MODUL 3 Sistem Kendali PID

1. JUDUL PRAKTIKUM

Sistem Kendali PID

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

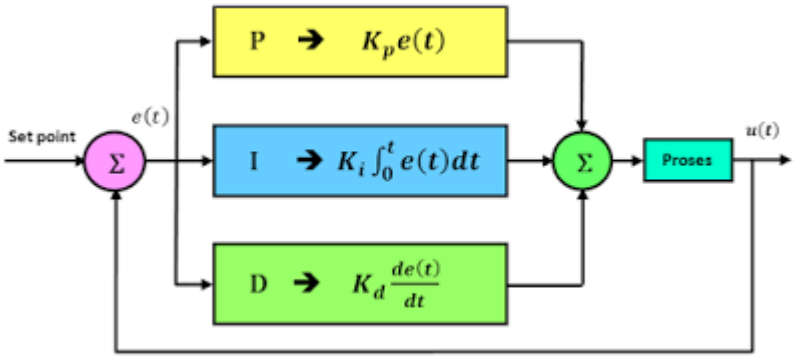
Perangkat Lunak :

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:

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

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

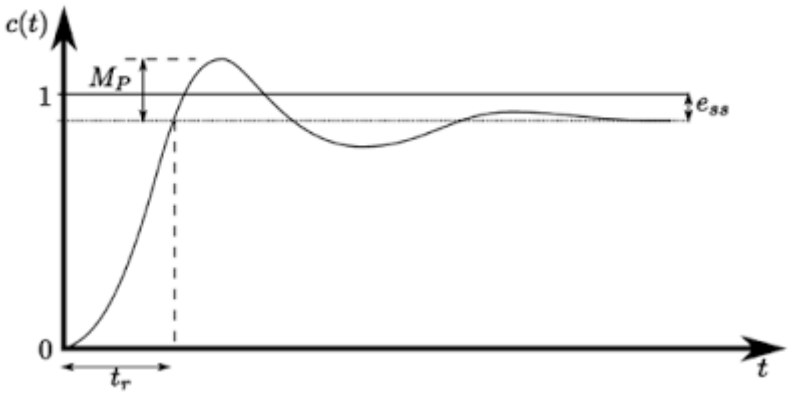
Apabila kita terapkan transformasi Laplace pada persamaan (4) di atas, maka penulisannya adalah sebagai berikut:

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

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

Respon Sistem Kendali PID

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



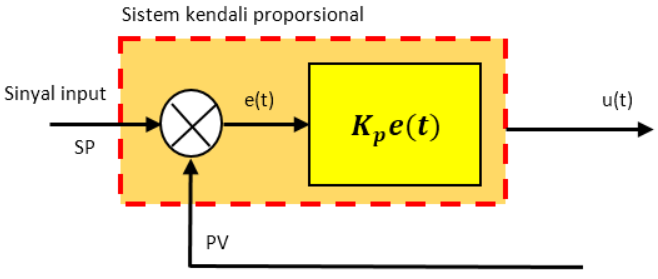
Gambar 2 Sinyal respon sistem kendali PID

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:

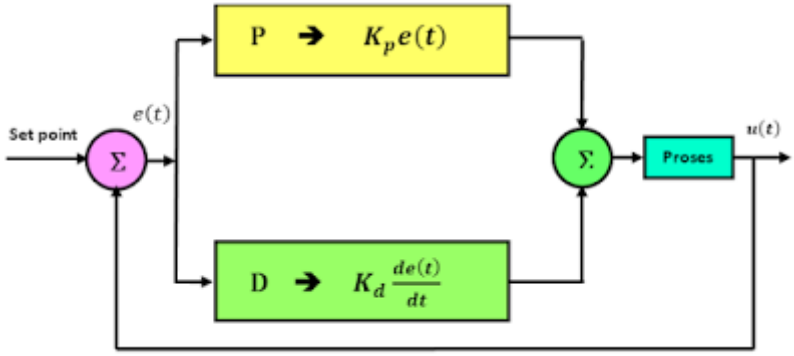
$$e(t) = SP - PV$$ ..... (2)

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. Pengertian Sistem Kendali PID Kasus PD (Proportional-Derivative)

Teknik kendali proporsional-derivatif (PD) adalah pengendali yang merupakan gabungan antara teknik kendali proporsional (P) dengan teknik kendali derivatif (D). Gambar 1 merupakan gambar diagram blok sistem kendali PD.



Gambar 4 Diagram blok sistem kendali PD.

Persamaan hubungan antara keluaran sistem dengan sinyal error pada kombinasi aksi kendali proporsional-derivative adalah sebagai berikut.

$$u(t) = K_p e(t) + K_d \frac{d}{dt} e(t)$$ ……. (1)

$$u(t) = K_p e(t) + K_p T_d \frac{de(t)}{dt}$$ …….(2)

Dalam penerapannya di software, kondisi ideal pada robot adalah bergerak maju lurus mengikuti garis, dengan kata lain error = 0 . Dari sini dapat diasumsikan bahwa Set Point (SP) / kondisi ideal adalah saat SP = 0. Nilai sensor yang dibaca oleh sensor disebut *Process Variable* (PV) / nilai aktual pembacaan. Menyimpangnya posisi robot dari garis disebut sebagai *error* (e), yang didapat dari $e = SP - PV$. Dengan mengetahui besar *error*, mikrokontroler dapat memberikan nilai PWM motor kiri dan kanan yang sesuai agar dapat menuju ke posisi ideal (SP = 0). Besarnya nilai PWM ini dapat diperoleh dengan menggunakan kontrol Proporsional (P), dimana $P = e \times Kp$ (Kp adalah konstanta proporsional yang nilainya diset sendiri dari hasil *tuning*/*trial and error*).

Jika pergerakan robot masih terlihat bergelombang, dapat ditambahkan parameter kontrol Derivatif (D). Kontrol D digunakan untuk mengukur seberapa cepat robot bergerak dari kiri ke kanan atau dari kanan ke kiri. Semakin cepat bergerak dari satu sisi ke sisi lainnya, maka semakin besar nilai D. Konstanta D (K_d) digunakan untuk menambah atau mengurangi imbas dari derivatif. Dengan mendapatkan nilai Kd yang tepat pergerakan sisi ke sisi yang bergelombang akibat dari kontrol proporsional dapat diminimalisasi. Dengan mendapatkan nilai K_d yang tepat pergerakan sisi ke sisi yang bergelombang akibat dari kontrol proporsional bisa diminimalisasi. Nilai D didapat dari $D = Kd/Ts \times rate$, dimana Ts adalah time sampling atau waktu cuplik dan rate = e(n) – e(n-1). Dalam program, nilai error (SP – PV) saat itu menjadi nilai last_error, sehingga rate didapat dari error – last_error

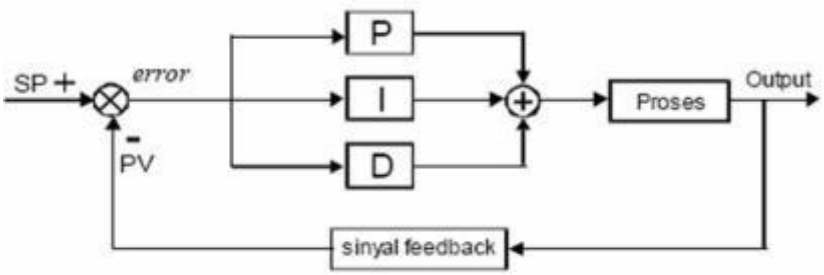
Ringkasan efek perubahan dari masing-masing parameter PID terhadap keluaran dari robot *line follower* dapat dilihat pada tabel berikut.

Tabel 1 Efek Perubahan Konstanta terhadap Parameter PID terhadap Keluaran Sistem

Parameter	***Rise Time***	***Overshoot***	***Settling Time***	***Steady State Error***
K_p	Menurun	Meningkat	Perubahan kecil	Menurun
K_i	Menurun	Meningkat	Meningkat	Menurun signifikan
K_d	Sedikit turun	Sedikit turun	Sedikit turun	Tidak ada efek

5.4. 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:

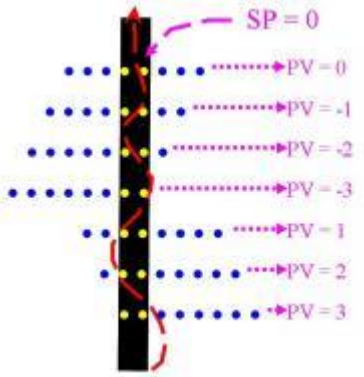


Dari blok diagram di atas dapat dijelaskan sebagai berikut

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)

$$error = SP - PV$$

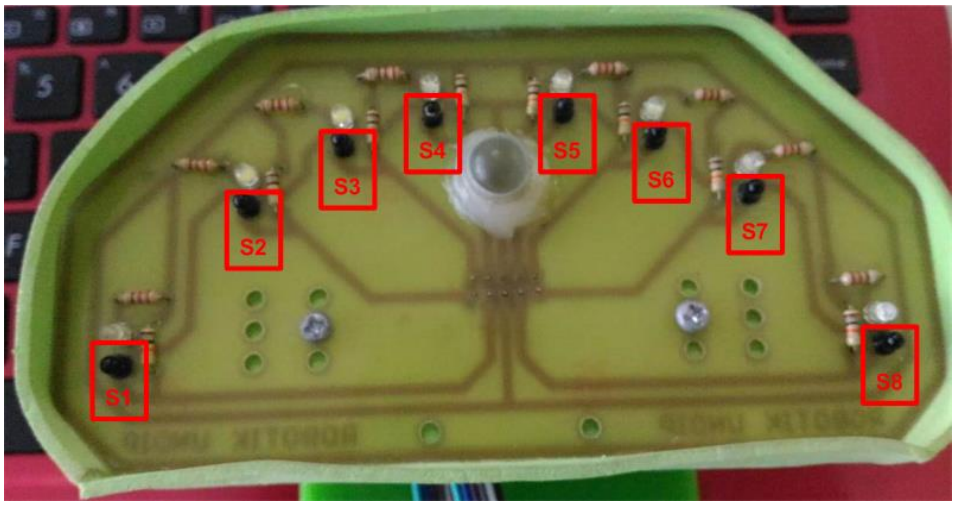
Ilustrasi pemberian bobot sensor (nilai kesalahan pembacaan sensor) pada robot line follower dapat dilihat pada gambar berikut.



PROSEDUR PRAKTIKUM

A. Percobaan dalam praktikum

1. Kasus Percobaan



Gambar 5 Contoh susunan dan urutan sensor pada robot line follower.

a. Modifikasi program sistem kendali pada praktikum sebelumnya dengan menambahkan sebuah sub program untuk melakukan kalibrasi sensor secara otomatis (*auto calibration*). Kalibrasi ini berfungsi untuk menentukan nilai pembacaan sensor untuk warna hitam dan putih atau warna lainnya. Apabila *push button* yang terhubung dengan pin D2 (pin switch Left Down (Kiri bawah) pada Gambar 2) ditekan, eksekusi program berikut :

- Tambahkan variabel **`sensor[0]`** sampai **`sensor[7]`**
- Tambahkan variabel **`NilaiMaxSensor[0]`** sampai **`NilaiMaxSensor[7]`** dengan nilai awal 1023
- Tambahkan variabel **`NilaiMinSensor[0]`** sampai **`NilaiMinSensor[7]`** dengan nilai awal 0
- Tambahkan variabel **`NilaiTengahSensor[0]`** sampai **`NilaiTengahSensor[7]`**
- Baca kondisi sensor 1 sampai 8 dan simpan di variabel **`sensor[0]`** sampai **`sensor[7]`**. Untuk nilai i dari 0 hingga 7 :
    - Jika nilai variabel **`sensor[i]`** > **`NilaiMinSensor[i]`** maka **`NilaiMinSensor[i]=sensor[i]`**
    - Jika nilai variabel **`sensor[i]`** < **`NilaiMaxSensor[i]`** maka **`NilaiMaxSensor[i]=sensor[i]`**
    - **`NilaiTengahSensor[i]`**= (**`NilaiMinSensor[i]+NilaiMaxSensor[i`**])/2
    - Saat proses kalibrasi dilakukan, LED LEFT (Pin D7) dan LED RIGHT (Pin D8) akan berkedap-kedip (*blinking*) sebagai indikator bahwa proses kalibrasi sedang berjalan.

b. Apabila proses kalibrasi sudah selesai dan tombol push button yang terhubung dengan pin D3 (pin switch Left Up/kiri atas pada Gambar 2) ditekan, maka eksekusi program berikut ini :
   - Inisialisasi variabel dengan tipe **`int Kp`** dengan nilai awal 15, **`int Kd`** dengan nilai awal : 5, **`int moveControl`** dengan nilai awal 0, **`int error`** dengan nilai awal 0, **`int lastError`** dengan nilai awal 0. **`int kecepatanMotorKanan`** dengan nilai awal 0, **`int kecepatanMotorKiri`** dengan nilai awal 0, **`int kecepatanSetPoint`** dengan nilai awal 150 dan **`int rate`** (selisih **`error`** dengan **`lastError`**).

c. Program harus dapat mendeteksi perubahan nilai pada sensor dan mengirimkannya ke serial monitor dengan ketentuan sebagai berikut.

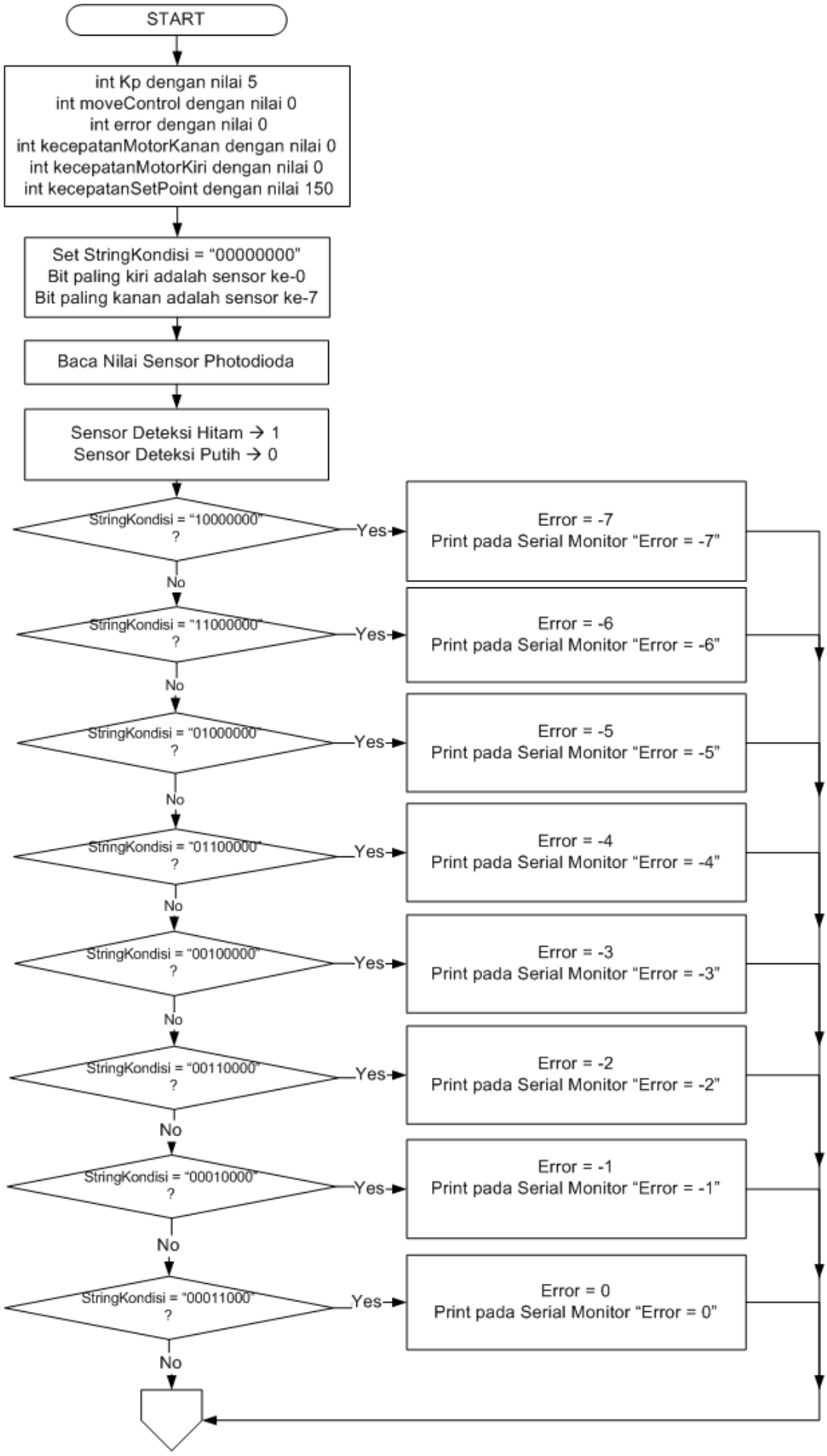
   - Jika kondisi sensor "10000000", error = -7, print di serial monitor error = -7,
   - Jika kondisi sensor "11000000", error = -6, print di serial monitor error = -6,
   - Jika kondisi sensor "01000000", error = -5, print di serial monitor error = -5,
   - Jika kondisi sensor "01100000", error = -4, print di serial monitor error = -4,
   - Jika kondisi sensor "00100000", error = -3, print di serial monitor error = -3,
   - Jika kondisi sensor "00110000", error = -2, print di serial monitor error = -2,
   - Jika kondisi sensor "00010000", error = -1, print di serial monitor error = -1,
   - Jika kondisi sensor "00011000", error = 0, print di serial monitor error = 0,
   - Jika kondisi sensor "00001000", error = 1, print di serial monitor error = 1,
   - Jika kondisi sensor "00001100", error = 2, print di serial monitor error = 2,
   - Jika kondisi sensor "00000100", error = 3, print di serial monitor error = 3,
   - Jika kondisi sensor "00000110", error = 4, print di serial monitor error = 4,
   - Jika kondisi sensor "00000010", error = 5, print di serial monitor error = 5,
   - Jika kondisi sensor "00000011", error = 6, print di serial monitor error = 6,
   - Jika kondisi sensor "00000001", error = 7, print di serial monitor error = 7,

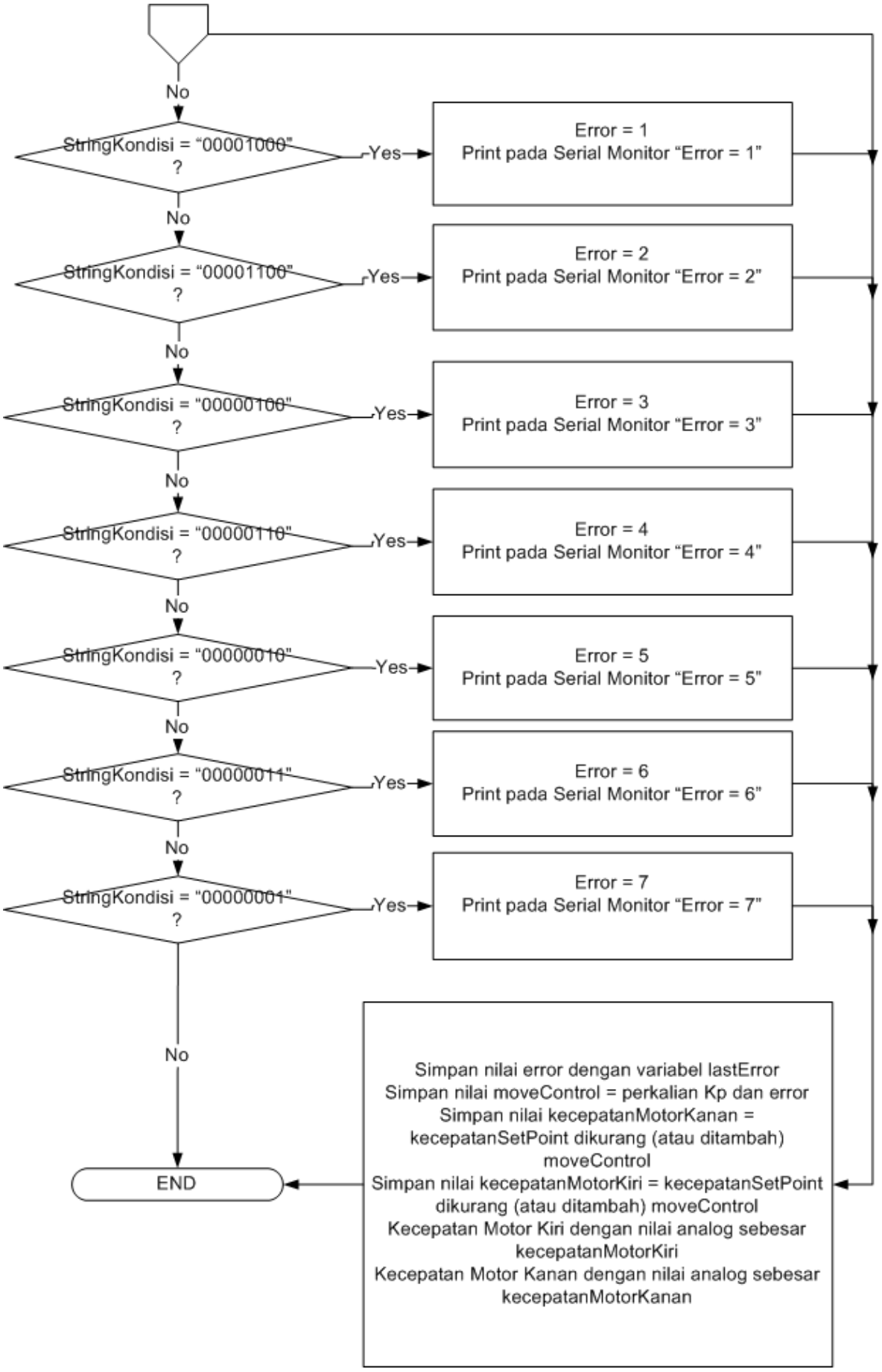
   Jelaskan fungsi dari variabel yang telah ditambahkan pada program di atas terhadap mekanisme sistem kendali pada robot *line follower*!

d. Kemudian tambahkan kode program dengan ketentuan sebagai berikut :
   1 Simpan nilai error saat ini dengan variabel **`Error`** dan nilai error sebelumnya dengan variable **`lastError`**.
   2 Simpan nilai selisih antara **`lastError`** dengan **`Error`** dalam variabel **`rate`**.
   3 Simpan nilai **`moveControl`** = perkalian **`Kp`** dan **`Error`** ditambah hasil perkalian **`Kd`** dan **`rate`**.
   4 Simpan nilai **`kecepatanMotorKanan`** = **`kecepatanSetPoint`** dikurang **`moveControl`**
   5 Simpan nilai **`kecepatanMotorKiri`** = **`kecepatanSetPoint`** ditambah **`moveControl`**
   6 Kecepatan Motor Kiri dengan nilai analog sebesar **`kecepatanMotorKiri`**
   7 Kecepatan Motor Kanan dengan nilai analog sebesar **`kecepatanMotorKanan`**

   Variasikan nilai konstanta Kp dan Kd pada program kemudian ujicoba dan amati hasil perubahan kedua konstanta tersebut!

   Jelaskan fungsi dari **`Kp`** dan **`Kd`** pada kode program di atas!


START
int Kp dengan nilai 5
int moveControl dengan nilai 0
int error dengan nilai 0
int kecepatanMotorKanan dengan nilai 0
int kecepatanMotorKiri dengan nilai 0
int kecepatanSetPoint dengan nilai 150
Set StringKondisi = "00000000"
Bit paling kiri adalah sensor ke-0
Bit paling kanan adalah sensor ke-7
Baca Nilai Sensor Photodioda
Sensor Deteksi Hitam → 1
Sensor Deteksi Putih → 0
StringKondisi = "10000000" ?
Yes
Error = -7
Print pada Serial Monitor "Error = -7"
No
StringKondisi = "11000000" ?
Yes
Error = -6
Print pada Serial Monitor "Error = -6"
No
StringKondisi = "01000000" ?
Yes
Error = -5
Print pada Serial Monitor "Error = -5"
No
StringKondisi = "01100000" ?
Yes
Error = -4
Print pada Serial Monitor "Error = -4"
No
StringKondisi = "00100000" ?
Yes
Error = -3
Print pada Serial Monitor "Error = -3"
No
StringKondisi = "00110000" ?
Yes
Error = -2
Print pada Serial Monitor "Error = -2"
No
StringKondisi = "00010000" ?
Yes
Error = -1
Print pada Serial Monitor "Error = -1"
No
StringKondisi = "00011000" ?
Yes
Error = 0
Print pada Serial Monitor "Error = 0"
No



Gambar 6 Flowchart sistem kendali PID kasus P.

e. *Screenshoot* keluaran serial monitor untuk setiap kondisi.
f. Isi tabel kebenaran dari sistem pada Tabel 2 dan tuliskan pada buku jurnal praktikum.

Tabel 2Tabel Kebenaran Sistem Kendali

Sensor								Error	lastError	Rate	Analog Value	
0	1	2	3	4	5	6	7				Motor Kiri	Motor Kanan

Fungsi Penyimpanan Pada EEPROM

g. Modifikasi program pada praktikum sebelumnya dengan variable dan fungsi berikut ini :
   - Tambahkan variabel integer dengan nama '**`state`**' dengan nilai awal adalah 0.
   - Tambahkan variabel integer dengan nama '**`setting`**' dengan nilai awal adalah 0.
   - Tambahkan variabel integer berupa array dengan nama '**`peka`**' dari 0 hingga 7 dengan nilai awal adalah 500.
   - Tambahkan variabel integer dengan nama '**`Kp`**' dari 0 hingga 7 dengan nilai awal adalah 20.
   - Tambahkan variabel integer dengan nama '**`Kd`**' dari 0 hingga 7 dengan nilai awal adalah 5.

   - Di dalam **`void setup`** tambahkan :
   - Baca nilai eeprom yang terdapat di EEPROM dengan alamat 0 hingga 7 (dengan ketentuan alamat 0 adalah untuk data nilai tengah kalibrasi sensor 1, alamat 1 untuk nilai tengah kalibrasi sensor 2, dan seterusnya hingga sensor 8) dengan perintah **`EEPROM.read(0)`** hingga **`EEPROM.read(7)`**, kemudian simpan didalam variabel **`peka[0]`** hingga **`peka[7]`** dengan mengkalikan dengan angka 4 (contoh : **`peka[0]=EEPROM.read[0]*4`**).
   - Baca nilai eeprom yang terdapat di EEPROM dengan alamat 8 (ketentuan alamat 8 adalah data nilai Kp), kemudian simpan didalam variabel **`Kp`**.

- Baca nilai eeprom yang terdapat di EEPROM dengan alamat 9 (ketentuan alamat 9 adalah data nilai Kd), kemudian simpan didalam variabel **`Kp`**.
- Tampilkan data **`peka[0]`** hingga **`peka[7], Kp`** dan **`Kd`** yang berasal dari data EEPROM ke Serial Monitor.

- Di dalam **`void loop`** ditambahkan *conditional* sebagai berikut :
  Referensi posisi push button dapat dilihat pada
- Jika tombol kiri belakang ditekan dan setting bernilai 0 maka **`'state'`** akan bernilai 1.
- Jika tombol kiri depan ditekan dan setting bernilai 0 maka **`'state'`** akan bernilai 2 dan 'setting' akan bernilai 1.
- Jika tombol kiri belakang ditekan dan setting bernilai 1 maka **`'state'`** akan bernilai 3.
- Jika tombol kiri depan ditekan dan setting bernilai 1 maka **`'state'`** akan bernilai 4.
- Jika tombol kanan belakang ditekan dan setting bernilai 1 maka **`'state'`** akan bernilai 5.
- Jika tombol kanan depan ditekan dan setting bernilai 1 maka **`'state'`** akan bernilai 6.

Perubahan state tersebut akan mengaktifkan sub program berikut :

- Jika **`'state'`** bernilai 1 maka proses *auto calibration* berlangsung.
- Jika **`'state'`** bernilai 2 maka nilai terakhir variabel **`'peka[0]`**' hingga **`'peka[7]`**' disimpan ke dalam EEPROM dengan alamat 0 hingga 7 dan menjalankan perintah proses *line follower* dengan sistem kendali PID kasus P dan D pada praktikum sebelumnya dengan nilai **`peka[0]`** hingga **`peka[7]`**, **`Kp`** dan **`Kd`** yang diperoleh dari data EEPROM dan atau berasal dari hasil autocallibration.
- Perintah **`EEPROM.write(alamat,value)`** dengan value yang dibagi dengan 4. Jelaskan mengapa value harus dikali dan dibagi 4!
  Contoh : **`EEPROM.write(0, peka[0]/4)`**

- Jika '**`state`**' bernilai 3 maka LED kiri akan menyala dan setelah 1 detik LED kiri akan mati, mengurangi nilai Kp(contoh : Kp=Kp-1;), menyimpan nilai Kp kedalam EEPROM pada alamat 8.
-
- Jika '**`state`**' bernilai 4 maka LED kiri akan menyala dan setelah 1 detik LED kiri akan mati, menambah nilai Kp(contoh : Kp=Kp+1;), menyimpan nilai Kp kedalam EEPROM pada alamat 8.
-

- Jika `'state'` bernilai 5 maka LED kanan akan menyala dan setelah 1 detik LED kiri akan mati, mengurangi nilai Kd(contoh : Kd=Kd-1;), menyimpan nilai Kp kedalam EEPROM pada alamat 9
-
- Jika `'state'` bernilai 6 maka LED kanan akan menyala dan setelah 1 detik LED kiri akan mati, menambah nilai Kd(contoh : Kd=Kd+1;), menyimpan nilai Kp kedalam EEPROM pada alamat 9

h. Variasikan nilai konstanta Kp dan Kd pada program kemudian ujicoba pada lintasan dan amati hasil perubahan kedua konstanta tersebut pada robot *line follower*! Apa yang menjadi kesimpulan dari hasil percobaan ini?

c. *Screenshoot* keluaran serial monitor untuk setiap kondisi. Cetak dan tempelkan pada buku jurnal praktikum.

6. Jurnal Praktikum

a. Jurnal pada Buku Praktikum harus memuat konten sebagai berikut :

- Judul Praktikum :
- Maksud dan Tujuan Praktikum :
- Peralatan dan Bahan Praktikum :
- Dasar Teori
- Foto Peralatan dan Bahan Praktikum :
- Hasil Praktikum (Tulis tangan kode program yang telah diberi komentar/penjelasan beserta foto hasil percobaan yang telah diberi nama dan NIM anggota kelompok)
- Kesimpulan Praktikum