TUGAS AKHIR – EE 184801

DESAIN KONTROLER BLDC MOTOR UNTUK KENDARAAN LISTRIK MENGGUNAKAN *FUZZY LOGIC CONTROLLER*

Deksaraka Danier
NRP 07111640000010

Dosen Pembimbing
Feby Agung Pamuji, ST., MT., Ph.D.
Dr. Ir. Soedibyo, MMT.

DEPARTEMEN TEKNIK ELEKTRO
Fakultas Teknologi Elektro dan Informatika Cerdas
Institut Teknologi Sepuluh Nopember
Surabaya 2020

TUGAS AKHIR – EE 184801

DESAIN KONTROLER BLDC MOTOR UNTUK KENDARAAN LISTRIK MENGGUNAKAN *FUZZY LOGIC CONTROLLER*

Deksaraka Danier
NRP 07111640000010

Dosen Pembimbing
Feby Agung Pamuji, ST., MT., Ph.D.
Dr. Ir. Soedibyo, MMT.

DEPARTEMEN TEKNIK ELEKTRO
Fakultas Teknologi Elektro dan Informatika Cerdas
Institut Teknologi Sepuluh Nopember
Surabaya 2020

FINAL PROJECT – EE 184801

BLDC MOTOR SPEED CONTROLLER DESIGN FOR ELECTRIC VEHICLE USING FUZZY LOGIC CONTROLLER

Deksaraka Danier
NRP 07111640000010

Supervisors
Feby Agung Pamuji, ST., MT., Ph.D.
Dr. Ir. Soedibyo, MMT.

DEPARTMENT OF ELECTRICAL ENGINEERING
Faculty of Electrical Technology
Institut Teknologi Sepuluh Nopember
Surabaya 2020

PERNYATAAN KEASLIAN
TUGAS AKHIR

Dengan ini saya menyatakan bahwa seluruh isi pada tugas akhir ini dengan judul "**DESAIN KONTROLLER BLDC MOTOR UNTUK KENDARAAN LISTRIK MENGGUNAKAN *FUZZY LOGIC CONTROLLER***" adalah merupakan hasil karya intelektual mandiri, diselesaikan tanpa menggunakan sumber materi yang tidak diizinkan dan bukan merupakan karya pihak lain yang saya akui sebagai karya sendiri.

Semua referensi yang dikutip maupun dirujuk telah ditulis secara lengkap pada daftar pustaka.

Apabila ternyata pernyataan ini tidak benar, saya bersedia menerima sanksi sesuai peraturan yang berlaku.

Surabaya, Juli 2020

Deksaraka Danier
07111640000010

DESAIN KONTROLLER BLDC MOTOR UNTUK KENDARAAN LISTRIK MENGGUNAKAN *FUZZY LOGIC CONTROLLER*

TUGAS AKHIR

Diajukan Guna Memenuhi Sebagian Persyaratan
Untuk Memperoleh Gelar Sarjana Teknik

Pada

Bidang Studi Teknik Sistem Tenaga
Departemen Teknik Elektro
Fakultas Teknologi Elektro dan Informatika Cerdas
Institut Teknologi Sepuluh Nopember

Menyetujui :

Dosen Pembimbing I

Feby Agung Pamuji, ST, MT, Ph.D
NIP. 198702062012121002

SURABAYA
JULI, 2020

DESAIN KONTROLLER BLDC MOTOR UNTUK KENDARAAN LISTRIK MENGGUNAKAN *FUZZY LOGIC CONTROLLER*

TUGAS AKHIR

Diajukan Guna Memenuhi Sebagian Persyaratan
Untuk Memperoleh Gelar Sarjana Teknik

Pada

Bidang Studi Teknik Sistem Tenaga
Departemen Teknik Elektro
Fakultas Teknologi Elektro dan Informatika Cerdas
Institut Teknologi Sepuluh Nopember

Menyetujui :

Dosen Pembimbing II

Dr. Ir. Soedibyo, MMT.
NIP. 195512071980031004

SURABAYA
JULI, 2020

DESAIN KONTROLLER BLDC MOTOR UNTUK KENDARAAN LISTRIK MENGGUNAKAN *FUZZY LOGIC CONTROLLER*

Nama : Deksaraka Danier
Dosen Pembimbing I : Feby Agung Pamuji, ST, MT, Ph.D
Dosen Pembimbing II : Dr. Ir. Soedibyo, MMT

ABSTRAK

Semakin bertambahnya jumlah kendaraan yang menggunakan bahan bakar fossil pada jaman sekarang mengakibatkan meningkatnya jumlah polusi udara. Untuk mengurangi jumlah polusi udara yang ada maka penggunaan kendaraan konvensional yang menghasilkan polusi udara harus dikurangi, salah satu caranya dengan menggantinya dengan kendaraan listrik. Adanya teknologi seperti motor listrik seperti Brushless DC Motor menjadi salah satu solusi pengganti mesin berbahan bakar fossil. Brushless DC Motor (BLDC) Motor merupakan motor sinkron tanpa sikat dengan permanen magnet pada bagian rotor, dan belitan pada bagian stator. Agar BLDC motor dapat berputar dengan kecepatan yang terkontrol, maka diperlukan sistem tertutup yang dapat memperbaiki kecepatan actual atau eror saat BLDC motor berputar. Pada tugas akhir ini metode kontrol kecepatan yang digunakan yaitu Pulse Width Modulation (PWM) lalu feedback kecepatan aktualnya akan dikontrol menggunakan fuzzy logic agar kecepatan aktualnya dapat diatur sesuai dengan kecepatan yang diinginkan. Maka dari itu dibuatlah simulasi serta implementasi dan didapatkan data bahwa hasil implementasi yang telah dibuat sudah mendekati dari hasil simulasi. Perbedaan hasil antara implementasi dan simulasi ini karena saat implementasi sensor kecepatan yang digunakan memiliki rata – rata error sebesar 2,11% sehingga mengakibatkan kecepatan actual yang dihasilkan juga berubah – ubah. Dari data hasil implementasi, error yang terjadi hingga 1.28 %

Kata kunci : Fuzzy Logic Controller, BLDC, PWM, Mikrokontroller

--- halaman ini sengaja dikosongkan ---

BLDC Motor Controller Design for Electric Vehicles Using Fuzzy Logic Controller

Name : Deksaraka Danier
Supervisor I : Feby Agung Pamuji, ST, MT, Ph.D
Supervisor II : Dr. Ir. Soedibyo, MMT

ABSTRACT

The increasing number of vehicles that use fossil fuels in recent times has resulted in an increase in the amount of air pollution. To reduce the amount of air pollution that exists, the use of conventional vehicles that produce air pollution must be reduced, one way to replace it with electric vehicles. The existence of technologies such as electric motors such as the Brushless DC Motor becomes one of the solutions to replace fossil-fueled engines. Brushless DC Motor (BLDC) The motor is a synchronous motor without a brush with permanent magnets on the rotor section, and windings on the stator section. In order for the BLDC motor to rotate at a controlled speed, a closed system is needed which can correct the actual speed or error when the BLDC motor is spinning. In this final project the speed control method used is Pulse Width Modulation (PWM) and then the actual speed feedback will be controlled using fuzzy logic so that the actual speed can be set according to the desired speed. Therefore the simulation and implementation are made and the data obtained that the results of the implementation that have been made are approaching from the simulation results. The difference between the results of the implementation and the simulation is because when implementing the speed sensor used has an average error of 2.11% so that the resulting actual speed also varies. From the data of the implementation, the error occurred was 1.28%

Keywords : Fuzzy Logic Controller, BLDC, PWM, Microcontroller

--- halaman ini sengaja dikosongkan ---

KATA PENGANTAR

Dengan menyebut nama Allah SWT yang Maha Pengasih lagi Maha Penyayang, kami panjatkan puja dan puji syukur atas kehadirat-Nya, yang telah melimpahkan rahmat, hidayah, dan inayah-Nya kepada kami, sehingga kami dapat menyelesaikan tugas akhir dengan judul "**DESAIN KONTROLLER BLDC MOTOR UNTUK KENDARAAN LISTRIK MENGGUNAKAN *FUZZY LOGIC CONTROLLER"*** dengan tepat waktu.

Tugas akhir ini disusun sebagai salah satu persyaratan untuk menyelesaikan pendidikan S1 pada Bidang Studi Teknik Sistem Tenaga, Departemen Teknik Elektro, Fakultas Teknologi Elektro, Institut Teknologi Sepuluh Nopember. Atas selesainya pembuatan laporan tugas akhir ini, penulis mengucapkan terima kasih kepada:

1. Keluarga Penulis terutama ayah, ibu, yang telah merawat sejak dari kecil hingga sekarang serta selalu mendukung baik secara moral dan material. Kakak dan adik saya yang selalu menjadi penyemangat penulis dalam menyelesaikan tugas akhir.
2. Bapak Dedet C. Riawan, ST., M.Eng., Ph.D selaku kepala Departemen Teknik Elektro, Fakultas Teknologi Elektro, Institut Teknologi Sepuluh Nopember.
3. Bapak Feby Agung Pamuji, ST, MT, Ph.D. dan Dr. Ir. Soedibyo, MMT selaku dosen pembimbing yang telah memberikan arahan dan petunjuk dalam menyelesaikan laporan tugas akhir.
4. Seluruh dosen dan karyawan Departemen Teknik Elektro ITS yang telah memberikan banyak ilmu pada saat perkulihaan maupun diluar perkuliahan.
5. Hasbi dan Dzulfikar yang telah meminjamkan peralatan untuk pembuatan prototype pada tugas akhir ini.
6. Teman-teman e-56 khususnya Dhiaul, Revo, Miftah, dan teman - teman lab Konversi Energi yang saling berjuang dan memberi semangat satu sama lain dalam pengerjaan tugas akhir.

Surabaya, Juli 2020

Penulis

--- halaman ini sengaja dikosongkan ---

DAFTAR ISI

DAFTAR GAMBAR

DAFTAR TABEL

--- halaman ini sengaja dikosongkan ---

BAB 1
PENDAHULUAN

1.1. Latar Belakang

Semakin bertambahnya jumlah kendaraan yang menggunakan bahan bakar fossil pada jaman sekarang mengakibatkan meningkatnya jumlah polusi udara. Untuk mengurangi jumlah polusi udara yang ada maka penggunaan kendaraan konvensional yang menghasilkan polusi udara harus dikurangi, salah satu caranya dengan menggantinya dengan kendaraan listrik.

Disisi lain, adanya teknologi Brushless motor DC yang merupakan salah satu jenis motor listrik tanpa menggunakan sikat dimana komutasinya secara elektrik berdasarkan posisi rotonya. Motor Brushless DC ini sangat banyak diaplikasikan pada bidang indurstri karena tingkat efisiensi dan keandalannya yang tinggi, motor Brushless DC juga memliki konstruksi sederhana, pemeliharaan yang mudah, dan memliki keunggulan rasio/inersia yang tinggi. Salah satu metode yang digunakan untuk mengatur kecepatan pada motor Brushless DC adalah dengan cara mengatur tegangan pada sisi stator menggunakan metode PWM. Keunggulan dari metode tersebut yaitu mempunyai struktur yang tidak rumit dan sudah umum diaplikasikan pada motor Brushless DC. Variasi kecepatan dari motor Brushless DC motor dapat diubah dengan mengatur duty cycle pada PWM.

Untuk mengatur duty cycle pada PWM ini diperlukan sistem kontrol tertutup atau close loop dengan menggunakan metode Fuzzy Logic Contreller. Metode digunakan sebagai respon feedback kecepatan actual motor yang kemudian dibandingkan dengan nilai set point yang telah ditentukan dan keluaran hasil perbandingan ini dapat digunakan untuk mengatur besarnya PWM agar mendapatkan kecepatan pada motor Brushless DC.

1.2. Perumusan Masalah

Perumusan masalah dari tugas akhir ini adalah:

1. Mensimulasi Fuzzy Logic Controller pada brushless DC
2. Mendesain program fuzzy logic speed control pada mikrokontroler menggunakan metode kontrol tegangan ESC dengan PWM untuk mengatur kecepatan pada Brushless DC

1.3. Tujuan Tugas Akhir

Tugas akhir ini bertjuan untuk mendapatkan prototype sistem implementasi dari kendaraan listrik menggunakan motor Brushless DC dengan metode fuzzy logic sebagai kontrol kecepatannya

1.4. Batasan Masalah

Pada penelitian ini, ditentukan batasan-batasan masalah sebagai berikut :

1. Beban yang digunakan pada BLDC motor tetap atau konstan
2. Motor brushless DC yang digunakan adalah A2212/13T
3. Sumber yang digunakan pada penelitian ini adalah adaptor 12 Volt, 2 Ampere
4. Penggunaan *electric speed control* sebagai pengaturan kecepatan motor brushless DC.
5. Mikrokontroller menggunakan *Arduino Uno*
6. Implementasi alat adalah sebuah prototype
7. Prototype didesain untuk kondisi atau keadaan jalan datar, bukan untuk kondisi jalan menanjak maupun menurun
8. Kontrol kecepatan menggunakan fuzzy logic

1.5. Metodelogi

Adapun metodelogi yang digunakan pada penelitian ini adalah sebagai berikut

1. Studi Literatur

Dalam Studi literature, penulis mempelajari dasar – dasar tentang motor bushless DC, mencakup prinsip kerja, jenis – jenis motor, serta analisis – analisis terkait dengan motor brushless DC. Selain itu juga mempelajari fuzzy logic controller, pengaturan PWM, electric speed control, dan sensor kecepatan.

Literatur yang digunakan dalam studi literatur ini diambil dari buku, jurnal ilmiah, serta artikel.

2. Identifikasi dan perancangan sistem

Pada tahap ini akan dilakukan identifikasi serta perancangan sistem yang akan dibuat. Berikut merupakan tahap – tahap identifikasi dan perancangan sistem :

a. Menentukan parameter – parameter pada sistem yang akan digunakan (Motor Brushless DC, tegangan suplai DC, ESC, Fuzzy Logic, sensor kecepatan)
b. Penentuan nilai – nilai dari komponen tiap alat
c. Skema kontrol untuk mengatur PWM menggunakan mikrokontroler arduino

3. Simulasi Sistem

Tahap ini adalah mensimulasikan sistem secara keseluruhan. Dimana setelah menentukan nilai – nilai komponen dan merancang sistem, hasil dari perancangan tersebut dapat digunakan pada tahap simulasi. Simulasi akan dilakukan dengan menggunakan software MATLAB. Data yang diperoleh saat identifikasi dan perancangan sistem akan dimasukkan ke tiap komponen sistem yang ada pada software, dan setelah itu simulasi dapat dilakukan

4. Analisis Data

Simulasi yang dilakukan akan menghasilkan data yang akan dianalisis seperti kecepatan yang dihasilkan brushless DC, respon logika fuzzy.

5. Implementasi Sistem

Simulasi yang telah dilakukan akan menentukan nilai – nilai yang akan sesuai dengan sistem control pada Brushless DC motor. Dan kemudian akan digunakan pada tahap perancangan prototype/hardware.

6. Penyusunan Laporan

Semua proses dan hasil penelitian ini akan ditulis dan disusun dalam laporan sebagai hasil penelitian

1.6. Sistematika Penulisan

Sistematika penulisan dibagi menjadi lima bab yang terdiri dari :

- Bab 1. Pendahuluan

  Bab pertama ini membahas mengenai latar belakang masalah, permasalahan, tujuan, metodologi, sistematika penulisan, dan relevansi.

- Bab 2. Tinjauan Pustaka

  Bab kedua ini menjelaskan tentang dasar teori yang berkaitan dengan tugas akhir ini, seperti dasar teori tentang kontroler fuzzy, brushless dc motor, sensor kecepatan, dan electronic speed controller (ESC).

- Bab 3. Desain dan Permodelan Sistem

  Bab ketiga ini menjelaskan tentang bagaimana skema yang akan digunakan untuk mengatur kecepatan motor brushless DC, mengatur PWM untuk ESC, dan feedback sistem dari mikrokontroller yang telah menerima referensi kecepatan dan feedback kecepatan actual dari sensor kecepatan.

- Bab 4. Hasil Simulasi dan Pembahasan

  Bab ini akan dibahas mengenai hasil perancangan system prototype dan analisis yang akan dibandingkan dengan perhitungan simulasi.

- Bab 5. Penutup

  Pada bab terakhir ini akan dipaparkan kesimpulan dari penelitian dan saran untuk penelitian ini, agar kedepannya dapat menjadi refensi yang berguna dalam pengembangan sistem kontrol kecepatan motor Brushless DC.

1.7. Relevansi

Dari hasil penelitian tugas akhir ini memiliki harapan agar dapat memberikan manfaat sebagai referensi bagi peneliti lain yang ingin melakukan penelitian tentang sistem konrol kecepatan motor brushless DC berbasis fuzzy

BAB 2
BRUSHLESS DC MOTOR, ELECTRONIC SPEED CONTROLLER, SENSOR INFRA MERAH, DAN FUZZY LOGIC

Pada bagian ini akan dipaparkan mengenai tentang tinjauan pustaka atau dasar teori Brushless DC Motor, dan teori pendukung terkait yang relevan terhadap tugas akhir ini, seperti mengenai kontroler fuzzy, kontrol tegangan berbasis PWM, dan sensor kecepatan.

2.1. Brushsless DC Motor

Motor BLDC memiliki konstruksi yang berbeda dengan motor dc pada umumnya, dimana motor ini memiliki tiga fasa yang bebeda sehingga masing-masing fasa memiliki sudut sebesar 120^0, sehingga motor BLDC menggunakan hall effect yang digunakan untuk mendeteksi posisi dimana rotor berada dengan menggunakan signal dari komutasi motor

Pada rotor *brushless* DC terdapat shaft dan permanent magnet. Motor *brushless* DC memiliki prinsip kerja berdasarkan gaya tarik dan gaya lawan antara kutub magnet. Arus melewati salah satu dari kumparan stator, dan menghasilkan kutub magnet yang akan menarik kutub yang berlawanan dari magnet permanent yang terdekat. dengan secara bergantian mengalirkan arus pada kumpara stator, maka akan menyebabkan rotor akan berputar.

Pada *brushless* DC tegangan tiap fasa pada kumparan motor dapat dirumuskan dalam persamaan :

$$u_x = R_x i_x + e_{\psi x} \qquad (1.1)$$

Dimana i, u, dan R merupakan parameter arus, tegangan, dan resistansi pada fasa x (fasa A, B, dan C). Sedangkan e_ψ adalah emf yang terinduksi pada fasa x. Besarnya emf yang terinduksi pada setiap fasa adalah sebanding dengan laju perubahan fluks.

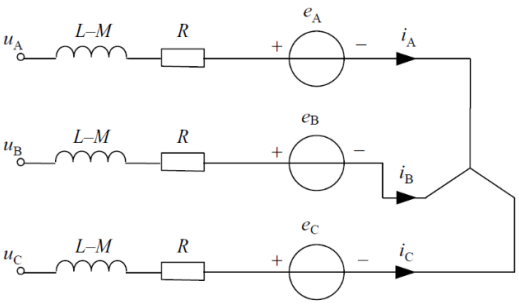
$$e_{\psi x} = \frac{d\psi_x}{dt} \qquad (1.2)$$

Besarnya fluks pada fasa A adalah :

$$\psi_A = L_A i_A + M_{AB} i_B + M_{AC} i_C + \psi_{PM}(\theta), \qquad (1.3)$$

Dimana ψ_{PM} merupakan *flux linkage* yang disebabkan oleh permanent magnet pada fasa A. Teta θ adalah sudut yang menyatakan posisi dari rotor, L_A adalah induktasi diri, M_{AB} dan M_{AC} adalah induktansi bersama fasa A dengan fasa B dan C. Besar dari $\psi_{PM}(\theta)$ bergantung pada distribusi medan magnet dari magnet permanen pada celah udara. Komponen radial dari medan magnet pada celah udara yang ditimbulkan oleh magnet permanen tedistribusi secara *trapezoidal* sepanjang permukaan dalam dari stator.

Pada motor Brushless DC 3 fasa memiliki rangkaian eqivalen pada sisi stator adalah sebagai berikut :



Gambar 2. 1 Rangkaian Ekivalen Motor BLDC

Pada rangkaian eqivalen tersebut berlaku hukum arus,

$$i_A + i_B + i_C = 0 \qquad (1.4)$$

Maka persamaan dapat diserdehanakan menjadi,

$$u_A = Ri_A + (L - M)\frac{di_A}{dt} + e_A \qquad (1.5)$$

Sehingga persamaan matriks tegangan fasa pada tiap kumparan stator motor BLDC adalah sebagai berikut :

$$\begin{bmatrix} u_A \\ u_A \\ u_A \end{bmatrix} = \begin{bmatrix} R & 0 & 0 \\ 0 & R & 0 \\ 0 & 0 & R \end{bmatrix} \begin{bmatrix} i_A \\ i_A \\ i_A \end{bmatrix} + \begin{bmatrix} L - M & 0 & 0 \\ 0 & L - M & 0 \\ 0 & 0 & L - M \end{bmatrix} \frac{d}{dx} \begin{bmatrix} i_A \\ i_A \\ i_A \end{bmatrix} + \begin{bmatrix} e_A \\ e_A \\ e_A \end{bmatrix} \qquad (1.6)$$

Sedangkan persamaan matriks untuk tegangan antar fasanya didapatkan dari pengurangan tegangan antar fasa dan didapatkan :

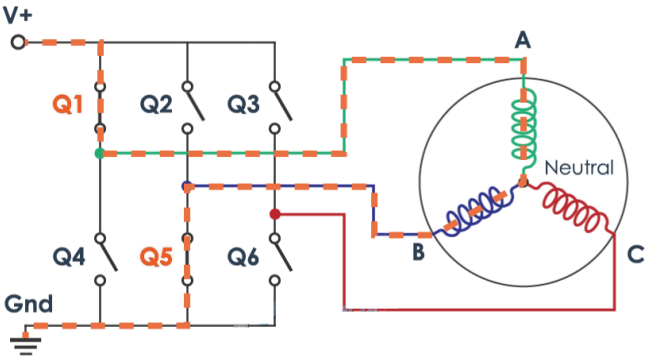
$$\begin{bmatrix} u_{AB} \\ u_{AC} \\ u_{CA} \end{bmatrix} = \begin{bmatrix} R & -R & 0 \\ 0 & R & -R \\ -R0 & 0 & R \end{bmatrix} \begin{bmatrix} i_A \\ i_A \\ i_A \end{bmatrix} + \begin{bmatrix} L - M & M - L & 0 \\ 0 & L - M & M - L \\ M - L & 0 & L - M \end{bmatrix} \frac{d}{dx} \begin{bmatrix} i_A \\ i_A \\ i_A \end{bmatrix} + \begin{bmatrix} e_A - e_B \\ e_B - e_C \\ e_C - e_A \end{bmatrix} \qquad (1.7)$$

2.2. *Electronic Speed Control* (ESC)

Sistem ESC brushless pada dasarnya menciptakan daya AC tiga fasa pada outputnya dengan cara pengaturan frekuensi, untuk menjalankan motor brushless. Pada ESC rotasi motor dideteksi dengan EMF atau dengan menggunakan hall sensor. Kontrol kecepatan pada ESC dapat diprogram melalui komputer dengan menggunakan mikrokontroler untuk menentukan batas tegangan rendah, pengaturan waktu, percepatan, pengereman, dan arah rotasi. Membalikkan arah motor juga dapat dilakukan dengan mengganti dua dari tiga output dari ESC ke motor.

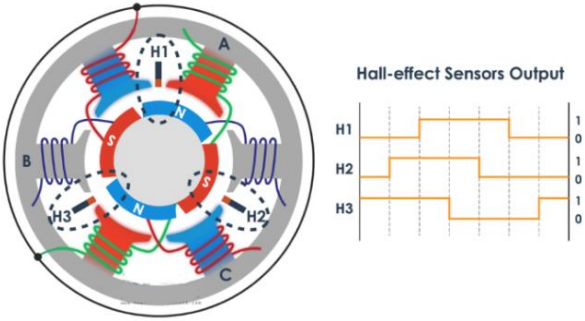
Rangkaian ESC terdiri dari switch yang telah diatur penyalaannya sehingga menghasilkan sinyal AC pada sisi output. Sebagai contoh sistem penalaan dapat dilihat pada gambar 2.2. Ketika fase A pada motor Brushless DC dihubungkan ke sumber tegangan positif dan pada sisi lainnya fase B dihubungkan dengan ground, lalu untuk fasa C dalam kondisi floating, maka arus akan mengalir dari sumber tegangan positif melalui fase A menuju titik netral kemudian melewati fase B dan akhirnya menuju ground. Jadi dengan hanya menggunakan satu aliran arus, empat

kutub yang berbeda dapat dihasilkan untuk menggerakan rotor. Dengan konfigurasi seperti ini, akan terbentuk konfigurasi star pada fasa-fasa motor Brushless DC seperti gambar 2.2, dimana titik netral secara internal meghubungkan masing-masing ujung fasa dan ujung lainnya keluar dari motor. Hal ini menyebabkan mengapa motor brushless dc memiliki tiga kabel. Jadi untuk menghasilkan siklus penuh sehingga dapat berputar, dua MOSFET yang benar harus diaktifkan pada setiap enam interval. Oleh karena itu dibutuhkan *electronic speed controller* (ESC).



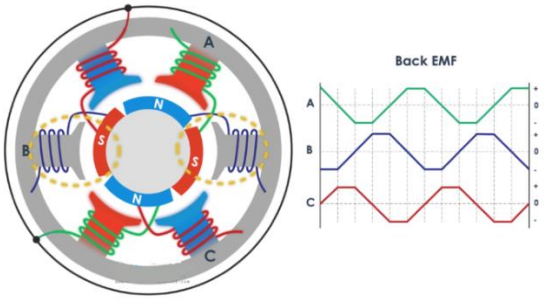
Gambar 2. 2. Contoh aliran arus pada fasa motor brushless

ESC berfungsi untuk mengendalikan kecepatan motor brushless dengan cara mengaktifkan MOSFET yang diperlukan untuk menghasilkan *rotating magnetic field* (RMF) sehingga motor dapat berputar. Semakin tinggi frekuensi masukan pada MOSFET, dengan kata lain semakin cepat ESC menempuh enam interval, maka semakin tinggi juga kecepatan motor Brushless DC yang akan dihasilkan. Untuk mengetahui fasa mana yang akan dilalui arus terlebih dahulu, dapat digunakan dengan cara mengetahui posisi rotor , terdapat dua metode umum yang digunakan untuk menentukan posisi rotor.



Gambar 2. 3. Keluaran pembacaan sensor hall-effect

Metode pertama adalah dengan menggunakan sensor *hall-effect* yang terletak pada stator dengan penempatan setiap 120 derajat atau 60 derajat antara satu sensor dengan sensor yang lainnya. Ketika rotor yang terdiri dari magnet permanen berputar, sensor *hall-effect* akan terinduksi medan magnet dan menghasilkan logika *high* untuk satu kutub dan *low* untuk kutub lainnya. Berdasarkan informasi ini ESC akan mengetahui saat untuk mengaktifkan interval komutasi selanjutnya.



Gambar 2. 4. Keluaran pembacaan EMF

Metode kedua untuk menentukan posisi rotor adalah dengan menggunakan *electromotive force* balik yang dihasilkan, atau disebut sebagai *back* EMF. *Back* EMF dihasilkan oleh keballikan dari proses menghasilkan medan magnet, yaitu saat menggerakan atau mengubah

medan magnet yang melewati kumparan maka akan menginduksi arus pada kumparan tersebut. Jadi ketika medan magnet pada rotor bergerak melewati kumparan yang sedang tidak aktif, medan magnet ini akan menginduksi arus pada kumparan tersebut sehingga akan timbul beda tegangan. ESC akan menangkap beda tegangan ini saat terjadi dan berdasarkan hal tersebut ESC akan memperhitungkan kapan interval selanjutnya harus terjadi.

2.3. Sensor Inframerah (IR Sensor)

Sensor inframerah merupakan sebuah komponen elektronika yang dapat mendeteksi cahaya inframerah. Saat ini terdapat banyak modul inframerah yang menggunakan chip detector inframerah digital yang didalamnya terdapat photodiode dan amplifier, sehingga output sensor yang menggunakan chip ini dapat menghasilkan sinyal berlogika 0 dan 1. Cara kerjanya yaitu, transmitter mengirimkan cahaya inframerah kemudian akan ditangkap oleh photodiode dan dikuatkan oleh amplifier, sehingga akan mengeluarkan output berlogika 1. Namun jika cahaya inframerah ini terhalang dan tidak bisa ditangkap oleh photodiode, maka output akan berlogika 0.

2.4. Logika Fuzzy

Kontrol kecepepatan motor dapat menggunakan beberapa metode, salah satunya adalah dengan menggunakan *fuzzy-logic* kontroler. Logika fuzzy telah banyak digunakan pada sistem otomasi dan kontrol. Logika fuzzy dapat digunakan untuk menyelesaikan permasalahan nonlinier dan ketidakpastian tanpa menggunakan model matematika. Logika fuzzy merupakan suatu teori himpunan logika yang dikembangkan untuk mengatasi konsep nilai yang terdapat diantara nilai kebenaran (true) dan kesalahan (false). Logika fuzzy berbeda dengan logika digital biasa atau *Boolean*. Logika digital biasa hanya mengenal dua keadaan yang tegas (*crips*), yaitu ‘ya’ atau ‘tidak’, ‘0’ atau ‘1’, dan ‘on’ atau ‘off’. Berbeda dengan logika digital biasa, logika fuzzy meniru cara berfikir manusia degan menggunakan konsep kesamaan suatu nilai. Dengan menggunakan logika fuzzy, nilai tidak lagi hanya bernilai ‘0’ dan ‘1’ tetapi seluruh kemungkinan diantara 0 dan 1.

Kesederhanaan konsep membuat logika fuzzy mudah dimengerti. Fuzzy tidak terpaku dengan satu keputusan, fleksibel, sehingga dapat memberi nilai toleransi pada ketidakpastian. Ada beberapa alasan mengapa memilih menggunakan logika fuzzy yaitu :

1. Konsep logika fuzzy mudah dimengerti. Konsep matematis dari logika yang sangat sederhana.
2. Sifat logika fuzzy yang fleksibel
3. Logika fuzzy mampu menggambarkan fungsi-fungsi linier yang bersifat kompleks.

2.4.1. Himpunan *Fuzzy*

Pada himpunan crips, nilai keanggotaan hanya terdapat 2 kemungkinan yaitu '0' dan '1'. Sebagai contoh, terdapat sebuah elemen x dalam suatu himpunan A, yang dinyatakan dalam derajat keanggotaan $\mu_A(x)$, memiliki dua nilai keanggotaan, yaitu:

a. Bernilai satu ($\mu_A(x) = 1$), yang berarti elemen x merupakan anggota penuh himpunan A;

b. Bernilai nol ($\mu_A(x) = 0$), yang berarti elemen x bukan merupakan anggota himpunan A.

Namun pada himpunan *fuzzy* yang menoleransi kesamaran, batas tegas yang memisahkan anggota dan bukan anggota pada himpunan tegas akan dieliminasi. Sehingga perpindahan antara anggota penuh dan bukan anggota akan terjadi berangsur-angsur karena keberadaan daerah *overlapping* pada himpunan *fuzzy*.

Terkadang kemiripan antara keanggotaan fuzzy dengan probabilitas menimbulkan kerancuan. Keduanya memiliki nilai pada interval [0,1], namun interpretasi nilainya sangat berbeda antara kedua kasus tersebut. Keanggotaan fuzzy memberikan suatu ukuran terhadap pendapat atau keputusan, sedangkan propabilitas mengindikasikan proporsi terhadap keseringan suatu hasil bernilai benar dalam jangka panjang. Misalnya, jika niliai keanggotaan suatu himpunan fuzzy MUDA adalah 0,9; maka tidak perlu dipermasalahkan berapa seringnya nilai itu diulang secara individul untuk mengharapkan suatu hasil yang hampir pasti muda. Di sisi lain, nilia probabilitas 0,9 muda berarti 10% dari himpunan tersebut diharapkan tidak muda.

Himpunan fuzzy memiliki 2 atribut, yaitu :

1. Linguistik, yaitu penamaan suatu grup yang mewakili suatu keadaan atau kondisi tertentu dengan menggunakan bahasa alami, seperti: MUDA, PAROBAYA, TUA
2. Numeris, yaitu suatu nilai (angka) menunjukkan ukuran dari suatu variabel seperti : 10, 25, 50 dan sebagainya.

Berikut beberapa hal yang perlu diketahui mengenai sistem *fuzzy*, yaitu:

1. Variabel *fuzzy*

   Variabel fuzzy merupakan variabel yang dibahas dalam sistem *fuzzy*. Contoh: temperatur, kecepatan, posisi, dll.

2. Himpunan *fuzzy*

   Himpunana fuzzy merupakan suatu kelompok yang mewakili suatu keadaan tertentu dalam variabel *fuzzy*. Umumnya ciri penamaan dalam himpunan *fuzzy* adalah linguistik. Linguistik merupakan penamaan yang nilainya berupa kata-kata, yang digunakan untuk memperkirakan nilai atau keadaan yang cukup sulit untuk dinyatakan secara eksak. Contoh: lambat, cepat, sangat cepat.

3. Semesta pembicaraan

   Semesta pembicaraan adalah keseluruhan nilai yang diperbolehkan untuk dioperasikan dalam suatu variabel *fuzzy*. Semesta pembicaraan merupakan himpunan bilangan real yang nilainya selalu bertambah secara monoton dari kiri ke kanan. Nilai semesta pembicaraan dapat berupa bilangan positif maupun bilangan negatif.

4. Domain

   Domain himpunan fuzzy adalah keseluruhan nilai yang diijinkan dalam semesta pembicaraan dan boleh dioperasikan dalam suatu himpunan fuzzy. Sama seperti semesta pembicaraan, domain merupakan himpunan bilangan real yang senantiasa naik (bertambah) secara monoton dari kiri ke kanan. Domain dapat berupa bilangan positif maupun bilangan negatif.

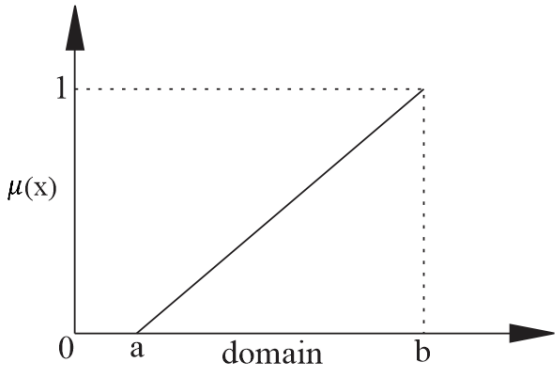
2.4.2. Fungsi Keanggotaan (*Membership Function*)

Fungsi keanggotaan (*membership function*) adalah suatu kurva yang menunjukkan pemetaan titik-titik *input* data ke dalam nilai keanggotaannya yang memiliki interval nilai antara 0 sampai 1. Salah satu cara yang digunakan untuk mendapatakan nilai keanggoataan adalah dengan melalui pendekatan fungsi. Berikut beberapa fungsi keanggotaan yang umum digunakan, yaitu :

2.4.2.1. Representasi linier

Pada representasi linear, pemetaan input ke derajat keanggotaannya digambarkan sebagai garis lurus. Bentuk ini paling sederhana dan menjadi pilihan yang baik untuk mendekati suatu konsep yang kurang jelas.

Ada 2 keadaan himpunan fuzzy yang linear. Pertama, kenaikan himpunan dimulai pada nilai domain yang memiliki derajat keanggotaan nol [0] bergerak ke kanan menuju ke nilai domain yang memiliki derajat keanggotaan lebih tinggi.



Gambar 2. 5. Fungsi Keanggotaan Linear Naik

Fungsi keanggotaan :

$$\mu[x] = \begin{cases} 0; & x \leq 0 \\ \frac{(x-a)}{(x-b)}, & a \leq x \leq b \\ 1; & x \geq b \end{cases} \quad (2.1)$$

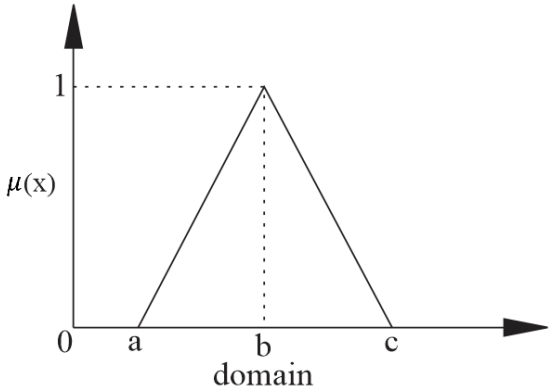
Kedua merupakan kebalikan yang pertama. Garis lurus di mulai dari nilai domain dengan derajat keanggotaan terteinggi pada sisi kiri, kemudian bergerak menurun ke nilai domain yang memiliki derajat keanggotaan lebih rendah.

$$\mu[x] = \begin{cases} \frac{(b-x)}{(b-a)}, & a \leq x \leq b \\ 0, & x \geq b \end{cases} \quad (2.2)$$

2.4.2.2. Representasi kurva segitiga

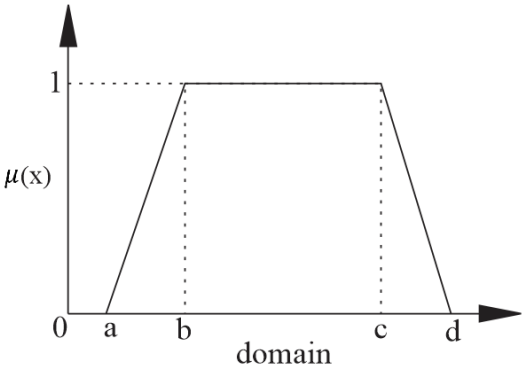
Fungsi segitiga merupakan gabungan antara dua fungsi linear seperti terlihat pada gambar. keanggotaan fungsi segitiga, yaitu:

$$\mu\,[x] = \begin{cases} 0\,; x \le a \\ \frac{(x-a)}{(b-a)} \quad ; a \le x \le b \\ \frac{(b-x)}{(c-b)} \quad ; b \le x \le c \\ 0, \; x \ge c \end{cases} \tag{2.3}$$



Gambar 2. 6. Kurva Representasi Segitiga

2.4.2.3. Representasi kurva trapesium



Gambar 2. 7. Kurva Representasi Trapesium

Kurva trapesium pada dasarnya seperti bentuk segitiga, dimana terdapat titik yang memiliki nilai keanggotaan 1.

Fungsi keanggotaan :

$$\mu\,[x] = \begin{cases} 0\,; x \le a \\ \frac{(x-a)}{(b-a)} \quad ;\ a \le x \le b \\ 1\,; b \le x \le c \\ \frac{(c-x)}{(d-c)} \quad ;\ c \le x \le d \\ 0,\ x \ge d \end{cases} \tag{2.4}$$

2.4.2.4. Representasi kurva bentuk bahu

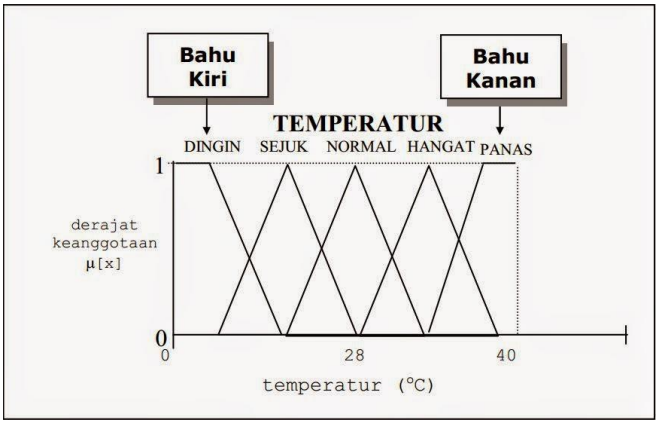
Daerah yang terletak di tengah-tengah suatu variabel yang direpresentasikan dalam bentuk segitiga, pada sisi kanan dan kirinya akan naik dan turun. Tetapi terkadang salah satu sisi dari variabel tersebut tidak mengalami perubahan. Sebagai contoh, apabila telah mencapai kondisi PANAS, kenaikan temperatur akan tetap berada pada kondisi PANAS. Himpunan fuzzy 'bahu', bukan segitiga, digunakan untuk mengakhiri variabel suatu daerah fuzzy. Bahu kiri bergerak dari 1 ke 0, sementara bahu kanan bergerak dari 0 ke 1. Gambar menunjukkan variabel TEMPERATUR dengan daerah bahunya.

Fungsi keanggotaan :

$$\mu dingin[x] = \begin{cases} 1; & a \leq x \leq b \\ \frac{(c-x)}{(c-b)}, & b \leq x \leq c \\ 0; & x \geq c \end{cases} \quad (2.5)$$

$$\mu sejuk[x] = \begin{cases} 0; & x \leq b \\ \frac{(x-b)}{(c-b)}, & b \leq x \leq c \\ \frac{(d-x)}{(d-c)}, & c \leq x \leq d \\ 0; & x \geq d \end{cases} \quad (2.6)$$

$$\mu normal[x] = \begin{cases} 0; & x \leq c \\ \frac{(x-c)}{(d-c)}, & c \leq x \leq d \\ \frac{(e-x)}{(e-d)}, & d \leq x \leq e \\ 0; & x \geq e \end{cases} \quad (2.7)$$



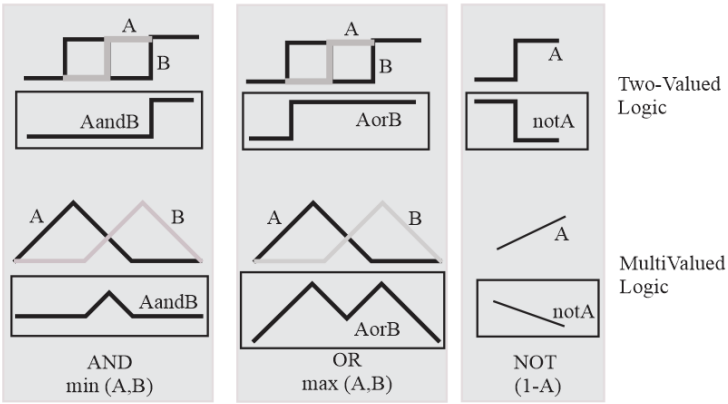
Gambar 2. 8. Daerah 'Bahu' Pada Variabel Temperatur

$$\mu hangat[x] = \begin{cases} 0; & x \leq d \\ \frac{(x-d)}{(e-d)}, & d \leq x \leq e \\ \frac{(f-x)}{(f-e)}, & e \leq x \leq f \\ 0; & x \geq f \end{cases} \quad (2.8)$$

$$\mu panas[x] = \begin{cases} 0; & x \leq e \\ \frac{(x-e)}{(f-e)}, & e \leq x \leq f \\ 1; & x \geq b \end{cases} \quad (2.9)$$

2.4.3. Operasi Himpunan *Fuzzy*

Berikut akan dijelaskan beberapa operator dasar yang umum digunakan untuk menggabungkan dan memodifikasi dua atau lebih himpunan *fuzzy* pada fungsi keanggotaan, yaitu operator interseksi, gabungan, dan komplemen. Masing-masing hasil operasi ditunjukkan oleh Gambar, kemudian akan dijelaskan sebagai berikut:



Gambar 2. 9. Operasi Himpunan Fuzzy

a. Operator interseksi (*AND*)

Interseksi himpunan *fuzzy A*dan himpunan *fuzzy B*yang didefinisikan dalam semesta pembicaraan U, dinotasikan sebagai $A \cap B$, memiliki dua nilai keanggotaan yaitu:

1. *Minimum*

Nilai keanggotaan *minimum* yaitu:

$$\mu_{A \cap B} = min \{\mu_A (u_i), \mu_B (u_i) : u_i \in U_i \quad (2.10)$$

2. *Algebraic product*

Nilai keanggotaan *product* yaitu:

$$\mu_{A \cap B} = \{\mu_A (u_i) \mu_B (u_i) : u_i \in U_i \quad (2.11)$$

b. Operator gabungan (*OR*)

Gabungan himpunan *fuzzy A*dan himpunan *fuzzy B*yang didefinisikan dalam semesta pembicaraan U, dinotasikan sebagai $A \cup B$, memiliki dua nilai keanggotaan yaitu:

1. *Maximum*
   Nilai keanggotaan *maximum* yaitu:

$$\mu_{A \cup B} = max \{\mu_A (u_i), \mu_B (u_i) : u_i \in U_i \quad (2.12)$$

2. *Algebraic Sum*

Nilai keanggotaan *algebraic sum* yaitu:

$$\mu_{A \cup B} = \{\mu_A (u_i) + \mu_B (u_i) - \mu_A (u_i) \mu_B (u_i) : u_i \in U_i \quad (2.13)$$

c. Operator komplemen (*NOT*)

Komplemen himpunan *fuzzy A* dengan fungsi keanggotaan $\mu_A (u_i)$, memiliki nilai keanggotaan yaitu:

$$\mu_{A'} (u_i) = 1 - \mu_A (u_i) \quad (2.14)$$

2.4.4. Sistem Inferensi *Fuzzy*

Inferensi *fuzzy* adalah cara penarikan kesimpulan pada sistem *fuzzy*. Dalam penalaran logika *fuzzy*, tipe yang umum digunakan untuk inferensi yaitu *generalised modus ponens* (GMP) atau disebut juga *direct reasoning*. Karena pada umumnya pemikiran manusia dilakukan dengan cara *modus ponen*. Pemetaan *input* ke *output* pada sistem *fuzzy* berada dalam sekumpulan aturan $kondisi \rightarrow aksi$ atau bentuk *IF-THEN* yang menggambarkan *modus ponen*, yaitu:

IF *premis* **THEN** *konsekuen*.

Premis merupakan kondisi dan konsekuen merupakan aksi. Umumnya *input* sistem *fuzzy* dihubungkan dengan premis sedangkan *output* sistem *fuzzy* dihubungkan dengan konsekuen.
Sistem *fuzzy* dibangun oleh empat bagian utama, yaitu:

1. Dasar-aturan
   Dasar-aturan merupakan sekumpulan aturan *IF-THEN* yang berisi kuantifikasi uraian linguistik pakar tentang cara mencapai kontrol yang baik.

2. Mekanisme inferensi
   Disebut juga inferensi *fuzzy*, yang menyamai cara pengambilan keputusan pakar dalam menerjemahkan dan mengaplikasikan pengetahuan tentang cara terbaik untuk mengontrol *plant*.

3. Fuzifikasi
   Mengonversi *input* kontroler kedalam bentuk informasi yang dapat dipahami oleh mekanisme inferensi untuk mengaktifkan dan mengaplikasikan aturan-aturan *IF-THEN*.

4. Defuzifikasi
   Mengonversi kesimpulan yang dihasilkan mekanisme inferensi menjadi *input* aktual untuk proses.

Terdapat beberapa langkah dalam mekanisme inferensi *fuzzy*, yaitu:

1. Fuzifikasi *input*

   Mengonversi *input* kontroler yang merupakan *input* tegas menjadi *input fuzzy* lalu memetakannya kedalam fungsi keanggotaan himpunan *fuzzy* dengan derajat keanggotaannya masing-masing.

2. Aplikasi operator *fuzzy*

   Mengombinasi dan memodifikasi ketika terdapat dua atau lebih himpunan *fuzzy* pada bagian premis.

3. Aplikasi metode implikasi

   Menghasilkan sebuah *output* konsekuen untuk masing-masing aturan.

4. Agregasi semua *output*

   Mengumpulkan semua *output* konsekuen aturan yang dihasilkan pada implikasi.

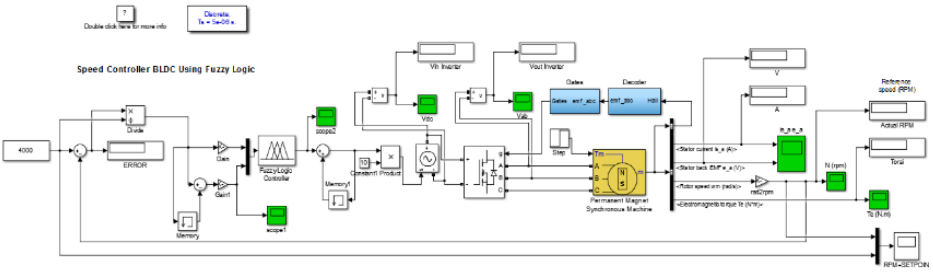
5. Defuzifikasi

   Mengonversi *output fuzzy* menjadi *output* tegas sebagai *input* proses selanjutnya.

BAB 3
PERENCANAAN SISTEM KONTROLLER BLDC MOTOR UNTUK KENDARAAN LISTRIK MENGGUNAKAN *FUZZY LOGIC CONTROLLER*

3.1. Simulasi Yang Dilakukan

Pada penelitian ini, simulasi dari tugas akhir yang berjudul **DESAIN KONTROLLER BLDC MOTOR UNTUK KENDARAAN LISTRIK MENGGUNAKAN *FUZZY LOGIC CONTROLLER*** akan dilakukan menggunakan Simulink, berikut adalah diagram blok simulasi yang dilakukan :



Gambar 3. 1. Simulasi Menggunakan Simulink

Pada gambar diatas, sistem mendapatkan nilai referensi kecepatan berupa konstanta tertentu yang diinput pada blok *Constant.* Kemudian nilai referensi tersebut akan dibandingkan dengan nilai RPM yang dihasilkan oleh *Permanent Magnet Synchronous Machine*, penggunaan *Permanent Magnet Synchronous Machine* sebagai representasi dari BLDC Motor pada simulasi menggunakan *Simulink.*

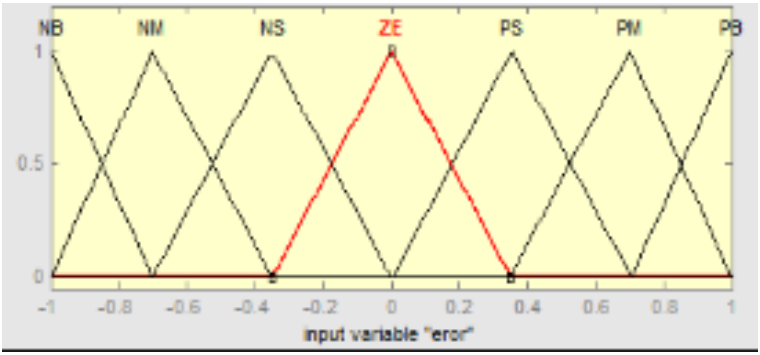
Perbandingan antara nilai referensi dan nilai RPM akan menghasilkan nilai *error*, kemudian nilai *error* tersebut dibagi dengan nilai referensi agar mendapatkan nilai *error'* dalam *range* 0 hingga 1. Nilai *error'* digunakan untuk input yang pertama yaitu *membership function for Input Variable Error* atau (*e*) pada penelitian ini dan memiliki domain -1 hingga 1 (dapat dilihat pada gambar 3.2. Membership Function for Input Variable Error)

Nilai *error'* juga akan digunakan untuk input kedua dari *fuzzy logic controller* yaitu input *delta error (de).* Nilai delta error ini didapatkan dengan cara, yaitu nilai *error'* saat ini akan dikurangi dengan *error'* sebelumnya.

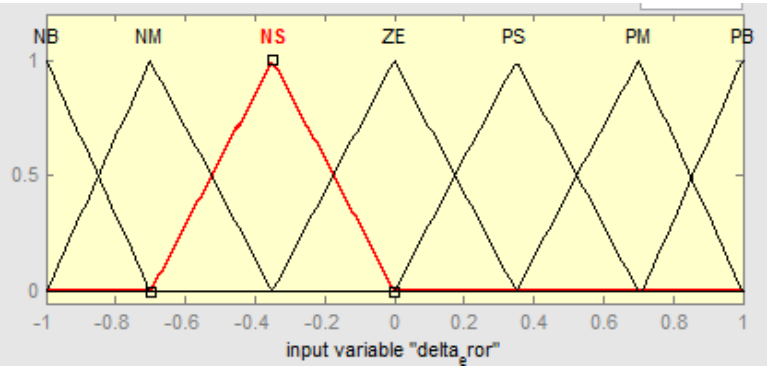
Kemudian kedua input tersebut (*e*) dan (*de*) akan diolah pada blok *fuzzy logic controller* yang sebelumnya telah diberi program .*FIS* berisi *Rule Fuzzy* (tabel XX Rule Fuzzy) sehingga menghasilkan output fuzzy yang kemudian akan dioleh pada blok selanjutnya sesuai gambar 3.1 Simulasi Menggunakan Simulink diatas.

3.2. Membership Function Input

Pada tugas akhir ini, sistem kontrol *fuzzy logic* menggunakan dua input, yaitu *Error* dan *Delta Error*. Input *Error* didefinisikan sebagai perbedaan nilai antara kecepatan referensi dengan kecepatan actual. Sedangkan input *Delta Error* didefinisikan sebagai perbedaan antara nilai error saat ini dan nilai error yang sebelumnya. Membership function dari *Error* dan *Delta Error* ini direpresentasikan menggunakan kurva segiata seperti gambar dibawah ini :



Gambar 3. 2. Membership Function for Input Variable Error



Gambar 3. 3. Membership Function for Input Variable Delta Error

3.3. Rule fuzzy

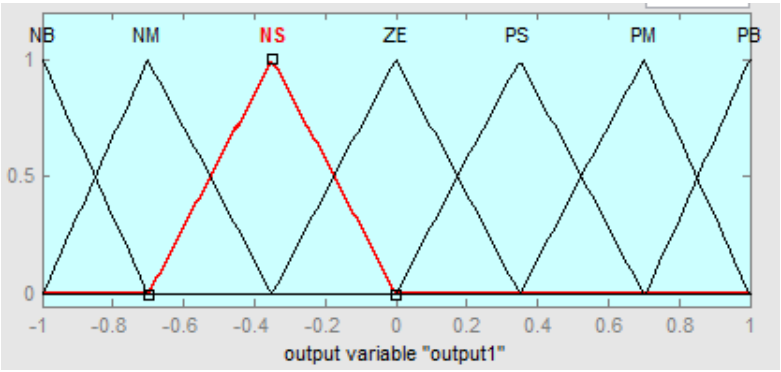
Langkah selanjutnya adalah menentukan rule fuzzy dengan menyesuaikan jumlah fungsi keanggotaan dari proses fuzzyfikasi. Logika fuzzy pada penelitian ini diharapkan agar kecepatan sebenarnya dari BLDC motor dapat mengikuti kecepatan referensi yang dikirim dari potensiometer. Sehingga dibuatlah rule fuzzy dengan tujuh buah fungsi keanggotan dari *error* dan tujuh buah fungsi keanggotaan dari *delta error*. Masing – masing fungsi yaitu Negative Big (NB), Negative Medium (NM), Negative Small (NS), Zero (ZE), Positive Small (PS), Positive Medium (PS), dan Positive Big (PB) seperti tabel dibawah ini :

Tabel 3. 1. Rule Fuzzy

e / de	NB	NM	NS	ZE	PS	PM	PB
NB	NB	NB	NB	NB	NM	NS	ZE
NM	NB	NB	NB	NM	NS	ZE	PS
NS	NB	NB	NM	NS	ZE	PS	PM
ZE	NB	NM	NS	ZE	PS	PM	PB
PS	NM	NS	ZE	PS	PM	PB	PB
PM	NS	ZE	PS	PM	PB	PB	PB
PB	ZE	PS	PM	PB	PB	PB	PB

3.4. Membership Function Output

Pada tugas akhir ini, sistem kontrol *fuzzy logic* menggunakan satu *output*. Membership function output ini direpresentasikan oleh 7 kurva segitiga, yaitu Negative Big (NB), Negative Medium (NM), Negative Small (NS), Zero (ZE), Positive Small (PS), Positive Medium (PS), dan Positive Big (PB) seperti tabel dibawah ini :



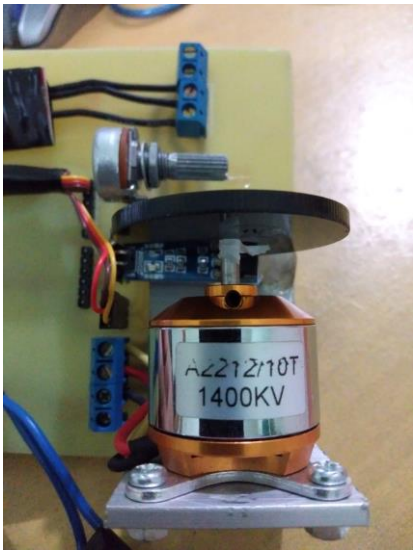
Gambar 3. 4. Membership Function Output

3.5. Spesifikasi Alat Yang Digunakan

Berikut adalah alat apa saja yang digunakan pada tugas akhir yang berjudul **DESAIN KONTROLLER BLDC MOTOR UNTUK KENDARAAN LISTRIK MENGGUNAKAN *FUZZY LOGIC CONTROLLER***

3.5.1 Motor Brushless DC

Pada penelitian ini, brushless DC motor yang digunakan adalah A2212/10T dengan spesifikasi sebagai berikut :



Gambar 3. 5. Burshless DC Motor A2212/10T

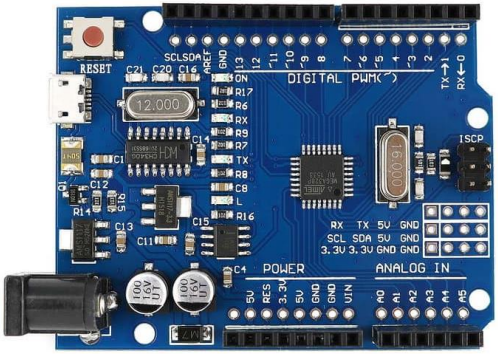
Tabel 3. 2. Spesifikasi Burshless DC Motor A2212/10T

Operating Voltage	7,2 – 12Volt (2-3 LiPo Battery)
RPM/V	1400 kV
Max Operating Temperature	±80^{o}C
Max Efficiency current	4 – 10 A

No Load Current	0.5A
Max Current	13A for 60 seconds
Max Watts	150W
Weight of Motor	50 – 60 gram
Size	28 mm dia x 28 mm bell length
Shaft Diameter	3.2 mm
Poles	14

3.5.2 Mikrokontroller

Penggunaan mikrokontroller pada penelitian ini menggunakan arduino uno karena ukuran file program ide arduino yang kami buat tidak lebih dari ukuran flash memory arduino uno, selain itu port atau pin digtal I/O yang dibutuhkan juga cukup untuk pembuatan alat pada penelitian ini. Berikut adalah spesifikasi arduino uno yang digunakan :



Gambar 3. 6. Arduino Uno

Tabel 3. 3. Spesifikasi Arduino Uno

Microcontroller	ATmega328P
Operating Voltage	5V
Input Voltage (recommended)	7-12V
Input Voltage (limit)	6-20V

Digital I/O Pins	14 (of which 6 provide PWM output)
PWM Digital I/O Pins	6
Analog Input Pins	6
DC Current per I/O Pin	20 mA
DC Current for 3.3V Pin	50 mA
Flash Memory	32 KB (ATmega328P) of which 0.5 KB used by bootloader
SRAM	2 KB (ATmega328P)
EEPROM	1 KB (ATmega328P)
Clock Speed	16 MHz
Length	68.6 mm
Width	53.4 mm
Weight	25 g

3.5.3. Electronic Speed Controller (ESC)

Agar kecepatan pada BLDC motor dapat diatur dengan mudah, diperlukan alat bernama electronic speed controller (ESC) sehingga pengaturan kecepatan BLDC motor dapat diprogram menggunakan mikrokontroller. Tegangan suplai menggunakan baterai yang kemudian akan diubah menjadi gelombang AC oleh electric speed control sebelum masuk ke dalam motor. Pada penelitian ini, ESC yang digunakan adalah Emax BLHeli 12A dengan spesifikasi sebagai berikut :

Tabel 3. 4. Spesifikasi Emax BLHeli 12A

Continuous Current	12A
Burst current (10S)	15A
Li-xx Battery (cell)	2-3
Dimension (L*W*H)	25mm x 20mm x 7mm
BEC Mode	Linear
BEC Output	1A/5V
Programmable	Yes

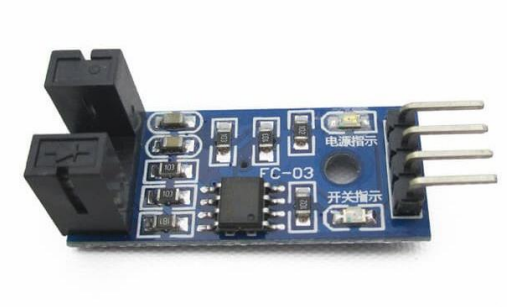
Gambar 3. 7. Emax BLHeli 12A

3.5.4. Sensor Kecepatan (IR Sensor)

Dalam dunia elektronika, salah satu jenis sensor inframerah (IR Sensor) merupakan *FC-03 Interrupter Sensor Module*. Sensor ini sering digunakan untuk membaca kecepatan pada motor DC yaitu menggunakan modul Sensor Inframerah seperti gambar dibawah ini dengan spesifikasi sebagai berikut :

Tabel 3. 5. Spesifikasi FC-03 Interrupter Sensor Module

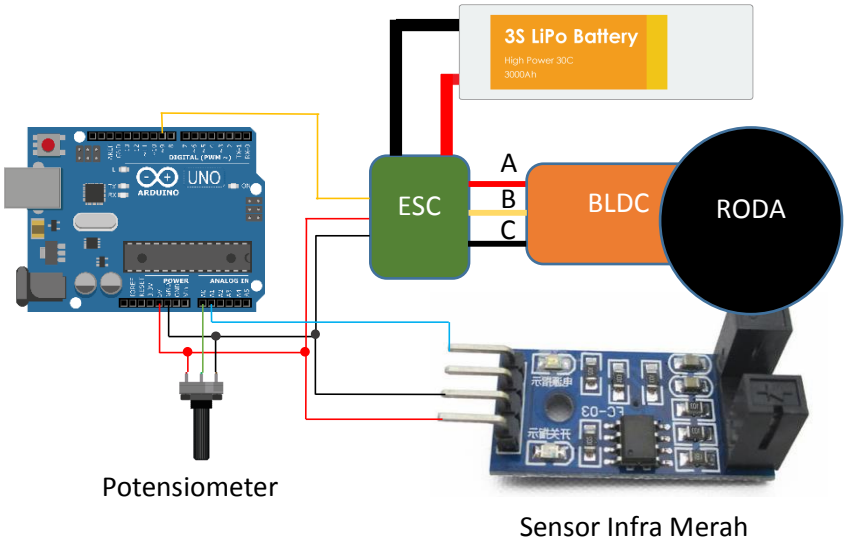
Jarak groove coupler	5mm
Tegangan input:	3.3 - 5V DC
Output Digital	0 dan 5V DC (0 dan 1)
Ukuran PCB	32 x 14mm



Gambar 3. 8. FC-03 Interrupter Sensor Module

3.6. Rangkaian Listrik

Gambar dibawah ini merupakan rangkaian listrik atau Wiring - Diagram dari implementasi alat atau prototype dari peneletian ini



Gambar 3. 9. Wiring Diagram dari Prototype

3.7. Kecepatan Referensi Menggunakan Potensiometer

Kecepatan referensi ini diatur menggunakan sinyal analog yang dikirimkan oleh potensiometer yang kemudian diubah dalam nilai digital oleh arduino sebelum dikirimkan menuju ESC. Nilai dari potensiometer adalah 0-1023 dalam bentuk analog akan diubah menjadi 0-180 dalam bentuk digital output.

ESC tidak akan merespon PWM apabila inputnya langsung dalam nilai tertentu. Oleh karena itu, perlunya nilai output PWM dari arduino yang masuk ke dalam ESC diinisiasikan dimulai dari nilai 0. Dan nilai PWM ini akan bertambah apabila nilai dari potensiometer juga bertambah.

3.8. Program Perhitungan Kecepatan dengan IR Sensor

Ketika BLDC motor berputar, arduino tidak dapat membaca kecepatan atau RPM yang dihasilkan, sehingga diperlukan bantu alat berupa sensor kecepatan. Sensor kecepatan yang digunakan pada penelitian ini berjenis IR sensor.

Alat ini akan mengirimkan sinyal LOW (0V) kepada arduino ketika IR sensor terhalang oleh suatu objek dan akan mengirimkan sinyal HIGH (5V) kepada arduino ketika IR sensor tidak terhalang oleh objek, kemudian akan diolah dan diprogram oleh arduino menjadi data berupa kecepatan atau RPM yang selanjutnya akan digunakan untuk dalam pembuatan sistem kontrol kecepatan menggunakan fuzzy logic

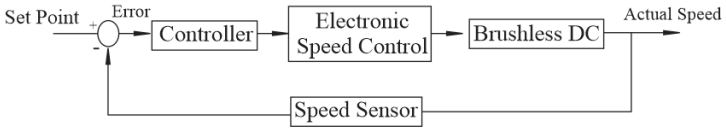
3.9. Program Kontrol Kecepatan BLDC dengan Fuzzy-logic

Setelah didapatkan nilai rpm pada motor, maka nilai kecepatan ini akan dibandingkan dengan sinyal referensi yang dikirimkan arduino ke dalam ESC. Logika fuzzy ini terjadi didalam arduino yang kemudian diolah menjadi sinyal output PWM. Tentunya nilai yang dibandingkan pada arduino bukanlah nilai PWM sebagai mana dijelaskan system kerja drive ESC pada bagian sebelumnya. Namun, nilai PWM yang terbangkit dari sinyal analog yang dikirimkan oleh potensiometer diubah kedalam bentuk referensi rpm dengan cara menskala hasil rpm yang dikirimkan kedalam ESC dan juga kecepatan yang terbangkit. Karena setiap kenaikan nilai PWM menyebabkan niali putaran bertambah dengan rentan nilai

yang mendekati sama. Maka nilai itu kemudian diskala dan dibuat sebagai nilai referensi masukan ke dalam ESC.

Adapun nilai PWM yang masuk ke dalam ESC akan selalu berubah-ubah dan tidak sama seperti pada kontrol ESC sebelumnya. Hal ini diakibatkan oleh eror pembacaan hall sensor yang digunakan.

Berikut merupakan blok diagram sistem kontrol kecepatan pada BLDC motor:



Gambar 3. 10. Diagram Blok Sistem Kontrol Kecepatan

Pada sistem kontrol kecepatan pada fuzzy logic control ini, nilai referensi berasal dari nilai potensiometer yang awalnya bernilai analog dari 0 – 1023 menjadi sebuah sinyal digital dari 0 – 180. Nilai potensiometer ini kemudian dikalkulasikan dengan cara menskalakan PWM input pada ESC dan kecepatan actual yang ditimbulkan oleh motor. Kecepatan referensi ini kemudian diubah dalam bentuk PWM sebelum masuk kedalam esc, hal ini dikarenakan untuk dapat memutar motor maka nilai yang diinputkan kedalam ESC harus mulai dari nilai yang kecil.

Pada PWM ini akan menyebabkan motor berputar yang kemudian putaran motor akan dibaca sensor kecepatan. Kecepatan actual ini akan dibandingkan dengan kecepatan referensi di dalam kontrol fuzzy. Pada FLC terdapat beberapa proses. Pertama yaitu fuzzyfikasi atau pengenalan eror dan delta error. Eror ini adalah perbandingan antara kecepatan actual dan referensi. Di dalam kontrol fuzzy telah ditetapkan rule-base atau suatu kondisi yang akan diakibatkan dari adanya proses fuzzyfikasi.

--- halaman ini sengaja dikosongkan ---

BAB 4
HASIL IMPLEMENTASI DAN PEMBAHASAN

Pada bab empat ini akan dibahas mengenai hasil pengujian dari implementasi simulasi yang telah dilakukan, berikut adalah hasil yang telah diperoleh :

4.1. Uji Akurasi Pembacaan IR Sensor

Untuk mengetahui parameter kecepatan pada penelitian ini, penulis menggunakan IR Sensor untuk mengetahui kecepatan putaran pada BLDC motor. Sensor ini lalu dibandingkan dengan alat tachometer (TM 300) yang ada di Laboratorium Konversi. Pengujian ini dilakukan dengan cara merubah nilai PWM yang dikirim ke ESC menggunakan arduino kemudian IR sensor akan membaca kecepatan putaran pada BLDC motor dan membandingkannya menggunakan TM 300. Berikut adalah hasil pengujian yang didapatkan :

Tabel 4. 1. Pengujian IR Sensor

PWM	Pembacaan Kecepatan		Error (%)
	IR Sensor (RPM)	TM 300 (RPM)	
14	1478.54	1382	6.53
19	1885.84	1832	2.85
26	2456.06	2506	2.03
36	3270.66	3422	4.63
46	4085.26	4105	0.48
53	4655.48	4580	1.62
64	5551.54	5639	1.58
72	6203.22	6181	0.36
87	7425.12	7355	0.94
96	8158.26	8165	0.08
Rata - Rata Error (%)			2.11

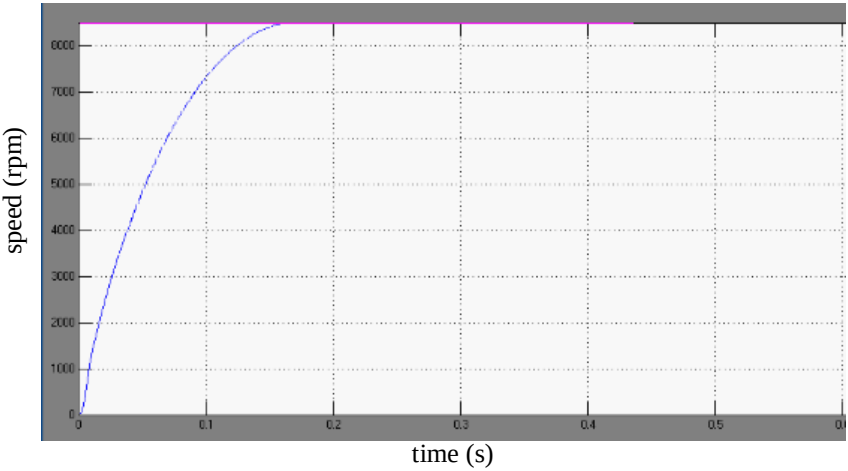
4.2. Pengujian Kontrol Kecepatan

Pengujian ini merupakan implementasi dari kontrol kecepatan menggunakan fuzzy logic. Hasil dari implementasi ini akan dibandingkan dengan hasil simulasi yang telah dilakukan untuk mengetahui perbedaan diantara keduanya dan untuk mengetahui berapa eror yang terjadi pada hasil implementasi. Berikut adalah hasil pengujian yang didapatkan :

4.2.1. Kecepatan 2800 RPM

4.2.1.1. Hasil simulasi kontrol kecepatan 2800 RPM

Pada simulasi ini, kecepatan reference diatur pada 2800 RPM dan direpresentasikan dengan garis berwarna merah. Hasilnya dapat terlihat bahwa kecepatan actual yang direpresentasikan dengan garis berwarna biru dapat mengikuti kecepatan yang diinginkan atau kecepatan reference seperti terlihat pada gambar dibawah ini :

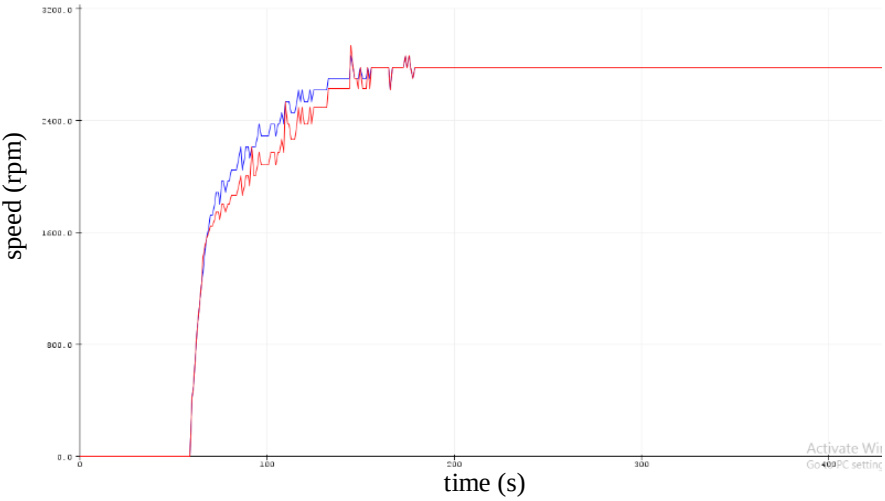


Gambar 4. 1. Hasil simulasi kontrol kecepatan 2800 RPM

4.2.1.2. Hasil implementasi kontrol kecepatan 2800 RPM

Pada implementasi ini dilakukan dengan sistem close loop atau menggunakan *fuzzy logic controller*, kecepatan reference diatur pada 2800 RPM menggunakan potensiometer dan direpresentasikan dengan garis berwarna biru. Hasilnya dapat terlihat bahwa kecepatan actual yang

direpresentasikan dengan garis berwarna merah dapat mengikuti kecepatan yang diinginkan atau kecepatan reference seperti terlihat pada gambar dibawah ini :



Gambar 4. 2. Hasil implementasi kontrol kecepatan 2800 RPM

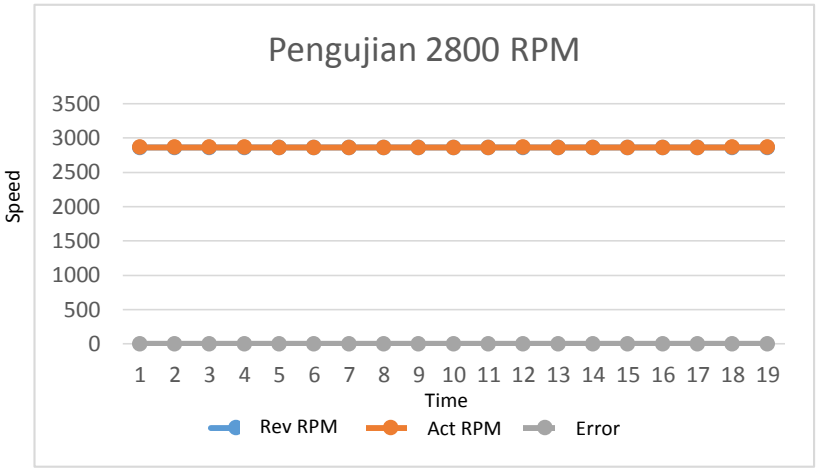
4.2.1.3. Tabel error pada implementasi kontrol kecepetan 2800 RPM

Nilai error dapat dihitung dengan mudah ketika kecepatan sudah dalam keadaan steady state. Berikut adalah nilai error pada implementasi kontrol kecepatan 2800 RPM :

Tabel 4. 2. Error Pada Implementasi Kontrol Kecepatan 2800 RPM

Reference RPM	Actual RPM	Error (%)
2862	2863	0.03
2862	2863	0.03
2862	2863	0.03
2862	2863	0.03
2862	2862	0.00
2862	2862	0.00
2862	2862	0.00
2862	2862	0.00
2862	2862	0.00
2862	2862	0.00
2862	2862	0.00
2862	2863	0.03
2862	2862	0.00
2862	2862	0.00
2862	2862	0.00
2862	2862	0.00
2862	2862	0.00
2862	2863	0.03
2862	2863	0.03
Rata - Rata Error (%)		0.01

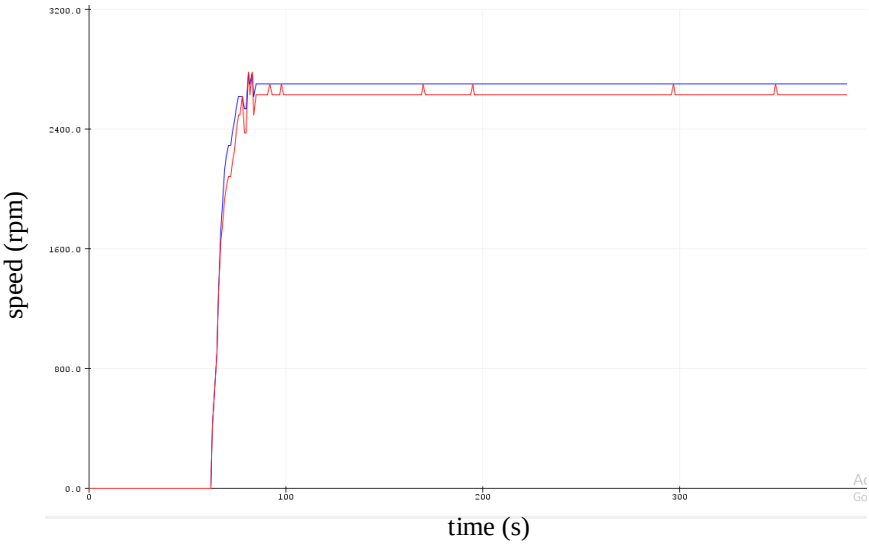
4.2.1.4. Grafik error pada implementasi kontrol kecepetan 2800 RPM



Gambar 4. 3. Grafik error pada implementasi kontrol kecepatan 2800 RPM

4.2.1.5. *Hasil implementasi kontrol kecepatan 4000 RPM tanpa fuzzy logic controller*

Pada implementasi ini dilakukan dengan sistem open loop atau tanpa menggunakan *fuzzy logic controller*, kecepatan reference diatur pada 2800 RPM menggunakan potensiometer dan direpresentasikan dengan garis berwarna biru. Hasilnya dapat terlihat bahwa kecepatan actual yang direpresentasikan dengan garis berwarna merah dapat mengikuti kecepatan yang diinginkan atau kecepatan reference seperti terlihat pada gambar dibawah ini :



Gambar 4. 4. Hasil implementasi kontrol kecepatan 2800 RPM tanpa fuzzy logic controller

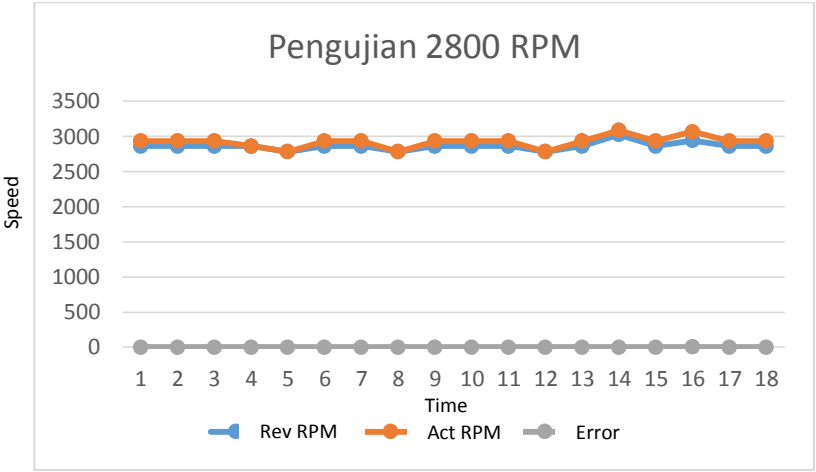
4.2.1.6. *Tabel error pada implementasi kontrol kecepetan 2800 RPM tanpa fuzzy logic controller*

Nilai error dapat dihitung dengan mudah ketika kecepatan sudah dalam keadaan steady state. Berikut adalah nilai error pada implementasi kontrol kecepatan 2800 RPM tanpa menggunakan fuzzy logic controller:

Tabel 4. 3. Error Pada Implementasi Kontrol Kecepatan 2800 RPM Tanpa Fuzzy Logic Controller

Reference RPM	Actual RPM	Error (%)
2862	2934	2.52
2862	2934	2.52
2862	2934	2.52
2862	2862	0.00
2781	2781	0.00
2862	2934	2.52
2862	2934	2.52
2781	2781	0.00
2862	2934	2.52
2862	2934	2.52
2862	2934	2.52
2781	2781	0.00
2862	2934	2.52
3025	3086	2.02
2862	2934	2.52
2944	3068	4.21
2862	2934	2.52
2862	2934	2.52
2862	2934	2.52
Rata - Rata Error (%)		2.05

4.2.1.7. Grafik error *pada implementasi kontrol kecepetan 2800 RPM tanpa fuzzy logic controller*

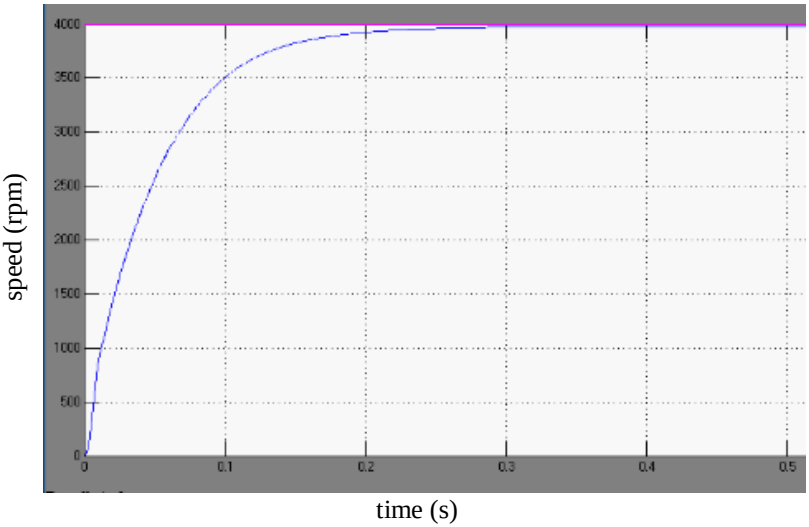


Gambar 4. 5. Grafik error pada implementasi kontrol kecepatan 2800 RPM tanpa Fuzzy Logic Controller

4.2.2. Kecepatan 4000 RPM

4.2.2.1. Hasil simulasi kontrol kecepatan 4000 RPM

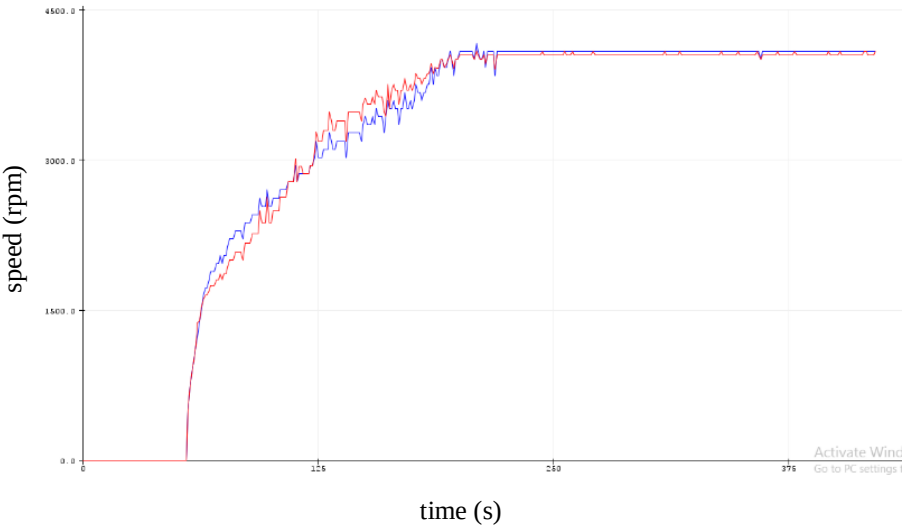
Pada simulasi ini, kecepatan reference diatur pada 4000 RPM dan direpresentasikan dengan garis berwarna merah. Hasilnya dapat terlihat bahwa kecepatan actual yang direpresentasikan dengan garis berwarna biru dapat mengikuti kecepatan yang diinginkan atau kecepatan reference seperti terlihat pada gambar dibawah ini



Gambar 4. 6. Hasil simulasi kontrol kecepatan 4000 RPM

4.2.2.2. Hasil implementasi kontrol kecepatan 4000 RPM

Pada implementasi ini dilakukan dengan sistem close loop atau menggunakan *fuzzy logic controller*, kecepatan reference diatur pada 4000 RPM menggunakan potensiometer dan direpresentasikan dengan garis berwarna biru. Hasilnya dapat terlihat bahwa kecepatan actual yang direpresentasikan dengan garis berwarna merah dapat mengikuti kecepatan yang diinginkan atau kecepatan reference seperti terlihat pada gambar dibawah ini



Gambar 4. 7. Hasil implementasi kontrol kecepatan 4000 RPM

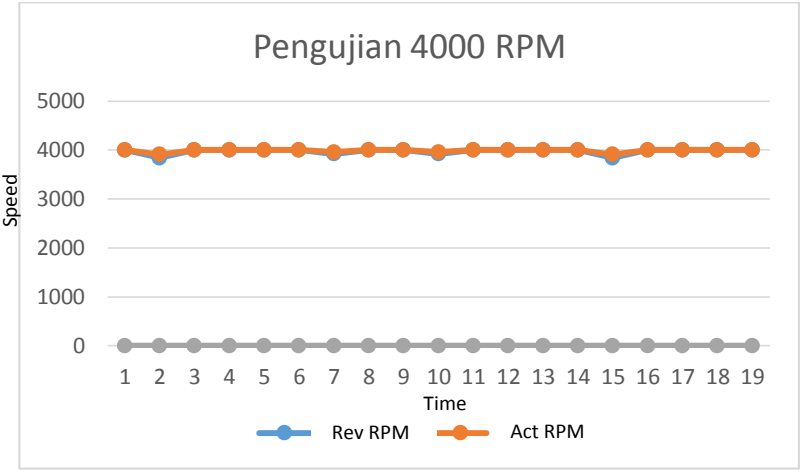
4.2.2.3. Tabel error pada implementasi kontrol kecepetan 4000 RPM

Nilai error dapat dihitung dengan mudah ketika kecepatan sudah dalam keadaan steady state. Berikut adalah nilai error pada implementasi kontrol kecepatan 4000 RPM :

Tabel 4. 4. Error Pada Implementasi Kontrol Kecepatan 4000 RPM

Reference RPM	Actual RPM	Error (%)
4003	4003	0.00
3840	3912	1.88
4003	4003	0.00
4003	4003	0.00
4003	4003	0.00
4003	4003	0.00
3921	3958	0.94
4003	4003	0.00
4003	4003	0.00
3921	3958	0.94
4003	4003	0.00
4003	4003	0.00
4003	4003	0.00
4003	4003	0.00
3840	3912	1.88
4003	4003	0.00
4003	4003	0.00
4003	4003	0.00
4003	4003	0.00
Rata - Rata Error (%)		0.30

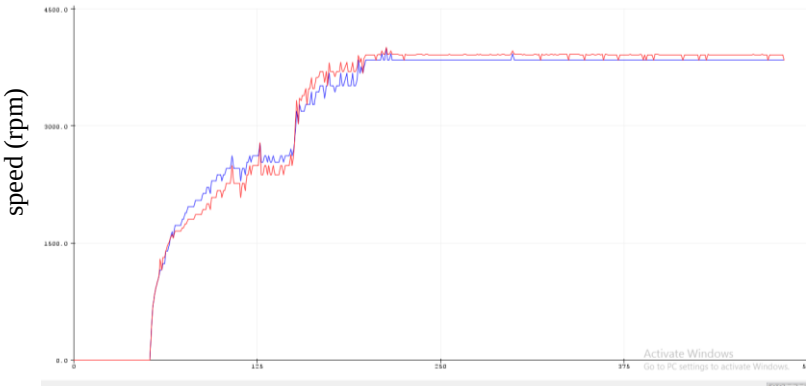
4.2.2.4. Grafik error pada implementasi kontrol kecepetan 4000 RPM



Gambar 4. 8. Grafik error pada implementasi kontrol kecepatan 4000 RPM

4.2.2.5. Hasil implementasi kontrol kecepatan 4000 RPM tanpa fuzzy logic controller

Pada implementasi ini dilakukan dengan sistem open loop atau tanpa menggunakan *fuzzy logic controller*, kecepatan reference diatur pada 4000 RPM menggunakan potensiometer dan direpresentasikan dengan garis berwarna biru. Hasilnya dapat terlihat bahwa kecepatan actual yang direpresentasikan dengan garis berwarna merah dapat mengikuti kecepatan yang diinginkan atau kecepatan reference seperti terlihat pada gambar dibawah ini



Gambar 4. 9. Hasil implementasi kontrol kecepatan 4000 RPM tanpa fuzzy logic controller

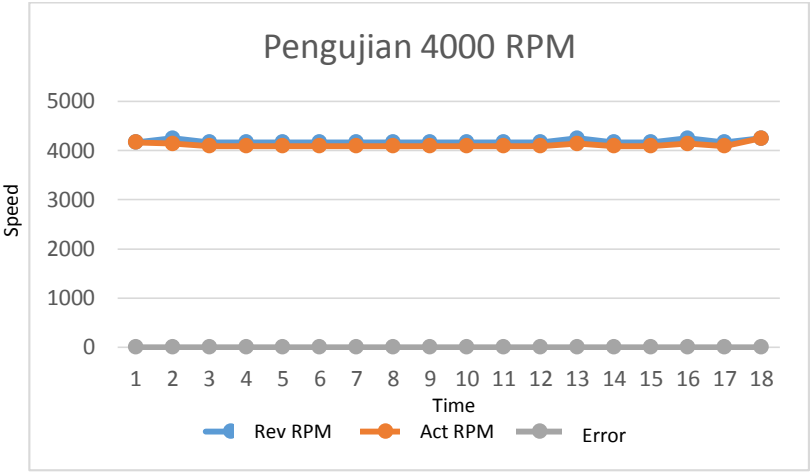
4.2.2.6. Error pada implementasi kontrol kecepatan 4000 RPM tanpa fuzzy logic controller

Nilai error dapat dihitung dengan mudah ketika kecepatan sudah dalam keadaan steady state. Berikut adalah nilai error pada implementasi kontrol kecepatan 4000 RPM :

Tabel 4. 5. Error Pada Implementasi Kontrol Kecepatan 4000 RPM Tanpa Fuzzy Logic Controller

Reference RPM	Actual RPM	Error (%)
4166	4166	0.00
4247	4139	2.54
4166	4090	1.82
4166	4090	1.82
4166	4090	1.82
4166	4090	1.82
4166	4090	1.82
4166	4090	1.82
4166	4090	1.82
4166	4090	1.82
4166	4090	1.82
4166	4090	1.82
4166	4090	1.82
4247	4139	2.54
4166	4090	1.82
4166	4090	1.82
4247	4139	2.54
4166	4090	1.82
4247	4247	0.00
4166	4090	1.82
Rata - rata Error (%)		1.75

4.2.2.7. Error pada implementasi kontrol kecepatan 4000 RPM tanpa fuzzy logic controller

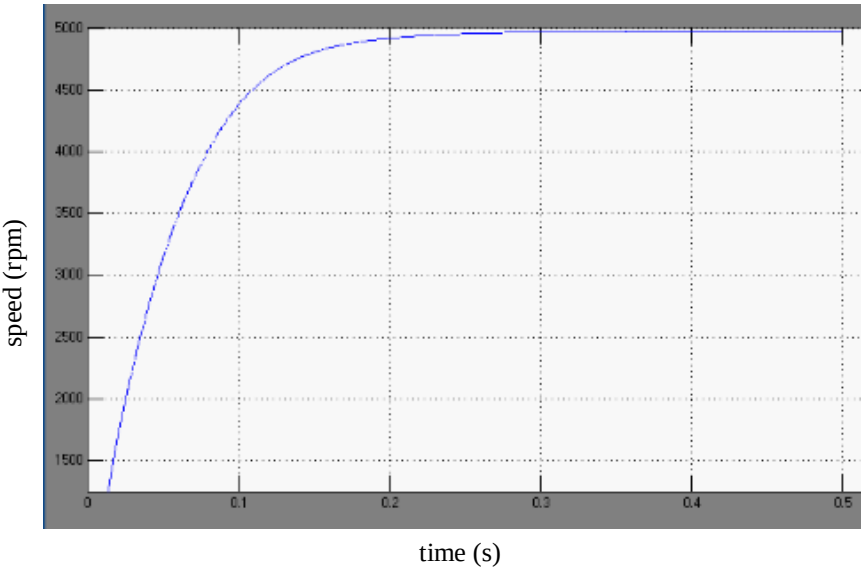


Gambar 4. 10. Grafik error pada implementasi kontrol kecepatan 4000 RPM tanpa fuzzy logic controller

4.2.3. Kecepatan 5000 RPM

4.2.3.1. Hasil simulasi kontrol kecepatan 5000 RPM

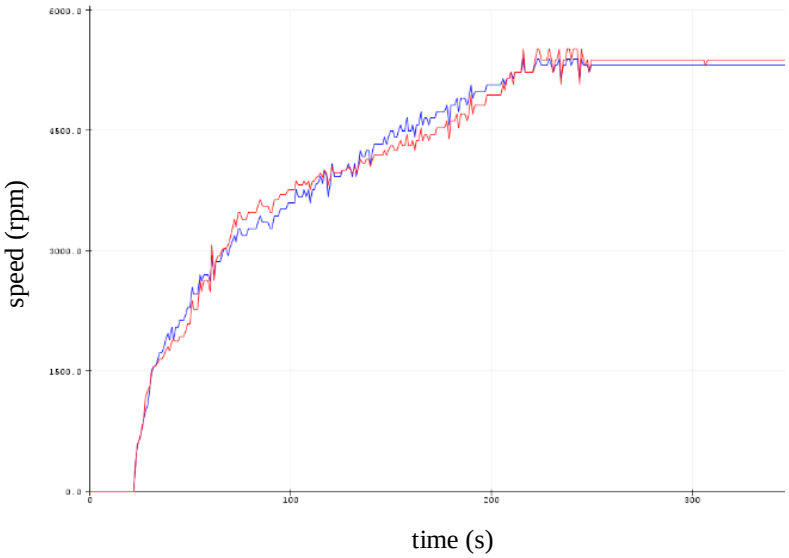
Pada simulasi ini, kecepatan reference diatur pada 5000 RPM dan direpresentasikan dengan garis berwarna merah. Hasilnya dapat terlihat bahwa kecepatan actual yang direpresentasikan dengan garis berwarna biru dapat mengikuti kecepatan yang diinginkan atau kecepatan reference seperti terlihat pada gambar dibawah ini



Gambar 4. 11. Hasil simulasi kontrol kecepatan 5000 RPM

4.2.3.2. Hasil implementasi kontrol kecepatan 5000 RPM

Pada implementasi ini dilakukan dengan sistem close loop atau menggunakan *fuzzy logic controller*, kecepatan reference diatur pada 5000 RPM menggunakan potensiometer dan direpresentasikan dengan garis berwarna biru. Hasilnya dapat terlihat bahwa kecepatan actual yang direpresentasikan dengan garis berwarna merah dapat mengikuti kecepatan yang diinginkan atau kecepatan reference seperti terlihat pada gambar dibawah ini :



Gambar 4. 12. Hasil implementasi kontrol kecepatan 5000 RPM

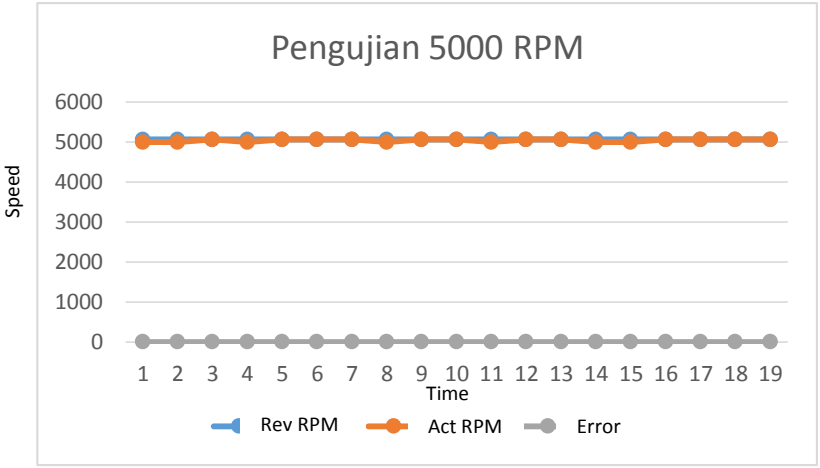
4.2.3.3. Error pada implementasi kontrol kecepetan 5000 RPM

Nilai error dapat dihitung dengan mudah ketika kecepatan sudah dalam keadaan steady state. Berikut adalah nilai error pada implementasi kontrol kecepatan 5000 RPM :

Tabel 4. 6. Error Pada Implementasi Kontrol Kecepatan 5000 RPM

Reference RPM	Actual RPM	Error (%)
5062	4995	1.32
5062	4995	1.32
5062	5062	0.00
5062	4995	1.32
5062	5062	0.00
5062	5062	0.00
5062	5062	0.00
5062	4995	1.32
5062	5062	0.00
5062	5062	0.00
5062	4995	1.32
5062	5062	0.00
5062	5062	0.00
5062	4995	1.32
5062	4995	1.32
5062	5062	0.00
5062	5062	0.00
5062	5062	0.00
5062	5062	0.00
Rata - Rata Error (%)		0.49

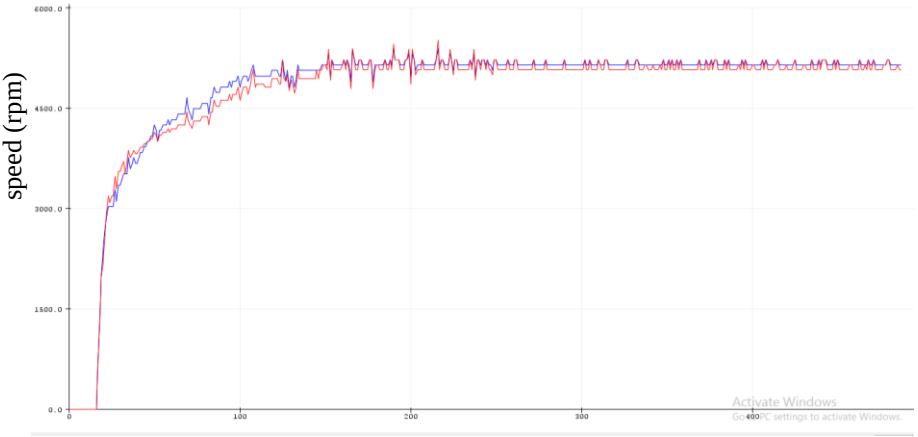
4.2.3.4. Error pada implementasi kontrol kecepetan 5000 RPM



Gambar 4. 13. Grafik error pada implementasi kontrol kecepatan 5000 RPM

4.2.2.5. Hasil implementasi kontrol kecepatan 5000 RPM tanpa fuzzy logic controller

Pada implementasi ini dilakukan dengan sistem open loop atau tanpa menggunakan *fuzzy logic controller*, kecepatan reference diatur pada 5000 RPM menggunakan potensiometer dan direpresentasikan dengan garis berwarna biru. Hasilnya dapat terlihat bahwa kecepatan actual yang direpresentasikan dengan garis berwarna merah dapat mengikuti kecepatan yang diinginkan atau kecepatan reference seperti terlihat pada gambar dibawah ini



Gambar 4. 14. Hasil implementasi kontrol kecepatan 5000 RPM tanpa fuzzy logic controller

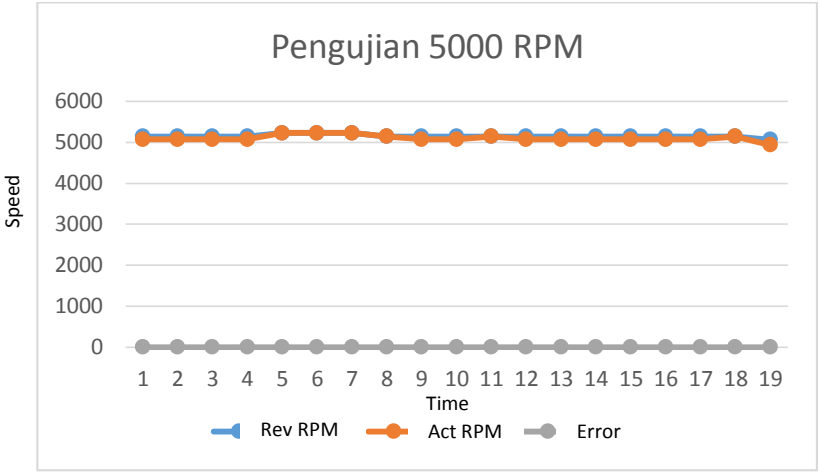
4.2.2.6. Error pada implementasi kontrol kecepatan 5000 RPM tanpa fuzzy logic controller

Nilai error dapat dihitung dengan mudah ketika kecepatan sudah dalam keadaan steady state. Berikut adalah nilai error pada implementasi kontrol kecepatan 5000 RPM :

Tabel 4. 7. Error Pada Implementasi Kontrol Kecepatan 5000 RPM Tanpa Fuzzy Logic Controller

Reference RPM	Actual RPM	Error (%)
5143	5072	1.38
5143	5072	1.38
5143	5072	1.38
5143	5072	1.38
5225	5225	0.00
5225	5225	0.00
5225	5225	0.00
5143	5143	0.00
5143	5072	1.38
5143	5072	1.38
5143	5143	0.00
5143	5072	1.38
5143	5072	1.38
5143	5072	1.38
5143	5072	1.38
5143	5072	1.38
5143	5072	1.38
5143	5143	0.00
5062	4937	2.47
Rata - Rata Error (%)		1.00

4.2.2.7. Error pada implementasi kontrol kecepatan 5000 RPM tanpa fuzzy logic controller

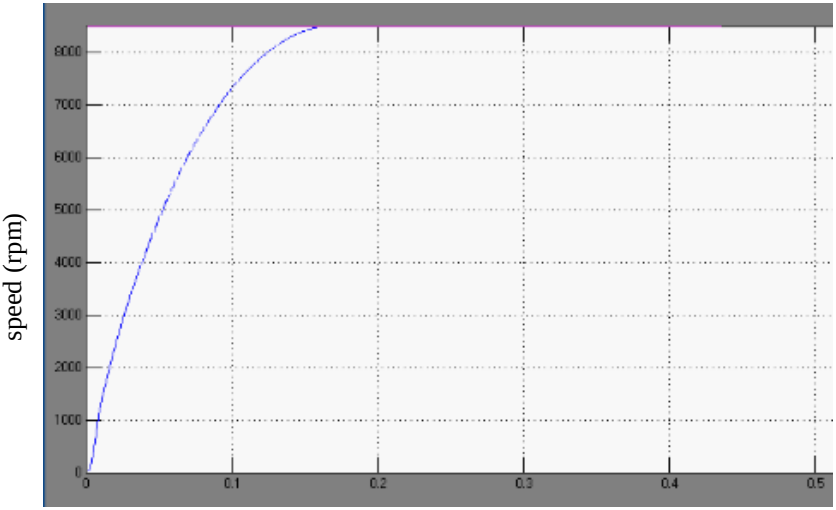


Gambar 4. 15. Grafik error pada implementasi kontrol kecepatan 5000 RPM tanpa fuzzy logic controller

4.2.4. Kecepatan 8500 RPM

4.2.4.1. Hasil simulasi kontrol kecepatan 8500 RPM

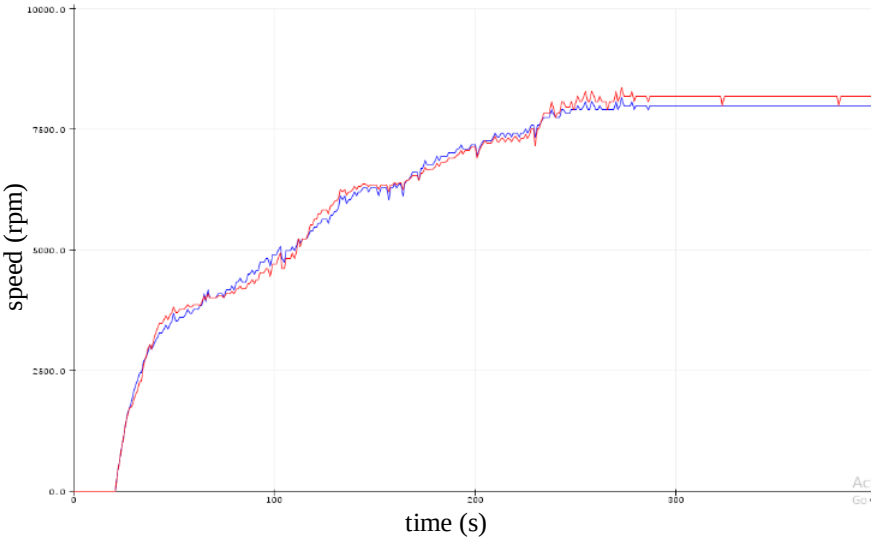
Pada simulasi ini, kecepatan reference diatur pada 5000 RPM dan direpresentasikan dengan garis berwarna merah. Hasilnya dapat terlihat bahwa kecepatan actual yang direpresentasikan dengan garis berwarna biru dapat mengikuti kecepatan yang diinginkan atau kecepatan reference seperti terlihat pada gambar dibawah ini



Gambar 4. 16. Hasil implementasi kontrol kecepatan 8500 RPM

4.2.4.2. Hasil implementasi kontrol kecepatan 8500 RPM

Pada implementasi ini dilakukan dengan sistem close loop atau menggunakan *fuzzy logic controller*, kecepatan reference diatur pada 5000 RPM menggunakan potensiometer dan direpresentasikan dengan garis berwarna biru. Hasilnya dapat terlihat bahwa kecepatan actual yang direpresentasikan dengan garis berwarna merah dapat mengikuti kecepatan yang diinginkan atau kecepatan reference seperti terlihat pada gambar dibawah ini :



Gambar 4. 17. Hasil Implementasi kontrol kecepatan 8500 RPM

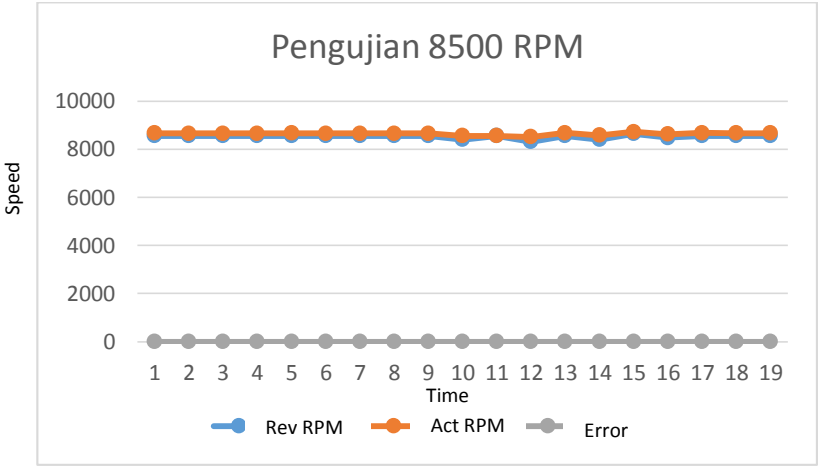
4.2.4.3. Error pada implementasi kontrol kecepetan 8500 RPM

Nilai error dapat dihitung dengan mudah ketika kecepatan sudah dalam keadaan steady state. Berikut adalah nilai error pada implementasi kontrol kecepatan 8500 RPM :

Tabel 4. 8. Error Pada Implementasi Kontrol Kecepatan 8500 RPM

Reference RPM	Actual RPM	Error (%)
8564	8668	1.21
8564	8657	1.09
8564	8657	1.09
8564	8657	1.09
8564	8668	1.21
8564	8657	1.09
8564	8657	1.09
8564	8657	1.09
8564	8657	1.09
8401	8557	1.86
8564	8564	0.00
8320	8518	2.38
8564	8686	1.42
8401	8584	2.18
8645	8730	0.98
8483	8624	1.66
8564	8677	1.32
8564	8668	1.21
8564	8668	1.21
Rata - Rata Error (%)		1.28

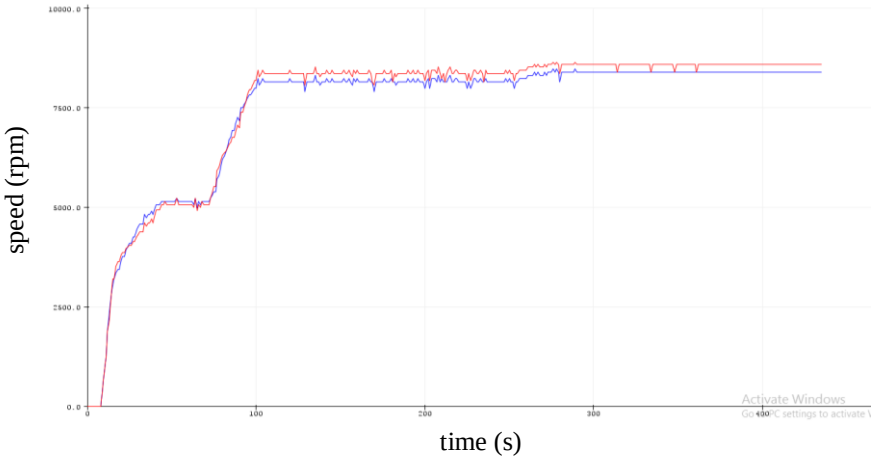
4.2.4.4. Grafik error pada implementasi kontrol kecepetan 8500 RPM



Gambar 4. 18. Grafik error pada implementasi kontrol kecepatan 8500 RPM

4.2.2.5. Hasil implementasi kontrol kecepatan 8500 RPM tanpa fuzzy logic controller

Pada implementasi ini dilakukan dengan sistem open loop atau tanpa menggunakan *fuzzy logic controller*, kecepatan reference diatur pada 5000 RPM menggunakan potensiometer dan direpresentasikan dengan garis berwarna biru. Hasilnya dapat terlihat bahwa kecepatan actual yang direpresentasikan dengan garis berwarna merah dapat mengikuti kecepatan yang diinginkan atau kecepatan reference seperti terlihat pada gambar dibawah ini :



Gambar 4. 19. Hasil Implementasi kontrol kecepatan 8500 RPM tanpa fuzzy logic controller

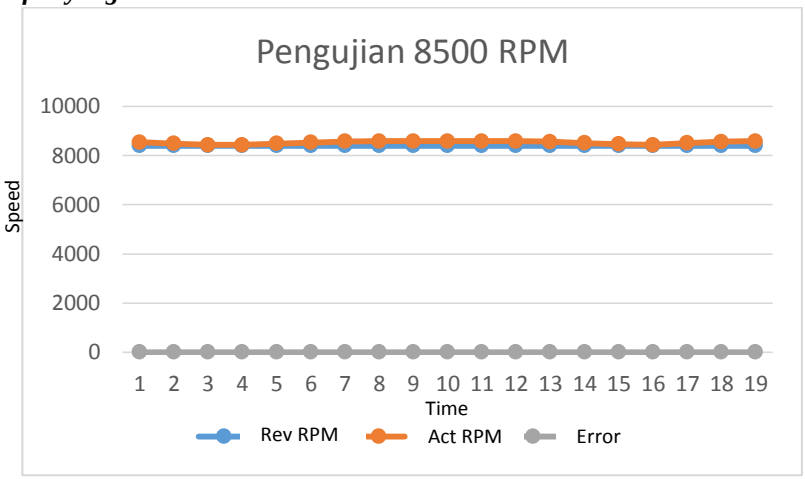
4.2.2.6. Error pada implementasi kontrol kecepatan 8500 RPM tanpa fuzzy logic controller

Nilai error dapat dihitung dengan mudah ketika kecepatan sudah dalam keadaan steady state. Berikut adalah nilai error pada implementasi kontrol kecepatan 8500 RPM :

Tabel 4. 9. Error Pada Implementasi Kontrol Kecepatan 8500 RPM tanpa fuzzy logic controller

Reference RPM	Actual RPM	Error (%)
8401	8541	1.67
8401	8483	0.98
8401	8432	0.37
8401	8432	0.37
8401	8483	0.98
8401	8523	1.45
8401	8571	2.02
8401	8584	2.18
8401	8584	2.18
8401	8584	2.18
8401	8584	2.18
8401	8584	2.18
8401	8557	1.86
8401	8504	1.23
8401	8459	0.69
8401	8432	0.37
8401	8504	1.23
8401	8557	1.86
8401	8584	2.18
Rata - Rata Error (%)		1.48

4.2.2.7. Error pada implementasi kontrol kecepatan 8500 RPM tanpa fuzzy logic controller



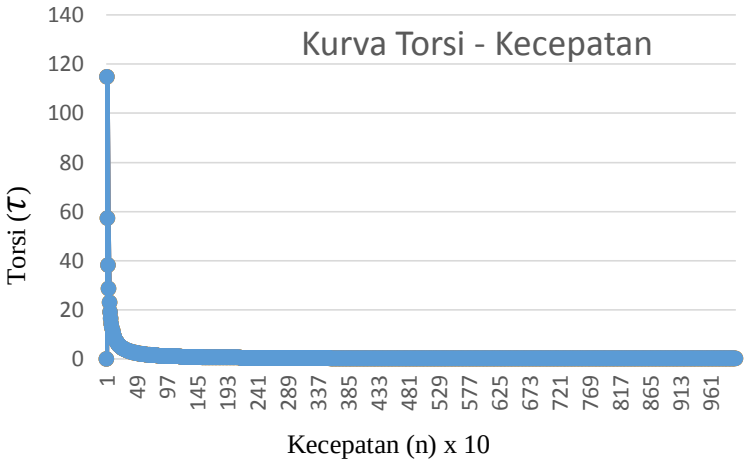
Gambar 4. 20. Grafik error pada implementasi kontrol kecepatan 8500 RPM

4.3. Pengujian Torsi

Pada penelitian ini, hanya dilakukan perubahan kecepatan sedangkan beban mekanis yang digunakan tetap. Sehingga torsi dapat dihitung menggunakan rumus :

$$\tau = \frac{120}{2\,\pi\,(\frac{n}{60})}$$

Dari rumus tersebut didapatkan data kurva karakteristik torsi – kecepatan seperti gambar ini :



Gambar 4. 21. Kurva Torsi – Kecepatan

Agar lebih jelas, berikut adalah tabel data torsi terhadap kecepatan 2800, 4000, 5000, dan 8500 RPM.

Tabel 4. 10. Tabel Torsi terhadap kecepatan 2800, 4000, 5000, dan 8500 RPM

RPM	Torsi
2800	0.41
4000	0.29
5000	0.23
8500	0.13
Rata - Rata Torsi	0.27

Bersdasarkan tabel diatas, diketahui bahwa rata – rata torsi yang dihasilkan pada kecepatan telah diuji yaitu sebesar 0.27 Nm

4.4. Analisis Data dan Pembahasan

Setelah melakukan beberapa pengujian dan mendapatkan data – data tersebut, maka akan dilakukan analisis data dan pembahasan seperti dibawah ini :

4.4.1. Analasis Uji Akurasi IR Sensor

Data uji akurasi IR sensor pada tabel 4.1 *Pengujian IR Sensor* menunjukkan bahwa rata – rata error dari pembacaan IR sensor yang digunakan pada penelitian ini yaitu 2.11%. Artinya modul IR sensor ini memliki akurasi yang tidak jauh berbeda dengan alat tachometer TM – 300. Dan penggunaan IR sensor untuk penelitian masih layak untuk digunakan karena error yang terjadi kurang dari 5% sehingga diharapkan hasil atau implementasi alat dari penelitian ini memiliki error yang kecil juga.

4.4.2. Analisis Pengujian Kontrol Kecepatan

Pada penelitian ini dilakukan pengujian dengan cara membandingkan antara hasil simulasi, implementasi alat menggunakan sistem close loop (menggunakan fuzzy logic), dan implementasi alat yang menggunakan sistem open loop (tidak menggunakan fuzzy logic).

Berdasarkan gambar grafik pengujian masing - masing, antara hasil simulasi, implementasi alat menggunakan sistem close loop dan implementasi alat yang menggunakan sistem open loop memiliki output yang sesuai harapan yaitu output (kecepatan actual) dapat mengikuti input (kecepatan referensi).

Namun dari pengujian diatas, didapatkan data bahwa implementasi alat yang menggunakan sistem close loop (dengan fuzzy logic) memiliki rata – rata error yang lebih kecil jika dibandingkan dengan sistem open loop (tanpa fuzzy logic). Agar perbedaannya lebih jelas, dapat dilihat tabel dibawah ini :

Tabel 4. 11. Perbedaan Error Antara Simulasi, Implementasi Alat Dengan FCL, dan Implementasi Alat Tanpa FCL

RPM	Error (%)		
	Simulasi	Implementasi Alat dengan FCL	Implementasi Alat tanpa FCL
2800	0.52	0.01	2.05
4000	0.52	0.30	1.75
5000	0.52	0.49	1.00
8500	0.00001	1.28	1.48
Rata – Rata Error (%)	0.39	0.52	1.57

Pada tabel diatas, dapat dilihat bahwa hasil dari simulasi memiliki nilai error yang lebih stabil karena sistem bekerja dalam keadaan ideal atau tanpa gangguan dari luar sistem.

Jika melihat hasil dari implementasi alat dengan FCL atau sistem close loop memiliki nilai error yang lebih besar ketika nilai RPM dinaikkan. Hal ini dikarenakan alat mengalami vibrasi saat nilai RPM semakin tinggi sehingga mengakibatkan pembacaan kecepatan oleh IR Sensor juga ternggangguu akibat vibrasi tersebut dan ouput yang dihasilkan juga semakin jelek.

Sedangkan hasil dari implementasi alat tanpa FCL atau sistem open loop memiliki nilai error yang lebih besar jika dibandingkan dengan hasil simulasi maupun implementasi alat dengan sistem close loop. Selain itu nilai error yang dihasilkan juga berubah – ubah tidak bergantung pada nilai RPM yang diberikan. Hasil ini terjadi akibat tidak adanya perbaikan terhadap nilai error yang dihasilkan atau feedback kepada sistem dan merupakan kelemahan utama dari sistem open loop.

4.4.3. Analisis Pengujian Torsi

Berdasarkan pengujian torsi yang telah dilakukan, torsi yang dihasilkan pada 2800 RPM sebesar 0.41 Nm sedangkan torsi yang dihasilkan pada 8500 RPM sebesar 0.13 Nm, sehingga didapatkan penurunan torsi sebesar 68.29%.

4.4.4. Penggunaan Prototype Kendaraan Untuk Kondisi Jalan Menanjak dan Menurun

Untuk kondisi jalan menanjak, kendaraan harus membutuhkan torsi yang besar agar mampu menggerakkan kendaraan keatas tanpa hambatan, caranya yaitu dengan mengatur gas atau RPM yang pas agar menghasilkan torsi terbesar dari kendaraan tersebut.

Pada prototype kendaraan atau implementasi alat ini, dilakukan uji coba atau tes jalan dengan asumsi beban tetap dan untuk penggunaan di jalan yang menanjak, maka diperlukan torsi yang paling besar yaitu 0.41 Nm sehingga prototype kendaraan ini harus menjaga kecepatannya stabil pada 2800 RPM agar kendaraan dapat melewati jalanan menanjak. Inilah pentingnya kontrol kecepatan pada sebuah kendaraan agar RPM dapat mudah diatur sehingga torsi pun juga mudah diatur.

Sedangkan pada kondisi jalan menurun, kendaraan tidak memerlukan torsi yang besar, cukup memanfaatkan gaya grafitasi kendaaran dapat mudah melewati jalan menurun. Tetapi jika kecepatan tetap diberikan meskipun jalanan menurun akan menyebabkan potensi bahaya pada kendaraan karena memilkik kecepatan yang telalu kencang dan mengakibatkan kendaraan sulit dikontrol. Sehingga dalam kondisi jalan menurun, prototype kendaraan atau implementasi alat ini cukup mengatur kecepatan pada 0 RPM agar torsi yang dihasilkan juga 0 Nm.

--- halaman ini sengaja dikosongkan ---

BAB 5
KESIMPULAN DAN SARAN

5.1. Kesimpulan

Berdasarkan pembahasan diatas, dapat diambil beberapa kesimpulan yaitu :

1. Kontrol kecepatan menggunakan fuzzy terbukti dapat diimplementasikan dan memiliki hasil yang tidak jauh berbeda dengan hasil simulasi.
2. Hasil simulasi memiliki rata – rata error sebesar 0.39% sedangkan hasil implementasi alat memiliki rata - rata error sebesar 0.52%
3. Implementasi alat menggunakan sistem close loop (dengan *fuzzy logic controller*) memiliki error yang lebih kecil dibandingkan dengan sistem open loop (tanpa *fuzzy logic controller*)
4. Pengaturan kecepatan pada RPM tinggi akan mengakibatkan menurunnya torsi sebesar 68.29%

5.2. Saran

1. Agar pembacaan kecepatan lebih akurat, disarankan agar menggunakan BLDC yang telah *built in* sensor kecepatan karena penggunaan sensor tambahan biasanya memiliki eror yang lebih besar
2. Untuk hasil kontrol yang lebih bagus, disarankan agar menambah jumlah fungsi keanggotaan dari fuzzyfikasi sehingga rule fuzzy juga ikut bertambah dan hasil outputnya juga semakin baik
3. Menggunakan perbedaan beban agar prototype alat dapat menirukan kondisi sebenarnya ketika dipakai berkendara.

--- halaman ini sengaja dikosongkan ---

DAFTAR PUSTAKA

[1] J. N. Ansari and S. L, "Speed Control of BLDC motor for Electric Vehicle," *International Journal of Engineering Research & Technology (IJERT)*, vol. 3, no. 5, pp. 1666 - 1671, 2014.

[2] Hartono, "Optimization of Tsukamoto Fuzzy Inference System using Fuzzy Grid Partition," *International Journal of Computer Science and Network*, vol. 5, p. 6, 5 October 2016.

[3] J. Zhao and Y. Yu, "Brushless DC Motor Fundamentals," 2011.

[4] A. Jaya, M. B. Fauziah, E. Purwanto, F. D. Murdianto, M. R. Rusli and G. Prabowo, "Design of PID-Logic for Speed Control of Brushless DC Motor in Dynamic Electric Vehicle to Improve Steady-State Performance," in *International Electronics Symposium on Engineering Technology and Application (IES-ETA)*, Surabaya, 2017.

[5] S. K, "Design of Fuzzy Logic Controller for Speed Control of Sensorless BLDC Motor Drive," in *International Conference on Control, Power, Communication and Computing Technologies (ICCPCCT)*, Kochi, 2018.

[6] M. Mohan, R. K. P and G. S, "Speed Control of Brushless DC Motor Using Fuzzy Based Controllers," *IRJET*, vol. 2, pp. 875-881, 2015.

[7] T. Mathew and C. A. Sam, "Closed Loop Control of BLDC Motor Using a Fuzzy Logic Controller and Single Current Sensor," in *International Conference on Advanced Computing and Communication System (ICACCS)*, Cochin, 2013.

[8] P. P. Wach, Dynamics and Control of Electrical Drives, Verlag Berlin Heidelberg: Springer, 2011.

[9] . K. Manikandan and B. V. Premkumar, "Adaptive Neuro Fuzzy Inference System based Speed Controller for Brushless DC Motor," *Elsevier Neurocomputing*, vol. 138, pp. 260-270, 2014.

[10] H. Nasution, "Implementasi Logika Fuzzy pada Sistem Kecerdasan Buatan," *ELKHA*, vol. 4, p. 2, 2012.

[11] T. and A. H. Wijaya, "Remote Fuzzy Logic Control System For a DC Motor Speed Control," *Jurnal Teknik Elektro*, vol. 2, pp. 8-12, 2002.

[12] L. K. Agrawal, B. K. Chauhan and K. B. G, "Speed Control of Brushless DC Motor Using Fuzzy," *International Journal of Pure and Applied Mathematics*, vol. 119, pp. 2689-2696, 2018.

LAMPIRAN

A. Program Arduino Menggunakan Fuzzy Logic Controller

```
#include <Servo.h>

#include <LiquidCrystal.h>


LiquidCrystal lcd(12, 11, 5, 4, 3, 2); //
inisialisasi pin LCD 16x2


Servo ESC;                      // create servo
object to control the ESC

int potValue;                   // pin
(potensiometer)


const int IrSensor = 0;   // input IR sensor pin
0


const int toESC =  9;   //output ke ESC pin 9


int photoState = 0;

long RPM = 0;

unsigned long timerrevCalc = 0;

unsigned long duration;
```

```
//--------------------------------------//
//Deklarasi variable untuk Fuzzy Logic //
//--------------------------------------//

float input = 0;            // nilai input dari
pembacaan potensiometer
float output = 0;           // nilai output dari
pembacaan RPM
float eror = 0;             // eror = input -
output

float rule00, rule01, rule02, rule03, rule04,
rule05, rule06 = 0;
float rule10, rule11, rule12, rule13, rule14,
rule15, rule16 = 0;
float rule20, rule21, rule22, rule23, rule24,
rule25, rule26 = 0;
float rule30, rule31, rule32, rule33, rule34,
rule35, rule36 = 0;
float rule40, rule41, rule42, rule43, rule44,
rule45, rule46 = 0;
float rule50, rule51, rule52, rule53, rule54,
rule55, rule56 = 0;
```

```
float rule60, rule61, rule62, rule63, rule64,
rule65, rule66 = 0;


float NBin, NMin, NSin, ZEin, PSin, PMin, PBin =
0; //himpunan input fuzzy

float NBer, NMer, NSer, ZEer, PSer, PMer, PBer =
0; //himpunan eror fuzzy

float out0, out1, out2, out3, out4, out5, out6 =
0; //himpunan uotput fuzzy

float z0, z1, z2, z3, z4, z5, z6 = 0; //himpunan
uotput fuzzy




//-----------------------------------------------
------//




void setup() {

  Serial.begin(9600); //

  // Attach the ESC on pin 10

  ESC.attach(9,1000,2000); // (pin, min pulse
width, max pulse width in microseconds)

  pinMode(toESC, OUTPUT);

  pinMode(IrSensor, INPUT);
```

```
  // set up the LCD's number of columns and
rows:
  lcd.begin(16, 2);
  // Print a message to the LCD.
  lcd.print("hello, world!");
}
void loop() {


 //---------------//
 // Pembacaan RPM //
 //---------------//


  duration = pulseIn(IrSensor, HIGH, 100000);
//times the amount of microseconds the notch is
exposing the IR, Times out after 100000 uS.
Raise the timeout for slower RPM readings.
  timerrevCalc = duration * 50.7801; //See above
  RPM = 60000000/timerrevCalc; //See above
 // Serial.print ("RPM =");
  //Serial.print (RPM);
  //Serial.print (" ");
```

```
  //-------------------------//
  //   Fuzzyfikasi Input    //
  //-------------------------//


  potValue = analogRead(A1);    // pin A1, reads
the value of the potentiometer (value between 0
and 1023)
  potValue = map(potValue, 0, 1023, 0, 180);
// scale it to use it with the servo library
(value between 0 and 180)
input = potValue;
//input = 50;
 if (input <= 0 )
              {
                NBin = 1;
              }
     else if ((input > 0) && (input <=30))
              {
                NBin = 1-((input - (0))/(30));
              }
        else  {
                NBin =0;
              }
```

```
if ((input > 0) && (input <=30))
            {
              NMin = (input - (0))/(30);
            }
   else if ((input > 30) && (input <= 60))
            {
              NMin = 1- ((input - (30))/(30));
            }
      else  {
              NMin = 0;
            }

if ((input > 30) && (input <=60))
            {
              NSin = (input - (30))/(30);
            }
   else if ((input > 60) && (input <= 90))
            {
              NSin = 1- ((input - (60))/(30));
            }
      else  {
              NSin = 0;
```

```
                }

  if ((input > 60) && (input <=90))
                {
                  ZEin = (input - (60))/(30);
                }
     else if ((input > 90) && (input <= 120))
                {
                  ZEin = 1- ((input - (90))/(30));
                }
        else  {
                  ZEin = 0;
                }

  if ((input > 90) && (input <=120))
                {
                  PSin = (input - (90))/(30);
                }
     else if ((input > 120) && (input <= 150))
                {
                  PSin = 1- ((input -
(120))/(30));
                }
```

```
        else  {
                PSin = 0;
              }

  if ((input > 120) && (input <=150))
              {
                PMin = (input - (120))/(30);
              }
     else if ((input > 150) && (input <= 180))
              {
                PMin = 1- ((input -
(150))/(30));
              }
        else  {
                PMin = 0;
              }

  if ((input > 150) && (input <=180))
              {
                PBin = (input - (150))/(30);
              }
     else if (input <= 150)
              {
```

```
                PBin = 0;
              }
        else  {
                PBin = 1;
              }

//------------------------//
//   Fuzzyfikasi Eror     //
//------------------------//

 //mengubah nilai RPM ke PWM, DARI DATA HASIL
REGRESI RPM DAN PWM
  x = (RPM - 6.115)/2.160;
  eror = x;
 if (eror <= 0 )
              {
                NBer = 1;
              }
     else if ((eror > 0) && (eror <=30))
              {
                NBer = 1-((eror - (0))/(30));
              }
        else  {
```

```
                NBer =0;
              }

if ((eror > 0) && (eror <=30))
              {
                NMer = (eror - (0))/(30);
              }
   else if ((eror > 30) && (eror <= 60))
              {
                NMer = 1- ((eror - (30))/(30));
              }
      else  {
                NMer = 0;
              }

if ((eror > 30) && (eror <=60))
              {
                NSer = (eror - (30))/(30);
              }
   else if ((eror > 60) && (eror <= 90))
              {
                NSer = 1- ((eror - (60))/(30));
```

```
            }
      else  {
              NSer = 0;
            }

if ((eror > 60) && (eror <=90))
            {
              ZEer = (eror - (60))/(30);
            }
   else if ((eror > 90) && (eror <= 120))
            {
              ZEer = 1- ((eror - (90))/(30));
            }
      else  {
              ZEer = 0;
            }

if ((eror > 90) && (eror <=120))
            {
              PSer = (eror - (90))/(30);
            }
   else if ((eror > 120) && (eror <= 150))
```

```
            {
              PSer = 1- ((eror - (120))/(30));
            }
      else  {
              PSer = 0;
            }

if ((eror > 120) && (eror <=150))
            {
              PMer = (eror - (120))/(30);
            }
   else if ((eror > 150) && (eror <= 180))
            {
              PMer = 1- ((eror - (150))/(30));
            }
      else  {
              PMer = 0;
            }

if ((eror > 150) && (eror <=180))
            {
              PBer = (eror - (150))/(30);
```

```
                }
      else if (eror <= 150)
                {
                  PBer = 0;
                }
         else  {
                  PBer = 1;
                }


//------------------------//
//   Fuzzyfikasi Output   //
//------------------------//

  out0 = 0;
  out1 = 30;
  out2 = 60;
  out3 = 90;
  out4 = 120;
  out5 = 150;
  out6 = 180;
```

```
//---------------------------//
//-------RULE FUZZY----------//
//---------------------------//

 rule00 = min(NBer, NBin);
 rule10 = min(NMer, NBin);
 rule20 = min(NSer, NBin);
 rule30 = min(ZEer, NBin);
 rule40 = min(PSer, NBin);
 rule50 = min(PMer, NBin);
 rule60 = min(PBer, NBin);

 rule01 = min(NBer, NMin);
 rule11 = min(NMer, NMin);
 rule21 = min(NSer, NMin);
 rule31 = min(ZEer, NMin);
 rule41 = min(PSer, NMin);
 rule51 = min(PMer, NMin);
 rule61 = min(PBer, NMin);

 rule02 = min(NBer, NSin);
```

```
rule12 = min(NMer, NSin);
rule22 = min(NSer, NSin);
rule32 = min(ZEer, NSin);
rule42 = min(PSer, NSin);
rule52 = min(PMer, NSin);
rule62 = min(PBer, NSin);

rule03 = min(NBer, ZEin);
rule13 = min(NMer, ZEin);
rule23 = min(NSer, ZEin);
rule33 = min(ZEer, ZEin);
rule43 = min(PSer, ZEin);
rule53 = min(PMer, ZEin);
rule63 = min(PBer, ZEin);


rule04 = min(NBer, PSin);
rule14 = min(NMer, PSin);
rule24 = min(NSer, PSin);
rule34 = min(ZEer, PSin);
rule44 = min(PSer, PSin);
rule54 = min(PMer, PSin);
```

```
rule64 = min(PBer, PSin);

rule05 = min(NBer, PMin);
rule15 = min(NMer, PMin);
rule25 = min(NSer, PMin);
rule35 = min(ZEer, PMin);
rule45 = min(PSer, PMin);
rule55 = min(PMer, PMin);
rule65 = min(PBer, PMin);

rule06 = min(NBer, PBin);
rule16 = min(NMer, PBin);
rule26 = min(NSer, PBin);
rule36 = min(ZEer, PBin);
rule46 = min(PSer, PBin);
rule56 = min(PMer, PBin);
rule66 = min(PBer, PBin);
```

```
//---------------------------//
//-------Defuzzifikasi-------//
//---------------------------//


 // metode sugeno
float z =0 ;
 float pwm =0;
 float pembagiZ =0;
 float temp =0;


  long revRPM;
  revRPM = input*8.969 - 66.96;
  if (revRPM >= 0){revRPM;}
  else if (revRPM < 0){revRPM=0;}


//z = (rule00 * NBout)+(rule10 * NBout)+(rule20
* NBout)+(rule30 * NBout)+(rule40 *
NMout)+(rule50 * NSout)+(rule60 * ZEout)+(rule01
* NBout)+(rule11 * NBout)+(rule21 *
NBout)+(rule31 * NMout)+(rule41 * NSout)+(rule51
* ZEout)+(rule61 * PSout)+(rule02 *
NBout)+(rule12 * NBout)+(rule22 * NMout)+(rule32
* NSout)+(rule42 * ZEout)+(rule52 *
PSout)+(rule62 * PMout)+(rule03 * NBout)+(rule13
* NMout)+(rule23 * NSout)+(rule33 *
ZEout)+(rule43 * PSout)+(rule53 * PMout)+(rule63
* PBout)+(rule04 * NMout)+(rule14 *
```

```
NSout)+(rule24 * ZEout)+(rule34 * PSout)+(rule44
* PMout)+(rule54 * PBout)+(rule64 *
PBout)+(rule05 * NSout)+(rule15 * ZEout)+(rule25
* PSout)+(rule35 * PMout)+(rule45 *
PBout)+(rule55 * PBout)+(rule65 * PBout)+(rule06
* ZEout)+(rule16 * PSout)+(rule26 *
PMout)+(rule36 * PBout)+(rule46 * PBout)+(rule56
* PBout)+(rule66 * PBout);
```

```
z0 = (rule00 * out0) + (rule01 * out1) + (rule02
* out2) + (rule03 * out3) + (rule04 * out4) +
(rule05 * out5) + (rule06 * out6);

z1 = (rule10 * out0) + (rule11 * out1) + (rule12
* out2) + (rule13 * out3) + (rule14 * out4) +
(rule15 * out5) + (rule16 * out6);

z2 = (rule20 * out0) + (rule21 * out1) + (rule22
* out2) + (rule23 * out3) + (rule24 * out4) +
(rule25 * out5) + (rule26 * out6);

z3 = (rule30 * out0) + (rule31 * out1) + (rule32
* out2) + (rule33 * out3) + (rule34 * out4) +
(rule35 * out5) + (rule36 * out6);

z4 = (rule40 * out0) + (rule41 * out1) + (rule42
* out2) + (rule43 * out3) + (rule44 * out4) +
(rule45 * out5) + (rule46 * out6);

z5 = (rule50 * out0) + (rule51 * out1) + (rule52
* out2) + (rule53 * out3) + (rule54 * out4) +
(rule55 * out5) + (rule56 * out6);

z6 = (rule60 * out0) + (rule61 * out1) + (rule62
* out2) + (rule63 * out3) + (rule64 * out4) +
(rule65 * out5) + (rule66 * out6);
```

```
float ztotal;

ztotal = z0 + z1 + z2 + z3 + z4  + z5 + z6;

  //pembagiZ =  NBout + NMout + NSout + ZEout +
PSout + PMout + PBout;


  pembagiZ = rule00 + rule01 + rule02 + rule03 +
rule04 + rule05 + rule06 + rule10 + rule11 +
rule12 + rule13 + rule14 + rule15 + rule16 +
rule20 + rule21 + rule22 + rule23 + rule24 +
rule25 + rule26 + rule30 + rule31 + rule32 +
rule33 + rule34 + rule35 + rule36 + rule40 +
rule41 + rule42 + rule43 + rule44 + rule45 +
rule46 + rule50 + rule51 + rule52 + rule53 +
rule54 + rule55 + rule56 + rule60 + rule61 +
rule62 + rule63 + rule64 + rule65 + rule66;




  temp = ztotal / pembagiZ;


  pwm = temp ;

  float hasil;

  hasil = input - pwm;

  if (hasil > 5) {
```

```
    ESC.write(input);
  }
  else if ( hasil < 5){
    ESC.write(pwm);    // Send the signal to the
ESC
  //ESC.write(potValue);
  }

 float RevInput;
RevInput = (81.45*input) + 338.0;

  if (RevInput > 338){RevInput;}
  else if (RevInput <= 338){RevInput=0;}

float Output;
Output = (81.45*pwm) + 338.0;


  if (Output > 338){Output;}
  else if (Output <= 338){Output=0;}

// Serial.print("input =");
 Serial.print(RevInput);
```

```
 Serial.print("  ");


 //Serial.print("output =");
Serial.print(Output);
Serial.println("  ");

}
```

B. Program Arduino Tanpa Menggunakan Fuzzy Logic Controller

```
include <Servo.h>

#include <LiquidCrystal.h>


LiquidCrystal lcd(12, 11, 5, 4, 3, 2); //
inisialisasi pin LCD 16x2


Servo ESC;                      // create servo
object to control the ESC

int potValue;                   // pin
(potensiometer)


const int IrSensor = 0;   // input IR sensor pin
0


const int toESC =  9;   //output ke ESC pin 9


int photoState = 0;

long RPM = 0;

unsigned long timerrevCalc = 0;

unsigned long duration;


float input = 0;            // nilai input dari
pembacaan potensiometer
```

```
float output = 0;           // nilai output dari
pembacaan RPM

void setup() {

  Serial.begin(9600); //

  // Attach the ESC on pin 10

  ESC.attach(9,1000,2000); // (pin, min pulse
width, max pulse width in microseconds)

  pinMode(toESC, OUTPUT);

  pinMode(IrSensor, INPUT);


  // set up the LCD's number of columns and
rows:

  lcd.begin(16, 2);

  // Print a message to the LCD.

  lcd.print("hello, world!");

}


void loop() {




  duration = pulseIn(IrSensor, HIGH, 100000);
//times the amount of microseconds the notch is
exposing the IR, Times out after 100000 uS.
Raise the timeout for slower RPM readings.

  timerrevCalc = duration * 50.7801; //See above
```

```
  RPM = 60000000/timerrevCalc; //See above



  potValue = analogRead(A1);    // pin A1, reads
the value of the potentiometer (value between 0
and 1023)
  potValue = map(potValue, 0, 1023, 0, 180);
// scale it to use it with the servo library
(value between 0 and 180)
input = potValue;


float RevInput;
RevInput = (81.45*input) + 338.0;


  if (RevInput > 338){RevInput;}
  else if (RevInput <= 338){RevInput=0;}


float Output;
Output = (37.71*RPM) + 107.4;



  if (Output > 107.4){Output;}
  else if (Output <= 107.4){Output=0;}
}
```

C. Lampiran Alat

--- halaman ini sengaja dikosongkan ---

BIODATA PENULIS

Penulis, lahir di Kota Probolinggo pada tanggal 13 Desember 1997, merupakan anak kedua dari tiga bersaudara. Penulis menyelesaikan pendidikan dasar di SDN Sukabumi 2 Kota Probolinggo, lalu melanjutkan pendidikan tingkat menengah pertama di SMPN 1 Probolinggo dan Pendidikan tingkat menengah atas di SMAN 1 Probolinggo dengan jurusan ilmu pengetahuan alam. Memulai perkuliahan pada tahun 2016 di S1 Teknik Elektro, Fakultas Teknologi Elektro dan Informatika Cerdas. Selama masa perkuliahan, penulis mengambil program studi teknik sistem tenaga. Penulis juga aktif dalam kegiatan riset robot cerdas dan sering meraih penghargaan mewakili ITS dalam kegiatan Kontes Robot Indonesia baik tingkat regional maupun nasional.

Email : ddeksaraka13@gmail.com

WA : 081252146126

Line : deksarakadanier

--- halaman ini sengaja dikosongkan ---