MODUL 4
Sistem Kendali berbasis *Fuzzy logic* 1 : *Adaptive Speed Control*

1. JUDUL PRAKTIKUM

Sistem Kendali berbasis *Fuzzy logic* : *Adaptive Speed Control*

2. MAKSUD DAN TUJUAN

Maksud dan tujuan dari praktikum ini adalah :

1. Mahasiswa dapat memahami fungsi dan cara kerja *fuzzy logic* untuk mengendalikan kecepatan motor DC pada robot line follower
2. Mahasiswa dapat membuat program berbasis timer untuk melakukan algoritma *fuzzy logic* pada berbagai kondisi kecepatan.

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

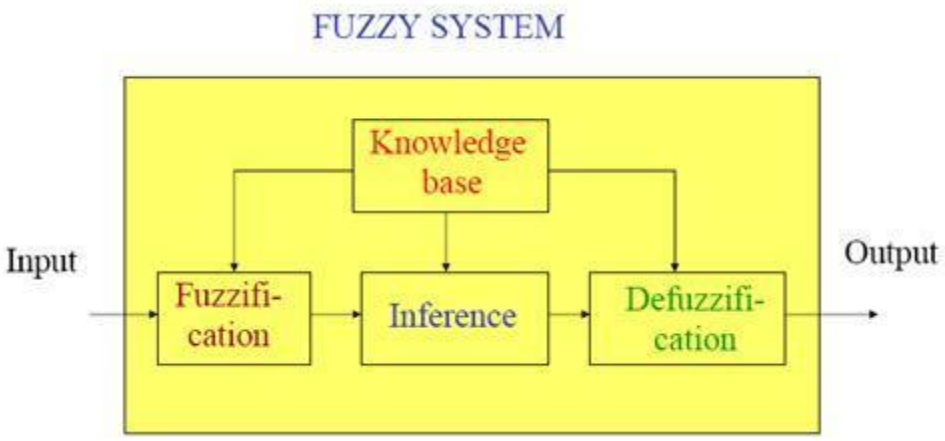
5.1. Sejarah Perkembangan *Fuzzy logic*

Fuzzy Set pertama kali diperkenalkan oleh Prof. Lotfi Zadeh, 1965 orang Iran yang menjadi guru besar di University of California at Berkeley dalam papernya yang monumental “Fuzzy Set”. Dalam paper tersebut dipaparkan ide dasar fuzzy set yang meliputi inclusion, union, intersection, complement, relation dan convexity.

Lotfi Zadeh mengatakan Integrasi Logika Fuzzy kedalam sistem informasi dan rekayasa proses adalah menghasilkan aplikasi seperti sistem kontrol, alat alat rumah tangga, dan sistem pengambil keputusan yang lebih fleksibel, mantap, dan canggih dibandingkan dengan sistem konvensional. Produk produk berikut telah menggunakan logika fuzzy dalam alat alat rumah tangga seperti mesin cuci, video dan kamera refleksi lensa tunggal, pendingin ruangan, oven microwave, dan banyak sistem diagnosa mandiri. Pelopor aplikasi fuzzy set dalam bidang kontrol, adalah Prof. Ebrahim Mamdani dkk dari Queen Mary College London. (masih dalam skala lab). Penerapan secara nyata di industri banyak dipelopori oleh para ahli dari Jepang misalnya Prof. Sugeno dkk dari Tokyo Institute of Technology.

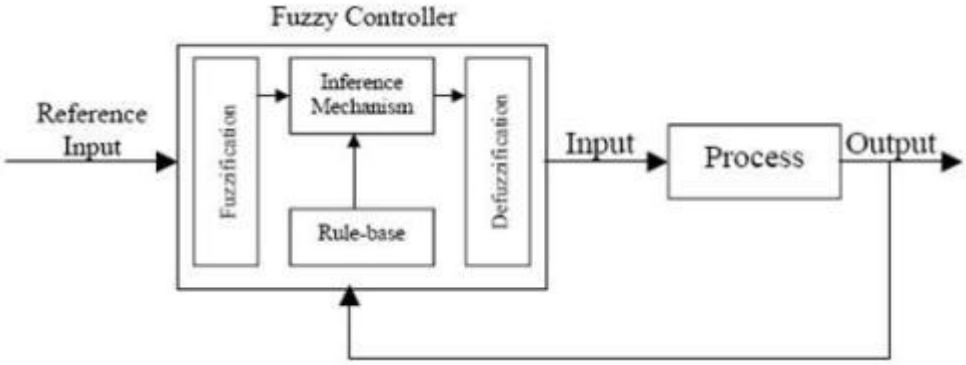
5.2. Tahap Pemodelan pada *Fuzzy logic* untuk Robot Line Follower

Proses fuzzy yang dilakukan pada sistem kendali robot line follower umumnya sama dengan proses fuzzy pada sistem kendali lainnya yaitu meliputi fuzzifikasi, evaluasi rule dan defuzzifikasi. Dengan menggunakan algoritma ini robot diharapkan dapat bergerak mengikuti jalur dengan sesuai dan cepat.



Gambar 1 Pemodelan Fuzzy logic.

Penerapan *fuzzy logic* pada sistem kendali dapat dilihat pada gambar berikut.

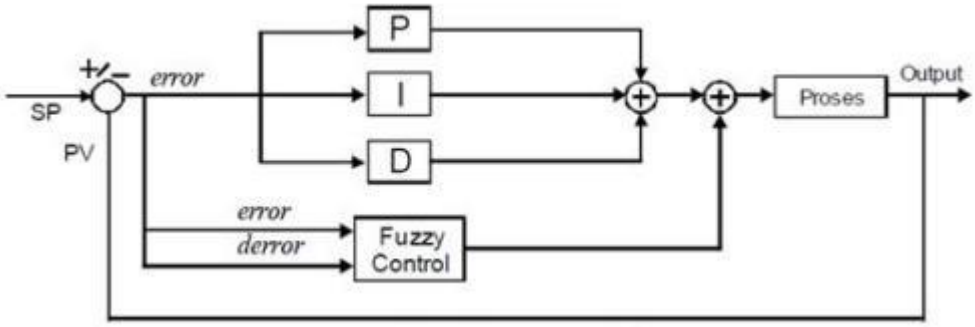


Gambar 2 Diagram blok sistem kendali berbasis fuzzy logic.

Dari gambar tersebut dapat dipahami bahwa di dalam *fuzzy logic* terdapat tiga hal terpenting yang harus diperhatikan yaitu:

1. **Fuzzifikasi** adalah proses untuk mengubah variabel non fuzzy (variabel numerik) menjadi variabel fuzzy (variabel linguistik).
2. **Inferencing (Ruled Based)** , pada umumnya aturan-aturan fuzzy dinyatakan dalam bentuk "**IF……THEN**" yang merupakan inti dari relasi fuzzy.
3. **Defuzzifikasi adalah** proses pengubahan data-data fuzzy tersebut menjadi data-data numerik yang dapat dikirimkan ke peralatan pengendalian yang dalam hal ini adalah mengubah kecepatan dari motor DC.

Diagram blok sistem kendali berbasis *fuzzy logic* untuk kendali kecepatan motor DC dapat dilihat pada Gambar 3.

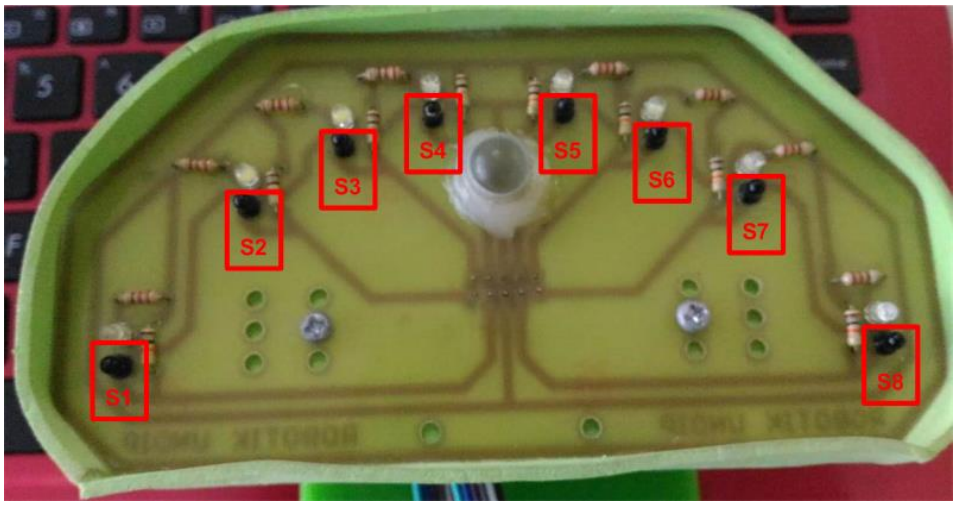


Gambar 3 Diagram blok sistem kendali fuzzy logic dan PID (hybrid).

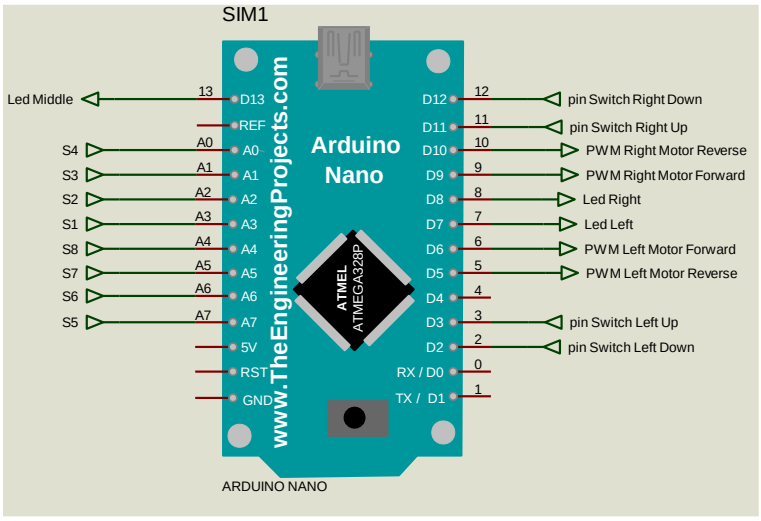
PROSEDUR PRAKTIKUM

A. Percobaan dalam praktikum

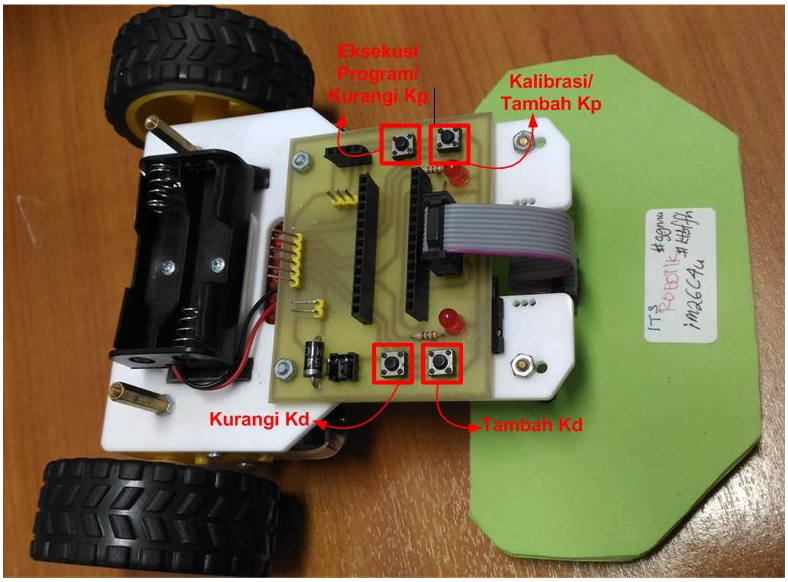
1. Kasus Percobaan



Gambar 4 Contoh susunan dan urutan sensor pada robot line follower.



Gambar 5 Pin Layout Arduino pada Robot Line Follower.



Gambar 6 Posisi dan fungsi push button pada robot line follower.

a. Kalibrasi kedua motor DC pada robot line follower dengan cara membuat program untuk menjalankan kedua motor tersebut dengan nilai PWM yang sama. Apabila terdapat perbedaan kecepatan di antara kedua motor tersebut, lakukan proses *tuning* dengan cara menambah atau mengurangi nilai PWM dari motor tersebut.
b. Modifikasi program pada praktikum sebelumnya dengan variable dan fungsi berikut ini :
   - Tambahkan variabel integer dengan nama '**`kecepatanNow`**'

   - 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

c. Gunakan fungsi **millis()** untuk menghitung jumlah milidetik semenjak program berjalan. Output dari fungsi ini memiliki tipe data **Unsigned Long**. Buatlah sub program dengan menggunakan fungsi millis() yang akan melakukan fungsi timer 2 detik (nilai timer dapat disesuaikan dengan kebutuhan dan kondisi di lapangan) secara berulang-ulang dan memeriksa nilai error saat ini. Apabila ketika nilai timer bernilai 2 detik, maka lakukan proses adaptive speed control sebagai berikut :

d. Cek nilai error, apabila nilai error saat ini = 0 maka tingkatkan nilai kecepatanSetPoint bertambah 20 (sesuaikan penambahan setpoint dengan kondisi) dan disimpan dalam variable **kecepatanNow**.

e. Apabila nilai error selain 0, maka nilai setpoint kembali ke 150.

f. Simpan nilai kecepatanMotorKanan = kecepatanNow dikurang moveControl

g. Simpan nilai kecepatanMotorKiri = kecepatanNow ditambah moveControl

h. Kecepatan Motor Kiri dengan nilai analog sebesar kecepatanMotorKiri

i. Kecepatan Motor Kanan dengan nilai analog sebesar kecepatanMotorKanan

j. Proses ini dilakukan secara berulang (counter) selama program berjalan.

k. Gunakan nilai Kp dan Kd yang paling optimal pada praktikum sebelumnya kemudian amati perubahan yang terlihat setelah ditambahkan algoritma *fuzzy logic*!

l. Sesuaikan nilai waktu pada timer/counter pada program kemudian ujicoba pada lintasan dan amati hasil perubahan nilai tersebut pada robot *line follower*! Apa yang menjadi kesimpulan dari hasil percobaan ini?

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

m. Isi tabel kebenaran dari sistem pada Tabel 1 dan tuliskan pada buku jurnal praktikum.

Tabel 1Tabel Kebenaran Sistem Kendali PD

Sensor								Kp	Kd	Error	kecepatanNow	Analog Value	
0	1	2	3	4	5	6	7					Motor Kiri	Motor Kanan

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