MODUL 7 Sistem Kendali PID : Kasus P,I, dan D

Mata Kuliah : Sistem Kendali

Kode Dosen : AJR

Kelas : D3TK-43-02

Anggota Kelompok :

1. Muhammad Yogi (6702194045)
2. M Rifki Arya Syahputra (6702190010)

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

A. 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 berbasis timer untuk melakukan algoritma PID.
3. Mahasiswa dapat menggunakan peripheral berupa push button untuk menambah konstanta Kp, Ki, dan Kd.

B. Alat dan Bahan

Alat dan Bahan :

a. Robot Kit Line Follower
b. Baterai LiPo 2-Cell 1300 mAh
c. Kabel Mini-USB
d. Arduino Nano
e. Battery Checker
f. Battery Balancer

Perangkat Lunak :

a. Software IDE Arduino
b. Software Proteus (untuk simulasi)

C. Teori dasar

Sistem Kendali PD

Jika dengan kasus P dan D telah mengendalikan pergerakan robot dengancukup smooth, maka penambahan Integratif menjadi opsional. Komponen Integratif (I) digunakan untuk mengakumulasi error dan mengetahui durasi error. Dengan menjumlahkan error disetiap pembacaan process value (PV) akan memberikan akumulasi offset yang harus diperbaiki sebelumnya. Saat robot bergerak menjauhi garis, maka nilai error akan bertambah. Semakin lama tidak mendapatkan set point (SP), maka semakin besar nilai I. Dengan mendapatkan nilai Ki yang tepat, imbas dari Integratif bisa dikurangi. Nilai akumulasi error didapat dari: error + last_error. Proses perhitungan integral secara intuitif melibatkan komponen waktu, sehingga implementasinya dibutuhkan suatu fungsi timer pada mikrokontroler. Dalam Arduino untuk menghitung waktu atau fungsi timer digunakan fungsi millis() menghasilkan jumlah milidetik semenjak program berjalan. Fungsi ini memiliki tipe data Unsigned Long. Jumlah milidetik yang dihasilkan ini harus dicatat dalam setiap perulangan program.

Nilai konstanta perhitungan PID di-tuning secara trial and error dan proses ini dilakukan dengan metode eksperimental. Nilai proporsional, derivative, dan integratif pada algoritma PID diujicoba hingga ditemukan hasil sistem yang stabil. Adapun cara yang dilakukan untuk men-tuning PID pada robot line follower adalah sebagai berikut:

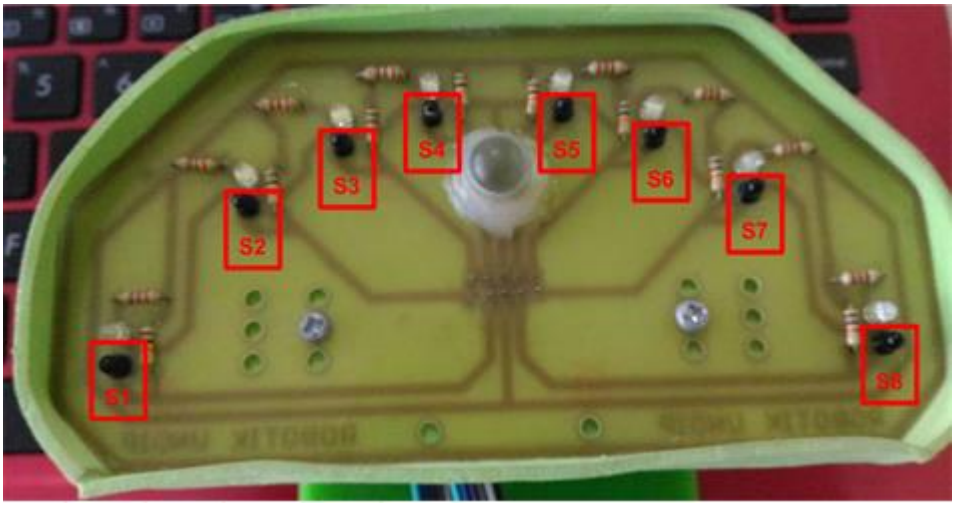
1. Langkah awal gunakan kontrol proporsional terlebih dahulu, abaikan konstanta integrative dan derivatifnya dengan memberikan nilai nol pada integratif dan derivatif.
2. Tambahkan terus konstanta proporsional maksimum hingga keadaan stabil namun robot masih berosilasi.

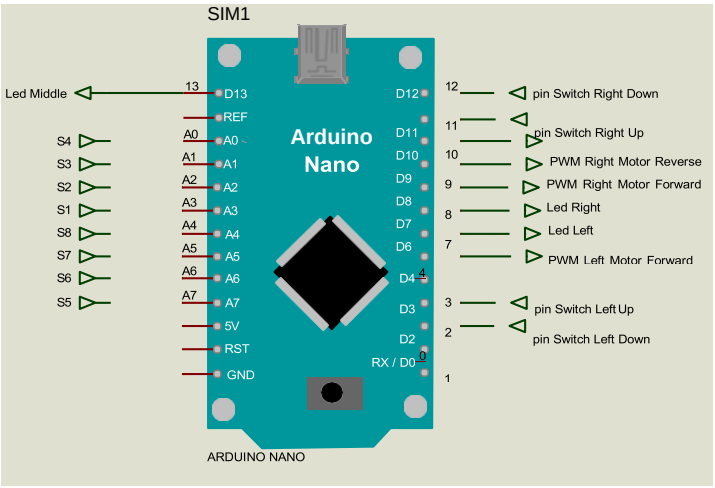
3. Untuk meredam osilasi, tambahkan konstanta derivatif dengan membagi dua nilai proporsional, amati keadaan sistem robot hingga stabil dan lebih responsif.
4. Jika sistem robot telah stabil, kontrol integratif dapat menjadi opsional, dalam artian jika ingin mencoba-coba tambahkan kontrol integratif tersebut, namun pemberian nilai integratif yang tidak tepat dapat membuat sistem robot menjadi tidak stabil.
5. Nilai set point kecepatan dan nilai batas bawah/atas memberikan patokan kecepatan robot.
6. Nilai time sampling (waktu cuplik) juga mempengaruhi perhitungan PID, tentunnya saat penggunaan kontrol integratif dan derivatif.
7. Periksa kembali perfoma sistem hingga mendapatkan hasil yang memuaskan.

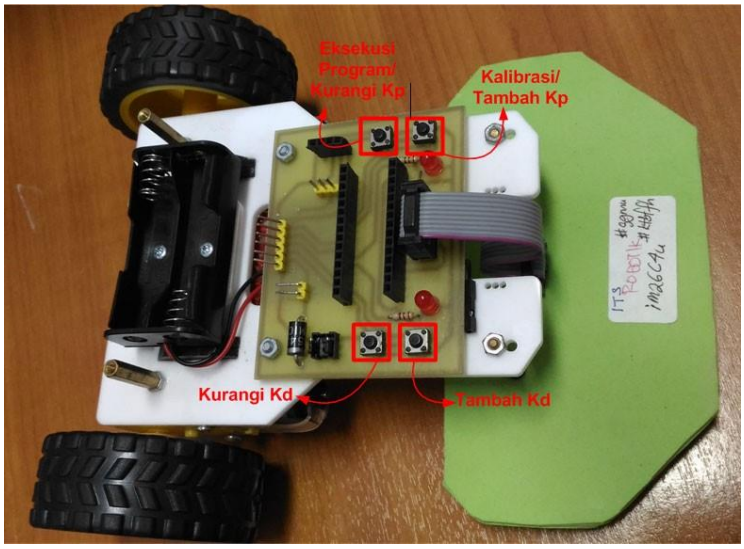
D. Hasil Percobaan

A. Percobaan dalam praktikum

1. Kasus Percobaan







a. 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 errSum
- Tambahkan variabel unsigned long SampleTime dengan nilai awal 1000. Angka 1000 menunjukkan nilai 1 detik.
- Tambahkan variabel now, lastTime, dan timeChange dengan tipe data unsigned long.
- Tambahkan variabel integer dengan nama 'Kp' dengan nilai awal adalah 20.
- Tambahkan variabel integer dengan nama 'Ki' dengan nilai awal adalah 20.
- Tambahkan variabel integer dengan nama 'Kd' 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.
- Jika tombol kanan dan kiri depan ditekan secara bersamaan dan setting bernilai 1 maka 'state' akan bernilai 7.
- Jika tombol kanan dan kiri belakang ditekan secara bersamaan dan setting bernilai 1 maka 'state' akan bernilai 8.
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 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 500

milidetik 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 500 milidetik 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 500 milidetik LED kiri akan mati, mengurangi nilai Kd(contoh : Kd=Kd-1;), menyimpan nilai Kd kedalam EEPROM pada alamat 9
- Jika 'state' bernilai 6 maka LED kanan akan menyala dan setelah 500 milidetik LED kiri akan mati, menambah nilai Kd(contoh : Kd=Kd+1;), menyimpan nilai Kd kedalam EEPROM pada alamat 10
- Jika 'state' bernilai 7 maka LED kiri dan kanan akan menyala dan setelah 500 milidetik LED kiri dan kanan akan mati, menambah nilai Ki(contoh : Ki=Ki+1;), menyimpan nilai Ki kedalam EEPROM pada alamat 11.
- Jika 'state' bernilai 8 maka LED kiri dan kanan akan menyala dan setelah 500 milidetik LED kiri dan kanan akan mati, mengurangi nilai Ki(contoh : Ki=Ki-1;), menyimpan nilai Ki kedalam EEPROM pada alamat 12.

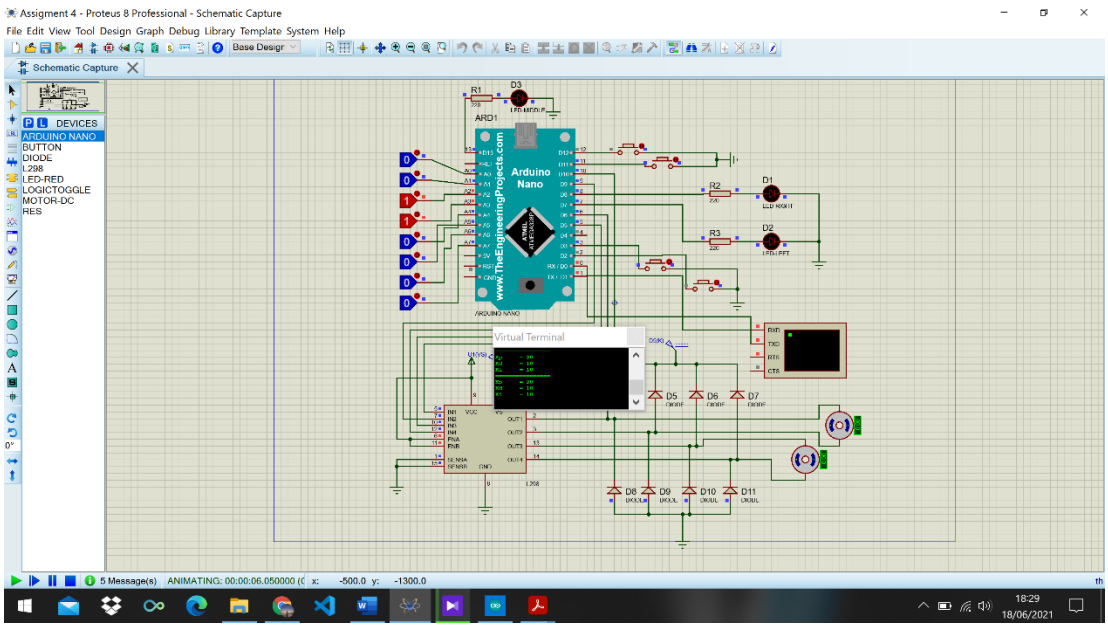
b. Gunakan fungsi millis() untuk menghitung jumlah milidetik semenjak program berjalan. Output dari fungsi ini memiliki tipe data Unsigned Long. Simpan output fungsi ini dalam variable now. Hitung selisih antara variabel waktu now dengan lastTime dalam variable timeChange.

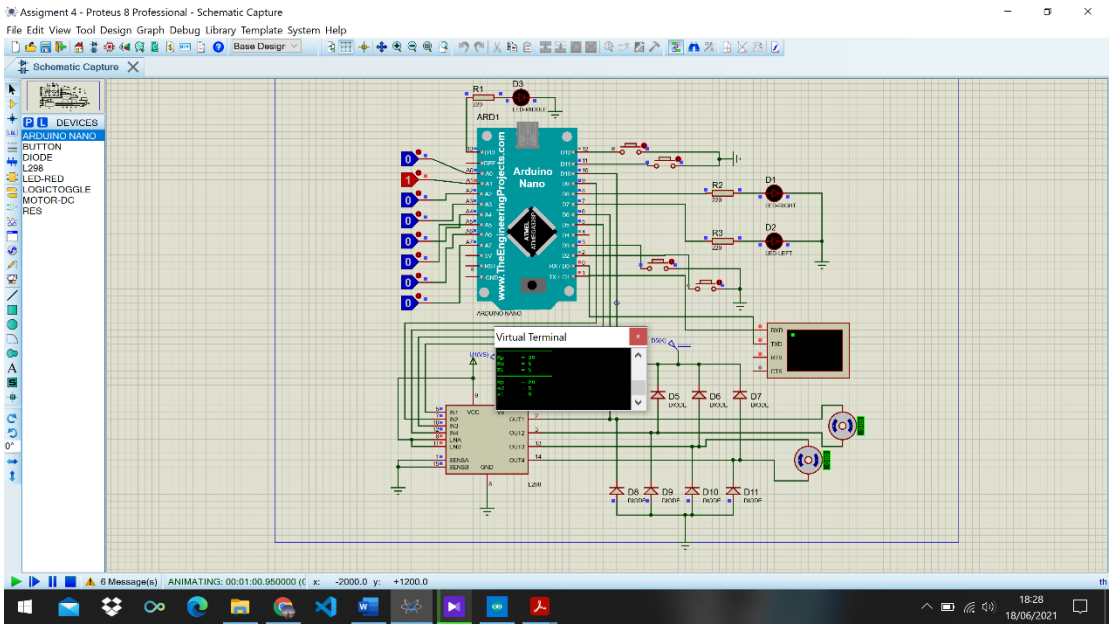
c. Apabila nilai timeChange lebih besar sama dengan nilai SampleTime, maka lakukan proses perhitungan algoritma PID adalah sebagai berikut :
- Simpan nilai error saat ini dengan variabel error dan nilai error sebelumnya dengan variable lastError.
- Simpan nilai selisih antara lastError dengan error dalam variabel rate.
- Simpan nilai penjumlahan antara hasil perkalian error dan timeChange dengan errSum dalam variable errSum. Hak ini menunjukkan bahwa nilai variabel errSum selalu di-update setiap saat.
- Simpan nilai moveControl =(Kp x Error)+(Ki x errSum)+(Kd x

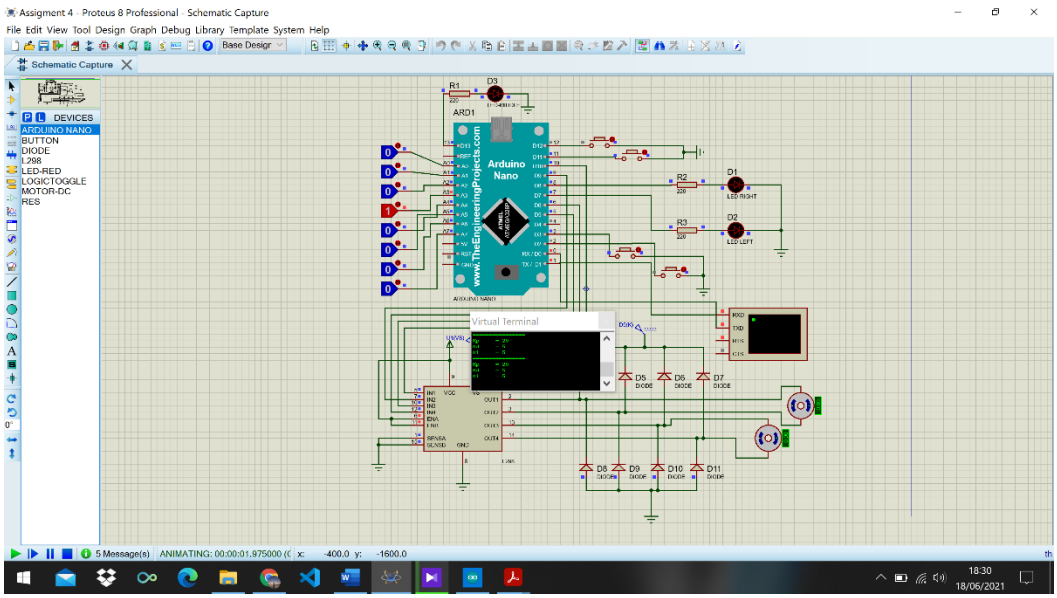
rate).

- Simpan nilai kecepatanMotorKanan = kecepatanSetPoint dikurang moveControl
- Simpan nilai kecepatanMotorKiri = kecepatanSetPoint ditambah moveControl
- Kecepatan Motor Kiri dengan nilai analog sebesar kecepatanMotorKiri
- Kecepatan Motor Kanan dengan nilai analog sebesar kecepatanMotorKanan

d. Variasikan nilai konstanta Kp, Ki, 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?







E. Kesimpulan

Kesimpulan pada praktikum ini kita mengerti Sistem kendali PID ini bertujuan untuk menentukan paramater aksi kendali Proportional, Integratif, Derivatif pada robot line follower, membuat rangkaian dengan memuncul kan error. Kp mampu untuk memperbaiki respon transien khususnya rise time dan settling time. Kd hanya berubah saat ada perubahan error sehingga saat error statis kontrol ini tidak akan bereaksi,

F. Link

Link github : https://github.com/rifkiaryas/Kelompok_M-Yogi-M-Rifki-Arya.git