada robot line follower. Proses ini dapat dilakukan dengan cara trial and error . Keunggulan cara ini plant tidak perlu diidentifikasi dan membuat model matematis plant. Hanya dengan cara mencoba memberikan konstanta P-I-D pada formula PID ehingga di peroleh hasil yang optimal, dengan mengacu pada karakteristik masing-masing kontrol P-I-D.
Tujuan penggunaan sistem kendali PID adalah untuk mengolah suatu sinyal kesalahan atau error, nilai error tersebut diolah dengan formula PID untuk dijadikan suatu sinyal kendali atau sinyal kontrol yang akan diteruskan ke aktuator. Diagram blok sistem umpan balik loop tertutup pada perancangan sistem kendali PID pada robot line follower dapat dilihat pada gambar berikut ini:

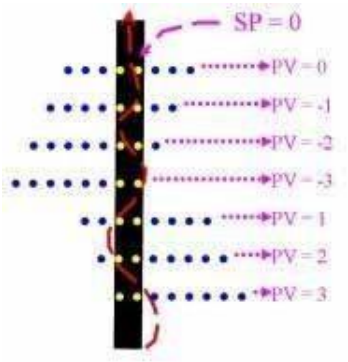


Dari blok diagram di atas dapat dijelaskan sebagai berikut

1. SP = *Set point*, suatu parameter nilai acuan atau nilai yang diinginkan.
2. PV = *Present Value*, nilai bobot pembacaan sensor saat itu atau variabel terukur yang di umpan balik oleh sensor (sinyal *feedback* dari sensor).
3. Error = nilai kesalahan, deviasi atau simpangan antar variabel terukur atau bobot sensor (PV) dengan nilai acuan (SP)

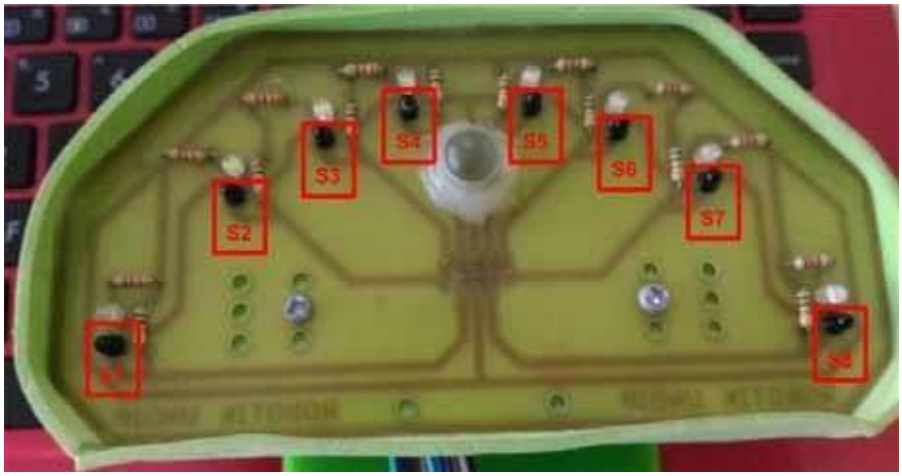
$$error = SP - PV$$

Ilustrasi pemberian bobot sensor (nilai kesalahan pembacaan sensor) pada robot line follower dapat dilihat pada gambar berikut.



PROSEDUR PRAKTIKUM

A. Percobaan dalam praktikum
   1. Peralatan dan Bahan Praktikum



Gambar 1 Contoh susunan dan urutan sensor pada robot line follower.

Code Program :

```
//pin buat sensor analog
int sensor1 = A0;
int sensor2 = A1;
int sensor3 = A2;
int sensor4 = A3;
int sensor5 = A4;
int sensor6 = A5;
int baca_sensor[6];

//pin buat enable
int pinEnable =4;
int pinEnable2 = 2;

//pin buat motor kiri
```

```
int motor_kiri1 = 5;
int motor_kiri2 = 6;

//pin buat motor kanan
int motor_kanan1 = 3;
int motor_kanan2 = 11;

//bantuan
int x;
char buffering [100];

//Error Code
int Last_Error = 0;
int error = 0;
int mtrSPDright, mtrSPDleft, setPointSPD = 255, PID_movement;

void setup(){
pinMode(sensor1, INPUT);
pinMode(sensor2, INPUT);
pinMode(sensor3, INPUT);
pinMode(sensor4, INPUT);
pinMode(sensor5, INPUT);
pinMode(sensor6, INPUT);

pinMode(pinEnable, OUTPUT);
pinMode(pinEnable2, OUTPUT);
pinMode(motor_kiri1, OUTPUT);
pinMode(motor_kiri2, OUTPUT);
pinMode(motor_kanan1, OUTPUT);
pinMode(motor_kanan2, OUTPUT);

Serial.begin(9600);
}

void readsensor() {
baca_sensor[0] = analogRead(sensor1);
baca_sensor[1] = analogRead(sensor2);
baca_sensor[2] = analogRead(sensor3);
baca_sensor[3] = analogRead(sensor4);
```

```
baca_sensor[4] = analogRead(sensor5);
baca_sensor[5] = analogRead(sensor6);

for (x = 0; x <= 5; x++){
Serial.println (baca_sensor[x]);
}
}

void try_PID(){
int kp = 30, ki = 30, kd = 0 ;
int rate = error - Last_Error;
Last_Error = error;
PID_movement = (kp * error) + (kd * rate) + (ki / rate);
mtrSPDright = setPointSPD - PID_movement;
mtrSPDleft = setPointSPD + PID_movement;

digitalWrite(pinEnable, HIGH);
analogWrite(motor_kiri1, mtrSPDleft);
analogWrite(motor_kiri2, 0);

digitalWrite(pinEnable2, HIGH);
analogWrite(motor_kanan1, mtrSPDright);
analogWrite(motor_kanan2, 0);
}

void If_Error(){
//just sensor 1 thats on
if (baca_sensor[0] < 34 && baca_sensor[1] > 34 &&
baca_sensor[2] > 34 && baca_sensor[3] > 34 &&
baca_sensor[4] > 34 && baca_sensor[5] > 34){

Last_Error = -2;
try_PID();
Serial.print ("Error Sensor Detect : ");
Serial.print (Last_Error);
Serial.print ("\n");
}

//just Sensor 2 thats on
```

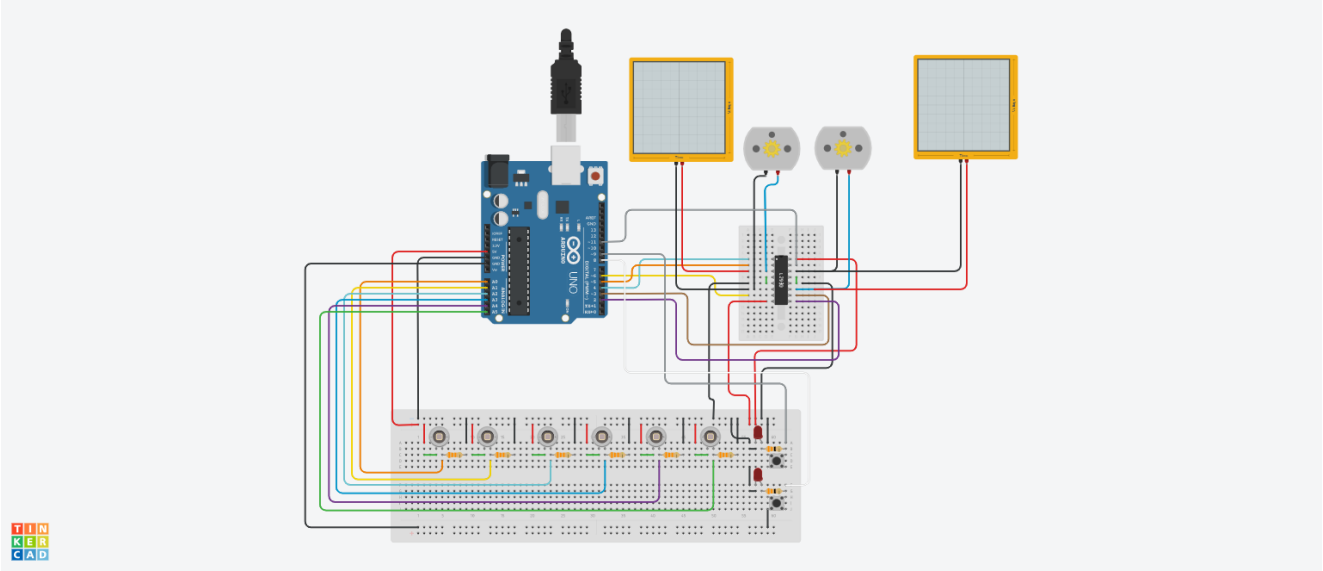
```
if (baca_sensor[0] > 34 && baca_sensor[1] < 34 &&
baca_sensor[2] > 34 && baca_sensor[3] > 34 &&
baca_sensor[4] > 34 && baca_sensor[5] > 34){

Last_Error = -1;
try_PID();
Serial.print ("Error Sensor Detect : ");
Serial.print (Last_Error);
Serial.print ("\n");
}

//just sensor 3 thats on
if (baca_sensor[0] > 34 && baca_sensor[1] > 34 &&
baca_sensor[2] < 34 && baca_sensor[3] > 34 &&
baca_sensor[4] > 34 && baca_sensor[5] > 34){

Last_Error = 0;
try_PID();
Serial.print ("Error Sensor Detect : ");
Serial.print (Last_Error);
Serial.print ("\n");
}

//just sensor 4 thats on
if (baca_sensor[0] > 34 && baca_sensor[1] > 34 &&
baca_sensor[2] > 34 && baca_sensor[3] < 34 &&
baca_sensor[4] > 34 && baca_sensor[5] > 34){

Last_Error = 0;
try_PID();
Serial.print ("Error Sensor Detect : ");
Serial.print (Last_Error);
Serial.print ("\n");
}

//just sensor 5 thats on
if (baca_sensor[0] > 34 && baca_sensor[1] > 34 &&
baca_sensor[2] > 34 && baca_sensor[3] > 34 &&
```

```
baca_sensor[4] < 34 && baca_sensor[5] > 34){

Last_Error = 1;
try_PID();
Serial.print ("Error Sensor Detect : ");
Serial.print (Last_Error);
Serial.print ("\n");
}

//just sensor 6 thats on
if (baca_sensor[0] > 34 && baca_sensor[1] > 34 &&
baca_sensor[2] > 34 && baca_sensor[3] > 34 &&
baca_sensor[4] > 34 && baca_sensor[5] < 34){

Last_Error = 2;
try_PID();
Serial.print ("Error Sensor Detect : ");
Serial.print (Last_Error);
Serial.print ("\n");
}
}

void loop(){

readsensor();
//Sensor 1 dan 2 mendeteksi gelap, sisanya terang >> Duty cycle 0% motor kiri, 50% motor kanan
if (baca_sensor[0] < 34 && baca_sensor[1] < 34 &&
baca_sensor[2] > 34 && baca_sensor[3] > 34 &&
baca_sensor[4] > 34 && baca_sensor[5] > 34){

Last_Error = -2;
try_PID();
}
//Sensor 2 dan 3 mendeteksi gelap, sisanya terang >> Duty cycle 20% motor kiri, 50% motor kanan
if (baca_sensor[0] > 34 && baca_sensor[1] < 34 &&
baca_sensor[2] < 34 && baca_sensor[3] > 34 &&
baca_sensor[4] > 34 && baca_sensor[5] > 34){

Last_Error = -1;
```

```
try_PID();
}
//Sensor 3 dan 4 mendeteksi gelap, sisanya terang >> Duty cycle 60% pada kedua motor (kedua motor aktif)
if (baca_sensor[0] > 34 && baca_sensor[1] > 34 &&
baca_sensor[2] < 34 && baca_sensor[3] < 34 &&
baca_sensor[4] > 34 && baca_sensor[5] > 34){

Last_Error = 0;
try_PID();
}
//Sensor 4 dan 5 mendeteksi gelap, sisanya terang >> Duty cycle 50% motor kiri, 20% motor kanan
if (baca_sensor[0] > 34 && baca_sensor[1] > 34 &&
baca_sensor[2] > 34 && baca_sensor[3] < 34 &&
baca_sensor[4] < 34 && baca_sensor[5] > 34){

Last_Error = 1;
try_PID();
}
//Sensor 5 dan 6 mendeteksi gelap, sisanya terang >> Duty cycle 50% Motor kiri, 0% motor kanan
if (baca_sensor[0] > 34 && baca_sensor[1] > 34 &&
baca_sensor[2] > 34 && baca_sensor[3] > 34 &&
baca_sensor[4] < 34 && baca_sensor[5] < 34){

Last_Error = 2;
try_PID();
}
//Semua sensor mendeteksi terang >> Duty cycle kedua motor 0% (semua motor mati)
if (baca_sensor[0] > 34 && baca_sensor[1] > 34 &&
baca_sensor[2] > 34 && baca_sensor[3] > 34 &&
baca_sensor[4] > 34 && baca_sensor[5] > 34){

//kedua motor mati
digitalWrite(pinEnable, HIGH);
analogWrite(motor_kiri1, 0);
analogWrite(motor_kiri2, 0);

digitalWrite(pinEnable2, HIGH);
```

```
analogWrite(motor_kanan1, 0);
analogWrite(motor_kanan2, 0);
}
If_Error();
}
```

Hasil Percobaan:



Tabel 1Tabel Kebenaran Sistem Kendali

Sensor						Error	Nilai Setpoint	Analog Value	
1	2	3	4	5	6			Motor Kiri	Motor Kanan
1	0	0	0	0	0	5	150	125(4688 Rpm)	175 (6843 Rpm)
1	1	0	0	0	0	4	150	130(4786 Rpm)	170(6775 Rpm)
0	1	0	0	0	0	3	150	135(4876 Rpm)	165(6624 Rpm)
0	1	1	0	0	0	2	150	140(5130 Rpm)	160(6578 Rpm)
0	0	1	0	0	0	1	150	145(5677 Rpm)	155(6245 Rpm)
0	0	1	1	0	0	0	150	150(5876 Rpm)	150(5876 Rpm)
0	0	0	1	0	0	-1	150	155(6245 Rpm)	145(5677 Rpm)
0	0	0	1	1	0	-2	150	160(6578 Rpm)	140(5130 Rpm)
0	0	0	0	1	0	-3	150	165(6624 Rpm)	135(4876 Rpm)
0	0	0	0	1	1	-4	150	170(6775 Rpm)	130(4786 Rpm)
0	0	0	0	0	1	-5	150	175(6843 Rpm)	125(4688 Rpm)

Kesimpulan Praktikum :

Dari kegiatan praktikum yang telah dijalankan di simulator tinkercad, dapat disimpulkan bahwa Sistem kendali PID digunakan untuk mengkoreksi error dari pengukuran variabel input (sensor) agar output sistem sesuai dengan nilai set point untuk menghasilkan error yang sekecil mungkin. Berdasarkan hasil percobaan yang telah dilakukan, didapatkan nilai parameter kontrol PID Kp= 5, Kd = 0, Ki=0 , Parameter tersebut digunakan dalam melakukan perhitungan PID dan mengatur kecepatan Motor kiri dan Kanan.