

## **MODUL 4 SISTEM KENDALI PID Kasus - P**



Mata Kuliah : Sistem Kendali

Dosen : MHI

Kelas : D3TK-43-03

Kelompok 17 DuetMaut :

- |                          |            |
|--------------------------|------------|
| 1. Ihsan Maulana Alzidni | 6702194020 |
| 2. Ajie Fauhad           | 6702194011 |

**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

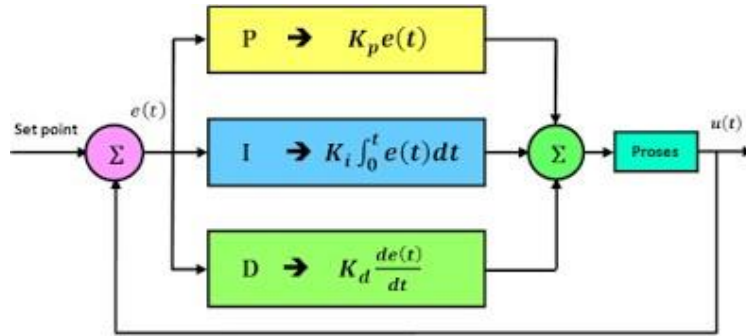
Perangkat Lunak :

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.

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

$$K_i = \frac{K_p}{T_i} \quad \dots\dots\dots (2)$$

$$K_d = K_p T_d \quad \dots\dots\dots (3)$$

Persamaan (2) dan (3) disubstitusikan ke dalam persamaan (1) maka akan menjadi:

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

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

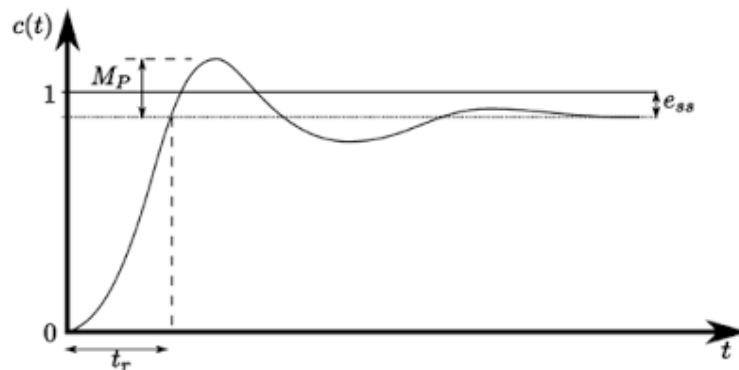
Apabila kita terapkan transformasi Laplace pada persamaan (4) di atas, maka penulisannya adalah sebagai berikut:

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

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

### Respon Sistem Kendali PID

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

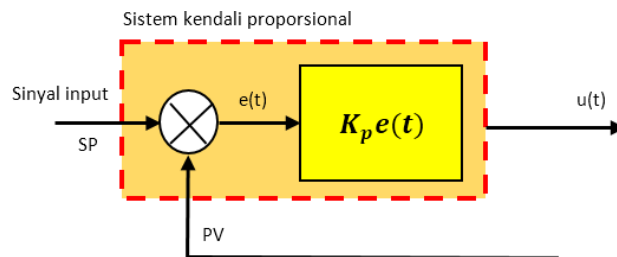


Gambar 2. Sinyal respon sistem kendali PID  
MP = maksimum overshoot, ess = steady state error, tr = rise time

Aksi kendali PID memiliki karakter mampu mengurangi rise time ( $t_r$ ), mengurangi overshoot maksimum (MP), dan menghilangkan kesalahan keadaan tunak atau steady-state errors ( $e_{ss}$ ).

### 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 state 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.

$$u(t) = K_p e(t) \quad \dots\dots\dots (1)$$

Sedangkan persamaan sinyal *error* -nya adalah:

$$e(t) = SP - PV \quad \dots\dots\dots (2)$$

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

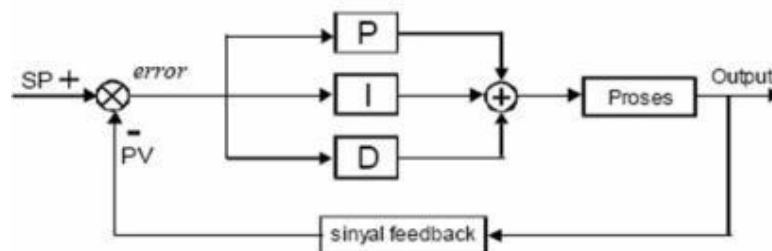
Dimana,

- $u(t)$  = 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:

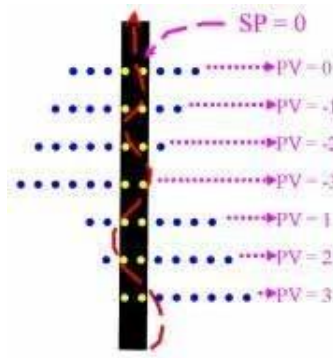


Dari blok diagram di atas dapat dijelaskan sebagai berikut

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)

$$error = SP - PV$$

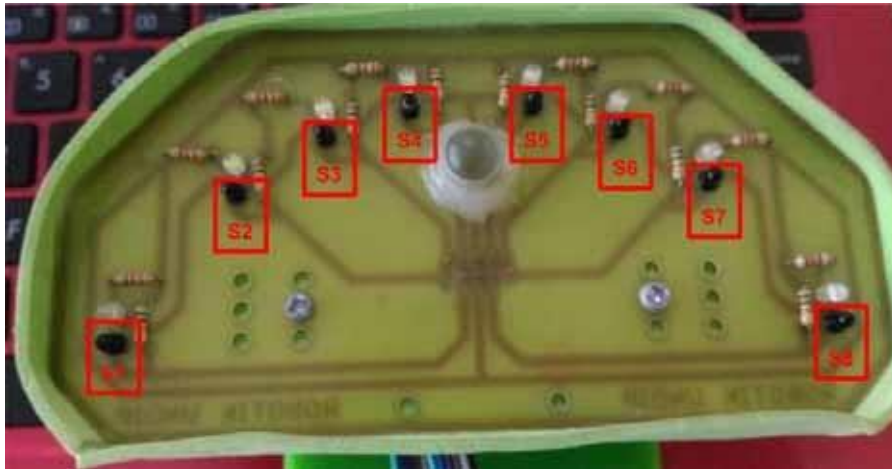
Ilustrasi pemberian bobot sensor (nilai kesalahan pembacaan sensor) pada robot line follower dapat dilihat pada gambar berikut.



## PROSEDUR PRAKTIKUM

### A. Percobaan dalam praktikum

#### 1. Peralatan dan Bahan Praktikum



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

Code Program :

```
//Modul Praktikum 4 - Sistem Kendali PWM
```

```
//Nama Tim : DuetMaut
```

```
//Nama Anggota 1 : Ihsan Maulana Alzidni 6702194020
```

```
//Nama Anggota 2 : Ajie Fauhad 6702194011
```

```
//Versi Program : 1.0
```

```
//Deklarasi Pin yang digunakan untuk sensor
```

```
int sensor1 = A0;
```

```
int sensor2 = A1;
```

```
int sensor3 = A2;  
int sensor4 = A3;  
int sensor5 = A4;  
int sensor6 = A5;  
int baca_sensor[6];  
int sendat[6];  
int sensorBit;  
int but1 = 8;  
int but2 = 9;
```

```
//Variable bantuan untuk menyimpan memory a.k.a Error Detection Memory(EDM)
```

```
//pin penggerak Motor  
int ENA = 4; //Pin 1&2 harus HIGH  
int ENB = 2; //Pin 3&4 harus HIGH
```

```
//pin Motorkiri  
int motorInput1 = 5; //input motor driver  
int motorInput2 = 6; //input motor driver
```

```
//pin Motorkanan  
int motorInput3 = 3;  
int motorInput4 = 11;
```

```
//Initial Speed of Motor  
int kecepatanSetPoint = 150;;
```

```
//Konstanta PID  
float Kp = 5 ;  
float Ki = 0 ;  
float Kd = 0 ;
```

```
float error = 0,P = 0, I = 0 , D = 0 , PID_value = 0;  
float lastError = 0;
```

```
void setup(){  
  // Keenam Sensor Photodiode sebagai INPUT yaitu sensor cahaya  
  pinMode(sensor1, INPUT);  
  pinMode(sensor2, INPUT);  
  pinMode(sensor3, INPUT);  
  pinMode(sensor4, INPUT);  
  pinMode(sensor5, INPUT);  
  pinMode(sensor6, INPUT);
```

```
  //Motor sebagai OOUTPUT penggerak  
  pinMode(motorInput1, OUTPUT);  
  pinMode(motorInput2, OUTPUT);  
  pinMode(motorInput3, OUTPUT);  
  pinMode(motorInput4, OUTPUT);
```

```
  pinMode(ENA, OUTPUT);  
  pinMode(ENB, OUTPUT);
```

```
  Serial.begin(9600);  
}
```

```
void loop(){  
  read_sensor_values();  
  Serial.print(error);  
  hitungPID();  
  motor_control();  
}
```

```
//Membaca sinyal analog dari sensor  
void read_sensor_values(){  
  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);
```

```
// 1 = Gelap 0 = Terang
```

```
// Case 1 0 0 0 0 0
```

```
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)  
{  
    error = 5;  
    Serial.print("\n");  
}
```

```
// Case 1 1 0 0 0 0
```

```
else 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){  
  
    error = 4;  
    Serial.print("\n");  
}
```

```
// Case 0 1 0 0 0 0
```

```
else 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){  
  
    error = 3;  
    Serial.print("\n");  
}
```

```
// Case 0 1 1 0 0 0
```

```
else 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){
```

```
    error = 2;
```

```
    Serial.print("\n");
```

```
}
```

```
// Case 0 0 0 1 0 0
```

```
else 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){
```

```
    error = 1;
```

```
    Serial.print("\n");
```

```
}
```

```
// Case 0 0 1 1 0 0
```

```
else 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){
```

```
    error = 0;
```

```
    Serial.print("\n");
```

```
}
```

```
// Case 0 0 0 1 0 0
```

```
else 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){
```

```
    error = -1;
```

```
    Serial.print("\n");
```

```
}
```

```

// Case 0 0 0 1 1 0
else 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){

    error = -2;
    Serial.print("\n");
}

```

```

// Case 0 0 0 0 1 0
else 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){

    error = -3;
    Serial.print("\n");
}

```

```

// Case 0 0 0 0 1 1
else 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){

    error = -4;
    Serial.print("\n");
}

```

```

// Case 0 0 0 0 0 1
else 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){

    error = -5;

```

```
Serial.print("\n");  
}
```

```
// Belokan kiri tajam
```

```
else 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){
```

```
    error = 10;  
    Serial.print("\n");  
}
```

```
//Belokan kanan tajam
```

```
    else 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){
```

```
        error = 11;  
        Serial.print("\n");  
}
```

```
//Putar Balik
```

```
    else 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){
```

```
        error = 12;  
        Serial.print("\n");  
}
```

```
//Garis Finish diam
```

```
    else 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){
```

```
error = 13;  
Serial.print("\n");  
}
```

```
/*// Menampilkan data sensor ke Serial Monitor  
// Data sensor 1-6  
// Formatting tampilkan sensor
```

```
for(int i=0; i<=5; i++){  
    Serial.print("Sensor");  
    Serial.print(i+1);  
    Serial.print(": ");  
    Serial.print(baca_sensor[i]);  
    Serial.print("\n");  
    delay(1000);  
}*/
```

```
}  
// Menghitung Nilai PID  
void hitungPID(){
```

```
    P = error;  
    I = error + lastError;  
    D = error - lastError;
```

```
PID_value = (Kp * P) + (Ki * I) + (Kd * D);
```

```
lastError = error;  
}
```

```
// Mengatur Kecepatan Motor
```

```
void motor_control()
```

```
{  
    int kecepatanMotorKiri = kecepatanSetPoint - PID_value;
```

```
int kecepatanMotorKanan = kecepatanSetPoint + PID_value;
```

```
// Kecepatan Motor agar tidak melebihi batas pwm
```

```
kecepatanMotorKiri = constrain(kecepatanMotorKiri, 0, 255);
```

```
kecepatanMotorKanan = constrain(kecepatanMotorKanan, 0, 255);
```

```
Serial.print(" ");
```

```
Serial.print(PID_value);
```

```
Serial.print("\t");
```

```
Serial.print(kecepatanMotorKiri);
```

```
Serial.print("\t");
```

```
Serial.println(kecepatanMotorKanan);
```

```
digitalWrite(ENA, HIGH);
```

```
analogWrite(motorInput1, kecepatanMotorKiri);
```

```
analogWrite(motorInput2, 0);
```

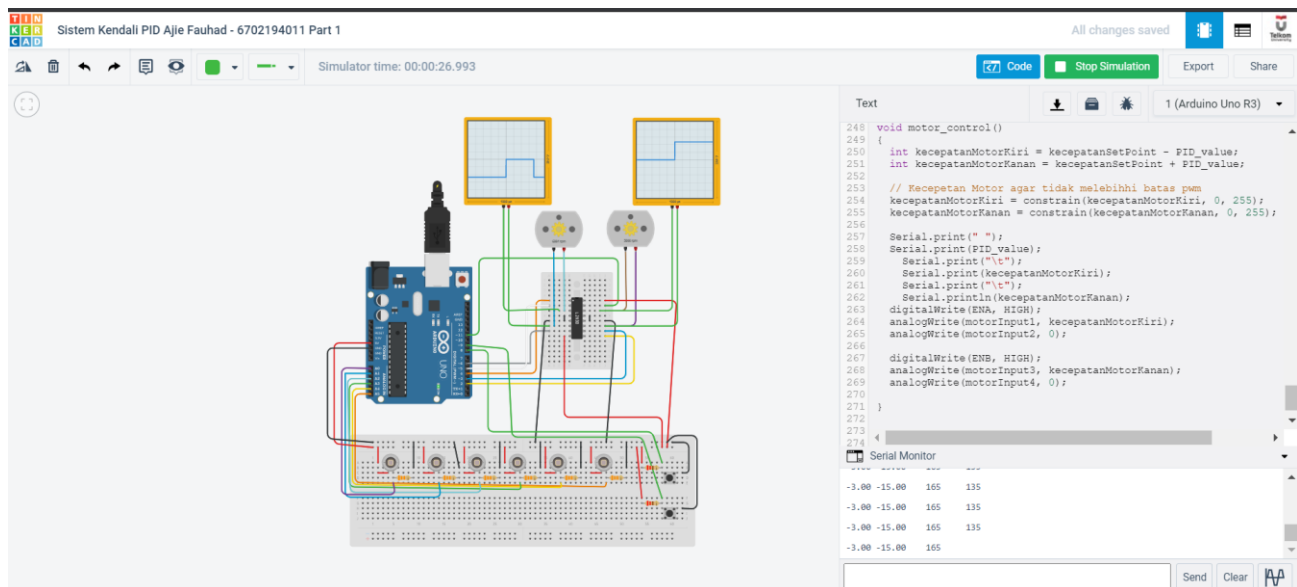
```
digitalWrite(ENB, HIGH);
```

```
analogWrite(motorInput3, kecepatanMotorKanan);
```

```
analogWrite(motorInput4, 0);
```

```
}
```

Hasil Percobaan:



Tabel 1 Tabel 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 :

Sistem kendali PID digunakan untuk mengoreksi error dari pengukuran variabel input (sensor) agar output sistem sesuai dengan nilai set point untuk menghasilkan error yang sekecil mungkin.

Berdasarkan hasil percobaan yang dilakukan, didapatkan nilai parameter kontrol PID  $K_p = 5$ ,  $K_d = 0$ ,  $K_i = 0$ , Parameter tersebut digunakan dalam melakukan perhitungan PID dan mengatur kecepatan Motor kiri dan Kanan.