DESAIN SISTEM KONTROL KECEPATAN MOTOR DC BERBEBAN MENGGUNAKAN *FUZZY LOGIC CONTROL SYSTEM* BERBASIS ARDUINO

TESIS

TEKNIK ELEKTRO KONSENTRASI SISTEM KONTROL ELEKTRONIK

Diajukan untuk memenuhi persyaratan
Memperoleh gelar Magister Teknik

ISMAIL
NIM 126060300111023

UNIVERSITAS BRAWIJAYA
FAKULTAS TEKNIK
MALANG
2017

DESAIN SISTEM KONTROL KECEPATAN MOTOR DC BERBEBAN MENGGUNAKAN *FUZZY LOGIC CONTROL SYSTEM* BERBASIS ARDUINO

TESIS

Oleh:

Nama	:	Ismail
NIM	:	126060300111023
Program Magister	:	Teknik Elektro
Minat	:	Sistem Kontrol Elektronik

Menyetujui
Komisi Pembimbing,

Ketua,	Anggota,
M. Aziz Muslim, ST., MT., Ph.D.	Dr. Ir. Bambang Siswojo, MT.
NIP. 19741203 200012 1 001	NIP. 19621211 198802 1 001

Mengetahui,
Ketua Program Studi Magister Teknik Elektro

Dr. Eng. Panca Mudjirahardjo, ST., MT.
NIP. 19700329 200012 1 001

JURUSAN TEKNIK ELEKTRO
FAKULTAS TEKNIK
UNIVERSITAS BRAWIJAYA
MALANG
2017

JUDUL TESIS:

DESAIN SISTEM KONTROL KECEPATAN MOTOR DC BERBEBAN MENGGUNAKAN FUZZY LOGIC CONTROL SYSTEM BERBASIS ARDUINO

Nama Mahasiswa	:	Ismail
NIM	:	126060300111023
Program Studi	:	Program Magister Teknik Elektro
Minat	:	Sistem Kontrol Elektronik

KOMISI PEMBIMBING :

Ketua	:	M. Aziz Muslim, ST., MT., Ph.D.
Anggota	:	Dr. Ir. Bambang Siswojo, MT.

TIM DOSEN PENGUJI :

Dosen Penguji 1	:	Dr. Ir. Ponco Siwindarto, M.Eng.Sc.
Dosen Penguji 2	:	Dr.Eng. Panca Mudjirahardjo, S.T., M.T.
Tanggal Ujian	:	17 November 2017
SK Penguji	:	1077 Tahun 2017

PERNYATAAN ORISINALITAS PENELITIAN TESIS

Saya menyatakan dengan sebenar-benarnya bahwa sepanjang pengetahuan saya dan berdasarkan hasil penelusuran berbagai karya ilmiah, gagasan dan masalah ilmiah yang diteliti dan diulas di dalam Naskah Tesis ini adalah asli dari pemikiran saya. Tidak terdapat karya ilmiah yang pernah diajukan oleh orang lain untuk memperoleh gelar akademik di suatu Perguruan Tinggi, dan tidak terdapat karya atau pendapat yang pernah ditulis atau diterbitkan oleh orang lain, kecuali yang secara tertulis dikutip dalam naskah ini dan disebutkan dalam sumber kutipan dan daftar pustaka.

Apabila ternyata di dalam naskah Tesis ini dapat dibuktikan terdapat unsur-unsur jiplakan, saya bersedia Tesis dibatalkan, serta diproses sesuai dengan peraturan perundang-undangan yang berlaku (UU No. 20 Tahun 2003, pasal 25 ayat 2 dan pasal 70).

Malang, November 2017

Mahasiswa,

ISMAIL

NIM: 126060300111023

Karya Ilmiah ini kutujukan kepada:

Istri tercinta, Mukayanah

serta ketiga anakku, Chica, Chici, dan Bagas

RIWAYAT HIDUP

Ismail, lahir di Sidoarjo pada 01 Juli 1964, anak dari **Bapak (Alm.) Abdul Rokhman** dan **Ibu (Alm.) Yatimah**. Bersekolah di **SDN Lajuk**, melanjutkan ke **SMP Bhayangkari Porong**, dan **STMN Sidoarjo**, lulus tahun 1984. Studi S1 di **Teknik Komputer Universitas Muhammadiyah Sidoarjo**. Pengalaman kerja sebagai staf pengajar Akademi Teknik Informatika pada tahun 2009-2012. Melanjutkan studi program magister (S2) di **Program Magister Teknik Elektro Jurusan Elektro Fakultas Teknik Universitas Brawijaya** pada tahun 2012-2017.

UCAPAN TERIMA KASIH

Dalam penyelesaian penelitian tesis ini, penulis banyak mendapatkan bantuan dari berbagai pihak. Untuk itu penulis menyampaikan ucapan terima kasih setulusnya kepada:

1. M. Azis Muslim, ST., MT., Ph.D. selaku pembimbing utama yang senantiasa memberikan arahan dan garis besar di setiap bimbingan sehingga benar-benar menyalakan semangat penulis dalam penelitian tesis ini.
2. Dr. Ir. Bambang Siswojo, MT. selaku pembimbing kedua yang selalu aktif memberikan masukan-masukan teknis sehingga esensi penelitian tesis ini benar-benar muncul ke permukaan.
3. Istri tercinta dan anak-anak yang senantiasa ikut mendukung semangat dan motivasi saya serta doa yang tak henti-hentinya sehingga saya bisa melaksanakan sidang kompre.

Malang, November 2017

Penulis

RINGKASAN

Ismail, Jurusan Teknik Elektro, Fakultas Teknik Universitas Brawijaya, Juni 2017, *Desain Sistem Kontrol Kecepatan Motor DC Berbeban Menggunakan Fuzzy Logic Control System Berbasis Arduino*, Dosen Pembimbing: M. Azis Muslim dan Bambang Siswojo.

Salah satu komponen mesin listrik yang memegang peranan penting pada proses industri adalah sistem penggerak, di mana motor DC merupakan salah satu jenis penggerak yang sangat banyak digunakan. Di dalam sistem penggerak motor DC tersebut sangat diperlukan pengendali untuk mendapatkan gerakan yang dibutuhkan berikut mempertahankan kecepatan putarannya sedapat mungkin sama sesuai dengan acuan yang diberikan. Selain itu, pengendali juga merupakan komponen yang berguna untuk menekan sinyal kesalahan sehingga dapat diperoleh unjuk kerja yang diinginkan. Dalam penelitian ini akan dilakukan perancangan sistem pengendalian kecepatan putar motor DC menggunakan perangkat keras kendali *microcontroller* menggunakan Arduino Mega 2560 untuk memenuhi spesifikasi kontrol pada aplikasi pengendalian skala laboratorium (*prototype*) dan *Fuzzy Control System* sebagai algoritma sistem pengendaliannya.

Fuzzy Control System dengan *Mamdani Inference Engine* digunakan karena kemampuannya untuk mudah dimodelkan menggunakan intuitif manusia, bersifat adaptif, tidak membutuhkan persamaan matematika yang kompleks, tidak terbatas pada sistem linier ataupun konstan, dan mudah disesuaikan dengan masukan dari manusia.

Menggunakan *microcontroller* Arduino Mega 2560 dan beberapa komponen pendukung, dibangun sebuah perangkat untuk melakukan pengujian sistem kendali menggunakan *Fuzzy Control System* dengan *Mamdani Inference Engine*. Perangkat tersebut dilengkapi dengan motor DC yang dikendalikan dan sensor kecepatan putarnya. Selain itu perangkat tersebut bisa diprogram dengan berbagai *set-point* dalam ranah waktu, dan dapat menerima perubahan penalaan parameter fuzzy mulai dari proses fuzzifikasi, penataan daftar aturan, sampai dengan defuzzifikasi. Untuk menerima dan melakukan analisa hasil pengujian, dibuat juga perangkat lunak *Data Logger* yang mampu menyimpan data dalam volume besar dan menampilkan melalui antarmuka grafis yang mudah dibaca.

Hasil pengujian menyatakan bahwa *overshoot* dan *ringing* merupakan besaran yang berkebalikan dengan *rise time*, namun dengan menggunakan *Fuzzy Control System* yang merupakan pengendali tidak linier besaran tersebut bisa diminimalkan, baik pada waktu beroperasi tanpa beban maupun dengan beban kualitatif tertentu. Selain itu penalaan parameter fuzzy menjadi sangat mudah dan intuitif berkat penggunaan *Mamdani Inference Engine*.

Kata kunci : *Fuzzy Control System*, Arduino, Motor DC, Kendali Kecepatan.

SUMMARY

Ismail, *Department of Electrical Engineering, Faculty of Engineering, University of Brawijaya, June 2017, Design of Loaded DC Motor Speed Control System using Arduino-Based Fuzzy Logic Control System, Academic Supervisor:* M. Aziz Muslim *and* Bambang Siswojo.

One of the electrical machinery components that play an important role in industrial processes is the drive system, in which the DC motor is one of the most widely used. In the DC motor drive system, it is necessary to control the required motion required as well as maintaining the rotation speed as much as possible in accordance with the given reference (set-points). In addition, the controller is also a useful component to suppress the error signal so that the desired performance can be obtained. This research designs system of DC motor rotation speed control using Arduino Mega 2560 microcontroller to fulfill control specification on laboratory scale control application and Fuzzy Control System as control system algorithm.

Fuzzy Control System with Mamdani Inference Engine is used because of its ability to be easily modeled using human intuitive, adaptive, does not require complex mathematical equations, not limited to linear or constant systems, and easily adapted to human input.

Using Arduino Mega 2560 microcontroller and several additional components, a device is then built for the control system testing using Fuzzy Control System with Mamdani Inference Engine. The device includes DC motor to be controlled and its rotation speed sensor. Moreover, the device receive angular speed program sequence and fuzzy setting from users. The program sequence consists of set-points within time domain, while the fuzzy setting consists of fuzzification parameters, customizable rule table, and defuzzification parameters. In order to overcome Arduino limitation due to limited memory capacity, an external Data Logger software has been made to accomodate high volume testing data, with an advanced easy-to-read graphical display interface.

As described by the testing result, it is shown that both overshoot and ringing variables are in opposite with rise time variable. Using Fuzzy Control System as a non-linear control method, however, those variables can be minimized, whether without or under specific qualitative loads. Furthermore, fuzzy parameters setting process could be done easily, thanks to the Mamdani Inference Engine.

Keywords : Fuzzy Control System, Arduino, DC Motor, Speed Control.

KATA PENGANTAR

Perkembangan teknologi, khususnya sistem kendali elektronik, telah mencapai taraf yang menggantikan evolusi teknologi mesin industri. Dewasa ini, pengembangan dunia industri lebih terlihat dari teknologi elektronik penunjangnya dibandingkan dengan teknologi mesin penggeraknya. Salah satu teknologi yang menunjang dunia industri adalah sistem pengendalian yang membantu mekanisme pengatur kecepatan pada motor listrik. Dengan mengambil topik tentang sistem kendali motor DC dan penerapannya menggunakan *fuzzy control system*, diharapkan dunia pendidikan bisa menyumbangkan ilmu terapan untuk mendukung teknologi permesinan.

Melalui penelitian ini dicoba untuk mengulas penerapan mekanisme pengendalian kecepatan putar motor DC menggunakan *fuzzy control system*. Penelitian ini juga diajukan sebagai bagian dari tugas akhir dalam rangka menyelesaikan studi di Program Magister Teknik Elektro di Universitas Brawijaya Malang bidang keahlian Sistem Kontrol Elektronika.

Diharapkan penelitian ini membuka jalan bagi penelitian lain untuk menyempurnakan hasil yang telah diperoleh. Tinjauan dan saran yang bersifat membangun sangat diharapkan demi peningkatan kesempurnaan dan lebih berguna bagi perkembangan ilmu pengetahuan khususnya dan bagi masyarakat umumnya.

Malang, November 2017

Penulis

DAFTAR ISI

Halaman

DAFTAR TABEL

DAFTAR GAMBAR

DAFTAR SIMBOL

Besaran Dasar	Satuan dan Singkatannya	Simbol
Set-Point (Kecepatan Diminta)	revolution per minute atau RPM	y_S
Process Value (Kecepatan Terukur)	revolution per minute atau RPM	y
Error (Kesalahan Kecepatan)	revolution per minute atau RPM	e
Delta Error (Selisih Kesalahan)	revolution per minute atau RPM	de
Control Action (Aksi Kendali)	-	u
Control Value (Nilai Kendali)	-	v
Resolusi Proses	micro second atau μs	r
Frekuensi Putar	Hertz atau Hz revolution per minute atau RPM	ω
Banyaknya Lubang Terdeteksi	-	H
Banyaknya Lubang *Optical Chopper*	-	C_H
Selisih Waktu Pengambilan Cuplikan Deteksi Lubang	second atau s	Δt
Waktu Cuplikan Saat Ini	-	t
Waktu Cuplikan Sebelumnya	-	t_{-1}
Tegangan Keluaran	Volt atau V	V
Nilai Pin Arduino	-	B

BAB I

PENDAHULUAN

1.1. Latar Belakang

Salah satu komponen mesin listrik yang memegang peranan penting pada proses industri adalah sistem penggerak, di mana motor DC merupakan salah satu jenis penggerak yang sangat banyak digunakan. Kebutuhan akan kecepatan putar motor DC yang bervariasi dan terprogram banyak dijumpai pada beberapa aplikasi, contohnya mesin cuci, mesin *food processor*, dan konveyor di pabrik. Dalam kasus ini, kecepatan putar motor DC yang dibutuhkan berubah-ubah sesuai dengan kebutuhan proses yang terprogram dalam *angular speed sequence program* dan perubahan (transisi) kecepatan tersebut harus terjadi dalam waktu yang cukup singkat. Sistem tersebut harus mampu melakukan hal-hal berikut:

1. Merespon kebutuhan perubahan kecepatan putar (*target speed*) sesingkat mungkin dan bersifat *predictable*.
2. Menghindari osilasi kecepatan putar setelah terjadi perubahan dari kecepatan yang sebelumnya.
3. Meminimalisir naik atau turunnya kecepatan putar motor DC dari kecepatan putar yang dibutuhkan (*target speed*) ketika terdapat perubahan beban.

Beberapa penelitian terdahulu menunjukkan hasil kualitatif, tetapi belum menunjukkan hasil kuantitatif. Seperti pada penelitian oleh Azadi (2012), yang menyatakan Pengendali Azadi mampu menstabilkan putaran motor DC dengan meminimalkan *overshoot* dan *undershoot*, tetapi belum menunjukkan besaran *overshoot* yang dihasilkan oleh sistem (Azadi, *et. al*., 2012). Penelitian tentang *neuro-fuzzy controller* menggunakan *particle swarm optimization* juga tidak memberikan hasil nominal tetang besaran *overshoot* dan *undershoot* serta ayunan, dan hanya memberikan hasil kualitatif serta digunakan sebagai sistem yang fleksibel (Farid, *et. al*., 2014).

Di dalam sistem penggerak motor DC tersebut sangat diperlukan pengendali untuk mendapatkan gerakan yang dibutuhkan berikut mempertahankan kecepatan putarannya sedapat mungkin sama sesuai dengan acuan yang diberikan. Selain itu, pengendali juga

merupakan komponen yang berguna untuk menekan sinyal kesalahan sehingga dapat diperoleh unjuk kerja yang diinginkan. Supaya dapat bekerja optimal, pengendali harus memiliki respon yang cepat dan secara terus menerus melakukan tugasnya. Karena pentingnya komponen pengendali, maka komponen ini memiliki sistem tersendiri yang disebut sistem pengendalian.

Pengendalian kecepatan pada motor DC sangat diperlukan terutama pada saat diperlukan kecepatan tertentu yang konstan dan bertahan pada nilai tertentu. Salah satu aplikasinya adalah sistem *variable pitch propeller* pada pesawat terbang, helikopter, UAV, atau kapal laut. Sistem tersebut membutuhkan kecepatan putar yang konstan dan dipertahankan tanpa mempedulikan daya dorong yang dihasilkan, karena variasi daya dorong didapatkan dari variasi sudut bilah (*propeller*) yang diatur dengan sebuah aktuator. Sistem ini memiliki beberapa kelebihan dibandingkan sistem *fixed pitch propeller* yang mengharuskan variasi kecepatan bilah untuk variasi daya dorong, di antaranya dengan putaran yang tetap bisa dihasilkan daya dorong bervariasi bahkan sampai memiliki arah berlawanan, serta kemampuannya untuk berbagi satu motor untuk menggerakkan banyak bilah. Bagaimanapun juga, kedua sistem tersebut sangat bergantung pada sistem pengendali kecepatan, seperti yang akan dibahas dalam penelitian ini.

Dalam mendesain sebuah sistem pengendalian, hal terpenting yang harus diperhatikan adalah spesifikasi unjuk kerja sistem yang dibutuhkan. Sistem pengendalian merupakan sistem dinamik, sehingga spesifikasi sistem yang dikendalikan harus bisa mengirim umpan balik (*feedback*) dengan resolusi waktu yang cukup tinggi. Algoritma pengendalian yang digunakan bervariasi sesuai dengan kebutuhan, misalnya *Proportional Integral Derivative* (PID), *Fuzzy Control System* (FCS), *Artificial Neural Network* (ANN), dan *Genetic Algorithm* (GA).

Perangkat keras kendali elektronik banyak digunakan sebagai sistem pengendalian motor DC. Perangkat keras kendali ada yang berbasis analog menggunakan rangkaian penguat operasional, serta ada yang berbasis digital menggunakan *Microcontroller* dan *Programmable Logic Controller* (PLC). *Microcontroller* dipilih dalam pengembangan aplikasi skala laboratorium karena harga kapasitas memori yang rendah.

Dalam penelitian ini akan dilakukan perancangan sistem pengendalian kecepatan putar motor DC menggunakan perangkat keras kendali *microcontroller* menggunakan Arduino Mega 2560 untuk memenuhi spesifikasi kontrol pada aplikasi pengendalian skala

laboratorium (*prototype*) dan *Fuzzy Control System* (*Mamdani Inference Engine*) sebagai algoritma sistem pengendaliannya. Penggunaan Arduino ditujukan karena sifatnya yang mudah diprogram dan sesuai dengan kebutuhan *prototyping* (pemodelan lab). Sedangkan *Fuzzy Control System* digunakan karena kemampuannya untuk mudah dimodelkan menggunakan intuitif manusia, bersifat adaptif, tidak membutuhkan persamaan matematika yang kompleks, tidak terbatas pada sistem linier ataupun konstan, dan mudah disesuaikan dengan masukan dari manusia.

Motor DC yang digunakan akan diberi urutan *set-point* yang bervariasi (kecepatan putar yang diminta dari rendah ke tinggi dan sebaliknya) dan beban yang bervariasi untuk kemudian diuji unjuk kerjanya dengan cara mengamati respon kecepatan putarannya.

1.2. Rumusan Masalah

Permasalahan yang ada dirumuskan sebagai berikut: Bagaimana merancang sistem pengendalian kecepatan putar motor DC menggunakan *Fuzzy Logic Control* berbasis Arduino.

1.3. Tujuan Penelitian

Tujuan dari penelitian ini adalah untuk merancang sistem pengendalian kecepatan putar motor DC menggunakan *Fuzzy Logic Control* berbasis Arduino guna mendapatkan kecepatan putar yang akurat, responsif, *overshoot*/*undershoot* yang minimal, dan *ringing* yang minimal sesuai dengan kebutuhan.

1.4. Batasan Masalah

Karena tujuan penelitian diperlukan untuk dibatasi, maka pembahasan penelitian ini memiliki ruang lingkup, antara lain:

1. Motor DC memiliki spesifikasi: tegangan 20 VDC, arus maksimal 220 mA, dan *output power* 4.4 W.
2. Algoritma yang digunakan untuk pengendali adalah *Fuzzy Control System Mamdani Inference Engine*.
3. Jenis-jenis *fuzzy set* untuk masukan dan keluaran yang digunakan adalah *triangle* dan *trapezoid*.

4. Maksimal *overlapping* dari *fuzzy set* adalah dua keanggotaan.
5. Masukan yang digunakan adalah *angular speed sequence program* dari operator, *start program button*, *emergency stop button*, dan umpan balik sensor kecepatan motor DC.
6. Keluaran yang digunakan adalah sinyal kendali menuju ke *driver* motor DC.
7. Perangkat keras pengendali yang digunakan adalah *microcontroller* Arduino Mega 2560.
8. Pembebanan pada motor DC dilakukan secara kualitatif (tidak dihitung dengan numerik) dan tidak terjadwal.

1.5. Manfaat Penelitian

Hasil dari penelitian ini diharapkan bermanfaat khususnya bagi pemakai yang membutuhkan kecepatan putar motor DC yang konstan sesuai dengan kebutuhan *set-point* terprogram dan memiliki respon cepat terhadap perubahan kebutuhan kecepatan atau perubahan beban.

Selain itu hasil penelitian ini juga bisa bermanfaat untuk para peneliti lain yang membutuhkan hubungan antara respon dan kehalusan dengan nilai pada parameter sistem kendali, khususnya dengan menggunakan *fuzzy control system*.

BAB II

TINJAUAN PUSTAKA

Pada bab ini dipaparkan beberapa dasar teori yang digunakan dalam penelitian. Dalam penelitian ini digunakan *fuzzy control system* pada Arduino untuk pengendalian kecepatan motor DC berbeban.

2.1. Sistem Pengendali

Sistem pengendali (atau sistem kendali, *control system*) adalah sebuah sistem yang terdiri dari sekumpulan perangkat yang mengelola, memerintah, mengarahkan, atau mengatur perilaku perangkat atau sistem lain untuk mencapai hasil yang diinginkan. Selain mendapatkan masukan berupa sekumpulan perintah dari operator, sistem pengendali juga menganalisa *feedback* (umpan balik) aksi objek untuk memperbaiki (memberikan koreksi) atas kesalahan yang timbul.

Sebuah sistem kendali bisa dikatakan memiliki empat fungsi, yaitu mengukur, membandingkan, menghitung, dan memperbaiki. Empat fungsi tersebut dilakukan oleh lima elemen, yaitu detektor, transduser, pemancar, pengendali, dan elemen kendali akhir. Fungsi pengukuran dilakukan oleh detektor, transduser, dan pemancar, yang biasanya dalam aplikasi praktis ketiga elemen tersebut berada dalam satu unit. Fungsi pembandingan dan penghitungan dilakukan oleh pengendali. Sedangkan fungsi perbaikan dilakukan terhadap elemen kendali akhir.

Keberhasilan sistem pengendali dapat diukur berdasarkan beberapa kriteria sebagai berikut:

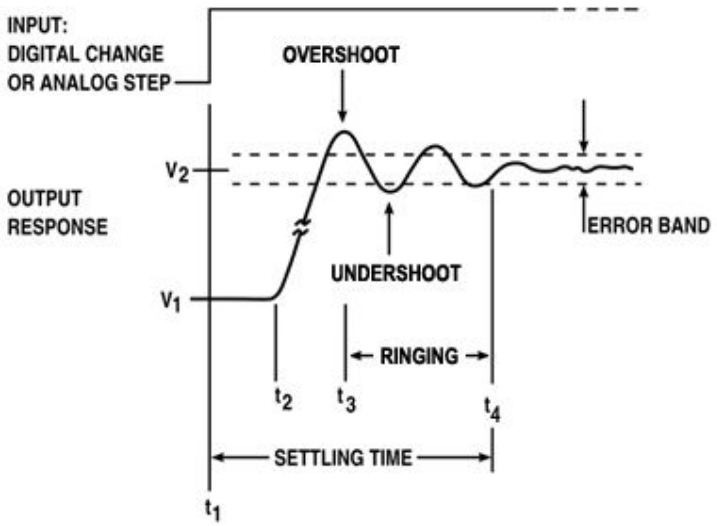
2.1.1. Overshoot dan Undershoot

Overshoot adalah satu kondisi di mana nilai sinyal atau fungsi melampaui *steady-state* (acuannya, atau *set-point* – nilai yang diinginkan). Lawannya, *undershoot*, terjadi ketika nilai sinyal atau fungsi lebih kecil dari acuannya. Baik *overshoot* maupun *undershoot* kebanyakan terjadi diikuti oleh ayunan (*ringing*, *oscillation*) yang diakibatkan oleh respon berlebihan berlawanan arah yang dilakukan oleh sebuah sistem kendali.

Dalam sebuah sistem kendali, *maximum overshoot* (atau *maximum undershoot*) didefinisikan sebagai nilai puncak maksimal dari kurva respon yang terukur terhadap respon yang diinginkan sistem. Baik *overshoot* maupun *undershoot* merupakan fenomena yang tidak diinginkan pada sebuah sistem kendali. Sistem kendali harus memenuhi syarat pengendalian di mana setiap *overshoot* (atau *undershoot*) yang terjadi harus memiliki nilai di bawah *maximum overshoot* (atau *maximum undershoot*). (Vukic, *et. al.*, 2003)

2.1.2. Ringing

Ringing adalah ayunan sinyal yang khususnya terjadi setelah ada perubahan masukan (*set-point*) yang mendadak. Pada beberapa kasus, *ringing* adalah fenomena yang tidak diinginkan, meskipun tidak selalu. Fenomena *ringing* sangat erat kaitannya dengan *overshoot*, yang dalam hal ini ringing terjadi mengikuti *overshoot*. (Vukic, *et. al.*, 2003)



Gambar 2.1 Sebuah ilustrasi pada fenomena *overshoot* yang diikuti oleh *ringing* dan *settling time*.
Sumber: Vukic, *et. al.*, 2003.

Urutan terjadinya fenomena tersebut adalah: *overshoot*, *ringing*, dan *settling time*, seperti pada Gambar 2.1.

2.1.3. Steady State

Sebuah sistem atau proses dikatakan pada keadaan *steady state* (stabil) jika variabel yang menentukan perilaku sistem atau proses dalam kondisi tidak berubah terhadap waktu.

Pada model waktu yang kontinyu, hal ini berarti variabel p dari sistem akan bernilai nol pada turunan parsial terhadap waktu, seperti pada Persamaan (2-1).

$$\frac{\partial p}{\partial t} = 0 \tag{2-1}$$

Dengan:

p = nilai proses

t = waktu (s)

Sedangkan pada model waktu diskrit, hal ini berarti selisih pertama pada setiap variabel p adalah nol dan seterusnya, seperti pada Persamaan (2-2).

$$p_t - p_{t-1} = 0 \tag{2-2}$$

Dengan:

p = nilai proses

t = waktu (s)

2.1.4. Steady State Error (Error Band)

Steady-state error didefinisikan sebagai selisih antara masukan (*set-point*) dan keluaran dari suatu sistem pada sebuah batas tertentu ketika besaran waktu menuju ketakhinggaan (yaitu ketika respon telah mencapai kondisi *steady-state*). *Steady state error* akan bergantung pada jenis masukan dan juga pada jenis sistem. *Steady-state error* disebut juga dengan *error band*.

2.1.5. Rising Time

Rising time (atau *rise time*, waktu naik) adalah waktu yang diperlukan oleh sebuah sinyal untuk mengubah nilainya dari suatu nilai rendah ke nilai tinggi yang diinginkan (*set-point*). Nilai ini dinyatakan sebagai rasio (perbandingan) atau persentase. Pada sebuah keluaran dari sistem, *rising time* yang dimilikinya secara umum bergantung pada *rising time* dari sinyal masukan dan pada karakteristik sistem itu sendiri. Pada dasarnya, sebuah sistem kendali harus bisa meminimalisasi efek *overshoot* sementara *rising time* yang dimiliki oleh objek yang dikendalikan memiliki nilai yang relatif besar.

Sebagai lawannya, *falling time* adalah sama dengan *rising time*, hanya saja berlaku dari suatu nilai tinggi ke nilai rendah yang diinginkan.

2.1.6. Settling Time

Settling time dari sebuah perangkat keluaran adalah waktu yang dibutuhkan mulai dari masukan acuan hingga sinyal yang ada memasuki tahap *steady-state* (atau sudah berada dalam kondisi *error steady state* yang ditoleransi). *Settling time* bergantung pada kemampuan respon sistem dan *time constant*. *Settling time* pada sebuah sistem yang dikedalikan harus dijaga sekecil mungkin oleh sistem kendali.

2.2. Fuzzy Logic

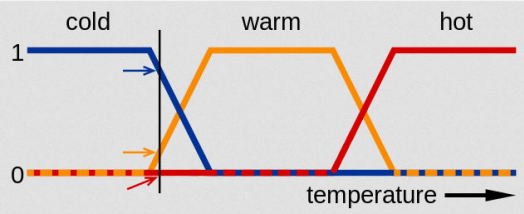
Fuzzy logic adalah sebuah bentuk dari logika bernilai banyak yang mana nilai kebenaran dari sebuah variabel jatuh antara 0 dan 1 sebagai bilangan nyata. Sebagai perbandingan, pada *boolean logic*, nilai kebenaran dari sebuah variabel jatuh hanya pada nilai 0 atau 1, sebagai bilangan bulat. Jadi sementara *boolean logic* hanya memiliki dua kemungkinan nilai, maka *fuzzy logic* memiliki semua nilai bilangan nyata yang mungkin antara 0 dan 1.

Fuzzy logic telah banyak digunakan untuk menangani konsep tentang kebenaran sebagian (*partial truth*). Selain dari itu, ketika variabel linguistik digunakan, derajat kebenaran yang muncul sebagai nilai variabel *fuzzy* bisa diatur oleh fungsi keanggotaan tertentu. (Bojadziev, *et. al.*, 2005)

Proses pada *fuzzy logic* terdiri dari bagian-bagian berikut:

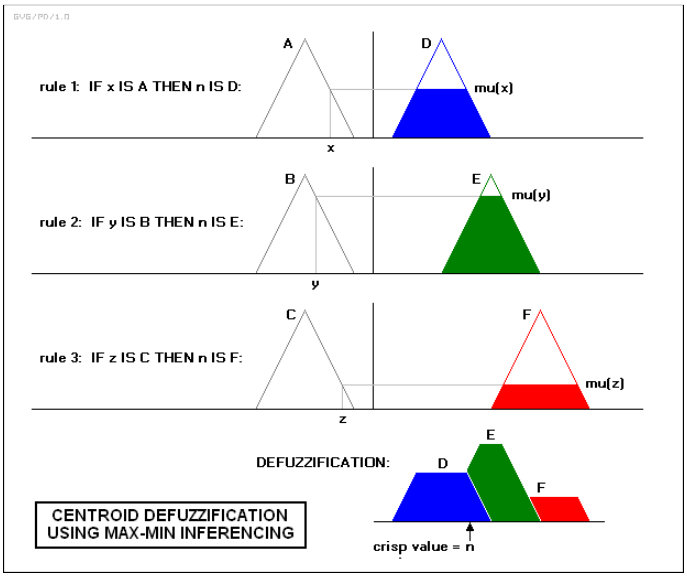
1. Fuzzifikasi semua nilai masukan pada fungsi keanggotaan.
2. Menjalankan semua aturan yang didefinisikan pada *rulebase* untuk menghitung fungsi keluaran *fuzzy*.
3. Defuzzifikasi fungsi keluaran *fuzzy* untuk mendapatkan nilai tegas.

Derajat keanggotaan harus didefinisikan terlebih dahulu pada sebuah fungsi keanggotaan yang dipetakan terhadap nilai masukan.



Gambar 2.2 Contoh fungsi keanggotaan pada temperatur.
Sumber: Bojadziev, *et. al.,* 2005.

Pada Gambar 2.2 ditunjukkan sebuah contoh fungsi keanggotaan pada temperatur, yang menjelaskan bahwa fungsi cold memiliki derajat 1 sampai pada temperatur tertentu, kemudian turun hingga 0 secara perlahan (tidak langsung) pada jangkauan tertentu sementara fungsi warm memiliki derajat dari 0 yang menaik hingga 1 pada jangkauan tersebut. Hal ini menunjukkan sifat derajat keanggotaan dari sebuah fungsi pada nilai masukan tertentu bisa berbagi dengan fungsi yang lainnya.



Gambar 2.3 Contoh max-min inferencing dan centroid defuzzification.
Sumber: Bojadziev, *et. al.*, 2005.

Gambar 2.3 menunjukkan contoh *max-min inferencing* yang menerjemahkan nilai variabel x, y, dan z menuju ke fungsi mu(x), mu(y), dan mu(z) sebagai derajat keanggotaan, kemudian melakukan *centroid defuzzification* untuk mendapatkan nilai tegas (*crisp value*).

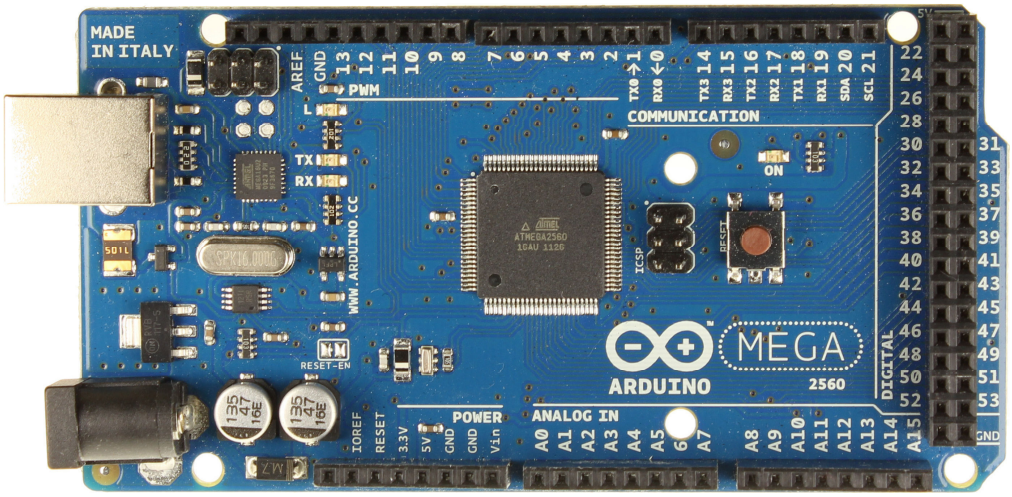
Ketika variabel dalam matematika biasanya mengambil nilai-nilai numerik, dalam aplikasi *fuzzy logic* sering digunakan nilai-nilai non-numerik untuk memfasilitasi ekspresi aturan dan fakta. Sebuah variabel linguistik seperti usia bisa jadi memiliki nilai seperti muda atau tua secara bersamaan. Nilai variabel linguistik dapat dimodifikasi melalui aturan linguistik diterapkan. Aturan linguistik dapat dikaitkan dengan fungsi-fungsi tertentu.

Operasi *fuzzification* dapat memetakan nilai masukan matematika dalam fungsi keanggotaan *fuzzy*. Dan sebaliknya proses *defuzzifying* dapat digunakan untuk memetakan fungsi keanggotaan keluaran yang bersifat kontinyu menjadi bersifat tegas yang dapat kemudian digunakan untuk tujuan keputusan atau pengendalian.

Sistem kendali yang berbasis pada *fuzzy logic* disebut dengan *fuzzy control system* (FCS), di mana kaidah-kaidah proses yang digunakan dalam melakukan pengendalian berdasarkan pada *fuzzy logic*. (Bojadziev, *et. al.*, 2005)

2.2.1. Arduino

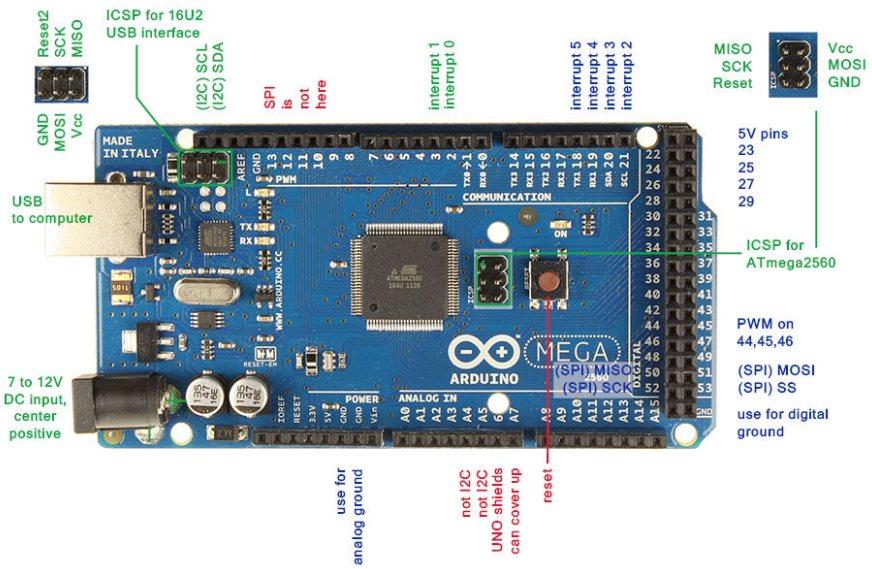
Arduino adalah perusahaan, proyek, dan komunitas pengguna perangkat keras komputer dan perangkat lunak yang mendesain dan memproduksi kit mikrokontroler untuk membangun perangkat digital dan objek interaktif yang memiliki kemampuan sensor dan kemampuan kendali fisik. Produk proyek Arduino didistribusikan sebagai perangkat keras dan perangkat lunak *open-source*, dengan lisensi menggunakan GNU *Lesser General Public License* (LGPL) atau GNU *General Public License* (GPL). Hal ini memungkinkan pembuatan papan rangkaian Arduino secara bebas dan pendistribusian perangkat lunak oleh siapa saja. Papan rangkaian Arduino tersedia secara komersial dalam bentuk terakit, atau sebagai kit yang bisa dibuat sendiri.



Gambar 2.4 Contoh sebuah perangkat keras Arduino Mega 2560.
Sumber: Banzi, 2011.

Papan rangkaian Arduino sendiri merupakan perangkat keras pengendali yang menggunakan *microcontroller*. Pada Arduino, rancangan papan rangkaian yang ada menggunakan berbagai *microprocessor* dan pengendali. Pada papan rangkaian ini disediakan serangkaian pin *input-output* baik yang digital maupun analog yang bisa dihubungkan menuju beragam papan rangkaian ekspansi, rangkaian lainnya, komputer, atau langsung pada perangkat sensor dan motor. Papan rangkaian ini menyajikan beberapa

antarmuka komunikasi serial, termasuk USB *Port* untuk berhubungan dengan komputer dalam pemrograman.



Gambar 2.5 Keterangan pin Arduino Mega 2560.
Sumber: Banzi, 2011.

Microcontroller pada Arduino dapat diprogram menggunakan C atau C++, dengan kemudahan yang disediakan oleh komunitas berupa IDE (*integrated development environment*) untuk melakukan pemrograman. Sekali program dituliskan ke papan rangkaian menggunakan USB *communication*, maka setiap kali papan rangkaian diberi daya program akan berjalan sampai daya diputus. (Banzi, 2011)

Tabel 2.1
Spesifikasi singkat Arduino Mega 2560.

Microcontroller	Atmega2560
Operating Voltage	5 V
Input Voltage (recommended)	7 V – 12 V
Input Voltage (limit)	6 V – 20 V
Digital I/O Pins	54 (14 with PWM output)
Analog Input Pins	16
DC Current per I/O Pin	40 mA
DC Current for 3.3 V Pin	50 mA
Flash Memory	256 KB of which 8 KB used by bootloader
SRAM	8 KB
EEPROM	4 KB
Clock Speed	16 MHz

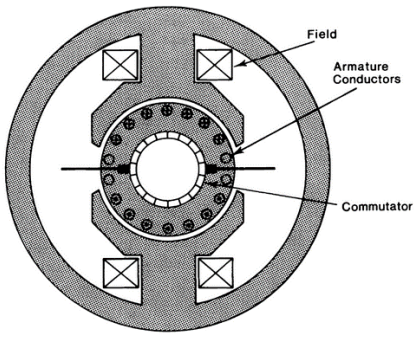
Sumber: http://www.robotshop.com/, 2016.

Arduino ada berbagai macam, bervariasi menurut tegangan pin *input-output*, *bit depth*, jumlah pin, dan *clock speed*. Pada penelitian ini digunakan Arduino Mega 2560, seperti

ditunjukkan pada Gambar 2.4 dan Gambar 2.5. Spesifikasi singkat dari Arduino Mega 2560 ditunjukkan pada Tabel 2.1.

2.2.2. Motor DC

Motor DC adalah peranti elektronik yang fungsinya mengubah arus listrik DC (searah) menjadi perwujudan putaran mekanik. Pada dasarnya, motor DC umumnya berbekal medan magnet untuk penggerak. Hampir semua jenis motor DC memiliki mekanisme di dalam, yang berupa elektromekanikal ataupun elektronik, yang secara ajeg mengubah arah arus pada interior motor DC. Kebanyakan jenis motor DC menghasilkan gerakan putar, jenis yang lain merupakan motor DC linier yang menghasilkan gaya dan gerakan dalam garis lurus (biasanya disebut dengan aktuator).



Gambar 2.6 Skematik 3D dari sebuah motor DC.
Sumber: Hughes, 1990.

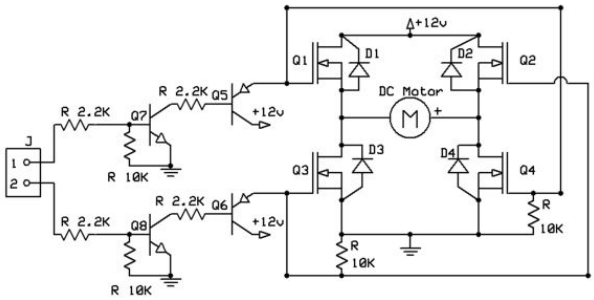
Putaran dari motor DC dapat dikontrol pada jangkauan yang cukup lebar, menggunakan variasi besarnya tegangan listrik atau variasi kekuatan arus listrik pada lilitan. Umumnya pengontrolan motor DC menggunakan variasi masukan pada tegangan listrik. (Hughes, 1990)

Gambar 2.7 Motor DC RS 555-555.
Sumber: Hughes, 1990.

Dalam penelitian ini digunakan RS 555-555 sebagai motor DC yang dikendalikan, seperti ditunjukkan pada Gambar 2.7, yang memiliki tegangan nominal 20 V, arus maksimal 220 mA, dan *output power* 4.4 W. Dalam kondisi tak berbeban, RS 555-555 mampu berputar hingga 3500 RPM.

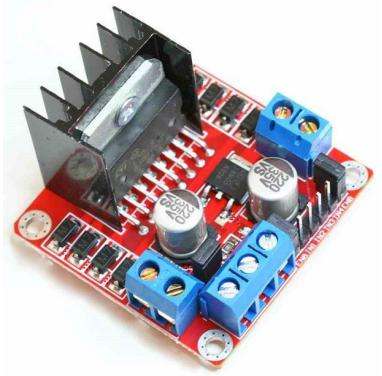
2.2.3. Driver Motor DC

Driver motor DC merupakan sebuah alat yang mengendalikan torsi motor DC. Sebuah *driver* motor DC merupakan penguat arus kecil (yang menjadi arus kendali) menjadi arus yang lebih besar untuk mencatu daya pada motor DC.



Gambar 2.8 Contoh sebuah skema rangkaian dari *driver* motor DC.
Sumber: Keljik, 2013.

Driver motor DC berupa rangkaian DIY (dikerjakan sendiri), dan disesuaikan dengan *control input*, tegangan, dan arus yang diperlukan oleh motor DC yang dikendalikan. *Driver* motor DC melakukan pengendalian dengan cara memvariasikan tegangan ke motor DC sesuai dengan kebutuhan torsi yang diinginkan.



Gambar 2.9 L298N DC Motor Driver.
Sumber: https://www.instructables.com/, 2017.

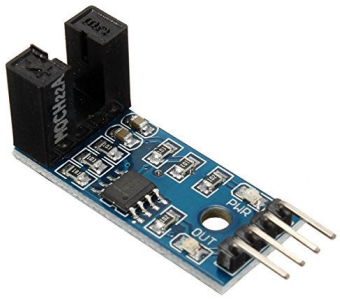
Pada penelitian ini, digunakan L298N sebagai *driver*, seperti yang ditunjukkan pada Gambar 2.9.

2.2.4. Speed Sensor

Speed sensor (sensor kecepatan) adalah sebuah alat yang mendeteksi kecepatan objek. Kecepatan yang dideteksi bisa berupa kecepatan putar, kecepatan linier, ataupun yang lainnya.

Gambar 2.10 Optical chopper.
Sumber: https://www.thorlabs.com/, 2017.

Dalam penelitian ini, digunakan detektor kecepatan putar untuk mendeteksi kecepatan putar motor DC yang dikendalikan. *Speed sensor* yang digunakan adalah *optical chopper* yang diapit oleh *opto switch*, yaitu sensor yang mengukur sinyal inframerah yang dipancarkan dari ujung satunya, tergantung pada sinyal yang diterima apakah terhalang atau tidak oleh *optical chopper*. Satuan untuk kecepatan putar yang digunakan adalah RPM (*revolutions per minute*), yaitu jumlah putaran yang terjadi setiap menitnya.

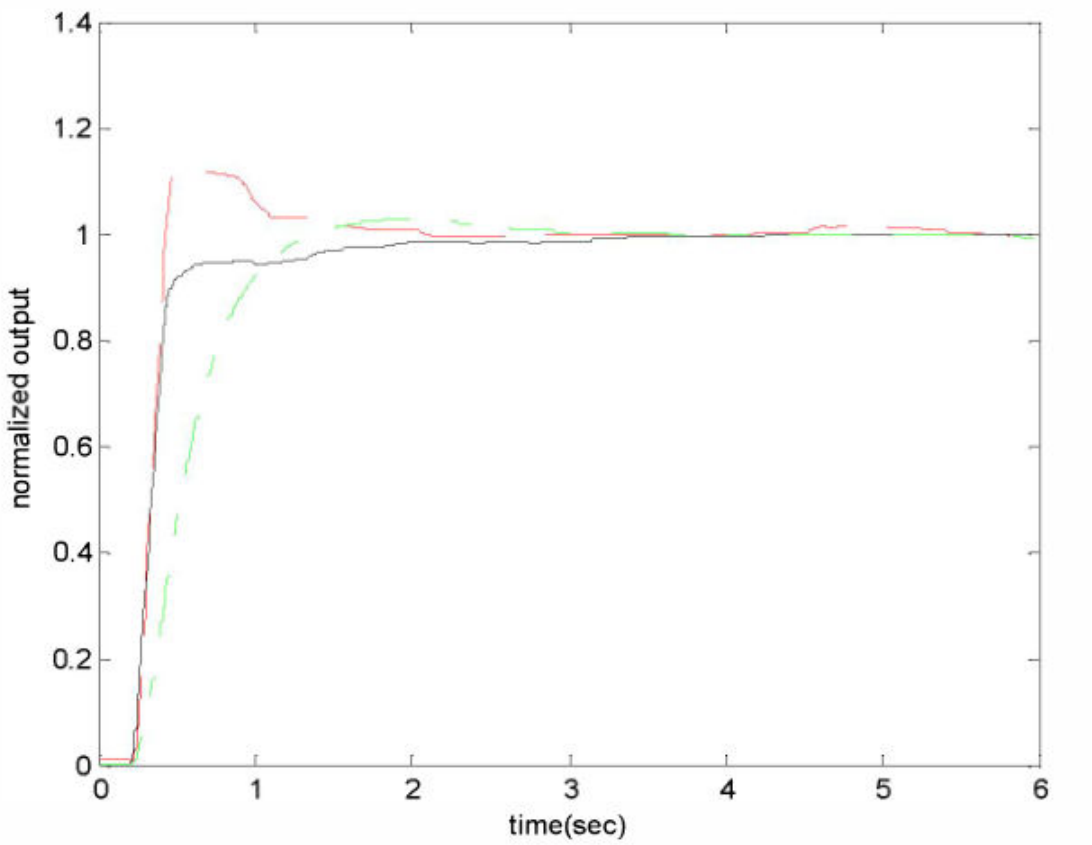


Gambar 2.11 Opto Switch LM393.
Sumber: https://www.banggood.com/, 2016.

Optical chopper yang digunakan adalah yang memiliki 20 lubang, sedangkan *opto switch* yang digunakan adalah LM393. Dengan 20 lubang yang ada, maka kecepatan transisi antara lubang dengan non-lubang harus dibagi dengan 20 untuk mendapatkan kecepatan putar dalam CPS.

2.3. Penelitian Sebelumnya

Beberapa penelitian terdahulu menunjukkan hasil kualitatif, tetapi belum menunjukkan kuantitatif. Seperti pada penelitian oleh Azadi (2012), yang menyatakan Pengendali Azadi mampu menstabilkan putaran motor DC dengan meminimalkan *overshoot* dan *undershoot*, tetapi belum menunjukkan besaran *overshoot* yang dihasilkan oleh sistem (Azadi, *et. al*., 2012).



Gambar 2.12 Hasil simulasi *Azadi Controller*.
Sumber: Azadi, *et. al.*, 2012.

Dari hasil simulasi pada Gambar 2.12, ditunjukkan perilaku sistem *Azadi Controller* tetapi tidak menunjukkan besaran nominal *rising time*, *overshoot*, dan *oscillation*.

Penelitian tentang *neuro-fuzzy controller* menggunakan *particle swarm optimization* juga tidak memberikan hasil nominal tetang besaran *overshoot* dan *undershoot* serta ayunan, dan hanya memberikan hasil kualitatif serta digunakan sebagai sistem yang fleksibel (Farid, *et. al*., 2014).

BAB III

KERANGKA KONSEP PENELITIAN

Kerangka konsep penelitian merupakan hubungan antar konsep pada permasalahan yang akan diteliti. Dengan kata lain, kerangka konsep ini akan berguna untuk menghubungkan topik yang dibahas secara rinci.

3.1. Deskripsi Masalah

Penelitian ini didasari pada kebutuhan akan kecepatan putar motor DC yang bisa diprogram sesuai dengan keperluan. Hal ini banyak dipakai pada aplikasi yang menggunakan motor DC sebagai bagian dari mesin yang memiliki kebutuhan kecepatan putar bervariasi dalam ranah waktu dan terprogram, contohnya adalah konveyor pabrik yang membutuhkan urutan kecepatan putar motor sesuai dengan program konveyor dan bisa dibebani dengan volume barang industri yang bervariasi. Pada kasus aplikasi seperti di atas, tidak ada perubahan kecepatan putar yang dilakukan manual atau tiba-tiba oleh operator. Semua urutan (*sequence*) kebutuhan perubahan kecepatan putar motor DC tersebut diprogram sebelum keseluruhan proses dimulai. Hal ini mengakibatkan proses perubahan kecepatan putar menjadi presisi sesuai kebutuhan tanpa adanya campur tangan operator sampai akhir dari proses tersebut.

Salah satu keuntungan yang didapatkan pada sistem seperti itu adalah terbukanya kemungkinan untuk meramalkan titik dalam waktu kapan motor DC tersebut harus mencapai kecepatan putar tertentu sesuai dengan program yang diberikan oleh *operator*, sehingga bisa mengoptimalkan waktu target perubahan kecepatan putar dengan cara melakukan perubahan kendali sebelum titik waktu target.

Perangkat keras pengendali yang digunakan dalam penelitian ini adalah pengendali Arduino (Arduino Mega 2560), karena kemudahannya untuk diterapkan pada aplikasi pengendalian skala laboratorium. Hasil penelitian ini bisa disebut sebagai *prototype*. Sedangkan algoritma pengendalian yang digunakan adalah *Fuzzy Control System* dengan *Mamdani Inference Engine*.

Masukan yang digunakan dalam penelitian ini adalah urutan kecepatan putar terprogram, yang juga diberi penanda waktu *start* menggunakan *button*. Sebagai tambahan, bila terjadi kondisi darurat bisa dilakukan pemberhentian dengan menggunakan *emergency button*. Tombol-tombol navigasi dan *keypad* juga dipakai sebagai fasilitas pendukung untuk melakukan pemrograman dan penalaan. Motor DC yang digunakan akan diberi beban bervariasi dan sistem akan diuji unjuk kerjanya dengan cara mengamati kestabilan kecepatan putarannya.

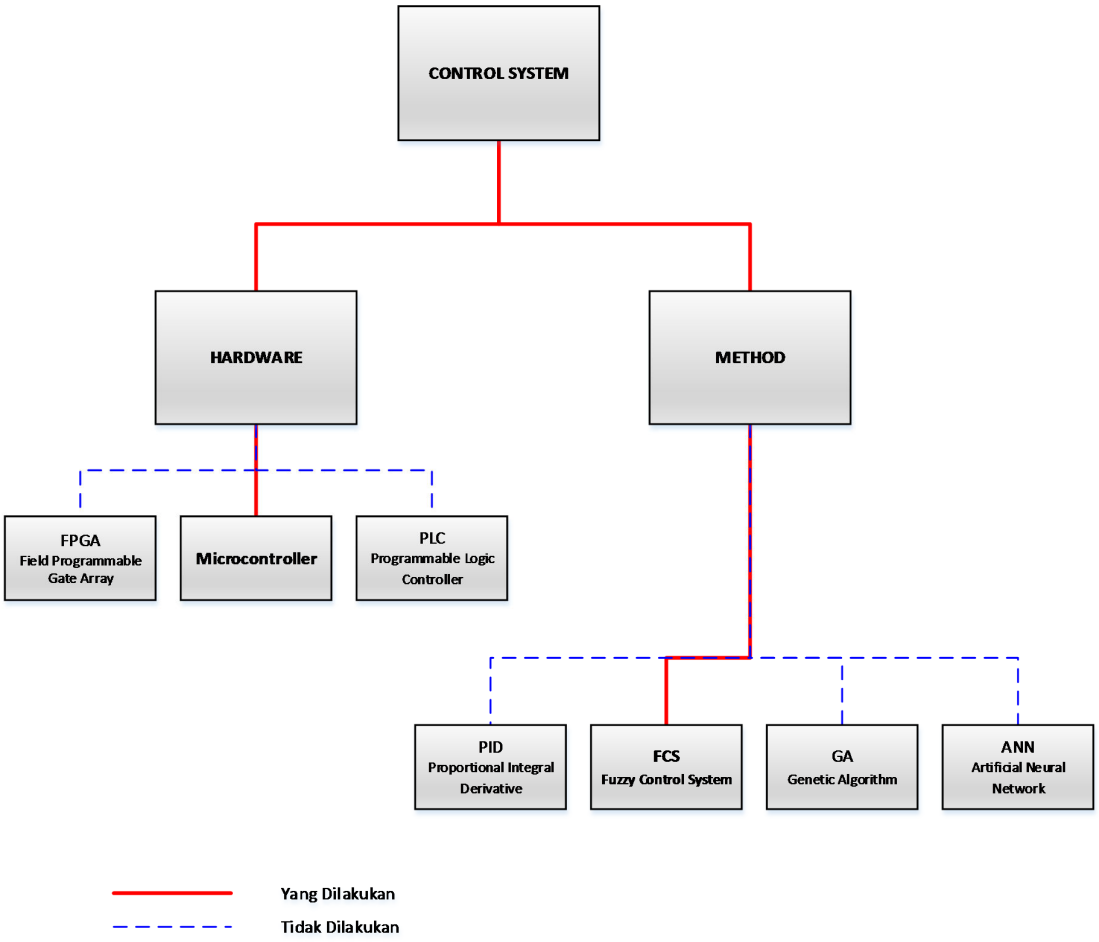
3.2. Definisi Operasional Variabel

Beberapa variabel yang akan diuji pada penelitian ini adalah sebagai berikut:

1. Ketepatan respon sistem terhadap permintaan kecepatan putar tertentu (*set-point*) sesuai dengan spesifikasi.
2. Kemampuan sistem untuk mempertahankan kecepatan putar terhadap variasi beban (*mechanical load*) yang diberikan secara acak.

3.3. Kerangka Teori Penelitian

Kerangka teori penelitian disusun sedemikian rupa sehingga dalam proses penyelesaian permasalahan dapat dilakukan secara sistematis dan terstruktur. Adapun kerangka tersebut ditunjukkan pada Gambar 3.1.



Gambar 3.1 Kerangka Teori Penelitian.
Sumber: Rencana Penelitian.

Berdasarkan Gambar 3.1. dapat dijelaskan bahwa dalam penelitian ini, perangkat keras yang digunakan adalah *microcontroller* (Arduino Mega 2560), sedangkan metode atau algoritma yang diterapkan dalam pengendalian motor DC tersebut adalah FCS.

3.4. Alat Penelitian

Dalam hal ini dibutuhkan beberapa alat-alat utama dan pendukung dalam melakukan proses pengembangan dan pengujian. Beberapa alat-alat utama tersebut, antara lain:

1. *Microcontroller*: Arduino Mega 2560.
2. Rangkaian *Driver* Motor DC L298N.
3. Motor DC RS 555-555 dengan spesifikasi: tegangan 20 VDC, arus maksimal 220 mA, dan *output power* 4.4 W.

4. Rangkaian *Speed Sensor*.
5. *Input Keypad*, *PushButtons*, LED *Indicators*, dan LCD 16x4 *char*.
6. Arduino IDE 1.8.1.
7. Borland Delphi 7.

Kemudian, beberapa alat-alat pendukung yang ada pada penelitian ini antara lain:

1. *Microsoft Windows 7*, sebagai *operating system*. Versi minimal yang bisa gunakan adalah *Microsoft Windows XP SP2*.
2. *Standard* PC dengan minimal bisa menjalankan *Microsoft Windows XP SP2*.

3.5. Hipotesis

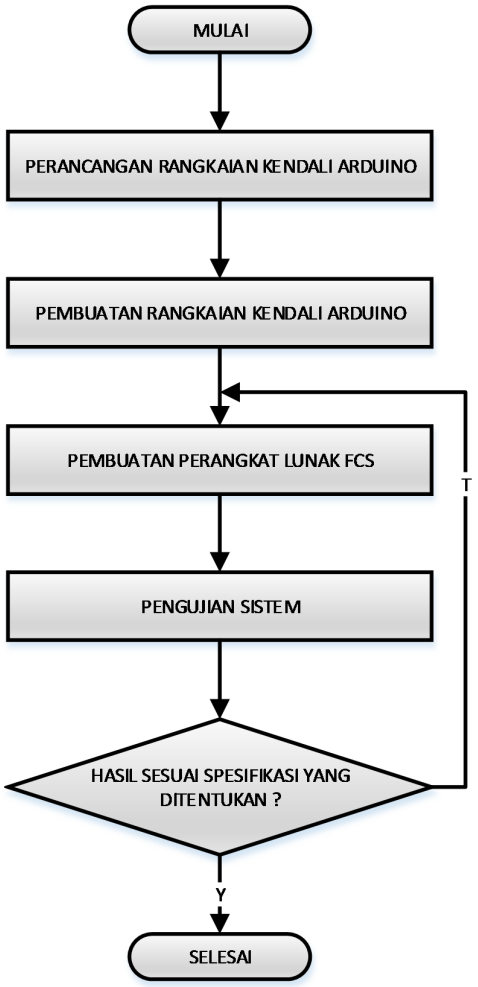
Berdasarkan hasil penelitian sebelumnya, diharapkan metode *Fuzzy Control System* dapat memberikan hasil pengendalian yang halus tetapi tetap responsif.

BAB IV

METODE PENELITIAN

Berikut ini dijelaskan cara memperoleh data dan variabel yang berguna dalam analisis, serta cara apa saja yang digunakan dalam penyelesaian masalah.

4.1. Metode Penelitian



Gambar 4.1 Diagram metode penelitian.
Sumber: Rencana Penelitian

Diagram pada Gambar 4.1 menjelaskan beberapa tahap yang dilakukan dalam penelitian. Pertama dilakukan perancangan perangkat keras yang diperlukan, yang diikuti

dengan pembuatan perangkat keras sesuai dengan spesifikasi. Setelah itu dilakukan pembuatan perangkat lunak kendali, yang dalam hal ini menggunakan *fuzzy control system*.

Langkah berikutnya yaitu pengujian sistem untuk mengetahui performa sistem sesuai spesifikasi. Setelah dilakukan pengujian, diketahui apakah performa sudah optimal. Bila belum optimal, maka akan dilakukan kembali pengujian sistem. Tetapi bila sudah optimal, maka proses penelitian ini dianggap selesai.

4.2. Spesifikasi Kendali

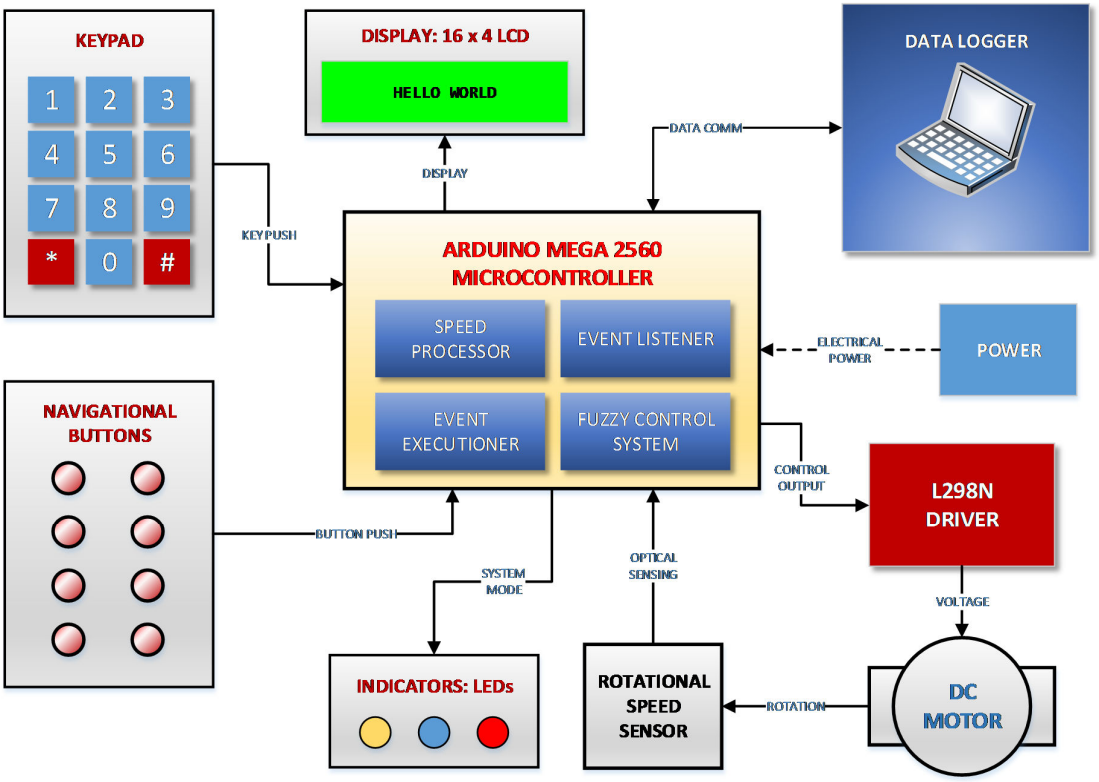
Beberapa spesifikasi yang diharapkan pada sistem kendali ini merupakan:

1. Menekan *overshoot*/*undershoot* hingga di bawah 20% dari pergantian *set-point*.
2. Menekan *overshoot*/*undershoot* hingga di bawah 30% dari perubahan kecepatan pada *set-point* yang sama sebagai hasil dari penambahan maupun pelepasan beban.
3. Menekan *rising time* hingga tidak melebihi dari 5 kali waktu aksi koreksi.
4. *Error band* ditentukan sebesar 5% dari kecepatan putar maksimal yang ditentukan.

Semua parameter spesifikasi kendali tersebut ditentukan berdasarkan nilai yang dipilih sebagai standar untuk melakukan pembandingan terhadap hasil aktual yang didapatkan sesuai dengan penalaan parameter *Fuzzy Logic* yang berlaku pada waktu proses pengujian.

4.3. Diagram Sistem

Berikut pada Gambar 4.2 dan Gambar 4.3 ditunjukkan struktur dari sistem yang akan dibangun.



Gambar 4.2 Diagram fisik sistem pengendali kecepatan putar motor DC dan pendukungnya.

Sumber: Rencana Penelitian.

Sesuai dengan diagram sistem fisik yang ada pada Gambar 4.2, maka dapat ditunjukkan bahwa terdapat empat bagian pada sistem fisik yang dibangun pada mikrokontroler Arduino Mega 2560, antara lain:

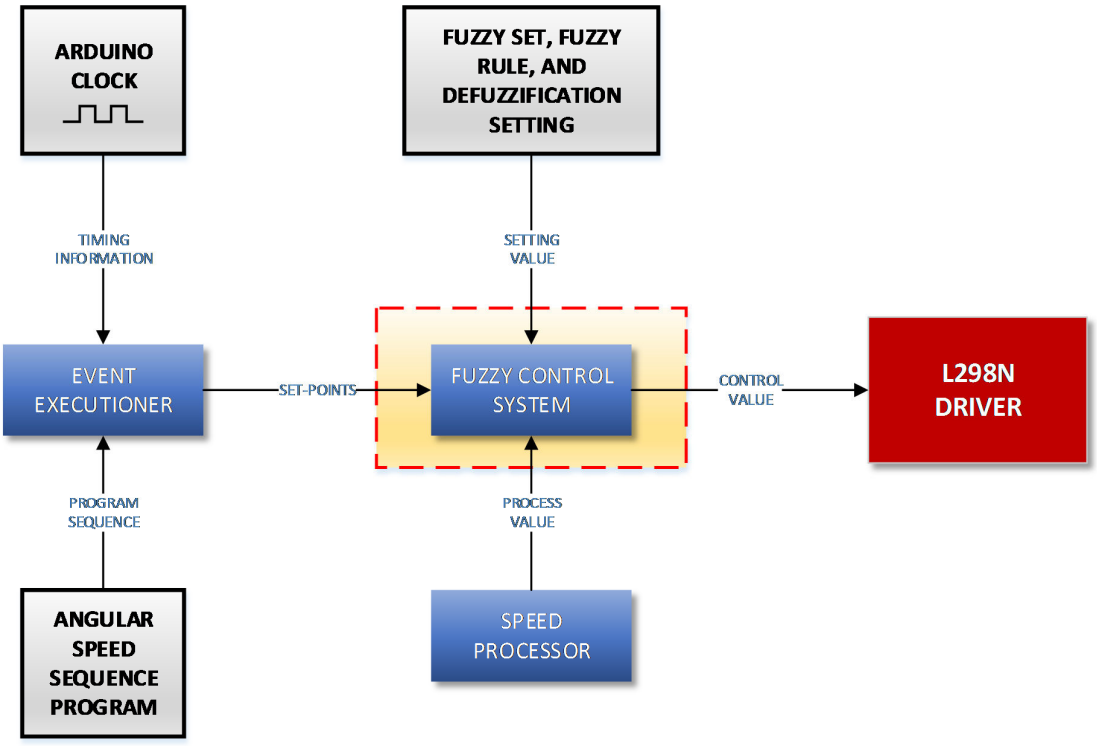
1. *Speed Processor*, yaitu bagian yang bertugas untuk menghitung kecepatan putar motor DC. Bagian ini mendapatkan prioritas pemrosesan yang paling utama, paling cepat, dan paling tinggi, mengingat pengambilan kecepatan putar menggunakan *opto chopper* berlubang 20, yang mana dalam satu putaran harus melakukan siklus pembacaan sensor 40 kali.
2. *Event Listener*, yaitu bagian yang bertugas membaca *event* penekanan tombol secara fisik maupun secara program melalui instruksi *serial communication*. Bagian ini mendapatkan prioritas pemrosesan kedua setelah *speed processor*, mengingat bagian ini harus dengan cepat mendapatkan pembacaan dari penekanan tombol.
3. *Event Executioner*, yaitu bagian yang bertugas melaksanakan perintah dari *event* penekanan tombol atau *event* lainnya, termasuk menampilkan data pada LCD,

menyalakan LED, membunyikan *buzzer*, dan memanggil FCS. Bagian ini mendapatkan prioritas pemrosesan ketiga.

4. FCS (*Fuzzy Control System*), yaitu bagian utama yang mengendalikan kecepatan putar motor DC berdasarkan *set-point* dengan menggunakan *fuzzy logic*. Bagian ini mendapatkan prioritas terakhir karena sifat dari kalkulasi fuzzy yang ringan.

Bagian FCS merupakan bagian utama, sedangkan bagian *speed processor*, *event listener*, dan *event executioner* merupakan bagian pendukung.

Masukan fisik yang diberikan ke dalam sistem secara manual oleh pengguna adalah 8 tombol navigasi dan 1 set *keypad* angka. Sedangkan masukan fisik yang diberikan secara otomatis adalah posisi sensor optik pada bagian sensor kecepatan putar motor DC. Keluaran fisik yang dihasilkan oleh sistem antara lain tampilan ke LCD, modus sistem ke LED, dan keluaran kendali (*control output*) ke *driver* motor DC. Satu keluaran fisik yang dihasilkan oleh sistem dikirim melalui *serial communication* ke perangkat lunak *Data Logger*.



Gambar 4.3 Diagram logik sistem pengendali kecepatan putar motor DC menggunakan *fuzzy control system*.

Sumber: Rencana Penelitian.

Gambar 4.3 menunjukkan diagram logik sistem pengendali kecepatan putar motor DC yang direncanakan menggunakan FCS. Dalam hal ini terdapat data penalaan FCS yang telah

diisikan oleh pengguna sebelumnya dan disimpan pada EEPROM (*electrically erasable programmable read-only memory*) Arduino. Selain itu juga terdapat data *set-point* pada waktu tertentu yang disebut dengan *angular speed sequence program* yang diisikan oleh pengguna dan disimpan pada EEPROM Arduino.

Seperti yang telah ditunjukkan pada Gambar 4.3, masukan yang dipakai pada bagian pengendali adalah:

1. *Set-points*, merupakan permintaan kecepatan putar motor DC sesuai dengan penjadwalan *angular speed sequence program*.
2. *Process Value*, merupakan *feedback* kecepatan putar motor DC hasil dari pembacaan sensor.
3. *Setting Values*, merupakan parameter FCS yang meliputi fuzzifikasi (fungsi keanggotaan), *fuzzy rules*, dan defuzzifikasi, sesuai dengan penalaan yang dilakukan oleh pengguna.

4.3.1. Angular Speed Sequence Program

Bagian ini adalah tempat penyimpanan berupa program urutan (*sequence*) yang bisa diakses oleh pemakai yang memungkinkan untuk memprogram urutan kecepatan motor DC yang dibutuhkan pada ranah waktu. Penulisan program urutan ini menggunakan variabel sebagai berikut:

1. TIME:ssss, merupakan penunjuk koordinat waktu mulainya *set-point* yang diinginkan dalam satuan detik (s). Nilai ssss merupakan nilai bulat yang berkisar dari 0 hingga 9999.
2. SPEED:vvvv, merupakan perintah *set-point* pada waktu TIME, menggunakan satuan RPM. Nilai vvvv merupakan nilai bulat yang berkisar dari 500 hingga 2500, atau bila di-*reset* akan mengembalikan nilai ke 0.

Program urutan ini harus diisi dulu oleh pemakai sebelum bisa digunakan. Program urutan ini akan tersimpan dalam EEPROM dari Arduino sehingga bila catu daya terputus program tetap bisa dibaca kembali.

4.3.2. Fuzzy Set, Fuzzy Rules, and Deffuzification Setting

Bagian ini adalah tempat penyimpanan berupa penalaan (*setting*) yang bisa diakses oleh pemakai yang memungkinkan untuk memprogram aturan dari semua parameter FCS. Penulisan penalaan ini menggunakan variabel sebagai berikut:

1. ERROR:n:eeee, merupakan penunjuk titik tengah derajat keanggotaan ke-n pada eeee RPM, yang menunjukkan *error* atau selisih antara *set-point* dengan *process value*. Nilai n adalah nilai bulat yang berkisar dari 0 hingga 6. Nilai eeee adalah nilai bulat berkisar dari -1000 hingga 1000.
2. DELTAERROR:n:eeee, merupakan penunjuk titik tengah derajat keanggotaan ke-n pada eeee RPM, yang menunjukkan *delta error* atau selisih antara *error* yang sekarang dengan *error* yang sebelumnya. Nilai n adalah nilai bulat yang berkisar dari 0 hingga 6. Nilai eeee adalah nilai bulat berkisar dari -1000 hingga 1000.
3. DEF:n:eee, merupakan penunjuk nilai tegas defuzzifikasi ke-n dengan menggunakan nilai koreksi eee. Nilai n adalah nilai bulat yang berkisar dari 0 hingga 6. Nilai eee adalah nilai bulat berkisar dari -127 hingga 127.
4. RULE:m:r, merupakan penunjuk nilai *fuzzy rule* pada tempat ke-m dan menggunakan nilai r. Nilai m berupa nilai bulat berkisar dari 0 hingga 48. Sedangkan nilai r merupakan nilai enumerasi dengan sebutan ANL, ANM, ANS, AZE, APS, APM, dan APL.

Nilai-nilai pada variabel tersebut akan dijelaskan pada bab berikutnya. Penalaan ini memiliki nilai awal sebelum diganti oleh pemakai, sehingga meskipun belum didefinisikan oleh pemakai tetap bisa digunakan. Penalaan ini akan tersimpan dalam EEPROM dari Arduino sehingga bila catu daya terputus penalaan tetap bisa dibaca kembali.

4.3.3. Keypad dan Navigasi

Bagian ini digunakan untuk memberikan masukan pemakai pada alat kendali dan pendukungnya. *Keypad* digunakan untuk mengisikan nilai yang dibutuhkan pada variabel, sedangkan tombol navigasi digunakan untuk melakukan navigasi menu pada perangkat, termasuk untuk menjalankan program urutan, memberhentikan program urutan, melakukan pemrograman urutan, dan melakukan penalaan.

4.3.4. Indikator LED dan Display LCD

Bagian ini digunakan untuk menampilkan keluaran dalam bentuk visual, antara lain penyalaan LED untuk menyatakan modus perangkat dan display LCD untuk menampilkan berbagai macam pesan dan antarmuka, termasuk menampilkan menu perangkat.

4.3.5. DC Motor Driver L298N

Bagian ini berupa rangkaian penguat arus dari keluaran Arduino Mega 2560 menuju ke arus yang lebih kuat dan tegangan yang disesuaikan dengan spesifikasi motor DC untuk memberi tenaga ke motor DC.

4.3.6. DC Motor

Bagian ini merupakan motor DC yang akan dikendalikan. Motor DC ini disimulasikan bisa diberi *set-point* dan beban yang bervariasi. Motor DC ini memiliki masukan arus dari rangkaian *DC Motor Driver*.

4.3.7. Rotational Speed Sensor

Bagian ini merupakan sensor untuk mendeteksi kecepatan putar motor DC. Sensor ini berupa *optical chopper* yang diapit oleh *opto switch*, dan hasil pembacaannya berupa sinyal terbuka atau tertutup, kemudian diolah oleh *speed processor*. Hasil dari pengukuran ini diberikan sebagai umpan balik pada sistem untuk mendapatkan pembacaan kecepatan putar motor DC yang akan dibandingkan dengan *set-point* pada *angular speed sequence program*.

4.3.8. Data Logger

Karena *internal memory* yang dimiliki oleh mikrokontroler Arduino Mega 2560 terbatas (8 KB), maka hasil dari eksekusi *angular speed sequence program* tidak bisa disimpan dalam perangkat. Oleh karena itu dibutuhkan perangkat luar yang menerima dan menyimpan hasil dari eksekusi. Dalam hal ini, digunakan PC yang dihubungkan dengan perangkat menggunakan kabel USB untuk *serial communication*. Program yang ada pada PC yang membaca *data logging* dari perangkat ini disebut dengan *Data Logger*.

BAB V

HASIL PENELITIAN DAN PEMBAHASAN

Pada bagian hasil ini akan diulas tentang konsep implementasi *fuzzy logic* pada pengembangan *software* pengendali motor DC, penghitungan kecepatan, dan menampilkan parameter *rising time*, *overshoot*/*undershoot*, dan *ringing*. Adapun pemvalidasian data akan disajikan pula dengan berdasarkan hasil eksekusi *software* tersebut. Pengujian penelitian tersebut dilakukan sampai hasil sesuai dengan spesifikasi tertentu.

5.1. Pemodelan Fuzzy Logic Pengendali Kecepatan

Langkah-langkah yang ditempuh untuk mendapatkan pemodelan FCS (*fuzzy control system*) pada kasus ini adalah:

1. Pengujian untuk mendapatkan performa kecepatan pemrosesan program dalam Arduino.
2. Menentukan kecepatan terendah (selain nol) dan kecepatan tertinggi yang dimungkinkan untuk mendapatkan layanan proses sesuai dengan performa kecepatan pemrosesan.
3. Mendapatkan fungsi pemetaan kendali (berupa nilai bilangan bulat sesuai dengan spesifikasi Arduino, lengkap dengan separasi antar nilainya) terhadap keluaran tegangan yang menuju ke motor DC (setelah melalui *driver*).
4. Menentukan spesifikasi kecepatan maksimum tak berbeban dan kecepatan maksimum berbeban ideal yang diharapkan dari motor DC.
5. Merancang *fuzzy controller*.
6. Merancang proses *defuzzification*.

5.1.1. Kecepatan Pemrosesan Arduino

Setiap perangkat keras Arduino memiliki spesifikasi kecepatan proses teoritis. Spesifikasi tersebut pasti berbeda dengan kondisi nyata karena beberapa faktor, di antaranya

adalah beban kalkulasi di setiap *process cycle* yang bervariasi. Berdasarkan *datasheet* yang ada, Arduino Mega 2560 memiliki resolusi proses maksimum sebesar 4µs. Untuk mendapatkan resolusi proses maksimum secara nyata maka dapat dilakukan dengan menggunakan kode berikut ini.

```
#include <LiquidCrystal_I2C.h>

unsigned long tickcount = 0, starttime, endtime;

LiquidCrystal_I2C MyLCD(0x3f, 16, 4, LCD_5x8DOTS);

void displaystatus()
{
  unsigned long deltatime = endtime - starttime;
  float resolution = float (deltatime) / float (tickcount);
  MyLCD.setCursor(0, 0);
  MyLCD.print("COUNT: ");
  MyLCD.print(tickcount); MyLCD.print("   ");
  MyLCD.setCursor(0, 1);
  MyLCD.print("RES: ");
  MyLCD.print(resolution, 2); MyLCD.print("   ");
  tickcount = 0;
  starttime = micros();
}

void setup()
{
  MyLCD.begin();
  MyLCD.clear();
  tickcount = 0;
  starttime = micros();
}

void loop()
{
  tickcount++;
  endtime = micros();
  if(tickcount >= 10000)
    displaystatus();
}
```

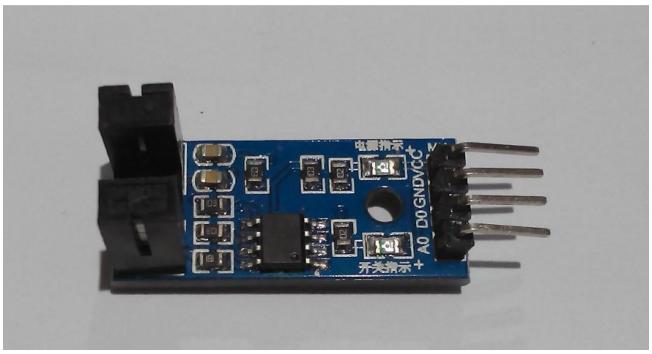
Setelah dilakukan eksekusi pada Arduino Mega 2560, ternyata didapatkan hasil pada *display* seperti pada Gambar 5.1.

Gambar 5.1 Hasil resolusi proses nyata maksimum dalam satuan µs.
Sumber: Hasil Penelitian.

Dari Gambar 5.1 ternyata didapatkan resolusi proses nyata maksimum adalah 5.72 µs. Hasil ini 43% lebih lambat daripada *datasheet* Arduino Mega 2560. Pada kondisi proses berbeban dengan menggunakan berbagai macam kalkulasi dan tidak memperhitungkan proses yang bersifat lambat dan mengunci (seperti menampilkan data pada *display*), diharapkan resolusi 100 kali lebih lambat bisa tercapai, yaitu sekitar 600 µs. Proses yang lambat dan bersifat mengunci hanya akan dieksekusi setiap 50000 µs sekali sehingga tidak terlalu mempengaruhi kerja sistem secara keseluruhan.

5.1.2. Kecepatan Putaran Terendah dan Tertinggi

Di dalam proses *sensoring* kecepatan putar motor DC, diperlukan *range* kecepatan yang mampu dihitung oleh proses Arduino.



Gambar 5.2 Perangkat *Opto Switch* LM393.
Sumber: Alat Kerja Penelitian.

Kecepatan putaran didapatkan dari perangkat *opto switch* LM 393 dan digabung dengan *optical chopper* 20 lubang dalam satu lingkaran. Sensor bekerja dengan cara menghitung waktu sela antara lubang yang meneruskan inframerah dari satu sisi diterima di sisi lain dengan halangan yang menghentikan inframerah yang sama.

Gambar 5.3 Optical Chopper 20 lubang pada poros motor DC.
Sumber: Alat Kerja Penelitian.

Sesuai dengan resolusi proses berbeban sebesar 600 μs, maka didapatkan kecepatan maksimum (dengan periode satu kali lubang dengan satu kali halangan) seperti pada Persamaan (5-1).

$$\omega = \frac{1000000}{2 \cdot r \cdot C_H} = \frac{1000000}{2 \cdot 600 \cdot 20} = 41.67Hz = 2500RPM \quad (5\text{-}1)$$

Dengan:

r = resolusi proses (μs)

C_H = banyaknya lubang pada *optical chopper*

ω = frekuensi putar (Hz, RPM)

Dengan kata lain, pada kecepatan yang lebih tinggi dari 2500 RPM tersebut, maka ketika sensor mengirimkan informasi pergantian antara lubang dengan halangan akan ada informasi yang tidak tercatat oleh proses karena melampaui resolusi proses, sehingga akan didapatkan pembacaan kecepatan yang tidak lagi akurat.

Pada saat melakukan proses lainnya yang bersifat relatif lambat dan mengunci yang dilakukan setiap 50000 μs, hasil pembacaan kecepatan sudah harus didapatkan untuk melakukan aksi sesuai data yang diterima. Hal ini bisa digunakan untuk mencari kecepatan minimum yang bisa dihitung seperti halnya pada Persamaan (5-2).

$$\omega = \frac{1000000}{2 \cdot r \cdot C_H} = \frac{1000000}{2 \cdot 50000 \cdot 20} = 0.5Hz = 30RPM \quad (5\text{-}2)$$

Dengan:

r = resolusi proses (μs)

C_H = banyaknya lubang pada *optical chopper*

ω = frekuensi putar (Hz, RPM)

Dengan kata lain, Persamaan (5-2) menunjukkan bahwa kecepatan minimum yang bisa dihitung oleh proses adalah 30 RPM. Sehingga jangkauan kecepatan putar yang bisa dibaca oleh proses adalah 30 RPM hingga 2500 RPM.

5.1.3. Fungsi Pemetaan

Fungsi pemetaan diperlukan untuk memetakan *range* pengendali terhadap tegangan keluaran yang dihasilkan *driver* L298N menuju motor DC. Setiap pin pengendali keluaran yang dimiliki oleh Arduino Mega 2560 memiliki dua tipe data:

1. Pin digital dengan kemungkinan nilai LOW dan HIGH.
2. Pin analog PWM (*pulse width modulation*) dengan jangkauan nilai dari 0 hingga 255 berupa bilangan bulat. Sehingga ada 256 kemungkinan nilai yang dipakai. Tegangan PWM yang keluar dari pin tersebut (tanpa menggunakan *driver*) adalah 0V hingga 5V.

Dalam hal ini digunakan pin analog 2 untuk melakukan pengendalian. Sedangkan untuk tegangan keluaran dari *driver* motor DC adalah 0 V hingga 20 V. Tegangan keluaran tersebut bisa memiliki polarisasi positif dan negatif. Pengendali polarisasi terdapat pada gabungan nilai pin digital 3 dan 4.

1. Bila pin 3 bernilai LOW dan pin 4 bernilai HIGH, maka polarisasi tegangan keluaran adalah positif, di mana motor DC akan berputar maju.
2. Bila pin 3 bernilai HIGH dan pin 4 bernilai LOW, maka polarisasi tegangan keluaran adalah negatif, di mana motor DC akan berputar mundur.
3. Bila pin 3 dan 4 masing-masing bernilai LOW, atau masing-masing bernilai HIGH, maka tegangan keluaran adalah 0 V, meskipun pin analog 2 bernilai tidak 0, di mana motor DC tidak berputar.

Dengan nilai pin 2 berkisar antara 0 hingga 255 yang menghasilkan tegangan keluaran 0 V hingga 20 V (V_{max} = 20 V), maka fungsi pemetaan nilai pin 2 analog terhadap tegangan keluaran ditunjukkan pada Persamaan (5-3).

$$f(V) = \left\lfloor \frac{V}{V_{max}} \cdot 255 \right\rfloor \quad (5\text{-}3)$$

Dengan:

V = tegangan keluaran (V)

V_{max} = tegangan keluaran maksimal (V)

f(V) = nilai pin 2 analog

Dalam hal ini digunakan tanda pembulatan ke bawah karena nilai f(V) harus berupa nilai bulat maksimal 255. Sedangkan fungsi pemetaan balik tegangan keluaran terhadap nilai pin 2 analog diberikan pada Persamaan (5-4).

$$f(B) = \frac{B}{255} \cdot V_{max} \quad (5\text{-}4)$$

Dengan:

B = nilai pin 2 analog

V_{max} = tegangan keluaran maksimal (V)

f(B) = tegangan keluaran (V)

5.1.4. Kecepatan Minimum dan Maksimum Motor DC (Tak Berbeban dan Berbeban)

Langkah selanjutnya adalah menentukan kecepatan putar minimum dan maksimum motor DC, baik dalam keadaan tak berbeban maupun keadaan berbeban ideal, dengan menggunakan tegangan keluaran 0 V hingga 20 V dari *driver* L298N. Sesuai dengan spesifikasi motor DC RS 555-555 yang dipakai, kecepatan maksimum motor tersebut adalah 3500 RPM. Sedangkan kecepatan maksimum motor DC yang diharapkan dengan kondisi beban ideal adalah 2500 RPM.

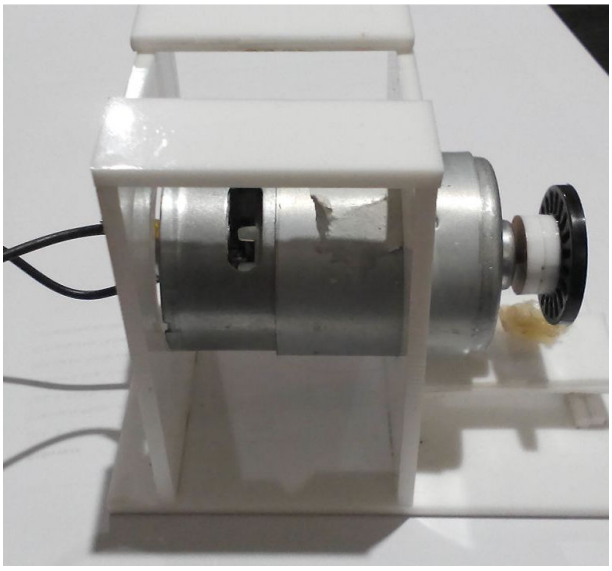
Sedangkan dari hasil percobaan pemberian tegangan *driver* pada motor DC, telah dilakukan pemberian tegangan kepada motor DC tanpa beban supaya bisa mulai berputar. Beberapa kali percobaan menambahkan tegangan tersebut dilakukan dan disajikan pada Tabel 5.1, yang mana didapatkan kecepatan minimum di mana motor DC ini mulai bergerak tanpa ada beban adalah 500 RPM.

Tabel 5.1
Percobaan pemberian tegangan minimum putar motor DC.

Percobaan ke	Nilai Kendali Pin Arduino	Nilai Tegangan Pin Arduino (V)	Nilai Tegangan Driver (V)	Kecepatan Putar (RPM)
1	12	0.24	0.56	499
2	10	0.20	0.47	488
3	12	0.24	0.56	494
4	14	0.28	0.66	500
5	12	0.24	0.56	496
6	11	0.22	0.52	490
7	13	0.25	0.61	498

Sumber: Hasil Penelitian.

Dengan diketahuinya kecepatan minimum motor DC (tanpa beban dan mulai bisa berputar) serta kecepatan maksimum motor DC (dengan beban ideal), maka bisa ditarik kesimpulan bahwa jangkauan kecepatan putar yang diharapkan adalah 500 RPM hingga 2500 RPM, dengan arah putar maju dan mundur.



Gambar 5.4 Motor DC yang dipakai dalam penelitian.
Sumber: Alat Kerja Penelitian.

Motor DC yang digunakan dalam penelitian ini disajikan dalam gambar 5.4, dan poros dari motor tersebut sudah tersambung dengan piringan 20 lubang sebagai bagian dari sensor kecepatan.

5.1.5. Merancang Fuzzy Controller

Tahap terakhir adalah merancang *fuzzy set*, *fuzzy rules*, dan *defuzzification* untuk mendapatkan keluaran berupa tegangan motor DC sesuai dengan masukan berupa

permintaan kecepatan putar (*set-point*) dan *feedback* dari sensor kecepatan putar (*process value*). Simbol yang digunakan variabel-variabel utama pada FCS ini antara lain:

1. *Set-Point*, dilambangkan dengan y_S.
2. *Process Value*, dilambangkan dengan y.
3. *Error*, dilambangkan dengan e.
4. *Delta Error*, dilambangkan dengan de.
5. *Control Action*, dilambangkan dengan u.
6. *Control Value*, dilambangkan dengan v.

Pengambilan data kecepatan dilakukan oleh *speed processor* dalam satuan RPM, sesuai dengan Persamaan (5-5).

$$\omega = \frac{H}{C_H \cdot (t_1 - t_0)} \cdot 60 \tag{5-5}$$

Dengan:

ω = kecepatan putar (RPM)
H = banyaknya lubang yang dideteksi
C_H = banyaknya lubang pada *optical chopper*
t_0 = waktu awal pengambilan cuplikan deteksi lubang (s)
t_1 = waktu akhir pengambilan cuplikan deteksi lubang (s)

Data kecepatan putar yang didapat bisa dikatakan adalah kecepatan sesaat pada waktu ini, yaitu fungsi kecepatan putar terhadap waktu t, meskipun dalam kenyataannya kecepatan putar yang didapatkan adalah kecepatan rata-rata. Persamaan (5-6) menunjukkan *process value* yang didapatkan dari kecepatan putar saat ini.

$$y(t) = \omega(t) = \frac{H(\Delta t)}{C_H \cdot \Delta t} \cdot 60 \tag{5-6}$$

$$y(t_{-1}) = \omega(t_{-1}) = \frac{H(\Delta t_{-1})}{C_H \cdot \Delta t_{-1}} \cdot 60 \tag{5-6}$$

Dengan:

y = *process value* (RPM)
ω = kecepatan putar (RPM)
H = banyaknya lubang yang dideteksi
C_H = banyaknya lubang pada *optical chopper*

Δt = selisih waktu pengambilan cuplikan deteksi lubang (s)

t = waktu cuplikan saat ini

t_{-1} = waktu cuplikan sebelumnya

Sebagai masukan, variabel *error* didefinisikan sebagai selisih antara *set-point* dengan *process value* sebagaimana ditunjukkan pada Persamaan (5-7).

$$e(t) = y_s(t) - y(t) \tag{5-7}$$

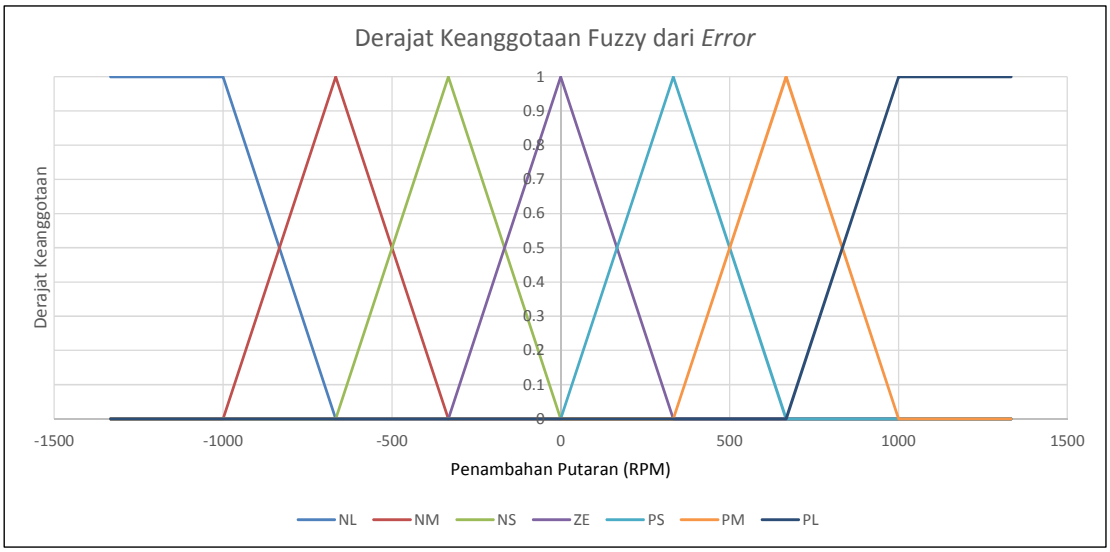
Dengan:

e = *error* (RPM)

y_S = *set-point* (RPM)

y = *process value* (RPM)

Langkah pertama dalam fuzzifikasi adalah menentukan fungsi derajat keanggotaan fuzzy berdasarkan masukan *error* dari kecepatan putar yang diinginkan, yang disajikan secara grafik dalam Gambar 5.5.



Gambar 5.5 Grafik derajat keanggotaan fuzzy berdasarkan *error* kecepatan yang diinginkan.

Sumber: Rancangan Penelitian.

Pada fungsi derajat keanggotaan tersebut terdapat tujuh macam keanggotaan, di antaranya adalah:

1. NL, yaitu *negative large*, atau sebuah nilai negatif yang besar.
2. NM, yaitu *negative medium*, atau sebuah nilai negatif yang menengah.

3. NS, yaitu *negative small*, atau sebuah nilai negatif yang kecil.

4. ZE, yaitu *zero*, atau nilai nol.

5. PS, yaitu *positive small*, atau sebuah nilai positif yang kecil.

6. PM, yaitu *positive medium*, atau sebuah nilai positif yang menengah.

7. PL, yaitu *positive large*, atau sebuah nilai positif yang besar.

Tujuh macam keanggotaan itu juga dipakai pada fungsi derajat keanggotaan fuzzy berdasarkan turunan (*derivative*) dari *error* dari kecepatan putar (*delta error*).

$$NL(e) = \begin{cases} 1; & e < -1000 \\ 1 - \frac{e+1000}{-667+1000}; & -1000 \leq e \leq -667 \\ 0; & e > -667 \end{cases} \quad (5\text{-}8)$$

$$NM(e) = \begin{cases} \frac{e+1000}{-667+1000}; & -1000 < e < -667 \\ 1 - \frac{e+667}{-333+667}; & -667 \leq e < -333 \\ 0; & -1000 \geq e \geq -333 \end{cases} \quad (5\text{-}9)$$

$$NS(e) = \begin{cases} \frac{e+667}{-333+667}; & -667 < e < -333 \\ 1 - \frac{e+333}{333}; & -333 \leq e < 0 \\ 0; & -667 \geq e \geq 0 \end{cases} \quad (5\text{-}10)$$

$$ZE(e) = \begin{cases} \frac{e+333}{333}; & -333 < e < 0 \\ 1 - \frac{e}{333}; & 0 \leq e < 333 \\ 0; & -333 \geq e \geq 333 \end{cases} \quad (5\text{-}11)$$

$$PS(e) = \begin{cases} \frac{e}{333}; & 0 < e < 333 \\ 1 - \frac{e-333}{667-333}; & 333 \leq e < 667 \\ 0; & 0 \geq e \geq 667 \end{cases} \quad (5\text{-}12)$$

$$PM(e) = \begin{cases} \frac{e-333}{667-333}; & 333 < e < 667 \\ 1 - \frac{e-667}{1000-667}; & 667 \leq e < 1000 \\ 0; & 333 \geq e \geq 1000 \end{cases} \quad (5\text{-}13)$$

$$PL(e) = \begin{cases} 1; & e > 1000 \\ \frac{e-667}{1000-667}; & 667 \leq e \leq 1000 \\ 0; & e < 667 \end{cases} \quad (5\text{-}14)$$

Dengan:

e = *error* (RPM)

NL = fungsi keanggotaan *negative large*

NM = fungsi keanggotaan *negative medium*

NS = fungsi keanggotaan *negative small*

ZE = fungsi keanggotaan *zero*

PS = fungsi keanggotaan *positive small*

PM = fungsi keanggotaan *positive medium*

PL = fungsi keanggotaan *positive large*

Persamaan (5-8) hingga Persamaan (5-14) merupakan fungsi derajat keanggotaan fuzzy berdasarkan masukan *error* dari kecepatan putar yang diinginkan. Langkah berikutnya adalah mendefinisikan fungsi keanggotaan *delta error*. Sebelumnya, Persamaan (5-15) menunjukkan definisi nilai *delta error*.

$$de(t) = e(t) - e(t_{-1}) \tag{5-15}$$

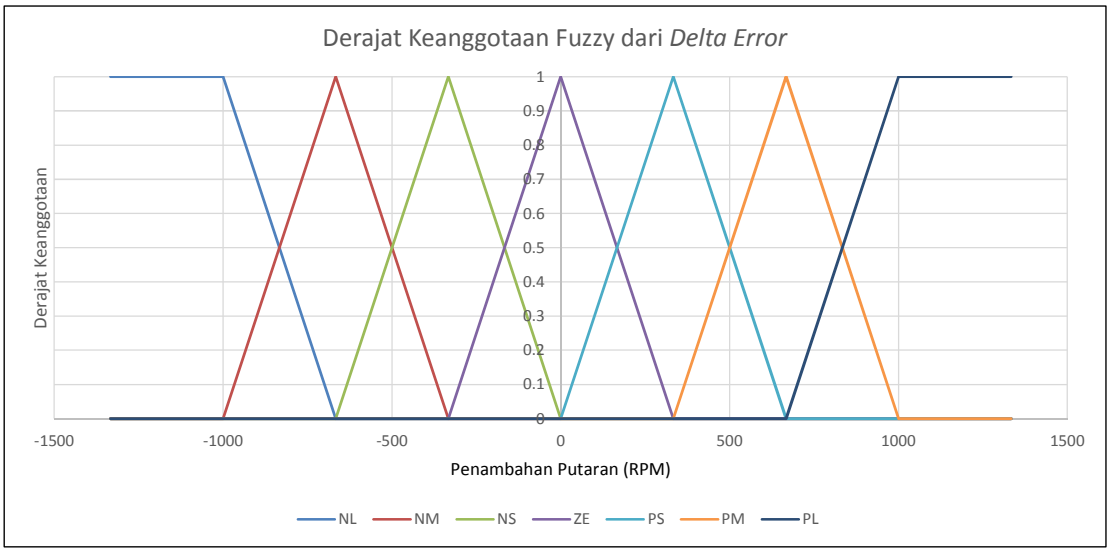
Dengan:

de = *delta error* (RPM)

e = *error* (RPM)

t = waktu cuplikan saat ini

t_{-1} = waktu cuplikan sebelumnya



Gambar 5.6 Grafik derajat keanggotaan fuzzy berdasarkan turunan dari *error* kecepatan yang diinginkan.

Sumber: Rancangan Penelitian.

Gambar 5.6 menunjukkan fungsi derajat keanggotaan *fuzzy* berdasarkan turunan dari *error* kecepatan yang diinginkan (*delta error*). Satuan maksimum untuk *error* dan *delta error* dari kecepatan yang diinginkan dibuat maksimum ± 1000 RPM, yaitu setengah dari *range* kecepatan putar motor DC maksimum (2500 RPM), agar transisi maksimum memerlukan minimal dua kali proses guna menghaluskan keluaran dan memberi kesempatan *feedback* lebih banyak sehingga menekan kemungkinan terjadinya *overshoot*/*undershoot* dan *ringing*.

$$NL(de) = \begin{cases} 1; & de < -1000 \\ 1 - \frac{de+1000}{-667+1000}; & -1000 \leq de \leq -667 \\ 0; & de > -667 \end{cases} \quad (5\text{-}16)$$

$$NM(de) = \begin{cases} \frac{de+1000}{-667+1000}; & -1000 < de < -667 \\ 1 - \frac{de+667}{-333+667}; & -667 \leq de < -333 \\ 0; & -1000 \geq de \geq -333 \end{cases} \quad (5\text{-}17)$$

$$NS(de) = \begin{cases} \frac{de+667}{-333+667}; & -667 < de < -333 \\ 1 - \frac{de+333}{333}; & -333 \leq de < 0 \\ 0; & -667 \geq\ de \geq 0 \end{cases} \quad (5\text{-}18)$$

$$Z(de) = \begin{cases} \frac{de+333}{333}; & -333 < de < 0 \\ 1 - \frac{de}{333}; & 0 \leq de < 333 \\ 0; & -333 \geq de \geq 333 \end{cases} \quad (5\text{-}19)$$

$$PS(de) = \begin{cases} \frac{de}{333}; & 0 < de < 333 \\ 1 - \frac{de-333}{667-333}; & 333 \leq de < 667 \\ 0; & 0 \geq de \geq 667 \end{cases} \quad (5\text{-}20)$$

$$PM(de) = \begin{cases} \frac{de-333}{667-333}; & 333 < de < 667 \\ 1 - \frac{de-667}{1000-667}; & 667 \leq de < 1000 \\ 0; & 333 \geq de \geq 1000 \end{cases} \quad (5\text{-}21)$$

$$PL(de) = \begin{cases} 1; & de > 1000 \\ \frac{de-667}{1000-667}; & 667 \leq de \leq 1000 \\ 0; & de < 667 \end{cases} \quad (5\text{-}22)$$

Dengan:

de = *delta error* (RPM)

NL = fungsi keanggotaan *negative large*

NM = fungsi keanggotaan *negative medium*

NS = fungsi keanggotaan *negative small*

ZE = fungsi keanggotaan *zero*

PS = fungsi keanggotaan *positive small*

PM = fungsi keanggotaan *positive medium*

PL = fungsi keanggotaan *positive large*

Persamaan (5-16) hingga Persamaan (5-22) merupakan fungsi derajat keanggotaan fuzzy berdasarkan masukan turunan dari *error* dari kecepatan putar yang diinginkan (*delta error*).

Tabel 5.2
Representasi tabulasi inferensi Mamdani dari *fuzzy rules*.

de / e	NL	NM	NS	ZE	PS	PM	PL
NL	APL	APL	APL	APL	APL	APM	APS
NM	APL	APM	APM	APM	APM	APS	APS
NS	APM	APS	APS	APS	APS	AZE	AZE
ZE	APS	APS	AZE	AZE	AZE	ANS	ANS
PS	AZE	AZE	ANS	ANS	ANS	ANS	ANM
PM	ANS	ANS	ANM	ANM	ANM	ANM	ANL
PL	ANS	ANM	ANL	ANL	ANL	ANL	ANL

Sumber: Rancangan Penelitian.

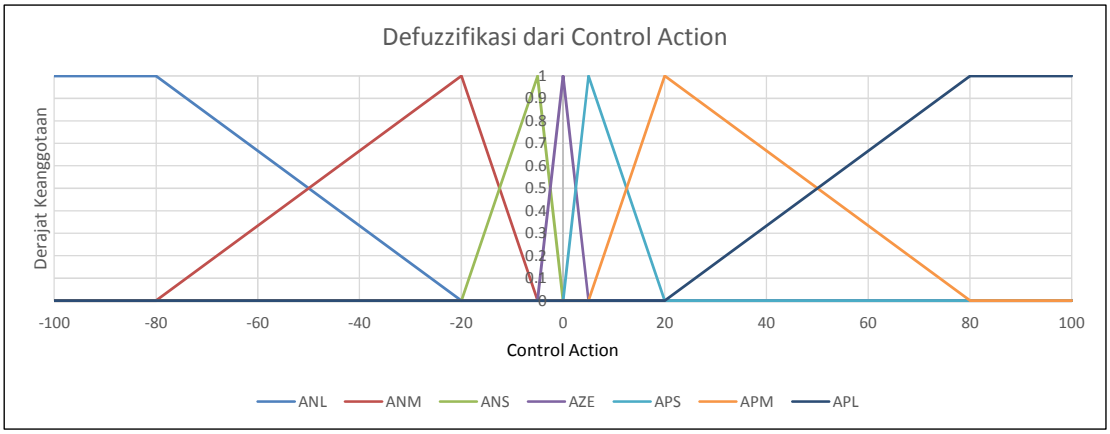
Proses *defuzzification* nantinya akan dilakukan dengan memberikan nilai tegas (*crisp value*) untuk diberikan sebagai keluaran. Nilai keluaran tersebut diberi klasifikasi sesuai dengan tujuh kategori seperti di bawah ini:

1. APL, *yaitu Add Positive Large*, atau menambahkan bilangan positif yang besar.
2. APM, yaitu *Add Positive Medium*, atau menambahkan bilangan positif yang menengah.
3. APS, yaitu *Add Positive Small*, atau menambahkan bilangan positif yang kecil.
4. AZE, yaitu *Add Zero*, atau tidak menambahkan apa-apa.
5. ANS, yaitu *Add Negative Small*, atau menambahkan bilangan negatif yang kecil.

6. ANM, yaitu *Add Negative Medium*, atau menambahkan bilangan negatif yang menengah.
7. ANL, yaitu *Add Negative Large*, atau menambahkan bilangan negatif yang besar.

Karena inferensi yang digunakan pada penelitian ini adalah Mamdani, maka digunakan representasi tabulasi seperti pada Tabel 5.2. Pada proses *defuzzification* terdapat *mapping* dari hasil keluaran Tabel 5.2 menuju ke nilai keluaran pin 2 analog Arduino Mega 2560. Oleh karena itu ditentukan kategori nilai tegas (*crisp value*) pada *defuzzification* seperti berikut ini dan ditunjukkan secara grafis pada Gambar 5.7:

1. APL, menggunakan nilai +080.
2. APM, menggunakan nilai +020.
3. APS, menggunakan nilai +005.
4. AZE, menggunakan nilai 000.
5. ANS, menggunakan nilai -005.
6. ANM, menggunakan nilai -020.
7. ANL, menggunakan nilai -080.



Gambar 5.7 Himpunan *fuzzy* dari aksi kendali.
Sumber: Rancangan Penelitian.

Berdasarkan masukan dari *error* dan *delta error*, maka disusun tabel aturan (*rule table*) untuk menentukan keluaran nilai kendali relatif dengan menimbang semua keanggotaan yang mungkin dari *error* dan *delta error*. Sesuai dengan sifat dari inferensi Mamdani, tabel ini merupakan bentuk intuitif manusia sebagai respon (reaksi) terhadap *error* dan *delta error*. Representasi dalam bentuk tabel dari *fuzzy rules* disajikan pada Tabel 5.2.

Dalam Tabel 5.2 ditunjukkan hasil dari pengambilan keputusan fuzzy yang berupa keluaran nilai keanggotaan fuzzy. Tabel 5.2 disusun secara intuitif berdasarkan *error* dengan pertimbangan sebagai berikut:

1. Bila *error* berupa bilangan negatif besar, maka tambahkan kendali bernilai positif besar; Bila *error* berupa bilangan negatif sedang, maka tambahkan kendali bernilai positif sedang; Bila *error* berupa bilangan negatif kecil, maka tambahkan kendali bernilai positif kecil.
2. Bila *error* nol, jangan tambahkan apa-apa.
3. Bila *error* berupa bilangan positif besar, maka tambahkan kendali bernilai negatif besar; Bila *error* berupa bilangan positif sedang, maka tambahkan kendali bernilai negatif sedang; Bila *error* berupa bilangan positif kecil, maka tambahkan kendali bernilai negatif kecil.

Tabel 5.3
Representasi pembobotan dari *fuzzy rules*.

de / e	NL	NM	NS	ZE	PS	PM	PL
NL	W_1	W_2	W_3	W_4	W_5	W_6	W_7
NM	W_8	W_9	W_{10}	W_{11}	W_{12}	W_{13}	W_{14}
NS	W_{15}	W_{16}	W_{17}	W_{18}	W_{19}	W_{20}	W_{21}
ZE	W_{22}	W_{23}	W_{24}	W_{25}	W_{26}	W_{27}	W_{28}
PS	W_{29}	W_{30}	W_{31}	W_{32}	W_{33}	W_{34}	W_{35}
PM	W_{36}	W_{37}	W_{38}	W_{39}	W_{40}	W_{41}	W_{42}
PL	W_{43}	W_{44}	W_{45}	W_{46}	W_{47}	W_{48}	W_{49}

Sumber: Rancangan Penelitian.

Selain itu, terdapat *delta error*, yang analog dengan besaran naiknya atau turunnya kecepatan dari kecepatan sebelumnya. Secara intuitif, maka pertimbangannya berdasarkan *delta-error* dan *error* adalah sebagai berikut:

1. Bila *error* berupa bilangan negatif besar, dan kecepatan sedang turun atau tetap, maka tambahkan kendali bernilai positif besar, agar segera mencapai *set-point*.
2. Bila *error* berupa bilangan negatif besar, tetapi kecepatan sedang naik cepat, maka penambahan kendali jangan terlalu besar, agar tidak terjadi *overshoot* berlebihan.

3. Bila *error* berupa bilangan positif besar, dan kecepatan sedang naik atau tetap, maka tambahkan kendali bernilai negatif besar, agar segera mencapai *set-point*.
4. Bila *error* berupa bilangan positif besar, tetapi kecepatan sedang turun cepat, maka pengurangan kendali jangan terlalu besar, agar tidak terjadi *undershoot* yang berlebihan.
5. Bila tidak ada *error*, dan kecepatan tetap, maka jangan menambahkan apapun.
6. Bila tidak ada *error*, tetapi kecepatan sedang naik cepat, maka kendali harus dikurangi karena kecepatan memiliki tren naik.
7. Bila tidak ada *error*, tetapi kecepatan sedang turun cepat, maka kendali harus ditambah karena kecepatan memiliki tren turun.

Sesuai dengan *rule table* pada Tabel 5.2, ketika masukan dari keanggotaan *error* dan *delta-error* digabungkan, maka dicari bobot masing-masing sel pada Tabel 5.2 dengan menggunakan operator nilai minimal antara kedua masukan tersebut. Untuk memudahkan, Tabel 5.3 ditampilkan untuk merepresentasikan pembobotan tersebut.

Pencarian bobot tiap sel dilakukan dengan cara mencari nilai minimal hasil silangan keanggotaan *error* dengan *delta-error*. Pencarian bobot disajikan sesuai dengan Persamaan (5-23) hingga Persamaan (5-71).

$$W_1 = \min(NL(e), NL(de)) \tag{5-23}$$

$$W_2 = \min(NL(e), NM(de)) \tag{5-24}$$

$$W_3 = \min(NL(e), NS(de)) \tag{5-25}$$

$$W_4 = \min(NL(e), ZE(de)) \tag{5-26}$$

$$W_5 = \min(NL(e), PS(de)) \tag{5-27}$$

$$W_6 = \min(NL(e), PM(de)) \tag{5-28}$$

$$W_7 = \min(NL(e), PL(de)) \tag{5-29}$$

$$W_8 = \min(NM(e), NL(de)) \tag{5-30}$$

$$W_9 = \min(NM(e), NM(de)) \tag{5-31}$$

$$W_{10} = \min(NM(e), NS(de)) \tag{5-32}$$

$$W_{11} = \min(NM(e), ZE(de)) \tag{5-33}$$

$$W_{12} = \min(NM(e), PS(de)) \quad (5\text{-}34)$$

$$W_{13} = \min(NM(e), PM(de)) \quad (5\text{-}35)$$

$$W_{14} = \min(NM(e), PL(de)) \quad (5\text{-}36)$$

$$W_{15} = \min(NS(e), NL(de)) \quad (5\text{-}37)$$

$$W_{16} = \min(NS(e), NM(de)) \quad (5\text{-}38)$$

$$W_{17} = \min(NS(e), NS(de)) \quad (5\text{-}39)$$

$$W_{18} = \min(NS(e), ZE(de)) \quad (5\text{-}40)$$

$$W_{19} = \min(NS(e), PS(de)) \quad (5\text{-}41)$$

$$W_{20} = \min(NS(e), PM(de)) \quad (5\text{-}42)$$

$$W_{21} = \min(NS(e), PL(de)) \quad (5\text{-}43)$$

$$W_{22} = \min(ZE(e), NL(de)) \quad (5\text{-}44)$$

$$W_{23} = \min(ZE(e), NM(de)) \quad (5\text{-}45)$$

$$W_{24} = \min(ZE(e), NS(de)) \quad (5\text{-}46)$$

$$W_{25} = \min(ZE(e), ZE(de)) \quad (5\text{-}47)$$

$$W_{26} = \min(ZE(e), PS(de)) \quad (5\text{-}48)$$

$$W_{27} = \min(ZE(e), PM(de)) \quad (5\text{-}49)$$

$$W_{28} = \min(ZE(e), PL(de)) \quad (5\text{-}50)$$

$$W_{29} = \min(PS(e), NL(de)) \quad (5\text{-}51)$$

$$W_{30} = \min(PS(e), NM(de)) \quad (5\text{-}52)$$

$$W_{31} = \min(PS(e), NS(de)) \quad (5\text{-}53)$$

$$W_{32} = \min(PS(e), ZE(de)) \quad (5\text{-}54)$$

$$W_{33} = \min(PS(e), PS(de)) \quad (5\text{-}55)$$

$$W_{34} = \min(PS(e), PM(de)) \quad (5\text{-}56)$$

$$W_{35} = \min(PS(e), PL(de)) \quad (5\text{-}57)$$

$$W_{36} = \min(PM(e), NL(de)) \quad (5\text{-}58)$$

$$W_{37} = \min(PM(e), NM(de)) \quad (5\text{-}59)$$

$$W_{38} = \min(PM(e), NS(de)) \quad (5\text{-}60)$$

$$W_{39} = \min(PM(e), ZE(de)) \quad (5\text{-}61)$$

$$W_{40} = \min(PM(e), PS(de)) \quad (5\text{-}62)$$

$$W_{41} = \min(PM(e), PM(de)) \quad (5\text{-}63)$$

$$W_{42} = \min(PM(e), PL(de)) \quad (5\text{-}64)$$

$$W_{43} = \min(PL(e), NL(de)) \quad (5\text{-}65)$$

$$W_{44} = \min(PL(e), NM(de)) \quad (5\text{-}66)$$

$$W_{45} = \min(PL(e), NS(de)) \quad (5\text{-}67)$$

$$W_{46} = \min(PL(e), ZE(de)) \quad (5\text{-}68)$$

$$W_{47} = \min(PL(e), PS(de)) \quad (5\text{-}69)$$

$$W_{48} = \min(PL(e), PM(de)) \quad (5\text{-}70)$$

$$W_{49} = \min(PL(e), PL(de)) \quad (5\text{-}71)$$

Dengan:

W_n = bobot sel ke-n
e = *error* (RPM)
de = *delta error* (RPM)
NL = fungsi keanggotaan *negative large*
NM = fungsi keanggotaan *negative medium*
NS = fungsi keanggotaan *negative small*
ZE = fungsi keanggotaan *zero*
PS = fungsi keanggotaan *positive small*
PM = fungsi keanggotaan *positive medium*
PL = fungsi keanggotaan *positive large*

Cara *defuzzification* yang digunakan dalam metode ini adalah *center of gravity*. Metode ini adalah melakukan perkalian masing-masing bobot dengan nilai tegas *defuzzification* (seperti pada Tabel 5.2), kemudian dijumlahkan, dan dibagi dengan jumlahan semua bobotnya.

$$WR_1 = W_1 \cdot APL + W_2 \cdot APL + W_3 \cdot APL + W_4 \cdot APL + W_5 \cdot APL + W_6 \cdot APM + W_7 \cdot APS \quad (5\text{-}72)$$

$$WR_2 = W_8 \cdot APL + W_9 \cdot APM + W_{10} \cdot APM + W_{11} \cdot APM + W_{12} \cdot APM + W_{13} \cdot APS + W_{14} \cdot APS \quad (5\text{-}73)$$

$$WR_3 = W_{15} \cdot APM + W_{16} \cdot APS + W_{17} \cdot APS + W_{18} \cdot APS + W_{19} \cdot APS + W_{20} \cdot AZE + W_{21} \cdot AZE \quad (5\text{-}74)$$

$$WR_4 = W_{22} \cdot APS + W_{23} \cdot APS + W_{24} \cdot AZE + W_{25} \cdot AZE + W_{26} \cdot AZE + W_{27} \cdot ANS + W_{28} \cdot ANS \quad (5\text{-}75)$$

$$WR_5 = W_{29} \cdot AZE + W_{30} \cdot AZE + W_{31} \cdot ANS + W_{32} \cdot ANS + W_{33} \cdot ANS + W_{34} \cdot ANS + W_{35} \cdot ANM \quad (5\text{-}76)$$

$$WR_6 = W_{36} \cdot ANS + W_{37} \cdot ANS + W_{38} \cdot ANM + W_{39} \cdot ANM + W_{40} \cdot ANM + W_{41} \cdot ANM + W_{42} \cdot ANL \quad (5\text{-}77)$$

$$WR_7 = W_{43} \cdot ANS + W_{44} \cdot ANM + W_{45} \cdot ANL + W_{46} \cdot ANL + W_{47} \cdot ANL + W_{48} \cdot ANL + W_{49} \cdot ANL \quad (5\text{-}78)$$

$$u = \frac{\sum_{x=1}^{7} WR_x}{\sum_{y=1}^{49} W_y} \quad (5\text{-}79)$$

Dengan:

u = *control action*

W_y = bobot sel ke-y

WR_x = jumlahan bobot dikalikan nilai tegas defuzzifikasi baris ke-x

ANL = nilai tegas *add negative large*

ANM = nilai tegas *add negative medium*

ANS = nilai tegas *add negative small*

AZE = nilai tegas *add zero*

APS = nilai tegas *add positive small*

APM = nilai tegas *add positive medium*

APL = nilai tegas *add positive large*

Dalam Persamaan (5-72) sampai Persamaan (5-78) dinyatakan cara mencari jumlah perkalian (*sum product*) antara bobot dengan nilai tegas tiap sel pada Tabel 5.3, kemudian dicari nilai u yang merupakan nilai tegas keluaran *control action* seperti pada Persamaan (5-79).

$$v(t) = \begin{cases} 0; & v(t_{-1}) + u(t) < 0 \\ v(t_{-1}) + u(t); & 0 < v(t_{-1}) + u(t) < 255 \\ 255; & v(t_{-1}) + u(t) > 255 \end{cases} \quad (5\text{-}80)$$

Dengan:

v = *control value*

u = *control action*

t = waktu cuplikan saat ini

t_{-1} = waktu cuplikan sebelumnya

Nilai *control action* (u) yang dihasilkan dipakai sebagai faktor penambah dari *control value* (v) arduino yang terdahulu. Hasil dari penambahan nilai itu akan dibatasi dengan nilai minimal sebesar 0 dan nilai maksimal sebesar 255, sebagaimana ditunjukkan pada Persamaan (5-80).

5.2. Perangkat Keras Kendali

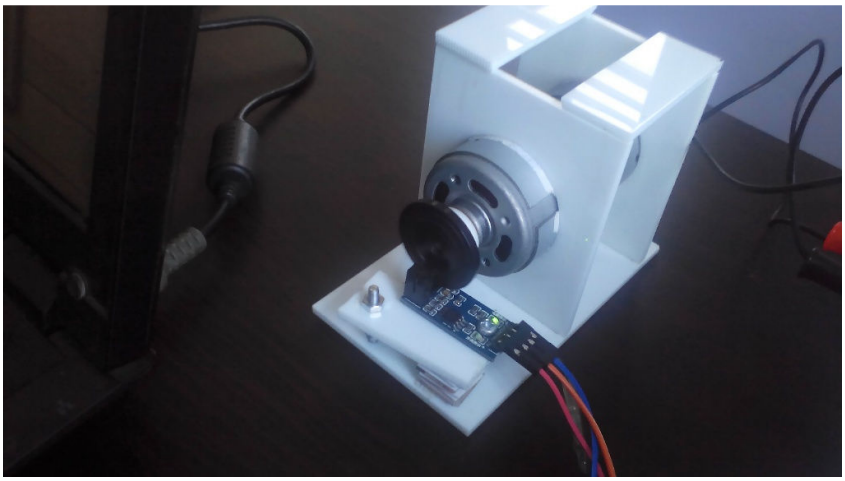
Blok diagram perangkat keras pengendali telah disajikan pada Bab IV Gambar 4.2 dan Gambar 4.3. Secara fisik, semua bagian dibuat menjadi satu kotak kecuali motor DC dan sensor kecepatan. Perangkat ini telah dibuat sehingga bisa beroperasi menggunakan catu daya listrik 220 VAC sebagaimana umum dipakai di Indonesia.

Gambar 5.8 Perangkat keras pengendali dalam posisi *Ready Mode*.
Sumber: Hasil Penelitian.

Pada Gambar 5.8 disajikan kotak perangkat keras pengendali yang akan digunakan, yang terdiri dari:

1. Arduino Mega 2560.
2. DC *Motor Driver* L298N.
3. AC-DC *Adapter* 220 VAC ke 20 VDC dan 5 VDC.
4. LCD untuk *display*.
5. *Keypad*.
6. 8 *Pushbutton*.

7. 3 LED.
8. *Output Jack* untuk Motor DC.
9. *Output Cable* untuk Sensor Kecepatan LM393.
10. *Input Cable* untuk 220VAC.
11. *Input Jack* untuk USB *Programming*.



Gambar 5.9 Perangkat motor DC dan *speed sensor*.
Sumber: Hasil Penelitian.

Pada gambar 5.9 disajikan motor DC yang terhubung dengan *driver* dan *speed sensor* yang terhubung dengan Arduino. Perangkat motor DC yang ada terdiri dari:

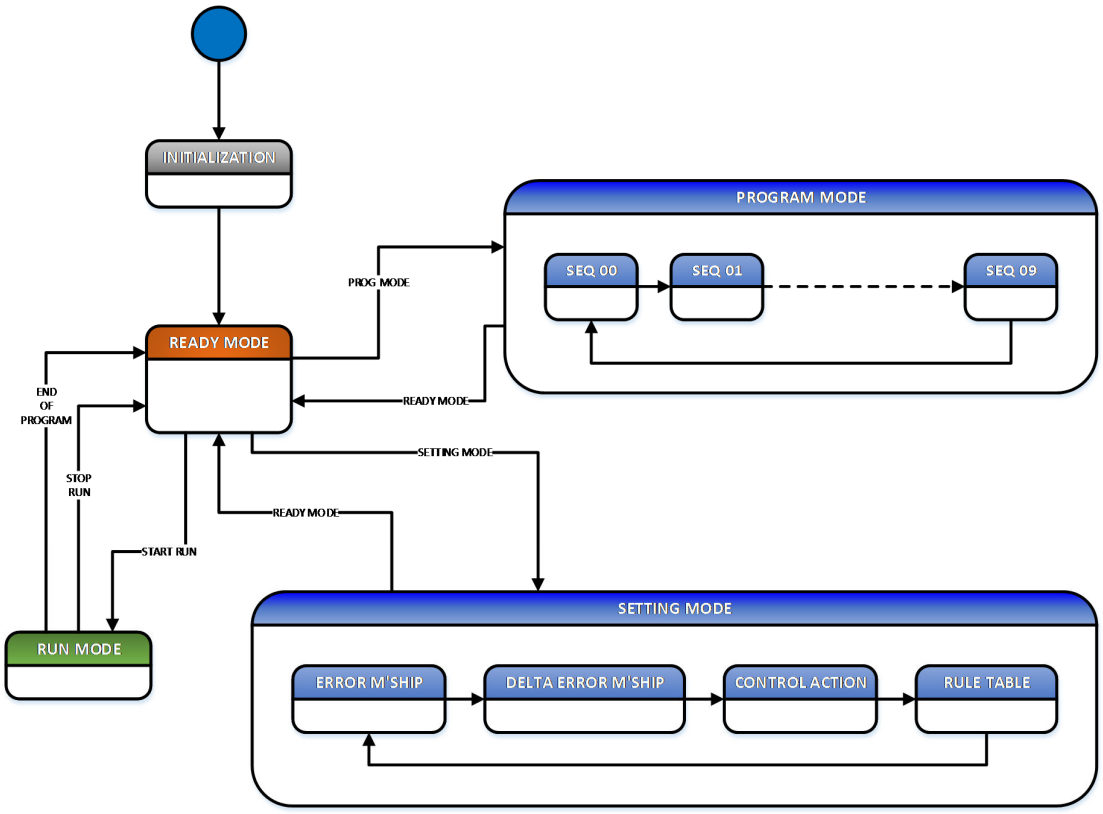
1. Motor DC RS 555-555 (*rated voltage*: 20V; *no load speed*: 3500 RPM).
2. *Speed Sensor* LM393.

Gambar 5.10 *USB port* pada perangkat pengendali.
Sumber: Hasil Penelitian.

Gambar 5.10 menyajikan *USB port* pada perangkat pengendali (Arduino) yang berfungsi untuk mengisikan program dari PC ke dalam Arduino dan sebagai alat komunikasi *serial* antara Arduino dengan PC. Dengan fasilitas komunikasi ini, akan dibangun suatu antarmuka antara PC dengan Arduino untuk keperluan *data logging* ke memori PC yang memiliki kapasitas lebih besar daripada Arduino.

5.3. Perangkat Lunak Pengendali Utama dan Pendukung

Sesuai dengan rancangan pada Bab IV, sistem menu pada perangkat dirancang sedemikian rupa sehingga memiliki fasilitas untuk pemakai melakukan pemrograman pada urutan *set-point* (*Angular Speed Sequence Program*), melakukan penalaan parameter fuzzy (*Fuzzy Set, Fuzzy Rules, and Deffuzification Setting*), dan menjalankan program.

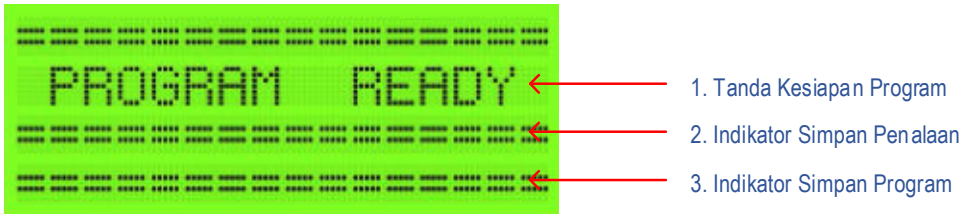


Gambar 5.11 Diagram Urutan Menu pada Perangkat.
Sumber: Rancangan Penelitian.

Secara umum, penjelasan Gambar 5.11 tentang menu pada perangkat memiliki beberapa modus:

1. *Ready Mode*, di mana perangkat siap untuk menerima perintah dari pemakai. Modus ini menyalakan LED warna oranye.
2. *Program Mode*, di mana perangkat dalam kondisi siap diprogram untuk menentukan urutan (*angular speed sequence program*). Pada modus ini, LED warna biru menyala.
3. *Setting Mode*, di mana perangkat dalam kondisi siap ditala untuk menentukan aturan fuzzy (*Fuzzy Set, Fuzzy Rules, and Deffuzification Setting*). Pada modus ini, LED warna biru akan berkedip.
4. *Run Mode*, di mana perangkat menjalankan program (*angular speed sequence program*) sesuai dengan aturan fuzzy yang ada, menjalankan aksi pemutaran motor DC, sembari mengirimkan *data timing*, *set-point*, dan *process value* ke *Data Logger* melalui *serial communication*. Pada modus ini, LED warna hijau akan berkedip.

Pada Gambar 5.11, tidak ada *final state* karena sifat dari *loop* pada perangkat Arduino yang tidak pernah berhenti menuju ke *final state*, kecuali terjadi pemutusan daya listrik. Sesuai dengan diagram urutan menu pada perangkat, maka berikut ini adalah contoh perangkat ketika dalam berbagai modus.



Gambar 5.12 Tampilan LCD perangkat pada posisi *Ready Mode* dan keterangannya.
Sumber: Rancangan Penelitian.

Gambar 5.8 menunjukkan perangkat dalam posisi *Ready Mode*. Pada modus tersebut, pemakai bisa menuju ke *Program Mode* atau *Setting Mode* dengan menekan tombol PROGRAM MODE atau SETTING MODE. Sedangkan *Run Mode* bisa dijalankan dengan menekan tombol START RUN dan hanya bisa dilakukan bila *angular speed sequence program* sudah diisi. Gambar 5.12 menunjukkan rancangan tampilan pada LCD ketika perangkat dalam posisi *Ready Mode*. Petunjuk rancangan tampilan sesuai dengan penomoran pada Gambar 5.12 adalah sebagai berikut:

1. Tanda Kesiapan Program, merupakan penanda apakah *angular speed sequence program* sudah diisi atau tidak. Bila sudah terisi maka akan tampil PROGRAM

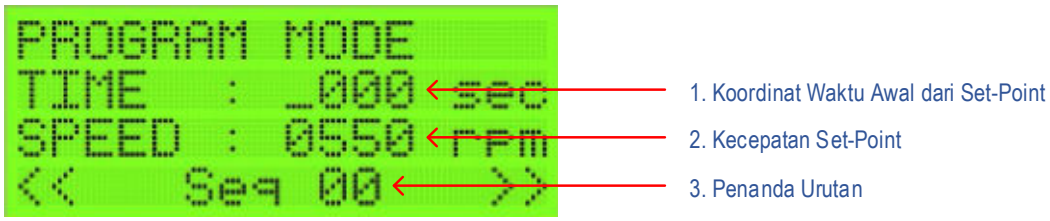
READY, namun bila belum terisi (atau sedang dalam posisi *reset*) maka akan tampil NO PROGRAM.

2. Indikator Simpan Penalaan, merupakan penanda apakah hasil penalaan sudah disimpan pada EEPROM atau belum. Ketika sudah tersimpan maka akan tampil tanda sama dengan (=) sepanjang layar, tetapi bila belum tersimpan maka akan tampil Sett Unsaved secara berkedip.
3. Indikator Simpan Program, merupakan penanda apakah *angular speed sequence program* sudah disimpan pada EEPROM atau belum. Ketika sudah tersimpan maka akan tampil tanda sama dengan (=) sepanjang layar, tetapi bila belum tersimpan maka akan tampil Data Unsaved secara berkedip.

Baik ketika *angular speed sequence program* belum disimpan atau penalaan belum disimpan, maka keduanya bisa disimpan dengan menekan tombol SAVE TO EEPROM ketika perangkat dalam posisi *Ready Mode*.

Gambar 5.13 Perangkat ketika dalam *Program Mode*.
Sumber: Hasil Penelitian.

Gambar 5.13 menunjukkan perangkat ketika sedang dalam *Program Mode*. Pada modus ini pemakai bisa melakukan pemrograman *sequence* dari *set-point* yang diinginkan (*angular speed sequence program*) dalam satuan RPM dengan waktu mulainya dalam satuan detik.



Gambar 5.14 Tampilan LCD perangkat pada posisi *Program Mode* dan keterangannya. Sumber: Rancangan Penelitian.

Pada Gambar 5.14 ditunjukkan tampilan LCD perangkat ketika pada posisi *Program Mode*, dengan petunjuk rancangan tampilan sebagai berikut:

1. Koordinat Waktu Awal dari *Set-Point*, yaitu variabel yang menunjukkan koordinat waktu awal dari set-point yang diinginkan. Variabel ini menunjukkan waktu relatif dari awal pemakai menekan tombol START RUN, dan memiliki satuan detik (*second*). Variabel ini bisa diubah menggunakan gabungan tombol NEXT CURSOR dengan DECREASE DIGIT dan INCREASE DIGIT atau menggunakan *keypad*. Variabel bisa diisi dengan nilai bulat mulai dari 0000 hingga 9999, dengan catatan nilai pada urutan setelahnya harus minimal 1 detik lebih besar daripada nilai pada urutan sebelumnya. Untuk menuju variabel Kecepatan *Set-Point*, pemakai bisa menggunakan tombol NEXT FIELD.

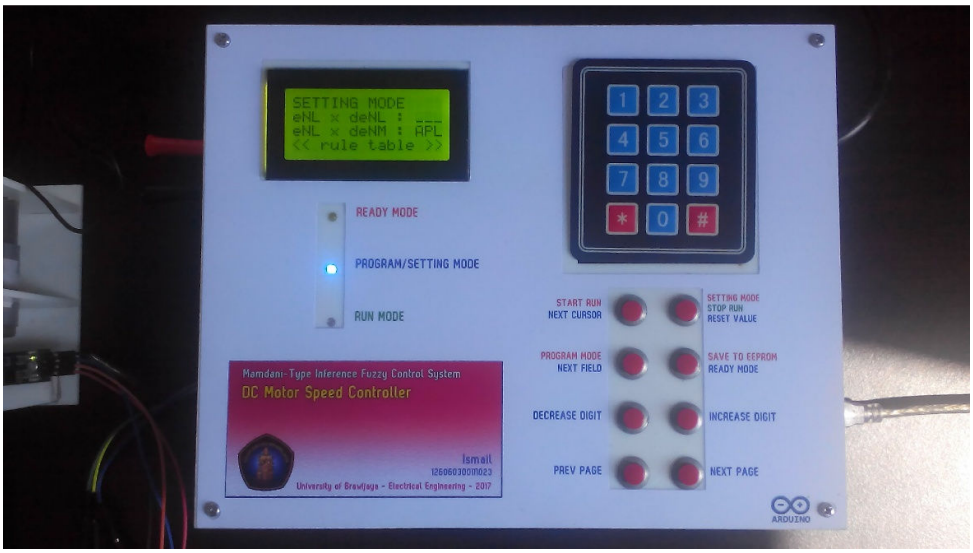
2. Kecepatan *Set-Point*, yaitu variabel yang menunjukkan *set-point* yang diinginkan pada koordinat waktu tertentu, dan memiliki satuan RPM. Variabel ini bisa diisi dengan nilai bulat mulai dari 0500 hingga 2500, atau di-*reset* sehingga bernilai 0000. Variabel ini bisa diubah menggunakan gabungan tombol NEXT CURSOR dengan DECREASE DIGIT dan INCREASE DIGIT atau menggunakan *keypad*. Tombol RESET VALUE bisa ditekan untuk melakukan *reset* pada variabel ini. Untuk menuju variabel Koordinat Waktu Awal dari *Set-Point*, pemakai bisa menggunakan tombol NEXT FIELD.

3. Penanda Urutan, yang merupakan penunjuk halaman atau urutan dari *angular speed sequence program*. Variabel ini disediakan mulai dari Seq 00 hingga Seq 09, di mana variabel *set-point* pada Seq 09 harus di-*reset* agar urutan memiliki kondisi berhentinya motor DC. Halaman yang sedang aktif bisa diubah dengan menggunakan tombol PREV PAGE dan NEXT PAGE.

Ketika dalam posisi *Program Mode*, pemakai bisa sewaktu-waktu menekan tombol READY MODE untuk kembali pada posisi *Ready Mode*. Tetapi program urutan yang telah diubah pada saat ini belum tersimpan pada EEPROM. Untuk menyimpan pemakai bisa menekan tombol SAVE TO EEPROM.

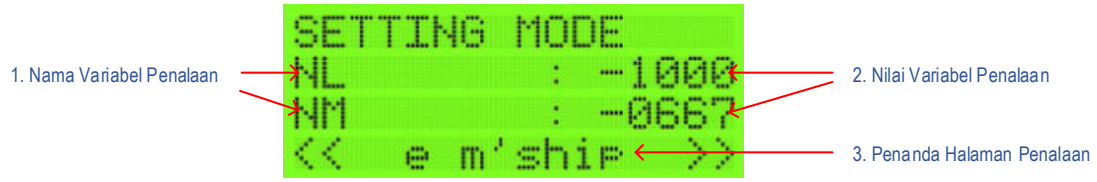
Sedangkan *Setting Mode* yang dialami perangkat ditunjukkan dalam Gambar 5.15. Dalam modus ini pemakai bisa mengisikan parameter *fuzzy* sesuai dengan kebutuhan percobaan. Parameter *fuzzy* antara lain:

1. *Error Membership*, yaitu mengatur titik-titik *error* (baik positif maupun negatif) dari 7 keanggotaan fuzzy.
2. *Delta-Error Membership*, yaitu mengatur titik-titik *delta error* (baik positif maupun negatif) dari 7 keanggotaan fuzzy.
3. *Control Action*, yaitu menentukan titik-titik dari 7 nilai tegas (*crisp value*) untuk keluaran penambahan nilai kendali motor DC.
4. *Rule Table*, yaitu pengaturan tabel aturan fuzzy, yang terdiri dari 49 sel macam pengaturan.



Gambar 5.15 Perangkat ketika sedang dalam *Setting Mode*.
Sumber: Hasil Penelitian.

Pada waktu perangkat sedang dalam posisi *Setting Mode*, maka tampilan pada LCD ditunjukkan pada Gambar 5.16.



Gambar 5.16 Tampilan LCD perangkat pada posisi *Setting Mode* dan keterangannya. Sumber: Rancangan Penelitian.

Berdasarkan Gambar 5.16, petunjuk rancangan tampilan pada waktu perangkat berada dalam posisi *Setting Mode* adalah sebagai berikut:

1. Nama Variabel Penalaan, yang merupakan nama variabel yang akan ditentukan nilainya. Pada halaman *Error Membership* (e m'ship) dan *Delta-Error Membership* (de m'ship) terdapat 7 variabel yaitu NL, NM, NS, ZE, PS, PM, dan PL. Pada halaman *Control Action* (cv mapping) terdapat 7 variabel yaitu ANL, ANM, ANS, AZE, APS, APM, dan APL. Sedangkan pada halaman *Rule Table* (rule table) terdapat 49 variabel, yaitu gabungan dari *Error Membership* dengan *Delta-Error Membership*. Setiap kali menekan tombol NEXT FIELD, karena kemampuan tampilan hanya bisa menampilkan 2 baris variabel setiap waktunya, maka tampilan akan menggulung ke atas.

2. Nilai Variabel Penalaan, yaitu merupakan isi atau nilai dari variabel penalaan. Pada halaman *Error Membership* (e m'ship) dan *Delta-Error Membership* (de m'ship) terdapat kemungkinan nilai bulat dari -1000 hingga +1000. Pada halaman *Control Action* (cv mapping) terdapat kemungkinan nilai bulat dari -127 hingga +127. Sedangkan pada halaman *Rule Table* (rule table) terdapat kemungkinan nilai enumerasi ANL, ANM, ANS, AZE, APS, APM, dan APL. Variabel ini bisa diubah menggunakan gabungan tombol NEXT CURSOR dengan DECREASE DIGIT dan INCREASE DIGIT atau menggunakan *keypad*.

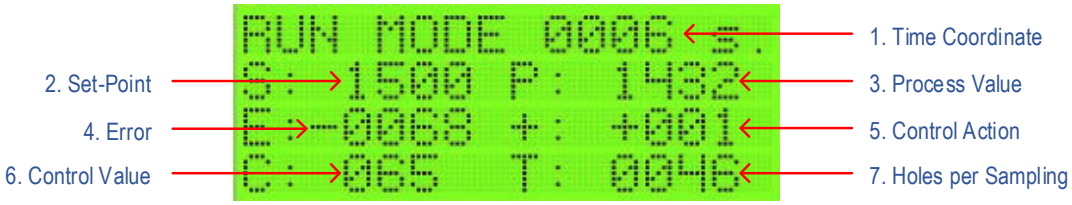
3. Penanda Halaman Penalaan, yang merupakan penunjuk halaman dari penalaan. Variabel ini disediakan nilainya meliputi e m'ship, de m'ship, cv mapping, dan rule table. Halaman yang sedang aktif bisa diubah dengan menggunakan tombol PREV PAGE dan NEXT PAGE. Untuk melakukan *reset* pada halaman yang sedang aktif bisa digunakan tombol RESET VALUE. Proses *reset* akan mengembalikan nilai variabel halaman yang sedang aktif pada nilai awal, yaitu nilai yang dibahas pada Sub Bab 5.1.5.

Ketika dalam posisi *Setting Mode*, pemakai bisa sewaktu-waktu menekan tombol READY MODE untuk kembali pada posisi *Ready Mode*. Tetapi penalaan yang telah diubah pada saat ini belum tersimpan pada EEPROM. Untuk menyimpannya pemakai bisa menekan tombol SAVE TO EEPROM.

Sedangkan ketika perangkat sedang dalam *Run Mode* diberikan contohnya pada Gambar 5.17. Pada modus ini perangkat menjalankan program sesuai dengan *set-point* yang diminta pada *angular speed sequence program* dan menjalankan respon sesuai dengan penalaan parameter fuzzy (*Fuzzy Set, Fuzzy Rules, and Deffuzification Setting*) yang diberikan.

Gambar 5.17 Perangkat ketika sedang dalam *Run Mode*.
Sumber: Hasil Penelitian.

Ketika perangkat sedang dalam posisi *Run Mode*, LCD akan menampilkan segala informasi yang berkenaan tentang kondisi motor DC yang dikendalikan dan semua variabel pengendalian, sekaligus mengirimkan informasi tersebut melalui *serial communication* menuju ke *Data Logger*.



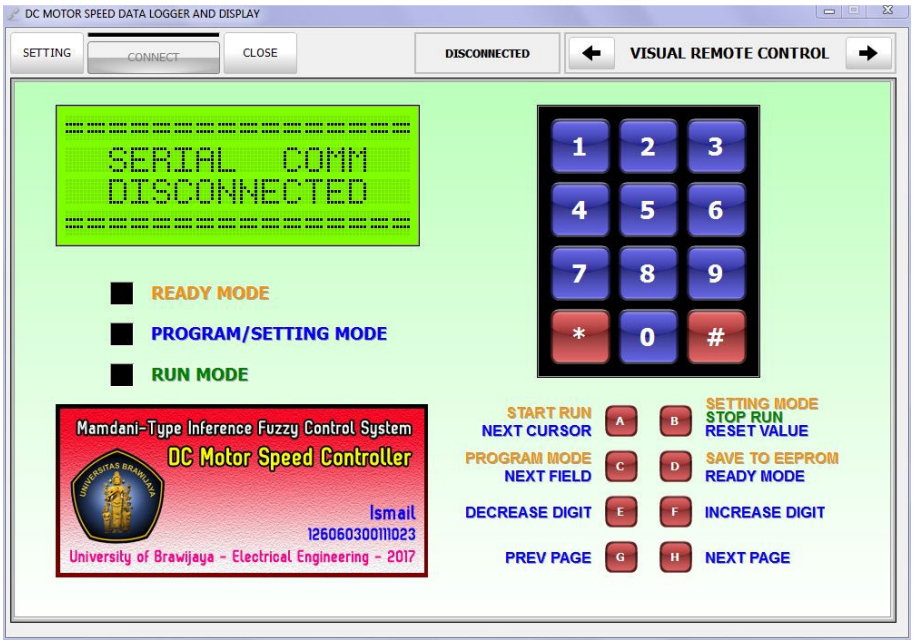
Gambar 5.18 Tampilan LCD perangkat pada posisi *Run Mode* dan keterangannya.
Sumber: Rancangan Penelitian.

Sesuai dengan Gambar 5.18, pada waktu perangkat berada dalam posisi *Run Mode*, LCD menampilkan beberapa variabel yang sedang aktif sebagai berikut:

1. *Time Coordinate*, yang menampilkan waktu relatif dari pemakai menekan tombol START RUN, yang ditampilkan dalam satuan detik (*second*).
2. *Set-Point*, yang menampilkan nilai kecepatan *set-point* motor DC yang berasal dari *angular speed sequence program* dalam satuan RPM.
3. *Process Value*, yang menampilkan nilai kecepatan motor DC yang sedang berjalan dalam satuan RPM.
4. *Error*, yang menampilkan selisih antar *process value* terhadap *set-point*, dan ditampilkan dalam satuan RPM.
5. *Control Action*, yaitu nilai penambahan atau pengurangan kendali yang menuju ke *driver* motor DC.
6. *Control Value*, yaitu nilai absolut kendali yang menuju ke *driver* motor DC.
7. *Holes per Sampling*, yaitu nilai banyaknya lubang pada *optical chopper* yang dideteksi setiap kali terjadi cuplikan pengembilan data kecepatan motor DC. Variabel ini ditampilkan hanya sebagai cadangan dan melakukan konfirmasi bahwa *process value* juga menampilkan kecepatan motor DC yang sesungguhnya.

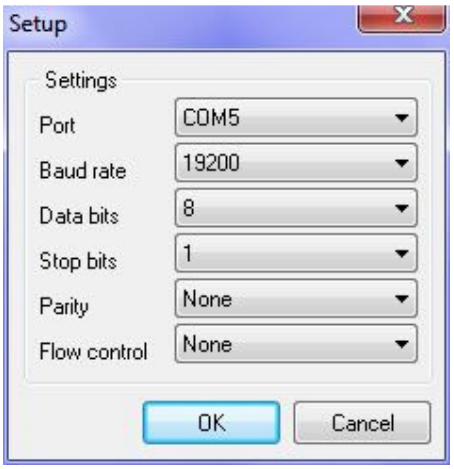
5.4. Perangkat Lunak Data Logger

Selain bisa berdiri sendiri dalam melakukan proses, perangkat juga dilengkapi dengan perangkat lunak pada PC yang bisa berkomunikasi dengan Arduino melalui *serial communication* menggunakan media *USB cable*. Perangkat lunak itu disebut dengan *Data Logger*. Berikut ini adalah contoh pengendalian, *monitoring*, dan *data logging* yang ada di PC.



Gambar 5.19 Tampilan perangkat lunak *Data Logger* sebelum disambungkan.
Sumber: Hasil Penelitian.

Pada Gambar 5.19 ditampilkan perangkat lunak *Data Logger* yang masih dalam tahap *disconnected* (belum tersambung) dengan Arduino. Tampilan itu merupakan tampilan ketika *Data Logger* baru dijalankan. Untuk melakukan penyambungan ke perangkat, harus dilakukan *setting port* dengan cara klik pada tombol SETTING.

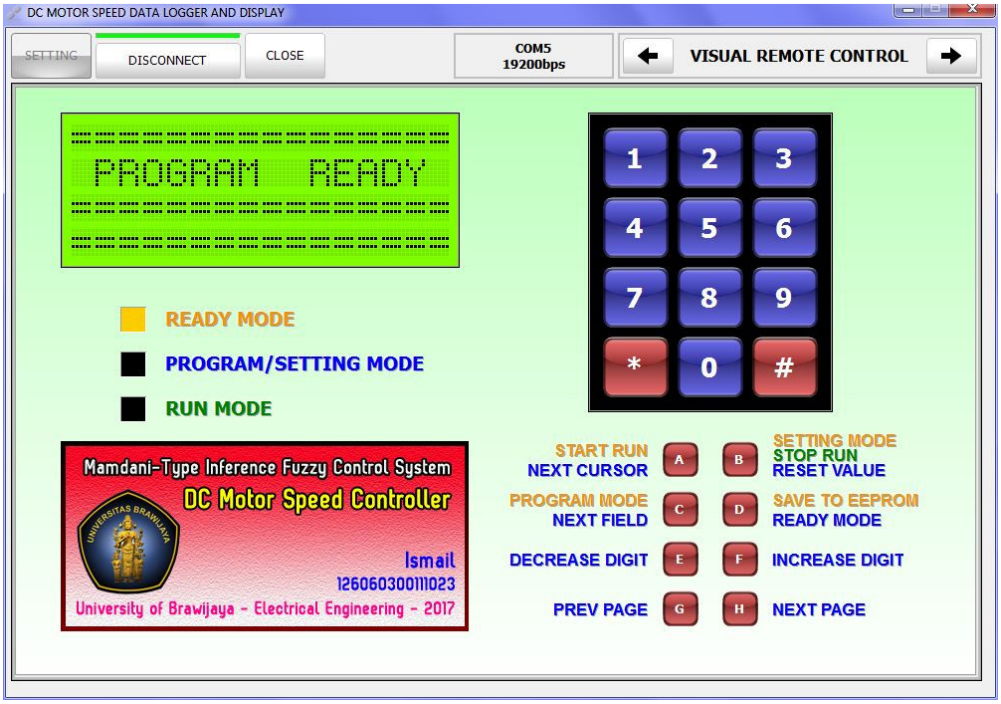


Gambar 5.20 Dialog Setup untuk menyambungkan ke perangkat.
Sumber: Hasil Penelitian.

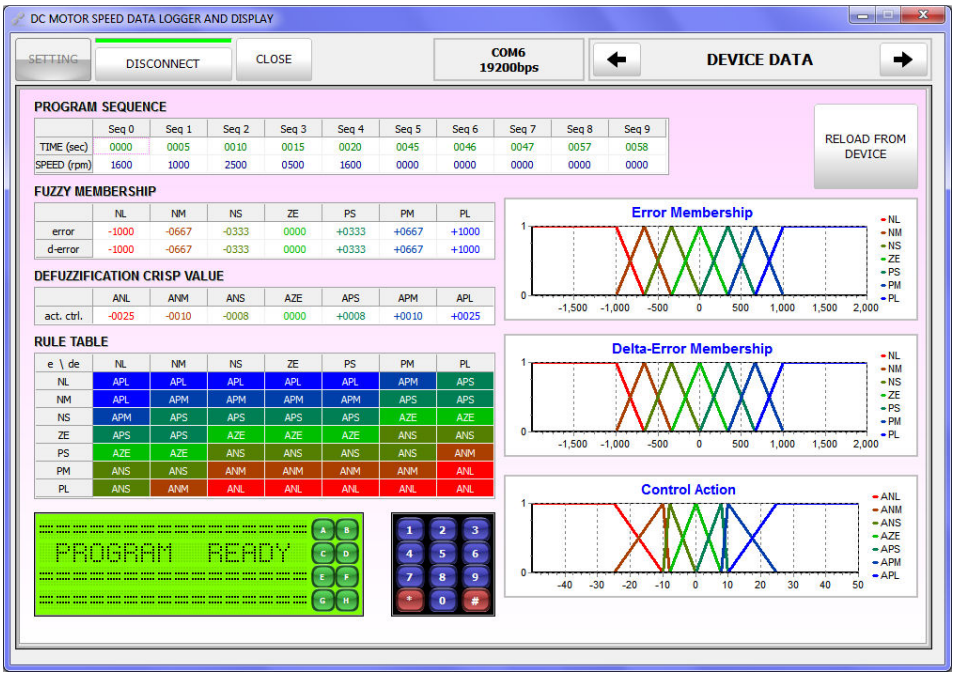
Pada Gambar 5.20 disajikan dialog Setup untuk memberikan penalaan sambungan ke perangkat. Port perlu diisi sesuai dengan *communication port* yang digunakan. Sedangkan Baud rate perlu diisi 19200 (*default*) karena perangkat juga sudah ditala dengan *baud rate*

19200. Setelah penalaan terisi, tombol OK bisa diklik untuk menyimpan hasil penalaan. Setelahnya, maka koneksi dengan perangkat keras kendali bisa dimulai dengan cara klik pada tombol CONNECT. Setelah tersambung, maka akan muncul tampilan seperti pada Gambar 5.21. Sekali tersambung, apa yang tampil pada LCD pada perangkat akan tampil juga pada replikanya di dalam *Data Logger*.

Data Logger terdiri dari beberapa halaman yang bisa dipilih oleh pemakai. Pada halaman awal, disajikan *remote control* atau replika seperti pada Gambar 5.22. Sedangkan pada halaman kedua (seperti pada Gambar 5.23) disajikan program (*angular speed sequence program*) dan penalaan (*Fuzzy Set, Fuzzy Rules, and Deffuzification Setting*) yang ada dalam perangkat, dan sebagian parameter disajikan secara visual. Ketika program atau penalaan dalam perangkat diubah, maka tampilan di *Data Logger* akan mengikuti secara otomatis. Halaman berikutnya akan tampil secara otomatis ketika perangkat memasuki *Run Mode*.

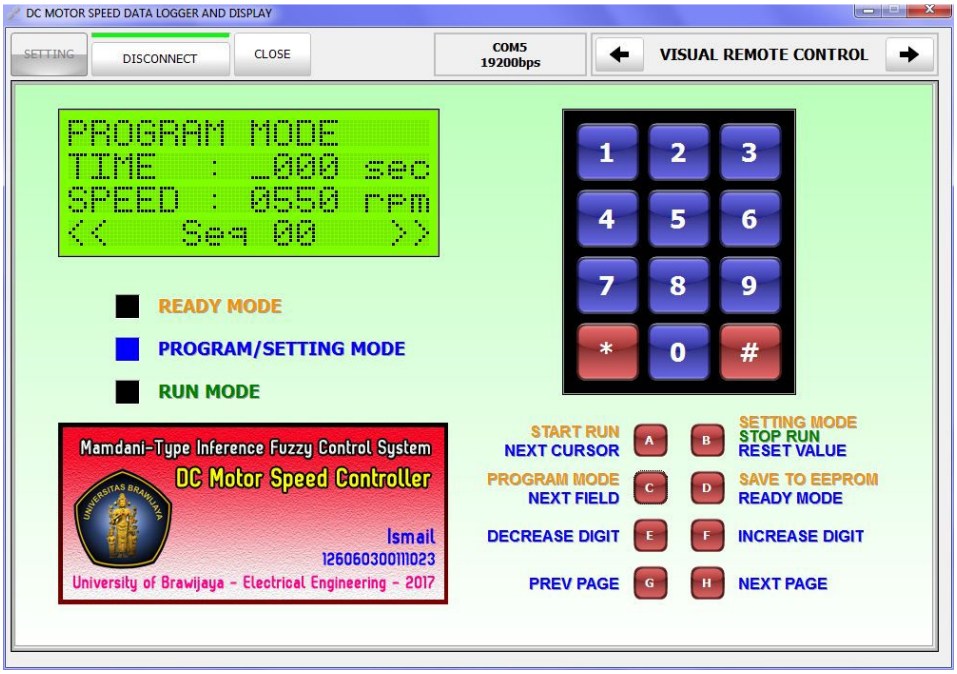


Gambar 5.21 Perangkat yang sudah tersambung menampilkan apa yang ada di LCD. Sumber: Hasil Penelitian.



Gambar 5.22 Tampilan program dan penalaan perangkat pada *Data Logger*.

Sumber: Hasil Penelitian.



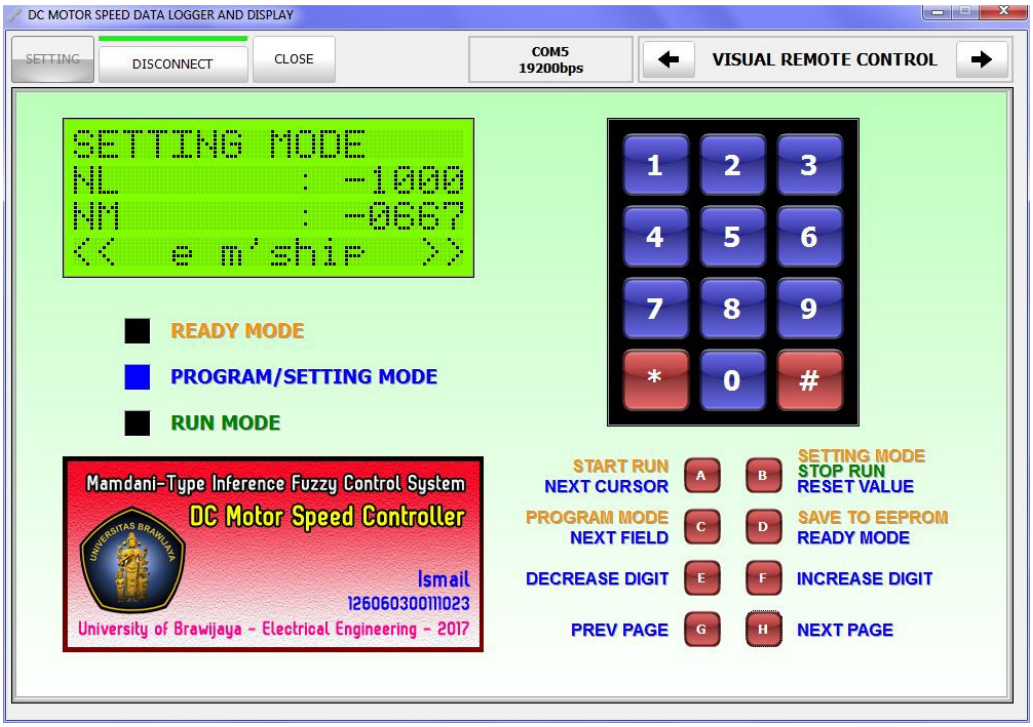
Gambar 5.23 Perangkat ketika memasuki *Program Mode*, seperti dilihat pada *Data Logger*.

Sumber: Hasil Penelitian.

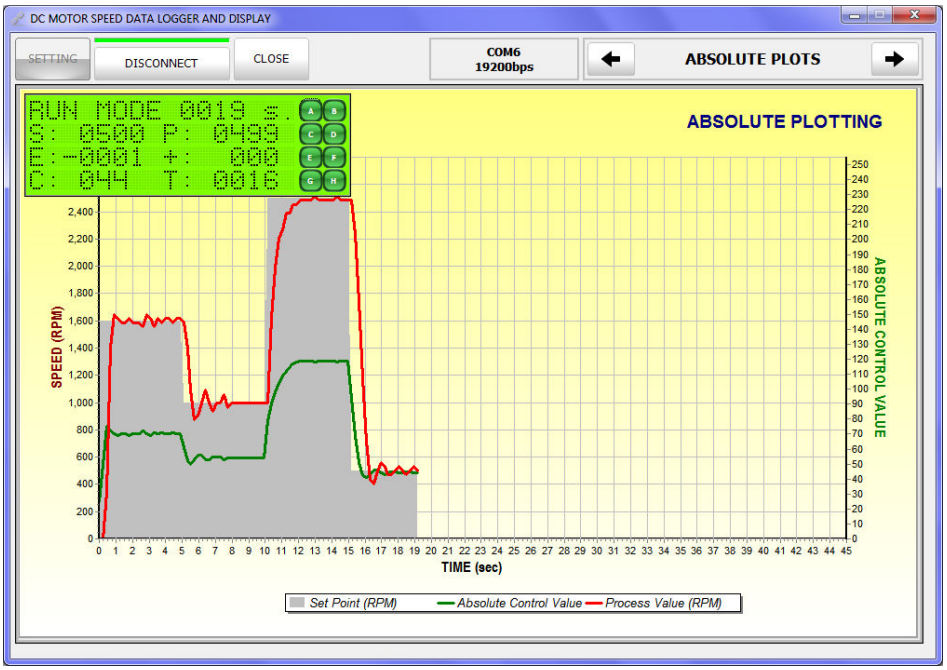
Pada halaman awal ketika perangkat memasuki posisi *Program Mode* maka akan tampil seperti Gambar 5.23. Sedangkan Gambar 5.24 adalah tampilannya ketika memasuki posisi

Setting Mode. Berikutnya ketika program dijalankan (memasuki *Run Mode*), halaman yang tampil adalah halaman ketiga. Meskipun program dalam kondisi berjalan dan otomatis menampilkan halaman ketiga, pemakai masih bisa bebas memilih halaman lainnya.

Halaman ketiga pada *Data Logger* adalah halaman *plotting live data* yang terdiri dari: *set-point* (dalam RPM), *process value* (dalam RPM), dan *absolute control value* (dengan nilai dari 0 hingga 255), seperti halnya ditunjukkan pada Gambar 5.25. Halaman keempat bisa dipilih ketika akan menampilkan *plotting live data* yang terdiri dari: *error* (dalam RPM), *delta error* atau percepatan (dalam RPM kali 10), dan *relative control value* (dengan nilai dari -255 hingga +255), seperti terlihat pada Gambar 5.26.

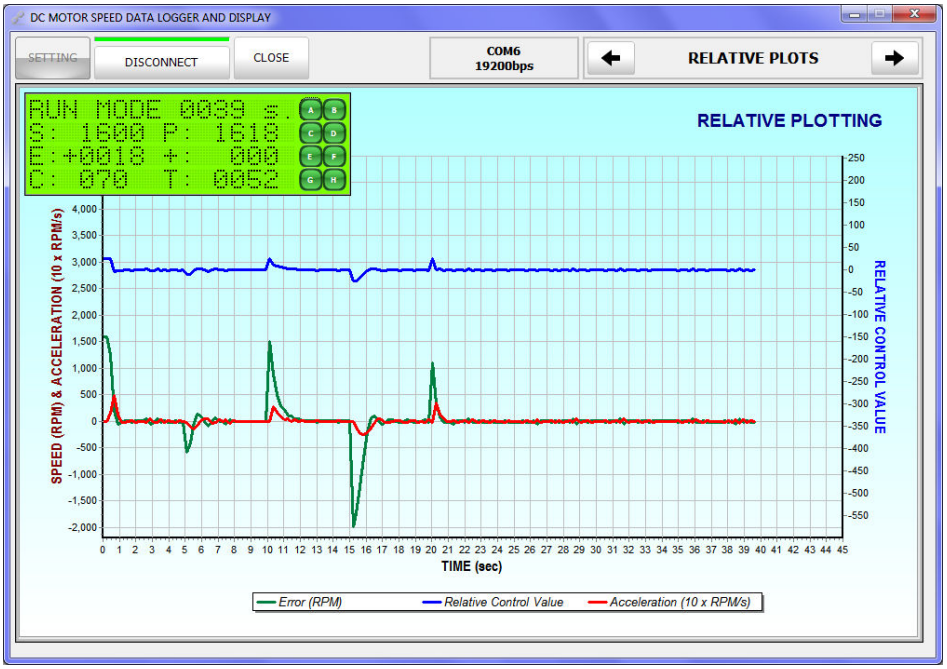


Gambar 5.24 Perangkat memasuki *Setting Mode* dilihat dari *Data Logger*.
Sumber: Hasil Penelitian.



Gambar 5.25 Ketika memasuki *Run Mode*, otomatis halaman pada *Data Logger* berpindah ke halaman ketiga, yaitu *plotting* absolut.

Sumber: Hasil Penelitian.



Gambar 5.26 Halaman keempat dari *Data Logger*, yaitu *plotting* relatif.

Sumber: Hasil Penelitian.

Gambar 5.25 menunjukkan *plotting* absolut dari hasil *data logging*, sedangkan Gambar 5.26 menunjukkan *plotting* relatifnya. Baik halaman ketiga maupun keempat bisa digunakan

untuk melakukan analisis data dengan cara mengarahkan kursor *mouse* ke area grafik. Cuplikan data bisa diperbesar (*zoom in*) dengan cara melakukan *drag* kiri ke kanan pada bagian yang akan ditampilkan, sedangkan untuk kembali ke pembesaran awal (*zoom* 100%) adalah dengan cara *drag* kanan ke kiri pada area grafik.

5.5. Parameter Fuzzy Control System untuk Pengujian

Dengan mudahnya mengubah *setting* parameter FCS dan *Data Logger* yang bekerja secara *real-time*, maka perangkat ini sangat ideal digunakan untuk menguji parameter FCS yang dibutuhkan. Nilai parameter FCS awal (*default*) telah disimpan sebelumnya pada EEPROM perangkat keras, sehingga bila sewaktu-waktu diperlukan parameter tersebut bisa dikembalikan ke nilai awalnya (*reset*).

Angular speed sequence program yang dibutuhkan adalah yang bisa menguji parameter untuk beberapa skenario yang dibutuhkan pada motor DC nantinya. Beberapa skenario di antaranya adalah:

1. *Start* motor dari berhenti menuju putaran rendah, menengah, dan tinggi.
2. Menaikkan kecepatan putaran motor dari putaran rendah ke putaran menengah dan tinggi.
3. Menaikkan kecepatan putaran motor dari putaran menengah ke putaran tinggi.
4. Menurunkan kecepatan putaran motor dari putaran tinggi ke putaran menengah atau rendah.
5. Menurunkan kecepatan putaran motor dari putaran menengah ke putaran menengah ke putaran rendah.
6. Memberikan beban pada waktu putaran tinggi, menengah, atau rendah.
7. Melepaskan beban pada waktu putaran tinggi, menengah, atau rendah.

Sesuai dengan spesifikasi kendali pada Sub Bab 4.2, yang diharapkan dari sistem kendali ini adalah sebagai berikut:

1. Menekan *overshoot*/*undershoot* hingga di bawah 20% dari pergantian *set-point*.
2. Menekan *overshoot*/*undershoot* hingga di bawah 30% dari perubahan kecepatan pada *set-point* yang sama sebagai hasil dari penambahan maupun pelepasan beban.

3. Menekan *rising time* hingga tidak melebihi dari 5 kali waktu aksi koreksi. Waktu aksi koreksi adalah 200 milidetik.
4. *Error band* ditentukan sebesar 5% dari kecepatan putar maksimal yang ditentukan. Nilai *error band* adalah 125 RPM.

Parameter *adjustment* (pengaturan) yang akan dibuat tetap antara lain:

1. Diagram derajat keanggotaan dari *error* (Gambar 5.5), karena sudah disesuaikan dengan jangkauan kecepatan putaran motor DC.
2. Diagram derajat keanggotaan dari *delta-error* (Gambar 5.6), karena sudah disesuaikan dengan jangkauan perubahan kecepatan putaran motor DC.
3. Tabulasi dari *fuzzy rules* (Tabel 5.2), karena sudah dibuat berdasarkan intuisi yang melibatkan penambahan atau pengurangan nilai kendali berdasarkan *error* dan *delta-error*.

Sedangkan parameter yang akan diatur sesuai pengamatan grafik kecepatan adalah:

1. Diagram derajat keanggotaan keluaran nilai kendali (Gambar 5.7).

5.6. Angular Speed Sequence Program untuk Skenario Pengujian

Sesuai dengan skenario pada Sub Bab 5.5, berikut ini disajikan tabel *angular speed sequence program* untuk beberapa pengujian FCS.

Tabel 5.4
Angular Speed Sequence Program untuk pengujian FCS.

S / T	0	5	10	15	20	25	30	35	40	45
S-01	500	1500	2500	1500	500	500	500	500	500	STOP
S-02	1500	2500	500	1000	2000	2500	1500	1500	1500	STOP
S-03	2500	500	2500	500	2500	2500	2500	2500	2500	STOP

Sumber: Rencana Pengujian.

Tabel 5.4 menyajikan *angular speed sequence program* untuk 3 jenis pengujian. Pengujian pertama (S-01) adalah pengujian *start* menuju ke putaran 500 RPM, kemudian mengubah putaran 1500 RPM dan 2500 RPM, kemudian turun lagi dan menguji beban di 500 RPM. Pengujian kedua (S-02) adalah *start* menuju putaran 1500 RPM kemudian naik ke 2500 RPM, kemudian turun ke 500 RPM dan naik bertahap lalu menguji beban di 1500 RPM. Pengujian ketiga (S-03) adalah *start* menuju ke 2500 RPM kemudian turun drastis ke

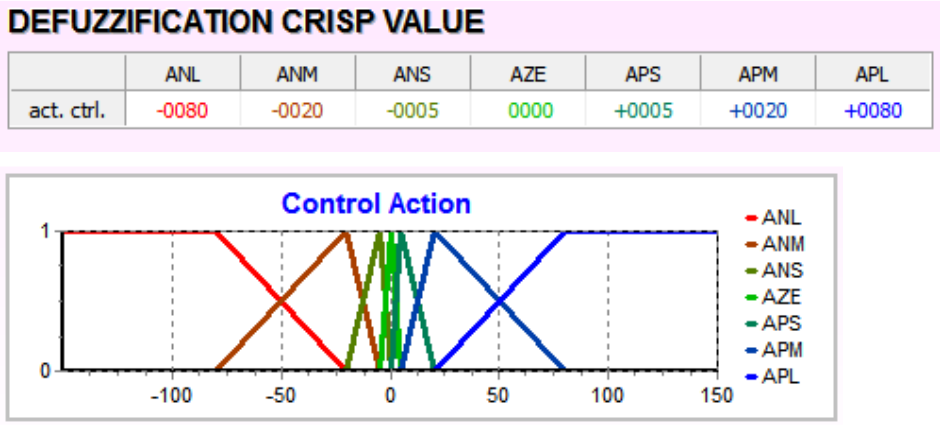
500 RPM, naik lagi ke 2500 RPM, turun ke 500 RPM, dan akhirnya menguji beban di putaran 2500 RPM. Bagian terarsir merah adalah bagian pengujian beban, di mana semua pengujian beban yang akan diberikan bersifat kualitatif karena perangkat pembebanan terjadwal dan kuantitatif belum tersedia pada alat.

5.7. Proses Pengujian

Pengujian dilakukan menggunakan parameter awal diagram derajat keanggotaan keluaran nilai kendali sesuai dengan Gambar 5.7, yaitu dengan urutan titik-titik nilai tegas APL +080, APM +020, APS +005, AZE 000, ANS -005, ANM -020, ANL -080.

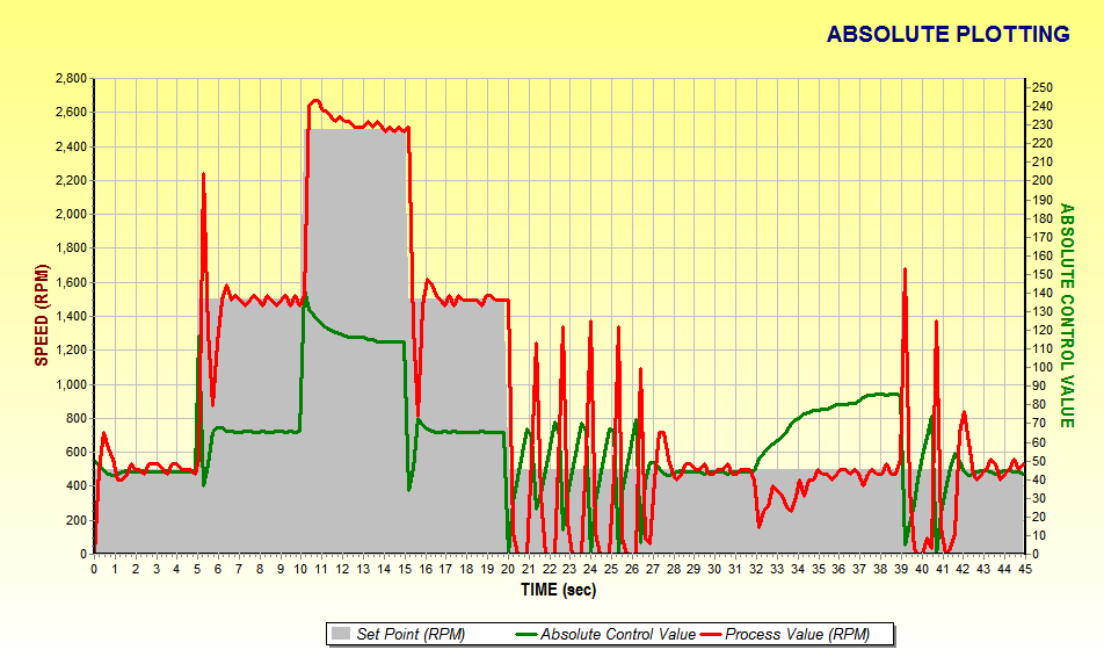
5.7.1. Pengujian S-01

Parameter awal pada semua pengujian akan menggunakan parameter *defuzzification* seperti pada Gambar 5.27.



Gambar 5.27 Parameter *defuzzification* awal.
Sumber: Proses Pengujian.

Sesuai dengan *angular speed sequence program* S-01 dan parameter *defuzzification* awal pada Gambar 5.27, maka Gambar 5.28 berikut ini adalah hasil pengujian S-01 menggunakan parameter awal.



Gambar 5.28 Hasil pengujian S-01 pertama, menggunakan parameter awal.
Sumber: Hasil Pengujian.

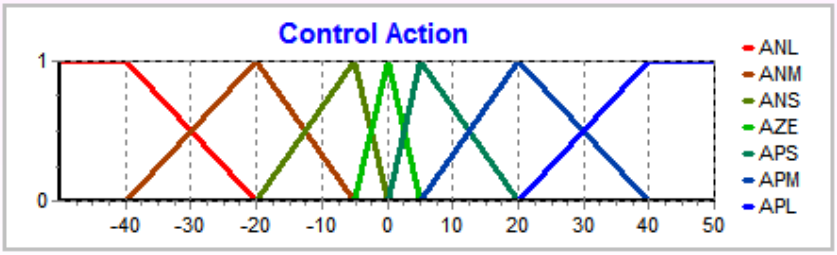
Dari Gambar 5.28 dapat dianalisis bahwa terjadi *undershoot* yang cukup besar ketika terjadi transisi kecepatan sedang (200 RPM di detik ke-0 – 40% , 700 RPM di detik ke-5 – 70%, dan 150 RPM di detik ke-10 – 15%). Kemudian terjadi *ringing* dengan amplitudo hingga 900 RPM – 36% – pada pada detik ke-20. *Rise time* yang terjadi cukup cepat, yaitu rata-rata 0.4 detik.

Pada detik ke-20, penurunan mengakibatkan ringing melampaui *error band* (125 RPM) diakibatkan oleh nilai absolut ANL yang terlalu besar (80). Oleh karena itu strategi pertama adalah menurunkan nilai absolut ANL menjadi 40. Hal yang sama juga terjadi ketika terjadi menaikkan nilai kendali yang mengakibatkan *overshoot* 70% di detik ke-5, maka nilai APL juga diturunkan menjadi 40.

Pembebanan dilakukan pada detik 32 dan dibebaskan kembali pada detik 39. Pada waktu terjadi pembebanan pada putaran rendah respon yang diberikan sistem cukup stabil, tetapi ketika terjadi pelepasan beban terjadi *overshoot* dan *undershoot* cukup besar yang disertai *ringing* sampai 3 kali siklus.

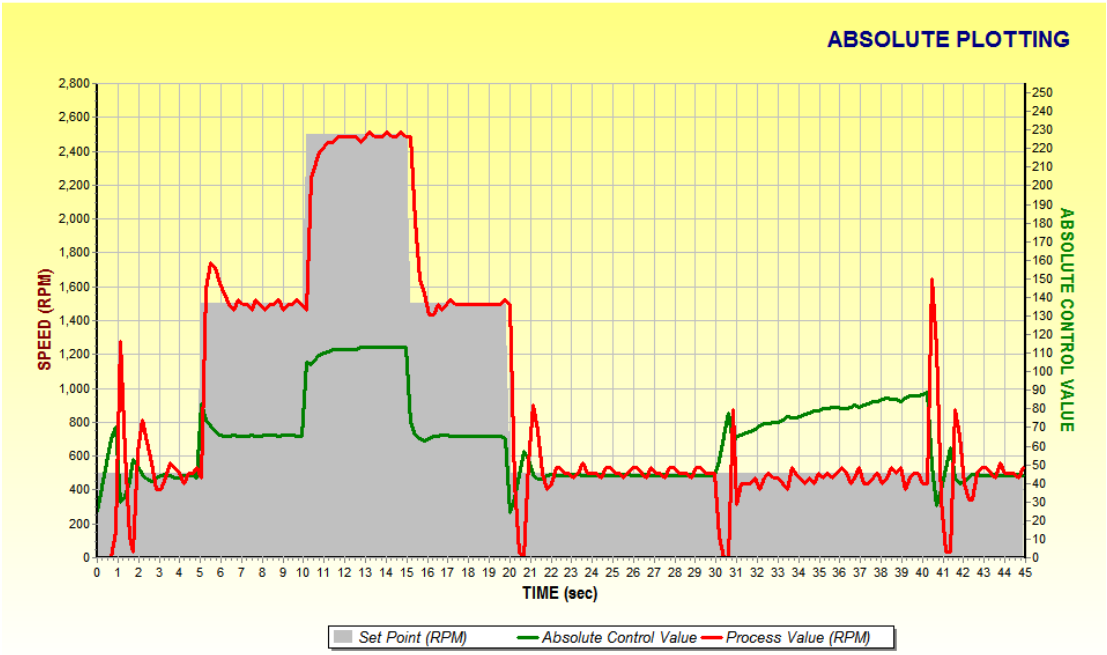
DEFUZZIFICATION CRISP VALUE

	ANL	ANM	ANS	AZE	APS	APM	APL
act. ctrl.	-0040	-0020	-0005	0000	+0005	+0020	+0040



Gambar 5.29 Pengubahan parameter *defuzzification* pertama pada S-01.
Sumber: Proses Pengujian.

Kemudian dilakukan pengujian lagi untuk S-01, yang dalam hal ini menggunakan parameter yang sudah diubah seperti pada Gambar 5.29.

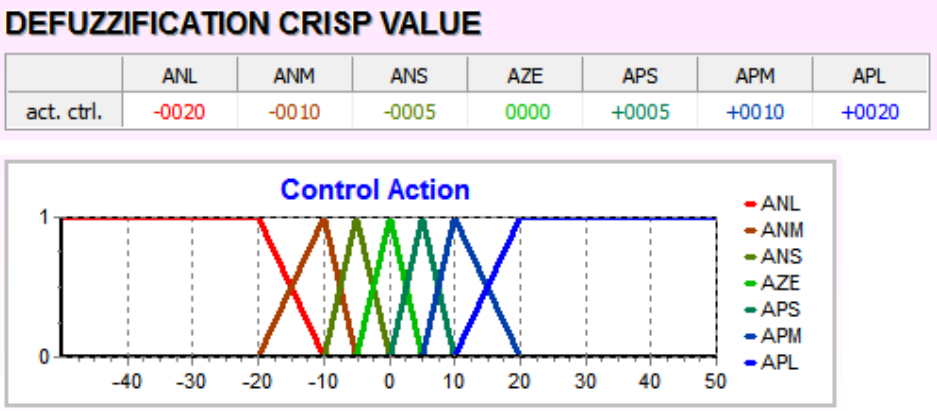


Gambar 5.30 Hasil pengujian S-01 kedua, menggunakan parameter perubahan pertama.
Sumber: Hasil Pengujian.

Gambar 5.30 menyatakan bahwa *rise time* detik ke-0 meningkat hingga 1 detik, namun *rise time* untuk detik ke-5 adalah 0.5 detik. *Rise time* pada detik ke-10 meningkat 2 detik, dan pada detik ke-15 meningkat hingga 1.2 detik. Sebaliknya *overshoot* pada detik ke-10 hilang sama sekali, pada detik ke-5 terjadi *overshoot* 250 RPM, namun pada detik ke-0 mencapai 800 RPM. *Error band* pada kasus ini cukup rendah di rata-rata 100 RPM.

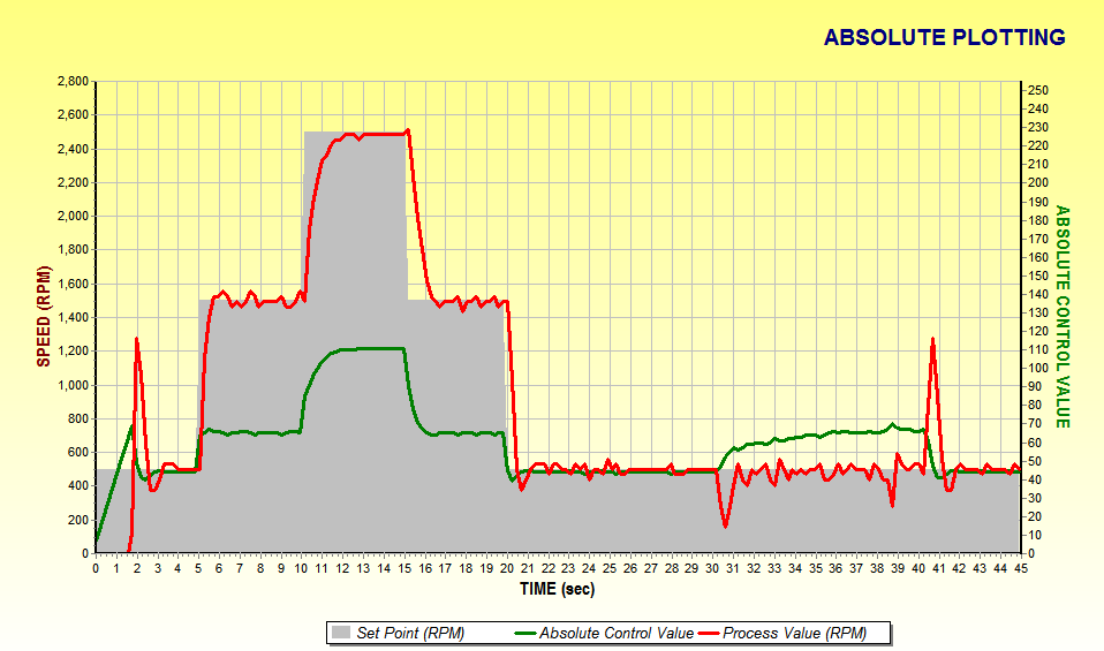
Pembebanan dilakukan pada detik 30 dan dibebaskan kembali pada detik 40. Pada waktu terjadi pembebanan pada putaran rendah respon yang diberikan sistem cukup stabil, dan ketika terjadi pelepasan beban terjadi *ringing* sampai 2 kali siklus dan tidak seliar pada parameter sebelumnya.

Sesuai dengan kasus pengujian pada perubahan parameter pertama ini, untuk menekan terjadinya *overshoot*, maka akan dicoba menurunkan lagi nilai absolut ANL dan APL menjadi 20, kemudian nilai absolut ANM dan APM diturunkan menjadi 10.



Gambar 5.31 Pengubahan parameter *defuzzification* kedua pada S-01.
Sumber: Proses Pengujian.

Dengan pengubahan pada Gambar 5.31, maka dilakukan pengujian lagi untuk *angular speed sequence program* yang sama.



Gambar 5.32 Hasil pengujian S-01 ketiga, menggunakan parameter perubahan kedua. Sumber: Hasil Pengujian.

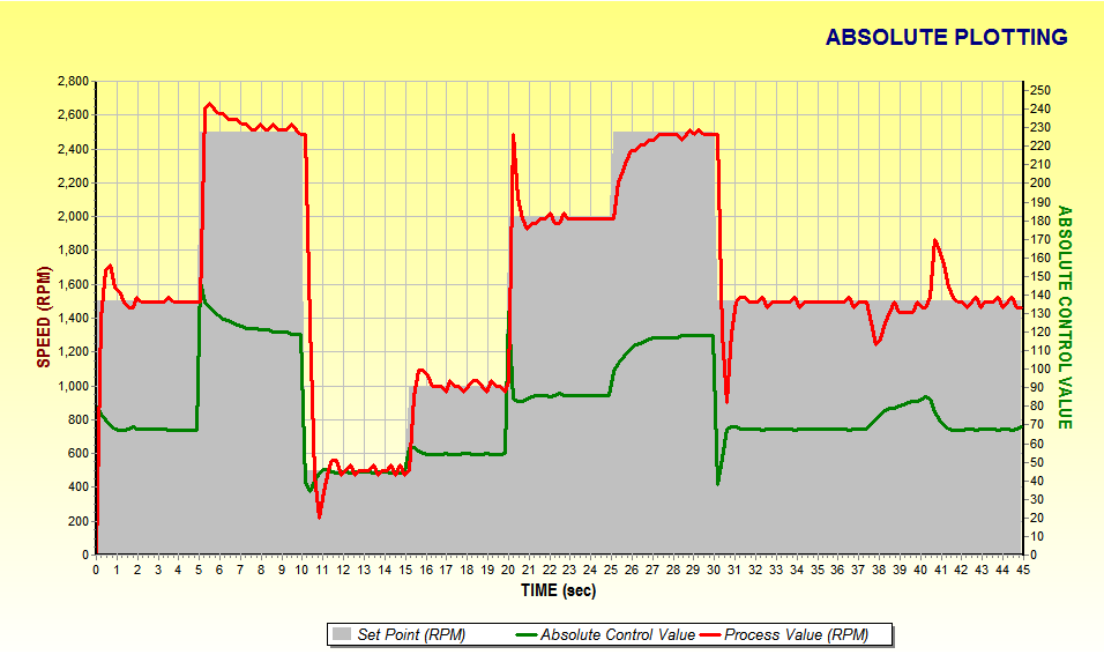
Gambar 5.32 menyatakan bahwa *rise time* detik ke-0 meningkat hingga 2 detik, namun *rise time* untuk detik ke-5 adalah 0.6 detik. *Rise time* pada detik ke-10 meningkat 2 detik, dan pada detik ke-15 meningkat hingga 1.5 detik. Sebaliknya *overshoot* pada detik ke-10 hilang sama sekali, pada detik ke-5 terjadi *overshoot* 50 RPM, namun pada detik ke-0 mencapai 800 RPM. *Error band* pada kasus ini cukup rendah di rata-rata 50 RPM.

Pembebanan dilakukan pada detik 30 dan dibebaskan kembali pada detik 40. Pada waktu terjadi pembebanan pada putaran rendah respon yang diberikan sistem cukup stabil, dan ketika terjadi pelepasan beban tidak terjadi ringing sama sekali.

Pada ketiga pengujian, dapat ditarik kesimpulan bahwa menurunkan nilai kendali untuk perubahan besar dan menengah akan menurunkan *overshoot* dan *error band*, tetapi berakibat *rise time* yang meningkat. Spesifikasi *error band* terpenuhi untuk percobaan kedua dan ketiga. Sedangkan spesifikasi *overshoot* pada putaran menengah-tinggi juga terpenuhi pada percobaan kedua dan ketiga. Khusus spesifikasi *overshoot start* ke putaran rendah tidak terpenuhi sama sekali pada ketiga percobaan, namun spesifikasi *overshoot* untuk penurunan ke putaran rendah terpenuhi pada percobaan ketiga. *Rise time* yang memenuhi spesifikasi hanya terdapat pada percobaan pertama.

5.7.2. Pengujian S-02

Sesuai dengan *angular speed sequence program* S-02 dan parameter *defuzzification* awal, maka berikut ini adalah hasil pengujian S-02 menggunakan parameter awal.



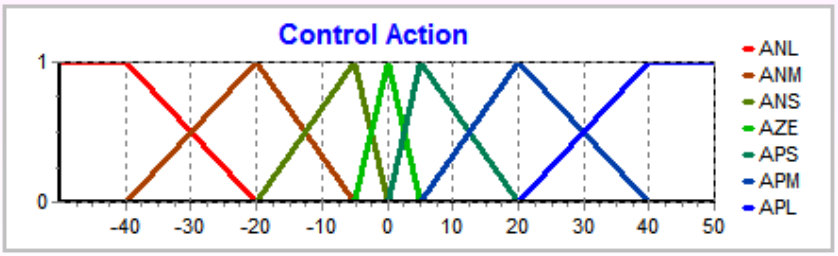
Gambar 5.33 Hasil pengujian S-02 pertama, menggunakan parameter awal.
Sumber: Hasil Pengujian.

Dari Gambar 5.33 dapat dianalisis bahwa pada detik ke-0 ketika *set-point* berada pada 1500 RPM, terjadi *overshoot* sebesar 200 RPM, kemudian pada detik ke-5 ketika *set-point* naik menjadi 2500 RPM, terjadi *overshoot* 150 RPM. *Undershoot* sebesar 300 RPM terjadi pada detik ke-10 ketika *set-point* turun menjadi putaran rendah 500 RPM. *Overshoot* cukup tinggi terjadi pada detik ke-20, ketika *set-point* naik dari 1000 RPM menuju 2000 RPM, yaitu sebesar 450 RPM. Sementara detik ke-25 tidak terjadi *overshoot*, dan detik ke 30 terjadi *undershoot* 500 RPM.

Pembebanan dilakukan pada detik 37 dan dibebaskan kembali pada detik 40. Pada waktu terjadi pembebanan pada putaran menengah respon yang diberikan sistem cukup stabil, dan ketika terjadi pelepasan beban tidak terjadi ringing sama sekali.

DEFUZZIFICATION CRISP VALUE

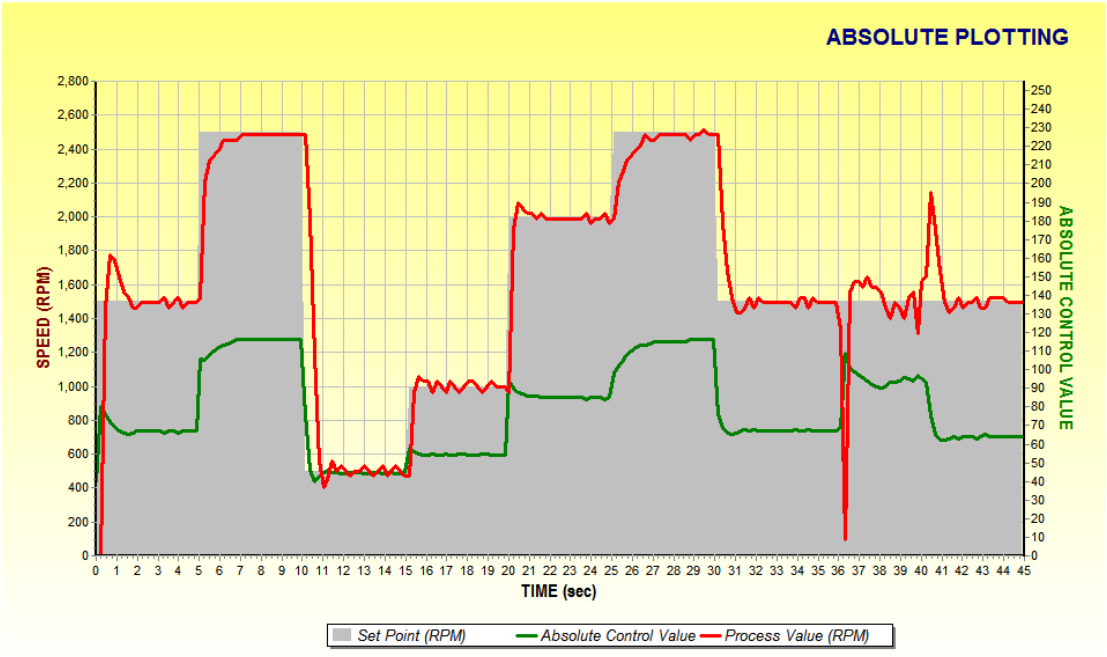
	ANL	ANM	ANS	AZE	APS	APM	APL
act. ctrl.	-0040	-0020	-0005	0000	+0005	+0020	+0040



Gambar 5.34 Pengubahan parameter *defuzzification* pertama pada S-02.
Sumber: Proses Pengujian.

Ringing yang terjadi dalam hal ini masih dalam toleransi *error-band*, yaitu kurang dari 50 RPM. Rata-rata *rise time* yang ditimbulkan cukup responsif, yaitu 0.5 detik. Parameter *defuzzification* kedua digunakan seperti halnya pada pengujian S-01, yaitu seperti ditunjukkan pada Gambar 5.34.

Kemudian dilakukan pengujian lagi untuk S-02, yang dalam hal ini menggunakan parameter yang sudah diubah, seperti ditunjukkan pada Gambar 5.35.



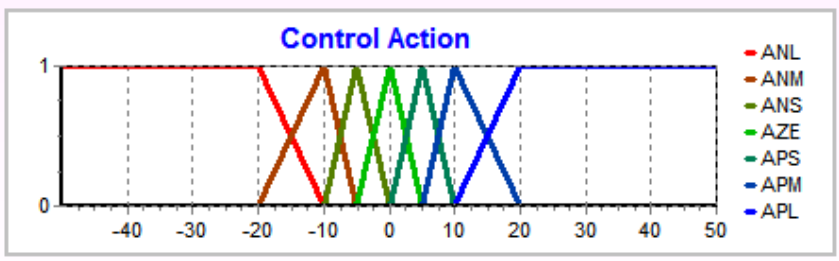
Gambar 5.35 Hasil pengujian S-02 kedua, menggunakan parameter perubahan pertama.
Sumber: Hasil Pengujian.

Gambar 5.35 menyatakan *overshoot* yang berlebihan sudah tidak lagi terjadi kecuali pada detik ke-0. *Ringing* yang terjadi juga masih dalam *error-band*, dengan nilai maksimum sekitar 50 RPM. Sedangkan *rise time* yang ditimbulkan meningkat menjadi 1 detik.

Pembebanan dilakukan pada detik 36 dan dibebaskan kembali pada detik 40. Pada waktu terjadi pembebanan pada putaran menengah respon yang diberikan sistem cukup stabil, dan ketika terjadi pelepasan beban tidak terjadi ringing sama sekali.

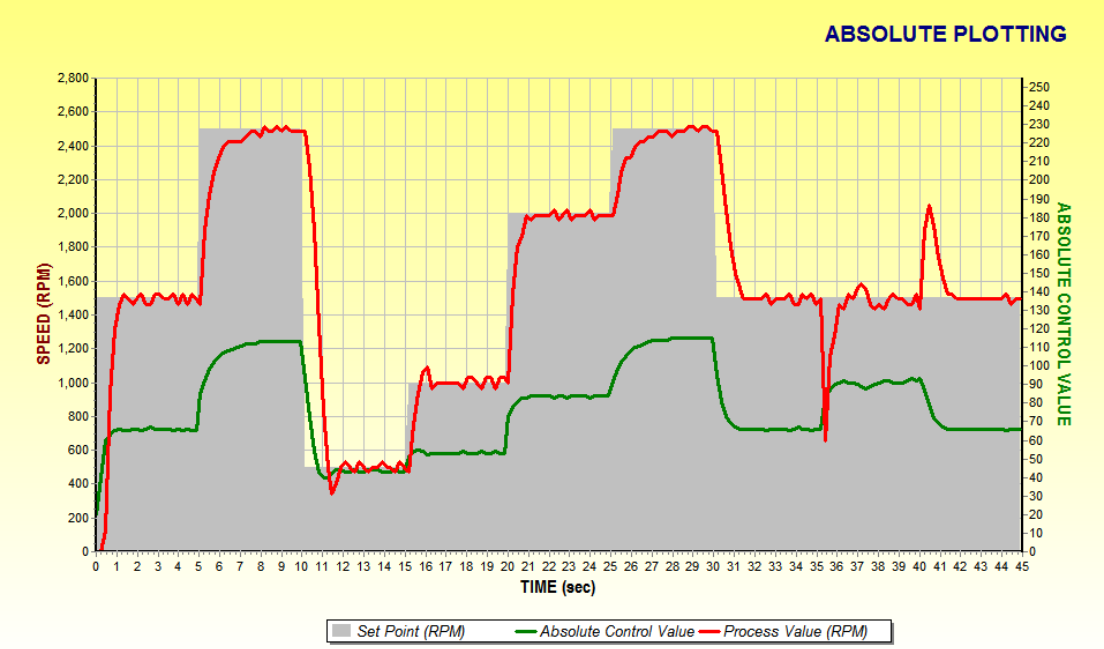
DEFUZZIFICATION CRISP VALUE

	ANL	ANM	ANS	AZE	APS	APM	APL
act. ctrl.	-0020	-0010	-0005	0000	+0005	+0010	+0020



Gambar 5.36 Pengubahan parameter *defuzzification* kedua pada S-02.
Sumber: Proses Pengujian.

Dengan pengubahan pada Gambar 5.36, maka dilakukan pengujian lagi untuk *angular speed sequence program* yang sama.



Gambar 5.37 Hasil pengujian S-02 ketiga, menggunakan parameter perubahan kedua. Sumber: Hasil Pengujian.

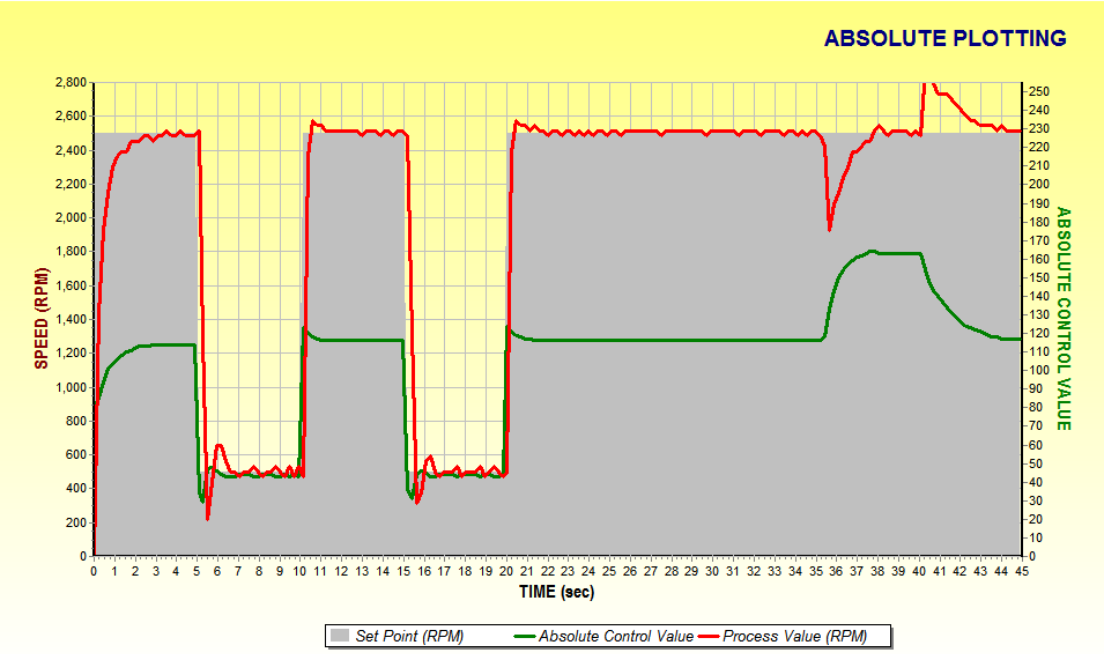
Gambar 5.37 menyatakan bahwa *rise time* rata-rata sekitar 1 detik hingga maksimum 1.5 detik. Semua *overshoot* dan *undershoot* telah berada dalam batas toleransi, dan ringing juga tidak lebih dari 50 RPM di semua bagian.

Pembebanan dilakukan pada detik 35 dan dibebaskan kembali pada detik 40. Pada waktu terjadi pembebanan pada putaran menengah respon yang diberikan sistem cukup stabil, dan ketika terjadi pelepasan beban tidak terjadi ringing sama sekali.

Seperti halnya yang terjadi pada pengujian S-01, pada ketiga pengujian S-02 dapat ditarik kesimpulan bahwa menurunkan nilai kendali untuk perubahan besar dan menengah akan menurunkan *overshoot* dan *error band*, tetapi berakibat *rise time* yang meningkat. Spesifikasi *error band* terpenuhi untuk percobaan kedua dan ketiga. Sedangkan spesifikasi *overshoot* pada putaran menengah-tinggi juga terpenuhi pada percobaan kedua dan ketiga. Khusus spesifikasi *overshoot start* ke putaran rendah tidak terpenuhi sama sekali pada ketiga percobaan, namun spesifikasi *overshoot* untuk penurunan ke putaran rendah terpenuhi pada percobaan ketiga. *Rise time* yang memenuhi spesifikasi hanya terdapat pada percobaan pertama.

5.7.3. Pengujian S-03

Sesuai dengan *program sequence* S-03 dan parameter *defuzzification* awal, maka berikut ini adalah hasil pengujian S-03 menggunakan parameter awal.



Gambar 5.38 Hasil pengujian S-03 pertama, menggunakan parameter awal.
Sumber: Hasil Pengujian.

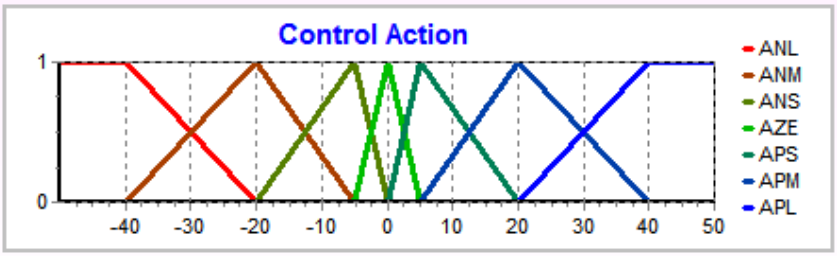
Dari Gambar 5.38 dapat dianalisis bahwa tidak terjadi *overshoot* berlebihan sama sekali, kecuali *undershoot* yang terjadi pada detik ke-5 dan detik ke-15. Ringing yang terjadi semuanya di batas toleransi. *Rise time* rata-rata terjadi pada waktu yang cukup responsif, yaitu 0.5 detik, kecuali ketika pada detik ke-0 *rise time* menjadi sekitar 2 detik.

Pembebanan dilakukan pada detik 35 dan dibebaskan kembali pada detik 40. Pada waktu terjadi pembebanan pada putaran tinggi respon yang diberikan sistem cukup stabil, dan ketika terjadi pelepasan beban tidak terjadi ringing sama sekali.

Seperti halnya pada pengujian S-01 dan S-02, dicoba menurunkan nilai absolut ANL menjadi 40, sementara nilai absolut APL juga diturunkan menjadi 40.

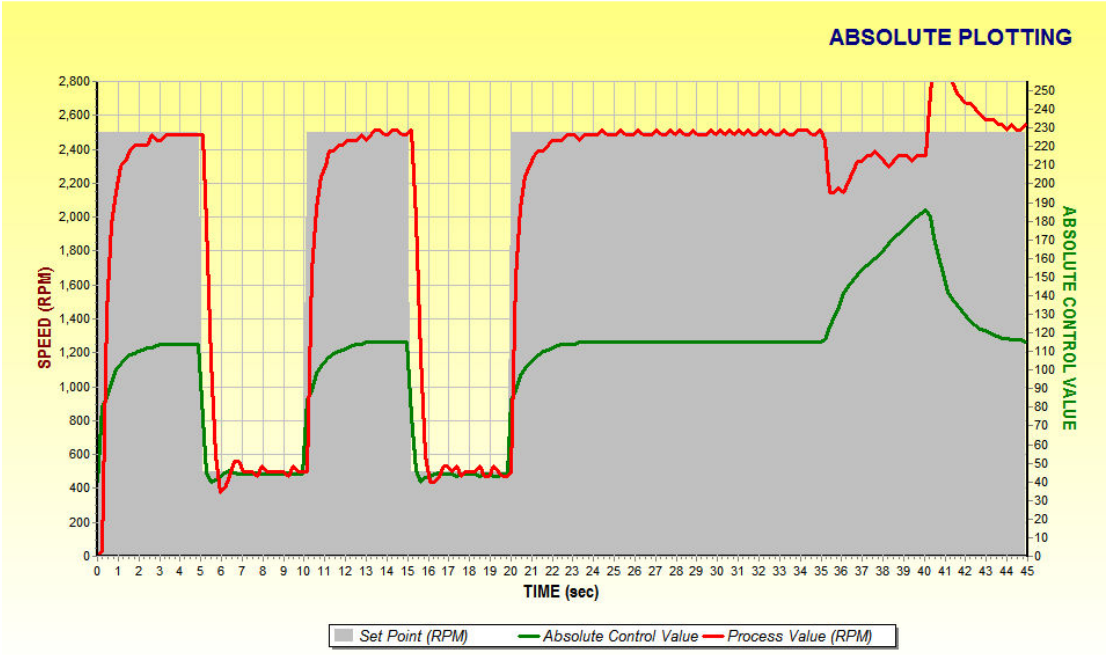
DEFUZZIFICATION CRISP VALUE

	ANL	ANM	ANS	AZE	APS	APM	APL
act. ctrl.	-0040	-0020	-0005	0000	+0005	+0020	+0040



Gambar 5.39 Pengubahan parameter *defuzzification* pertama pada S-03.
Sumber: Proses Pengujian.

Kemudian dilakukan pengujian lagi untuk S-03, yang dalam hal ini menggunakan parameter yang sudah diubah seperti pada Gambar 5.39.

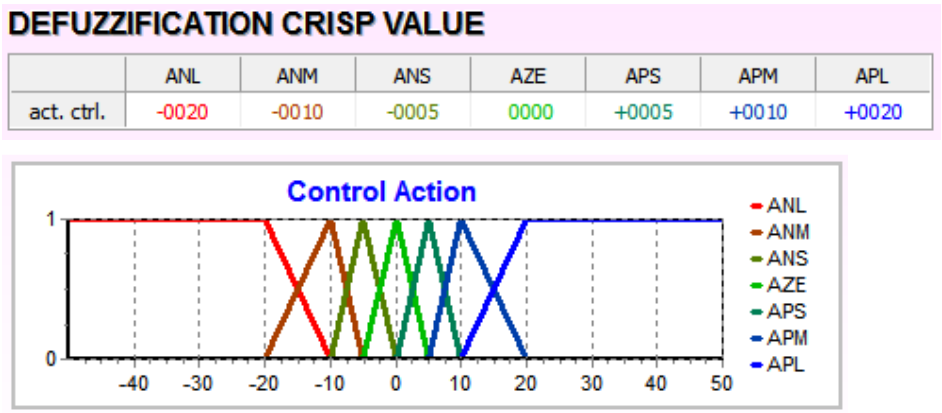


Gambar 5.40 Hasil pengujian S-03 kedua, menggunakan parameter perubahan pertama.
Sumber: Hasil Pengujian.

Gambar 5.40 menyatakan tidak ada *overshoot* dan *undershoot* yang berlebihan, semuanya berada di dalam batas toleransi. *Ringing* yang terjadi juga sama seperti pengujian pertama, tidak melebihi 50 RPM. Sementara *rise time* meningkat menjadi 2 detik maksimum. Pembebanan dilakukan pada detik 35 dan dibebaskan kembali pada detik 40.

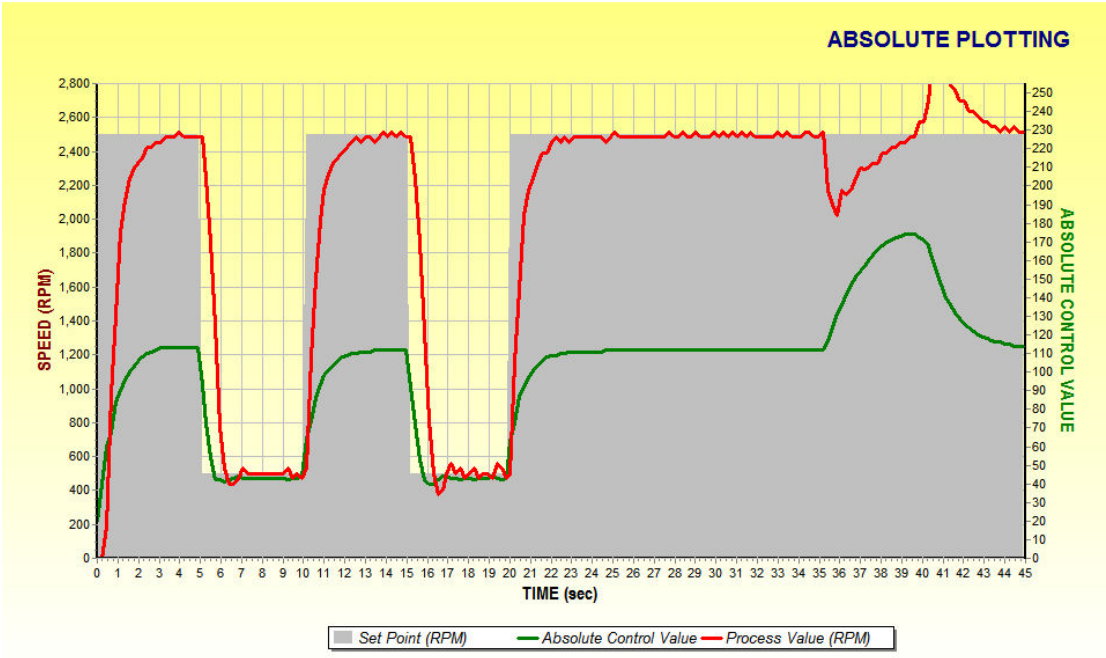
Pada waktu terjadi pembebanan pada putaran tinggi respon yang diberikan sistem cukup stabil, dan ketika terjadi pelepasan beban tidak terjadi *ringing* sama sekali.

Sesuai dengan kasus pengujian pada perubahan parameter pertama ini, akan dicoba menurunkan lagi nilai absolut ANL dan APL menjadi 20, kemudian nilai absolut ANM dan APM diturunkan menjadi 10.



Gambar 5.41 Pengubahan parameter *defuzzification* kedua pada S-03.
Sumber: Proses Pengujian.

Dengan pengubahan pada Gambar 5.41, maka dilakukan pengujian lagi untuk *angular speed sequence program* yang sama.



Gambar 5.42 Hasil pengujian S-03 ketiga, menggunakan parameter perubahan kedua.
Sumber: Hasil Pengujian.

Gambar 5.42 menyatakan bahwa tidak jauh berbeda dengan pengujian menggunakan parameter kedua, bahwa tidak terjadi *overshoot* dan *undershoot* di luar batas toleransi, sementara *ringing* yang terjadi lebih halus, sekitar 20 RPM. Sementara konsekuensi untuk *rise time* terjadi hingga maksimum 3 detik. Pembebanan dilakukan pada detik 35 dan dibebaskan kembali pada detik 40. Pada waktu terjadi pembebanan pada putaran tinggi respon yang diberikan sistem cukup stabil, dan ketika terjadi pelepasan beban tidak terjadi *ringing* sama sekali.

Pada ketiga pengujian, dapat ditarik kesimpulan bahwa menurunkan nilai kendali untuk perubahan besar dan menengah akan menurunkan *overshoot* dan *error band*, tetapi berakibat *rise time* yang meningkat. Spesifikasi *error band* terpenuhi untuk ketiga percobaan, sementara spesifikasi *overshoot* pada putaran menengah-tinggi juga terpenuhi pada ketiga percobaan. *Rise time* yang memenuhi spesifikasi hanya terdapat pada percobaan pertama.

5.8. Analisis Hasil Pengujian

Dari hasil pengujian, didapatkan analisis sebagai berikut:

1. Penggunaan parameter *defuzzification* [ANL/APL$\pm$080, ANM/APM$\pm$020, ANS/APS$\pm$005] mengakibatkan *overshoot* dan *ringing* rata-rata tidak memenuhi syarat namun mendapatkan *rise time* yang memenuhi syarat.
2. Penggunaan parameter *defuzzification* [ANL/APL$\pm$020, ANM/APM$\pm$010, ANS/APS$\pm$005] menekan *overshoot* dan *ringing* rata-rata hingga sudah memenuhi syarat namun mendapatkan *rise time* yang tidak memenuhi syarat.
3. *Overshoot* dan *ringing* merupakan kriteria yang analog dengan kehalusan respon sistem, bila dibandingkan dengan *rise time* yang analog dengan responsifnya sistem.
4. Antara *overshoot* dan *ringing* dengan *rise time* merupakan kriteria yang terkait saling berlawanan.
5. Dari total 9 pengujian yang dilakukan, hanya satu pengujian beban kualitatif yang menimbulkan *ringing* yang cukup mengganggu dan tidak sesuai dengan spesifikasi. Kedelapan pengujian (89%) menunjukkan respon sistem ini terhadap pembebanan kualitatif memenuhi spesifikasi.

6. Tidak semua aspek pengujian berhasil memenuhi spesifikasi kendali. Untuk spesifikasi pertama dan kedua, yaitu penekanan overshoot/undershoot tidak berhasil tercapai ketika spesifikasi ketiga, yaitu penekenan rising time, berhasil dilakukan. Sebaliknya, ketika penekanan rising time tidak berhasil dilakukan, maka penekanan overshoot/undershoot berhasil dilakukan.

BAB VI

KESIMPULAN DAN SARAN

6.1. Kesimpulan

Berdasarkan apapun hasil dari pengujian parameter FCS pada pengendalian motor DC berbeban, maka dalam hal ini ada beberapa kesimpulan dapat ditarik, di antaranya:

1. Pada *Fuzzy Control System* menggunakan *Mamdani Inference Engine*, pengaturan parameter tidak memerlukan pengetahuan matematika khusus dan bersifat intuitif. Performa yang didapatkan adalah sesuai dengan spesifikasi yang diharapkan di mana kriteria *overshoot* dan *ringing* berkebalikan dengan *rise time*.
2. Respon sistem terhadap pembebanan kualitatif yang bervariasi cukup bagus dan sesuai dengan spesifikasi.
3. Proses *defuzzification* bisa diatur untuk mendapatkan hasil antara minimal *rise time* dengan minimal *overshoot* dan *ringing*.

6.2. Saran

Beberapa saran yang kiranya bisa muncul antara lain adalah:

1. Diharapkan pada penelitian berikutnya aplikasi beban bisa dibuat dengan menggunakan beban kuantitatif.
2. Diharapkan pada penelitian berikutnya beban bisa dijadwalkan sebagaimana *set-point*.

DAFTAR PUSTAKA

Azadi, Sassan I., & Mosa, Nouri. 2012. *Utilizing Azadi Controller to Stabilize the Speed of a DC Motor*. Proceedings of the 2012 International Conference on Advanced Mechatronic Systems, Tokyo, Japan, p. 269-274.

Banzi, Massimo. 2011. *Getting Started with Arduino*. O'Reilly Media, Inc., California.

Bezdek, J. C., Ehrlich, R., & Full, W. 1984. *FCM: The Fuzzy C-Means Clustering Algorithm*. Computers & Geosciences, Vol. 10, No. 2, p. 191-203.

Bojadziev, George, & Bojadziev, Maria. 2005. *Fuzzy Sets, Fuzzy Logic, Applications*. World Scientific Publishing Co., Pte, Ltd., London.

Costa, Edson, & Serra, Ginalber. 2005. *Robust Takagi-Sugeno Fuzzy Control for Systems with Static Nonlinearity and Time-Varying Delay*. Federal Institute of Education, Science and Technology. Av. Getúlio Vargas, 04, Monte Castelo, CEP 65030-005, Sao Luis, MA, Brazil.

Farid, Ali Moltajaei, & Barakati, S. Masoud. 2014. *DC Motor Neuro-Fuzzy Controller Using PSO Identification*. The 22nd Iranian Conference on Electrical Engineering. p. 1162-1167.

Habiballa, H., Volna, E., Janosek, M., & Kotyrba, M. 2013. *Modelling and Reasoning with Fuzzy Logic Redundant Knowledge Bases*. In Proceedings 27th European Conference on Modelling and Simulation, ECMS, Norway, p. 361-366.

Hughes, Austin. 1990. *Electric Motors and Drives: Fundamentals, Types and Applications*. Jordan Hill, Oxford.

Jalůvka, M. 2015. *Using Fuzzy Logic for Autonomous Robot Control - 2D Mapping of Space (in Czech)*. Ostrava. Bachelor Thesis. Faculty of Science, University of Ostrava.

G., Feng. 2006. *A Survey on Analysis and Design of Model-Based Fuzzy Control Systems*. Fuzzy Systems, IEEE Transactions on. Vol. 14, No. 5. p. 676-697.

K., Tanaka, & M., Sano. 1993. *Concept of Stability Margin for Fuzzy Systems and Design of Robust Fuzzy Controllers in Fuzzy Systems*. Second IEEE International Conference on, Vol. 1. p. 29–34.

Keljik, Jeffrey J. 2013. *Electricity 4: AC/DC Motors, Controls, and Maintenance*. Delmar Cengage Learning, Clifton Park, New York.

Muslim, M. Aziz, Minggu, Desyderius, Saputra, Jeki, & Hasanah, Rini Nur. 2015. *Comparison Analysis Between Fuzzy and Fuzzified-PID Methods on Gun-Barrel Motion Control*. ARPN Journal of Engineering and Applied Sciences. Asian Research Publishing Network.

Passino, Kevin M., & Yurkovich, Stephen. 1998. *Fuzzy Control*. Addison Wesley Longman, Menlo Park, CA.

Volna, Eva, Kotyrba, Martin, & Jaluvka, Michal. 2016. *Intelligent Robot's Behavior Based on Fuzzy Control System*. Department of Informatics and Computers University of Ostrava Ostrava, Czech Republic.

Vukic, Zoran, Kuljaca, Ljubomir, Donlagic, Dali, & Tesnjak, Sejid. 2003, *Nonlinear Control System*. Marcel Dekker, Inc., New York.

Widodo, Bambang. 2008. *Simulasi Pengendali PID Fuzzy pada Sistem Pengaturan Kecepatan Motor Arus Searah*. Jurnal Sains dan Teknologi EMAS, Vol. 18, No. 3. P. 175-190.

Wierman, Mark J. 2010. *An Introduction to the Mathematics of Uncertainty Including Set Theory, Logic, Probability, Fuzzy Sets, Rough Sets, and Evidence Theory*. Creighton University. Omaha, Nebraska.

Zadeh, L. A. 1965. *Fuzzy Sets*. Information and Control, No. 8, pp. 338–353.

https://www.banggood.com diakses 10 Februari 2017.

https://www.instructables.com diakses 2 April 2017.

http://www.robotshop.com diakses 4 April 2017.

https://www.thorlabs.com diakses 4 April 2017.