PENGENDALIAN POSISI MENGGUNAKAN METODE LOGIKA FUZZY

Diajukan untuk memenuhi Tugas Akhir Diploma 3

Program D3 Instrumentasi Elektronika

Oleh :

Irma Furaida
2304210529

PROGRAM D3 FISIKA INSTRUMENTASI ELEKTRONIKA

FAKULTAS MATEMATIKA DAN ILMU PENGETAHUAN ALAM

UNIVERSITAS INDONESIA

DEPOK

2007

LEMBAR PENGESAHAN

Nama	: Irma Furaida
NPM	: 2304210529
Program Studi	: D3 Fisika
Jurusan	: Fisika Instrumentasi Elektronika
Tanggal Sidang	: 10 Juli 2007
Judul Tugas Akhir	:

PENGENDALIAN POSISI MENGGUNAKAN METODE LOGIKA FUZZY

Tugas Akhir ini telah diperiksa dan disetujui oleh :

PEMBIMBING I

(Drs. Arief Sudarmaji, M.T.)

PENGUJI I **PENGUJI II**

(**Dr. Prawito**) (**Supriyanto, S.Si**)

KATA PENGANTAR

Alhamdulillah, syukur kehadirat Allah SWT yang telah memberikan karunia-Nya yang begitu besar sehingga penulis mampu menyelesaikan tugas akhir ini dengan sebaik-baiknya. Hanya dengan rahmat dan kasih sayang-Nya, penulis memiliki motivasi lebih untuk menyelesaikan tugas akhir ini dengan segala keterbatasan yang dimiliki. Shalawat serta salam tak lupa penulis haturkan kepada Rasulullah Muhammad SAW beserta keluarga, sahabat, serta para pengikutnya yang telah istiqomah mengikuti jalan kemulian ini.

Tugas akhir yang berjudul "Pengendalian Posisi Menggunakan Metode Logika Fuzzy" ini ditujukan untuk memenuhi salah satu persyaratan untuk memperoleh gelar Diploma pada Program Diploma 3 Fisika Instrumentsi Elektronika dan Industri, Jurusan Fisika, Fakultas Matematika dan Ilmu Pengetahuan Alam, Universitas Indonesia.

Penulis hanya dapat berusaha semaksimal mungkin dengan segala keterbatasan yang ada. Namun, hal ini memberikan sebuah pelajaran berharga akan konsistensi dalam meraih cita-cita dan ketertarikan akan sebuah bidang.

Pada kesempatan ini pula penulis dengan kesungguhan dan ketulusan hati ingin menyampaikan ucapan terimakasih yang sebesar-besarnya kepada semua pihak yang telah membantu dalam menyelesaikan tugas akhir ini:

1. Allah SWT yang telah memberikan kesehatan, rizki, akal sehat dan hidayah kepada umat manusia.
2. Kedua orang tua tercinta Saifudin dan Mutmainah, adik-adikku tersayang Muhammad Iqbal Musyaffa, Khayatun Nufus, dan Falasifa Qonita, yang telah memberikan dukungan moril dan materiil, doa, cinta dan kasih sayangnya.
3. Bapak Dr. Prawito selaku ketua Program Diploma 3 Fisika Instrumentasi.

4. Bapak Drs. Arief Sudarmaji, MT selaku Dosen pembimbing yang telah memberikan petunjuk dan bimbingan dalam penyelesaian tugas akhir ini.
5. Dosen-dosen yang telah memberikan banyak ilmu selama menjalani kehidupan di kampus FMIPA UI.
6. Bapak Dwi Riyanto yang telah membantu dalam pembuatan mekanik tugas akhir ini.
7. Tante Awan, Om Yusuf, Om Hamid, dan Tante Sumi yang telah memberikan tempat tinggal dan selalu direpotkan oleh penulis selama kuliah serta bimbingan dan doanya.
8. Dobby Kurniawan yang telah memberikan doa dan motivasi agar selalu bersemangat dan tidak putus asa.
9. Seluruh Keluarga besar di Tegal yang telah memberikan doanya.
10. Sahabat-sahabatku, Tia, Riza, Pitri, Eka, Isti, Mba Nung, Nina, Eva, Adi, Gilang, Wawan, Iwonx, Isnendi, Mali, Yaniz, Lukni, Husni yang telah memberikan semangat dan doanya.
11. Kawan-kawan satu tim, Rika dan Yuni yang memberikan dukungan dan semangat.
12. Kawan-kawan seperjuangan Instrumentasi angkatan 2004 Hagi, Dewi, Arta, Sailor, Beni, Cakra, Gege, Sanggita, Tanti, Ican, Hamdan, Haeril, Rahmat, Vai, Slamet, Aryo, Abdul, Franki, Cahyo, Seno, Fajar, Usman, Yudith, Lindra, Ridho, Wahyu, Eka, Iko, Widdy, Joker, Erwin yang telah memberikan semangat, dukungan, dan pertemanannya selama ini.
13. Seluruh Rekan-rekan Instrumentasi angkatan 2002, 2003,dan 2005.
14. Seluruh Keluarga besar Departemen Fisika FMIPA UI.
15. Semua pihak yang secara tidak langsung ikut terlibat dalam pembuatan tugas akhir ini yang tidak saya sebutkan satu persatu, semoga amal baik yang telah dilakukan dibalas oleh ALLAH SWT.

Akhir kata penulis menyadari keterbatasannya, oleh karena itu kritik dan saran senantiasa diharapkan untuk dikemudian hari. Semoga ALLAH SWT senantiasa membalas dengan kebaikannya.

Depok, Juli 2007

Irma Furaida

ABSTRAK

Dalam penelitian ini aplikasi kendali logika *fuzzy* digunakan pada pengendalian posisi untuk pengaturan kecepatan motor dc. Kendali logika *fuzzy* diimplementasikan pada *Personal Computer (PC)* dan programnya dibuat dengan bahasa visual basic 6.0. Komunikasi yang digunakan adalah komunikasi parallel antara PC dengan *device* yang digunakan. Pengaturan kecepatan motor dc dilakukan dengan menggunakan metode *Pulse Width Modulation (PWM)*. Sistem logika *fuzzy* mempunyai 2 *crisp input* yaitu *error* dan perubahan *error* dan mempunyai 1 *crisp output*. Metode defuzzifikasi yang digunakan adalah *mean of maxima*. Jumlah label dari *membership function* adalah 7 label. Respon sistem ditampilkan dalam bentuk grafik antara *Setting Point (SP), Process Variable (PV),* dan *Manipulated Variable (MV)*. Hasil pengujian menunjukkan bahwa sistem telah bekerja dengan baik walaupun SP berubah-ubah.

Kata Kunci : Kendali Logika *Fuzzy*, *Fuzzy Inference*, motor dc

DAFTAR ISI

Halaman

DAFTAR GAMBAR

Halaman

DAFTAR TABEL

Halaman

BAB 1
PENDAHULUAN

1.1 Latar Belakang

Pesatnya perkembangan dunia akhir-akhir ini tidak lepas dari semakin beragamnya perangkat instrument yang digunakan dalam menunjang aktivitas produksi dunia industri. Hal ini juga turut mendorong timbulnya berbagai macam alat instrumentasi dengan fungsi yang relative sama, namun dengan kualitas dan kehandalan yang berbeda. Dalam bab ini penulis akan membahas tentang latar belakang, tujuan, batasan masalah dan deskripksi singkat dari modul pengendali posisi yang penulis buat.

Dengan kemajuan teknologi dan perkembangan ilmu pengetahuan yang semakin pesat, maka tuntutan akan kebutuhan peralatan dan perlengkapan yang lebih cepat, sederhana, akurat dan ekonomis semakin meningkat yang kemudian menghasilkan perkembangan baru dalam perencanaan dan pemakaian. Semakin luas pula kebutuhan yang menuntut kita untuk meningkatkan kualitas ilmu pengetahuan khususnya dalam bidang pengendalian. Sistem kendali merupakan salah satu alat instrumentasi yang sangat penting dalam dunia industri. Sistem ini merupakan sebuah sistem yang terdiri atas salah satu atau beberapa peralatan yang berfungsi untuk mengendalikan sistem lain yang berhubungan dengan sebuah proses.

Sistem pengendalian telah banyak aplikasinya di bidang industri ataupun di bidang elektronika. Seperti pada industri tekstil dan perancangan robot sistem pengendalian khususnya pengendalian posisi sangatlah dibutuhkan. Karena semakin banyak kebutuhan yang menggunakan sistem pengendali maka dibutuhkan suatu sistem pengendali yang stabil, akurat, dan mempunyai kesalahan yang relatif kecil. Pengendalian posisi tersebut memanfaatkan metode *fuzzy logic*, dimana penerapan aplikasi program menggunakan bahasa pemrograman *Visual Basic* (VB). Bila suatu sistem tersebut tidak berjalan sesuai dengan yang diharapkan, maka sistem pengendali ini dapat mengendalikan proses tersebut sehingga sistem dapat berjalan kembali sesuai dengan yang diharapkan.

Dari penjelasan diatas, penulis akan membuat suatu sistem pengendalian posisi dalam skala laboratorium. Dimana sistem pengendali posisi dibuat dengan menggunakan sistem digital sehingga kita cukup mengaturnya lewat komputer. Dan diharapkan sistem ini akan memperkecil kesulitan yang mungkin terjadi pada saat pengendalian posisi dan bekerja secara efektif dan efisien.

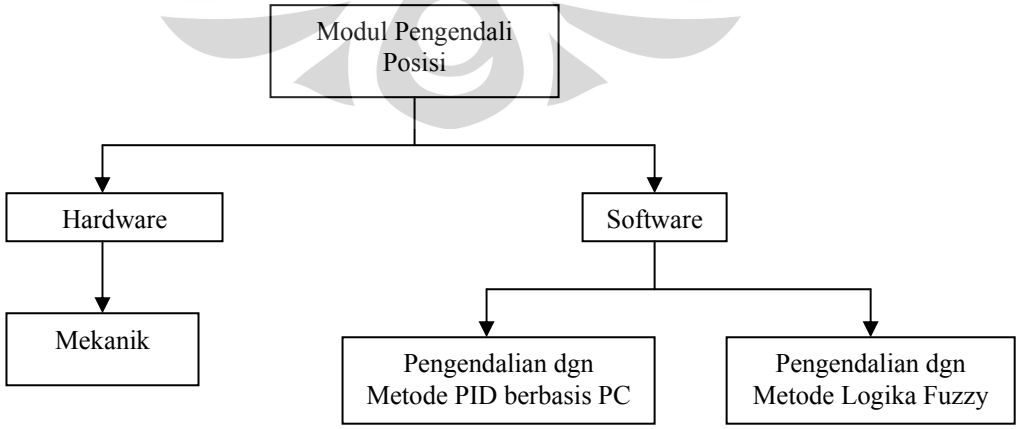
1.2 Tujuan Penelitian

Adapun tujuan yang ingin dicapai dari penelitian ini antara lain adalah sebagai berikut :

- Untuk membuat suatu modul pengendali posisi yang berskala laboratorium dengan menggunakan metode Logika Fuzzy.

1.3 Batasan Masalah

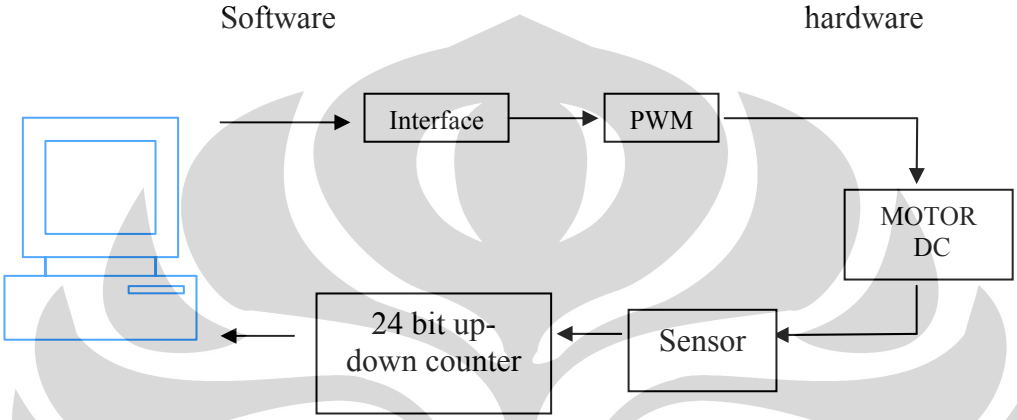
Pada modul ini ada dua komponen pendukung yaitu hardware dan software. Software pada modul ini digunakan untuk mengendalikan posisi dengan menggunakan komputer sebagai alat bantu. Ada 2 metode yang digunakan untuk mengendalikan modul ini yaitu Metode dengan menggunakan PID berbasis PC dan Metode dengan menggunakan Logika Fuzzy. Untuk itu, penulis akan membahas lebih dalam tentang pengendalian posisi dengan metode Logika Fuzzy. Untuk menghubungkan antara software dan hardware kita menggunakan rangkaian interface dan port paralel sebagai komunikasi. Untuk lebih jelas kita dapat lihat pada gambar dibawah ini :



Gambar 1.1 Blok Diagram Modul Pengendalian Posisi

1.4 Deskripsi Singkat

Sistem ini bekerja jika komputer mengirim suatu sinyal PWM (Pulse Width Modulation) pada actuator, yang pada modul ini berupa motor DC, maka motor akan bergerak dan menggerakan drat pada as sehingga konversi akan berputar ke linier. Saat drat berputar maka sensor akan membaca putaran dari drat dalam bentuk pulsa-pulsa listrik.



Gambar 1.2. Blok Pengendalian posisi

Data yang berupa pulsa-pulsa akan masuk ke rangkaian interface. Pada sistem ini rangkaian interface akan menerima data dari komputer dan mengirimnya ke rangkaian PWM sehingga kecepatan motor dapat diatur.

Sensor, motor DC, dan drat diletakkan pada satu as sehingga saat motor bergerak maka drat juga akan berputar. Saat drat berputar maka sensor akan menghitung pulsa yang dihasilkan oleh drat. Dari pulsa-pulsa ini kita akan melihat perpindahan posisi yang dihasilkan oleh motor.

1.5 Metode Penulisan

Dalam penulisan penelitian ini, penulis membuat urutan cara penulisan yang secara garis besar diuraikan sebagai berikut :

1.4.1 Bab 1.Pendahuluan

Pada bab ini penulis akan membahas tentang latar belakang yang mendasari pembuatan dari modul ini. Lalu tujuan dari penelitian ini serta batasan masalah, deskripsi singkat tentang modul ini dan metode penulisan.

1.4.2 Bab 2. Teori Dasar

Pada bab ini akan membahas tentang teori dasar yang sangat diperlukan agar pembaca lebih mengerti tentang modul ini. Teori yang ditulis antara lain Motor DC, PWM, Sensor dan Logika Fuzzy.

1.4.3 Bab 3. Cara kerja rangkaian dan listing program

Bab ini akan menjelaskan cara kerja dari rangkaian interface serta listing program yang digunakan pada modul pengendali posisi ini.

1.4.4 Bab 4. Hasil dan Analisa

Bab ini akan membahas hasil yang diperoleh dari pengujian yang dilakukan pada modul ini yang dapat digunakan sebagai acuan dari modul ini. Serta menganalisa modul ini agar lebih mengerti cara kerja yang sebenarnya.

1.4.5 Bab 5. Kesimpulan dan saran

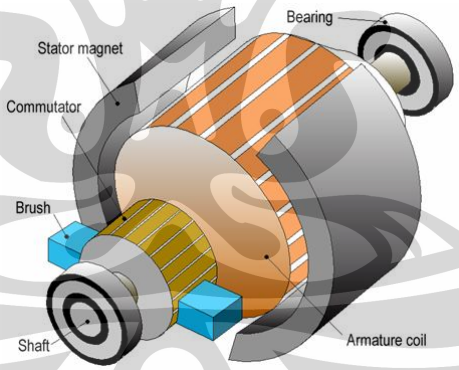
Bab ini menampilkan saran dan kesimpulan yang diperoleh pada penelitian modul ini. Sehingga dapat diketahui sesuai dengan tujuan yang diinginkan atau tidak.

BAB 2
TEORI DASAR

Perancangan modul pengendali posisi dengan menggunakan metode Logika Fuzzy ini sebagaimana tercantum dalam tujuan penelitian, ada beberapa pemahaman dasar yang sangat perlu dipahami terlebih dahulu. Beberapa pemahaman dasar tersebut antara lain : Motor DC, PWM (*Pulse Width Modulation*), Sensor, dan Logika Fuzzy.

2.1. Motor DC

Motor arus searah (DC) berfungsi mengubah energi listrik menjadi energi mekanik, dalam hal ini energi listrik yang diubah adalah listrik arus searah atau DC (Direct current). Prinsip keja motor arus searah berdasarkan pada penghantar tersebut akan mengalami gaya. Gaya tersebut menimbulkan torsi yang akan menghasilkan rotasi mekanik, sehingga motor akan berputar.



Gambar 2.1. Motor DC

Pada Motor DC didesain untuk memanfaatkan gaya magnet agar menghasilkan gerak berputar yang kontinyu dan disusun oleh komponen-komponen :

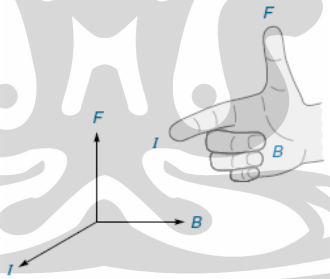
- Stator magnet digunakan sebagai penghasil gaya magnet permanen. Dibentuk menyesuaikan bentuk housing motor dengan setengah lingkaran atau satu lingkaran penuh.

- Armature coil digunakan sebagai kumpulan penghantar (konduktor) yang digulung sedemikian rupa hingga dapat menghasilkan torsi yang optimum. Duduk pada yoke yang dipasang permanen terhadap shaft.
- Commutator digunakan sebagai jalur masuk dan keluarnya arus listrik pada armature coil. Terbuat dari tembaga yang tersekat antar segmen oleh bahan isolator seperti mika.
- Brush digunakan sebagai medium penyalur arus listrik dari sumber listrik ke commutator. Terbuat dari tembaga atau carbon dan dedesain untuk lebih mudah aus dibandingkan dengan commutator.
- Bearing digunakan sebagai penyangga shaft pada housing motor.

Gaya yang dihasilkan motor dc tergantung pada :

a. Kekuatan pada medan magnet.
b. Besarnya arus yang mengalir pada penghantar.
c. Panjang kawat penghantar yang berada dalam medan magnet.

Apabila panjang kumparan rotor L dialiri arus listrik sebesar I dan terletak diantara kutub magnet utara dan selatan dengan kerapatan fluks sebesar B, maka kumparan rotor tersebut mendapat gaya F sebesar :



Gambar 2.2 Gaya Medan Magnet

$$F = B * I * L \qquad (2.1)$$

Keterangan :

F = Gaya Lorentz (Newton)

B = Kerapatan Fluks Magnet (Weber / m^2)

I = Arus Listrik (Ampere)

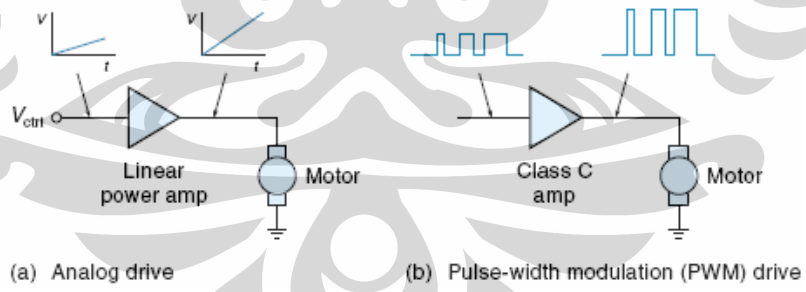
L = Panjang sisi kumparan rotor (m)

2.2 PWM (Pulse Width Modulation)

PWM adalah suatu teknik yang digunakan untuk mengontrol kerja dari suatu alat atau menghasilkan suatu tegangan DC yang variabel. Rangkaian PWM adalah rangkaian yang lebar pulsa tegangan keluarannya dapat diatur atau dimodulasi oleh sebuah sinyal tegangan modulasi. Disamping itu kita dapat menghasilkan suatu sinyal PWM dengan menentukan frekuensi dan waktu dari variabel ON dan OFF. Pemodulasian sinyal yang beragam dapat menghasilkan duty cycle yang diinginkan.

Dalam sistem PWM, power di suplai ke motor dalam bentuk pulsa dc pada tegangan tertentu. Lebar pulsa bervariasi untuk mengontrol kecepatan motor. Pulsa yang lebih lebar, tegangan dc rata-rata yang lebih tinggi diperbolehkan untuk motor. Frekuensi dari pulsa yang cukup tinggi dapat menginduktansi rata-rata motor sehingga dapat menggerakkan motor secara baik. Sistem ini memiliki 2 keuntungan bila dibandingkan dengan analog drive yaitu :

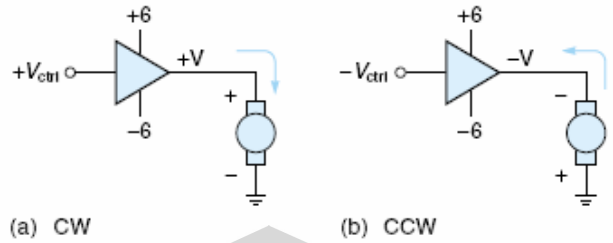
1. Power amplifier dapat menjadi tipe kelas C yang efisien.
2. DAC tidak dibutuhkan karena amplifier baik On atau Off dapat dikendalikan secara langsung dengan sinyal digital.



Gambar 2.3. Cara Pengendalian Motor

Untuk mengubah arah rotasi dari PWM motor, polaritas dari tegangan yang digunakan adalah berlawanan. Satu cara untuk dapat melakukannya sehingga sebuah motor driver mampu mengeluarkan tegangan positif dan tegangan negatif. Ketika tegangannya positif dan ground maka motor akan bergerak searah jarum jam (CW). Ketika tegangannya negatif dan ground maka

polaritas tegangan pada terminal motor berlawanan sehingga motor bergerak berlawanan arah jarum jam (CCW).



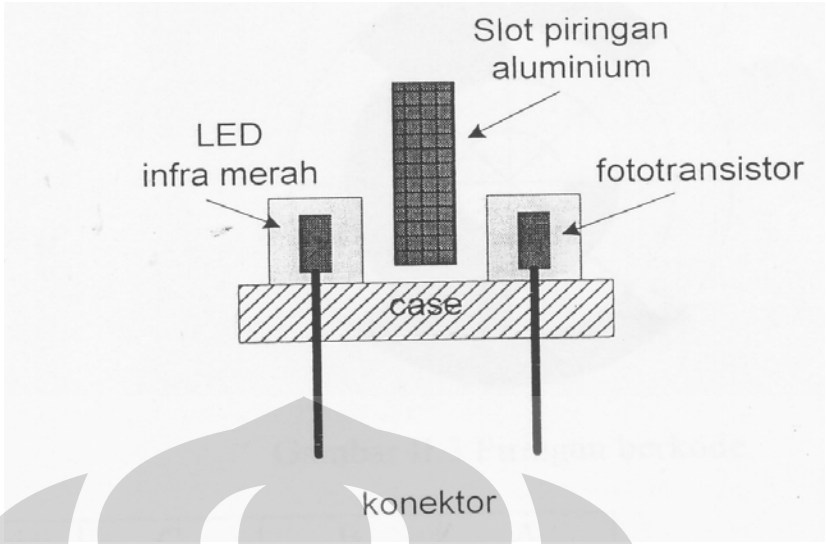
Gambar 2.4. Arah Putaran Motor DC

2.3. Sensor

Sensor merupakan besaran yang dapat mengubah besaran fisis menjadi besaran mekanis. Snsor juga dapat dikatakan suatu alat yang digunakan untuk mengubah suatu energi (gerak, panas dan kimia)menjadi suatu energi listrik. Sensor dapat terjadi karena adanya keinginan manusia yang ingin melakukan segala sesuatu dengan cepat, efisien, mudah dan praktis. Sensor mekanis adalah sensor yang mendeteksi perubahan gerak mekanis, seperti perpindahan atau pergeseran atau posisi, gerak lurus dan melingkar, tekanan, aliran, level dsb. Pada tugas akhir ini penulis menggunakan sensor shaft encoder.

2.3.1 Shaft Encoder/ Sensor Putaran

Shaft encoder merupakan suatu sensor yang digunakan untuk menghitung berapa banyak motor melakukan cacah/ menghitung dalam satu putaran atau dengan kata lain mengubah putaran mekanis menjadi data digital. Pengcounteran yang dimaksudkan adalah berasal dari piringan yang ada pada motor .Shaft encoder ini pada umumnya digunakan untuk menghitung berapa banyak putaran dalam persekian menit (Rpm).



Gambar 2.5. Sensor shaft encoder

Seperti pada gambar di atas slot piringan alumunium terhubung dengan lengan motor sehingga ketika motor berputar maka slot piringan tersebut akan berputar pula.

Dari sensor shaft encoder didapatkan data-data berbentuk pulsa dan selanjutnya pulsa-pulsa tersebut diolah menjadi up-down counter. Up-down counter disini menandakan adanya pergeseran posisi dari penunjuk perpindahan dari konstruksi mekanik. Jika penunjuk bergerak maju, maka rangkaian akan mencacah naik (up counter). Sedangkan apabila penunjuk bergerak mundur rangkaian akan mencacah turun (down counter).

2.4. Logika Fuzzy

Logika dalam percakapan sehari-hari berarti ”menurut akal”. Tetapi logika sebagai istilah berarti suatu metoda atau teknik yang diciptakan untuk meneliti ketepatan penalaran. Penalaran adalah suatu bentuk pemikiran. Penalaran membedakan cara perhitungan aritmatika. Sehingga logika sendiri berarti pengambilan kesimpulan yang tepat, menguraikan hubungan dalil-dalil yang menyangkut sebab-akibat, penyangkalan, perbedaan, perubahan dan lain sebagainya.

Fuzzy berarti kabur, tidak tegas. Tetapi disini diartikan sebagai nilai kebenaran yang tidak tegas. Nilai kebenaran yang tegas adalah benar atau salah, nol atau satu. Nilai kebenaran yang tidak tegas adalah sembarang nilai diantara benar dan salah, antara nol dan satu.

Terdapat dua pengertian mengenai *fuzzy logic* yaitu :

a. *Fuzzy Logic* adalah ilmu logika yang memakai proposisi, deklarasi atau himpunan *fuzzy*.
b. *Fuzzy Logic* adalah alat yang memakai ilmu logika seperti no. 1 diatas.

Ilmu *fuzzy logic* diperlukan untuk mengerti cara kerja alat yang memakai ilmu *fuzzy logic* itu. Ilmu tentang logika *fuzzy* dan himpunan *fuzzy* adalah ilmu yang jelas, pasti, dapat dimanipulasi dengan konsisten intuitif serta bermanfaat.

Sehingga dapat ditarik kesimpulan bahwa *fuzzy logic* berarti cara manarik kesimpulan dengan penalaran dari proposisi yang bernilai kebenaran tidak tegas (*fuzzy*) atau dapat berarti proses pengambilan keputusan berdasarkan dalil-dalil dan berdasarkan masukkan yang dikelompokkan dan dengan tingkat keanggotaan kelompok yang berubah. *Logika fuzzy* juga berarti alat yang memakai ilmu *fuzzy logic*. Biasanya sebagai ilmu kita sebut *fuzzy logic* dan alat yang menerapkan *fuzzy logic* disebut sistem *fuzzy* atau sistem logika *fuzzy*. Dimana sistem *fuzzy* memetakan *input* menjadi *output*.

Terdapat beberapa prinsip pokok sistem *fuzzy* yaitu antara lain :

a. *Input* berbentuk proposisi, deklarasi atau himpunan yang bersifat *fuzzy* yaitu mempunyai nilai kebenaran atau derajat keanggotaan yang tidak hanya nol atau satu, tetapi sembarang nilai dari nol sampai satu.
b. *Input* diolah di dalam alat yang di program dengan prinsip-prinsip ilmu logika *fuzzy*.
c. *Output* juga berbentuk proposisi, deklarasi atau himpunan yang bersifat *fuzzy*.
d. Jika *input* belum berbentuk *fuzzy*, mereka diubah dulu, sebaiknya jika diinginkan *output* bukan *fuzzy* seperti nilai tegas, maka mereka diubah kembali.

Beberapa aplikasi *fuzzy logic* antara lain :

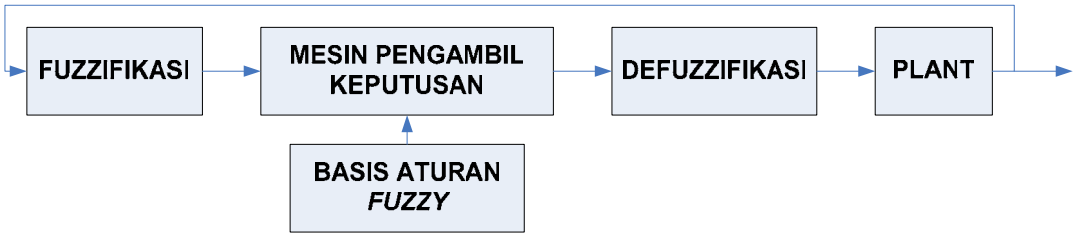
- Kontrol.

- Pendukung keputusan.
- Pengenalan citra.
- Bidang Psikologi.

Logika *fuzzy* merupakan suatu logika yang lebih dekat dengan cara berpikir manusia jika dibandingkan dengan logika klasik (*crisp*). Hal ini menjadi alasan penggunaan logika *fuzzy* dalam sistem kendali, karena pada dasarnya logika *fuzzy* menyediakan cara untuk menyatakan operasi dan aturan kendali pada suatu sistem dalam bentuk kata-kata. Hal tersebut menjadi sangat penting, karena manusia berpikir, mengemukakan ide dan mengambil keputusan dalam bentuk kata-kata. Dengan demikian pengendali logika *fuzzy* menggunakan strategi seorang operator yang ahli dalam mengendalikan suatu sistem, terutama sistem yang kompleks dan memiliki derajat ketidakpastian yang tinggi.

Pengendali logika *fuzzy* adalah algoritma yang mampu mentransformasikan sistem kendali linguistik menjadi strategi otomatis. Bagian terpenting dalam algoritma tersebut adalah kumpulan aturan kendali linguistik yang ditata dalam konsep relasi *fuzzy* dan aturan inferensi komposisional. Pada algoritma ini terjadi pergeseran pusat perhatian dari model sistem matematis menjadi model logika. Konfigurasi dasar pengendali logika *fuzzy* terdiri atas empat bagian utama yaitu :

1. Fuzzifikasi adalah bagian yang mentransformasikan data pengukuran (*Process Variable*) berupa bilangan *crisp* menjadi suatu nilai linguistik.
2. Basis Aturan *Fuzzy* (*Fuzzy Rule Base*) berisi pengetahuan mengenai operasi dari suatu proses.
3. Mesin Pengambil Keputusan (*Inference Engine*) adalah bagian terpenting dari pengendali logika *fuzzy* yang mempunyai kemampuan untuk mensimulasikan pengambil keputusan manusia dengan cara menggunakan penalaran aproksimasi untuk mendapatkan strategi kontrol yang diinginkan.
4. Defuzzifikasi berfungsi untuk mengubah keputusan atau aksi kontrol *fuzzy* yang dihasilkan *Inference Engine* menjadi *non-fuzzy* (*crisp*).



Gambar 2.6. Struktur dasar pengendali logika *fuzzy*.

2.4.1. Fuzzifikasi

Pada proses pengendalian data yang diamati dari sensor (besaran *crisp*) harus diubah menjadi suatu nilai linguistik agar dapat diolah lebih lanjut oleh mesin pengambil keputusan (*inference engine*). Proses transformasi ini disebut proses fuzzifikasi.

Fuzzifikasi merupakan proses pemetaan dari masukan yang diobservasi ke himpunan *fuzzy* dalam variasi himpunan semesta data masukan. Data masukan berupa data *crisp* dan fuzzifikasi memetakan skala masukan data *crisp* ke nilai himpunan *fuzzy* yang berhubungan dengan masukan sistem. Unit fuzzifikasi melakukan proses untuk mengubah nilai masukan (*crisp*) menjadi derajat keanggotaan pada suatu fungsi keanggotaan (*membership function*). Data masukan selalu berupa nilai numerik yang dibatasi oleh interval dari variabel *input* (dengan jangkauan tertentu) dan keluarannya berupa derajat *fuzzy* dari suatu fungsi keanggotaan (selalu dalam interval 0 dan 1). Data yang dipetakan dalam himpunan *fuzzy* diubah kedalam variabel linguistik. Proses fuzzifikasi diekspresikan sebagai :

$$x = fuzzifier(x_0)$$

Dimana :

x_0 adalah nilai vektor *crisp* dari variabel masukan proses.

x adalah vektor himpunan *fuzzy*.

fuzzifier adalah operator fuzzifikasi yang memetakan data *crisp* ke himpunan *fuzzy*.

Fungsi fuzzifikasi adalah :

- Pengukuran nilai variabel masukan.
- Pemetaan skala yang mengubah jangkauan nilai variabel nilai masukan ke dalam himpunan semesta yang bersesuaian.
- Fuzzifikasi yang mengubah data masukan ke dalam nilai linguistik yang dapat dipandang sebagai label dari himpunan *fuzzy*.

Terdapat beberapa metode untuk mendefinisikan masukan *crisp* ke dalam himpunan *fuzzy*, tergantung tipe yang digunakan dalam proses. Tipe sinyal masukan *crisp* bisa berupa sinyal diskrit atau sinyal kontinu. Selanjutnya masukan harus dibagi menjadi beberapa variabel dengan jangkauan tertentu. Banyaknya variabel akan menentukan kinerja sistem dan jumlah aturan kendali yang dapat diturunkan. Misalnya sistem yang memiliki dua masukan dengan masing-masing dibagi menjadi 5 variabel, maka jumlah aturan maksimum yang dapat diturunkan adalah 5 x 5 = 25 aturan.

Suatu cara fuzzifikasi yang sederhana dan sering digunakan pada pengendali logika *fuzzy* adalah dengan mengubah nilai *crisp* x_0 menjadi sebuah *fuzzy singleton* A dimana :

$$\mu(x) = 1, \text{ jika } x = x_0$$

$$\mu(x) = 0, \text{ lainnya}$$

Dengan demikian, untuk suatu nilai $x_i(t)$ pada saat t, nilai tersebut dipetakan (ditransformasikan) ke variabel *fuzzy* dengan derajat keanggotaan dan ke variabel *fuzzy* dengan derajat keanggotaan dan seterusnya terhadap variabel *fuzzy* yang lainnya.

2.4.2. Basis aturan *fuzzy*

Basis aturan dibentuk dari kumpulan pernyataan linguistik berdasarkan pengetahuan pakar. Pengetahuan pakar biasanya berbentuk aturan ***IF-THEN***, yang dapat diimplementasikan ke dalam kalimat kondisional *fuzzy*.

Pemilihan variabel proses dan variabel kendali pada aturan kendali *fuzzy* menetukan juga bentuk struktur aturan yang akan digunakan. Umumnya variabel

yang digunakan mencakup keadaan *derivative error* (DE), integral *error* dan bentuk-bentuk keadaan sejenisnya. Ada beberapa cara menurunkan aturan kendali *fuzzy*, diantaranya berdasarkan pengalaman pakar dan pengetahuan sistem kendali serta sistem pembelajaran (*adaptive*). Secara garis besar terdapat dua tipe aturan kendali *fuzzy* yaitu :

1. *State Evaluation Fuzzy Control Rules*

   Merupakan tipe yang sering digunakan dalam sistem kendali *fuzzy*. Sebagai contoh, jika diketahui tiga masukan dan satu keluaran maka bentuk aturannya :

   $$rule\ i : IF\ a\ is\ A_1;\ b\ is\ B_i;\ and\ c\ is\ C_i\ THEN\ d\ is\ D_1$$

   Dimana a, b dan c adalah variabel linguistik proses yang menyatakan keadaan variabel proses dan d adalah variabel linguistik yang menyatakan variabel kendali.

2. *Object Evaluation Fuzzy Control Rule*

   Tipe aturan ini disebut dengan kendali *fuzzy* prediktif, karena memuat aturan prediksi keadaan sekarang dan keadaan aksi kendali selanjutnya. Bentuk aturan ini yaitu :

   $$rule\ i : IF\ (c\ is\ C_i \rightarrow (a\ is\ A_i\ and\ b\ is\ B_i))\ THEN\ c\ is\ C_i$$

   Dalam bentuk kalimat, aturan ini berarti ”jika indeks unjuk kerja a adalah A_i dan indeks b adalah B_i, ketika keluaran kendali c adalah C_i, maka aturan ke i dipakai, dimana nilai C_i digunakan sebagai sinyal kendali”.

Tidak ada prinsip umum untuk menentukan jumlah aturan kendali *fuzzy* yang digunakan agar optimal. Karena jumlah aturan tergantung pada berbagai aspek seperti unjuk kerja pengendali, efisiensi perhitungan sifat kerja operator dan pemilihan jenis variabel linguistik.

2.4.3. Pengambil keputusan (*fuzzy inference*)

Terdapat beberapa cara untuk mengolah data input dalam menentukan aturan-aturan mana yang digunakan sesuai aksi kontrol yang harus dilakukan. Dari berbagai cara tersebut dua diantaranya adalah metode inferensi MAX-MIN dan metode inferensi MAX-DOT. Biasanya dalam sistem pengendalian, data yang dijadikan sebagai masukan adalah berbentuk bilangan *crisp*, yang harus diubah

menjadi *fuzzy singleton* melalui proses fuzzifikasi. Kemudian keadaan ini harus diolah oleh suatu algoritma pengambil keputusan (*inference engine*) misalnya dengan metode MAX-MIN (Mamdani) atau metode MAX-DOT (Larsen).
Misalkan pengendali *fuzzy logic* memiliki dua buah aturan dasar :

rule 1 : IF x is A_1 and y is B_1 THEN z is C_1

rule 2 : IF x is A_2 and y is B_2 THEN z is C_2

Dengan *firing strength* aturan ke-i didefinisikan sebagai α_i. Untuk masukan x_0 dan y_0, maka besar *firing strength* α_1 dan α_2 pada aturan tersebut didefinisikan sebagai:

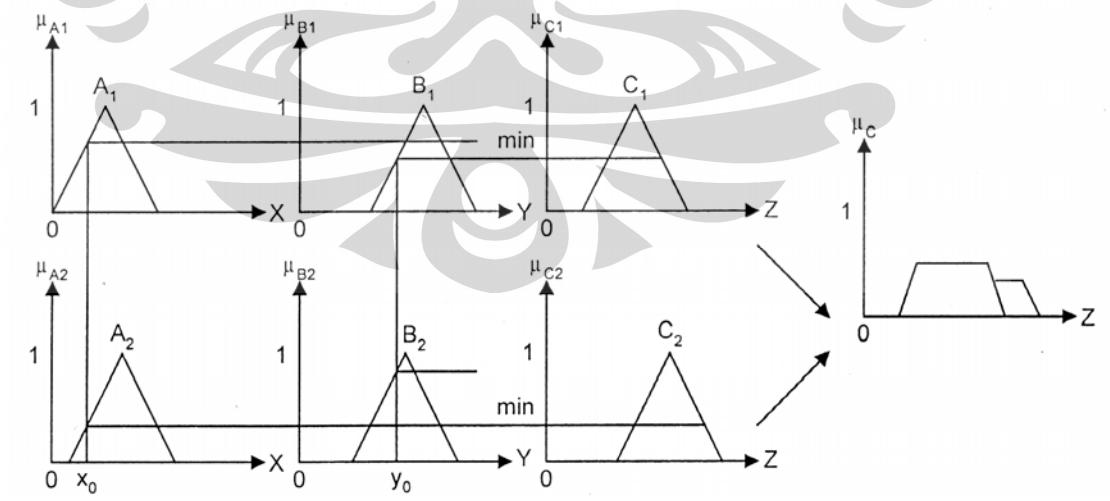
$$\alpha_1 = \mu_{A1}(x_0) \wedge \mu_{B1}(y_0)$$

$$\alpha_2 = \mu_{A2}(x_0) \wedge \mu_{B2}(y_0)$$

2.4.3.1. Metode MAX-MIN

Pada metode MAX-MIN digunakan operasi minimum Mamdani sebagai implikasi *fuzzy* dan untuk komposisi menggunakan komposisi maksimum. Pengambil keputusan untuk inferensi dari masukan x dan y adalah C dengan derajat keanggotaan yaitu :

$$\mu_C(w) = (\alpha_1 \wedge \mu_{C1}(w)) \vee (\alpha_2 \wedge \mu_{C2}(w))$$



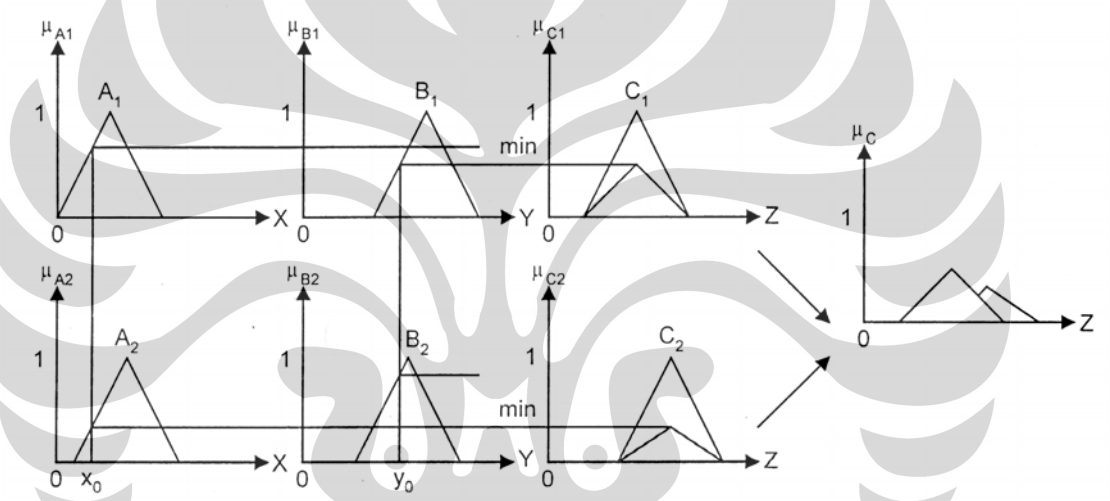
Gambar 2.7. Representasi diagram inferensi MAX-MIN (Mamdani).

2.4.3.2. Metode MAX-DOT

Metode ini menggunakan operasi komposisi maksimum dan operasi implikasi *algebraic product*. Disebut juga sebagai metode inferensi Larsen. Pengambilan keputusan untuk setiap aturan ke-i dapat dinyatakan dalam $\alpha_i \bullet \mu_{Ci}(w)$, sehingga hasil-hasil inferensi dari masukan x dan y adalah c dengan derajat keanggotaan :

$$\mu_C(w) = (\alpha_1 \bullet \mu_{C1}(w)) \vee (\alpha_2 \bullet \mu_{C2}(w))$$

dimana $\alpha_i = \mu_{Ai}(x_0) \bullet \mu_{Bi}(y_0)$ merupakan *firing strength* dari aturan kontrol ke-i.



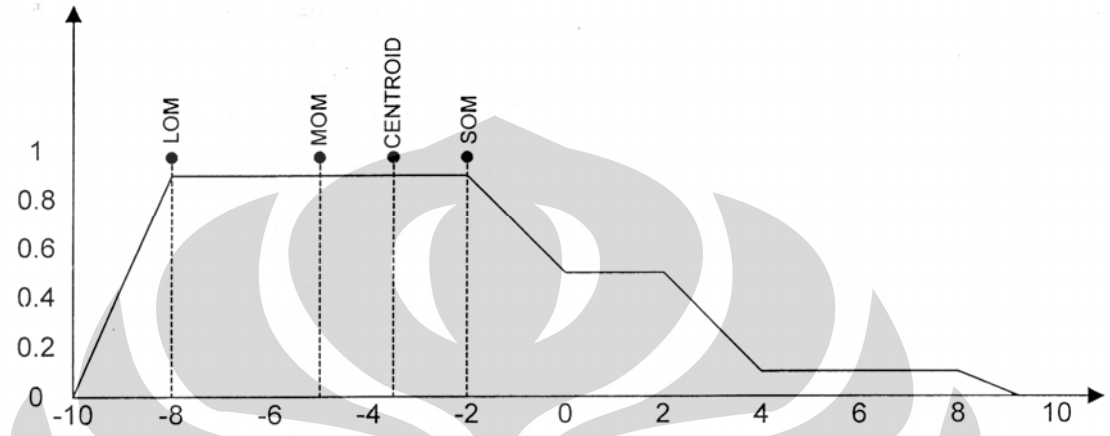
Gambar 2.8. Representasi diagram inferensi MAX-DOT (Larsen).

2.4.4. Defuzzifikasi

Seperti terlihat pada gambar 2.10 dan gambar 2.11, keluaran dari pengendali logika *fuzzy* masih merupakan gabungan variabel-variabel *fuzzy*, sedangkan pada aplikasi nyata, yang diperlukan ialah suatu nilai yang *crisp*. Dengan demikian diperlukan suatu proses yang memetakan gabungan variabel-variabel *fuzzy* tersebut menjadi suatu nilai *crisp*. Proses ini disebut defuzzifikasi.

Berbagai metode defuzzifikasi yang biasa digunakan diperlihatkan pada gambar 2.7. Gambar tersebut memperlihatkan perbandingan berbagai metode defuzzifikasi, dengan SOM (*Smallest of Maximum*), MOM (*Minimum of*

Maximum), LOM (*Largest of Maximum*), MOA (*Mean of Area*) dan *Centroid*. Metode yang kita pilih tergantung dari kecocokannya dengan pengendali yang akan kita buat, karena masing-masing memiliki kelebihan dan kekurangan.



Gambar 2.9. Perbandingan berbagai metode defuzzifikasi.

2.4.5. Metode perancangan pengendali logika *fuzzy*

Prinsip-prinsip dasar dari perancangan logika *fuzzy* meliputi :

1. Penentuan variabel linguistik masukan dan keluaran. Penentuan variabel linguistik dalam merancang pengendali logika *fuzzy* sangat tergantung pada banyaknya masukan dan keluaran yang terdapat pada sistem yang dikendalikan, biasanya variabel-variabel ini ditentukan berdasarkan jangkauan sinyal masukan maupun keluaran.
2. Pembagian daerah masukan dan keluaran ke dalam beberapa variabel *fuzzy* dan menentukan fungsi keanggotaan untuk tiap variabel *fuzzy* tersebut. Perlu diperhatikan banyaknya variabel *fuzzy* untuk tiap variabel linguistik masukan atau keluaran harus cukup banyak untuk dapat menghasilkan suatu penalaran yang baik, tetapi juga harus cukup sedikit supaya mempercepat proses perhitungan. Pemilihan fungsi keanggotaan yang berbeda-beda untuk tiap variabel *fuzzy* dapat saja dilakukan. Tidak dapat dikatakan bahwa suatu bentuk fungsi keanggotaan adalah paling baik, itu semua tergantung pada proses yang dikendalikan.

3. Penurunan aturan dasar kontrol. Penurunan aturan dasar pada proses pengendalian dengan pengendalian logika *fuzzy* biasanya dilakukan dengan pengetahuan pakar. Aturan-aturan ini sangat bergantung pada sifat atau karakter proses yang dikendalikan. Tidak ada patokan yang pasti mengenai banyaknya aturan yang harus dibentuk agar mencapai optimal.
4. Penentuan mekanisme pengambilan keputusan. Tidak ada suatu aturan yang paling baik atau cocok untuk semua proses. Aturan pengambilan keputusan yang paling baik untuk suatu proses biasanya di dapat setelah mencoba-coba beberapa aturan yang ada.
5. Penentuan metode defuzzifikasi. Demikian pula halnya dengan aturan atau metode defuzzifikasi. Tidak ada suatu metode yang paling baik yang dapat diterapkan pada semua jenis proses. Satu-satunya cara adalah dengan melakukan eksperimen sampai diperoleh hasil yang diinginkan.

BAB 3

CARA KERJA RANGKAIAN DAN LISTING PROGRAM

Pada bab ini akan dibahas mengenai cara kerja rangkaian interface dan listing program yang digunakan penulis dalam penyusunan alat "Pengendalian *posisi* menggunakan metode *logika fuzzy"*.

3.1 Rangkaian Interfacing

Rangkaian interfacing berfungsi untuk menyatukan dua protocol yang berbeda yaitu antara PC dengan hardware. Jalur komunikasi yang digunakan adalah port parallel (port printer). Port printer memiliki 8 pin yang berfungsi sebagai outpu dan 4 pin sebagai input. Data input dan output memiliki 2 sifat yaitu normal dan inverting. Normal yaitu kondisi data yang masuk ke port printer sama dengan data yang keluar pada hardware, sedangkan inverting yaitu kondisi data yang masuk ke port printer berkebalikan dengan kondisi yang dikeluarkan oleh hardware. Pada pembuatan rangkaian interfacing ini, penulis menggunakan 8 jalur data output dan 4 jalur data input normal yang berfungsi sebagai jalur pengiriman dan jalur penerimaan data dari hardware yang digunakan.

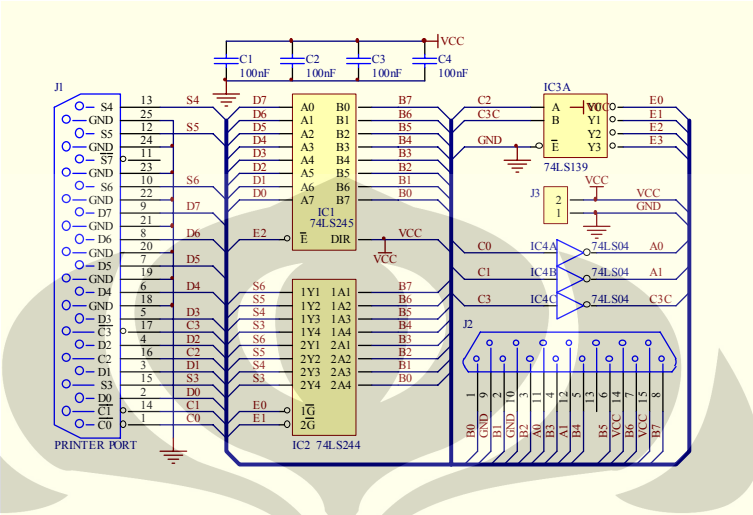
Pada rangkaian interfacing jalur pengiriman data yang digunakan adalah IC 74LS245 dan jalur penerimaan data digunakan IC 74LS244. Pada IC 74LS245 lajur D (D0-D7) terhubung dengan port printer sedangkan lajur B (B0-B7) terhubung dengan harware luar , dan jalur DIR terhubung dengan Vcc. Sehingga data akan mengalir dari lajur A ke lajur B. Jalur Enable (E) dapat dikendalikan dari PC, jalur ini berfungsi sebagai penentu kapan data dari lajur A dialirkan menuju lajur B.

Tabel 3.1. Tabel kebenaran IC 74LS25

INPUTS		OUTPUT
E	DIR	
L	L	Bus B to Bus A
L	H	Bus A to Bus B
H	X	Isolation

H=HIGH Voltage level
L=LOW Voltage level
X=Immaterial

Untuk lebih jelasnya kita dapat melihat gambar skematik rangkaian interfacing di bawah ini:



Gambar 3.1. Rangkaian Interfacing

IC 74LS244 merupakan IC *octal 3-state buffer* dimana IC ini akan menahan data terlebih dahulu, karena komputer tidak dapat menerima data 8 bit sekaligus. Sehingga konfigurasi pada sistem ini yaitu *output* IC 74LS244 dibagi 2 yaitu 4 bit *upper* dan 4 bit *lower* sehingga dapat di-*input*-kan ke komputer melalui pin-pin S_3-S_6 yang terdapat pada *port* LPT. Adapun tabel kebenarannya sebagai berikut :

Tabel 3.2. Tabel kebenaran IC 74LS244.

Inputs		Output
$\overline{G}$	A	Y
L	L	L
L	H	H
H	X	Z

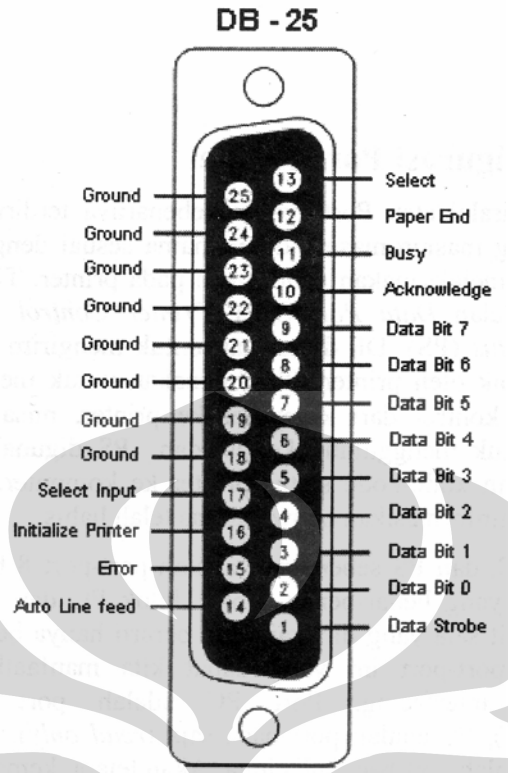
L = LOW Logic Level
H = HIGH Logic Level
X = Either LOW or HIGH Logic Level
Z = High Impedance

IC 74LS139 merupakan IC *1 of 4 decoder* atau *demultiplexer*, *input* IC ini didapat dari *port* LPT pada komputer.IC 74LS04 merupakan IC inverting, dimana IC ini mendapatkan input dari komputer yaitu pin C_0, C_1, dan C_3 berlogika aktif high. Sehingga urutan *output* dibalik pada setiap 4 bit *upper* dan 4 bit *lower* maka keluarannya akan sesuai dengan masukkannya.

Setelah melalui tahapan-tahapan tersebut maka data-data tersebut akan dialirkan ke komputer melalui *port* LPT yang akan diproses oleh VB. Semua data pengiriman maupun data penerimaan terhubung dengan hardware luar melalui DB 15.

3.2. Personal Computer (PC) dan Port Parallel

Pada perancangan sistem ini PC merupakan salah satu komponen penting, sebab pada PC ini terdapat perancangan *software* yang digunakan pada penelitian ini. Bahasa pemrograman yang terdapat pada PC ini yaitu menggunakan bahasa *Visual Basic* 6.0 yang digunakan sebagai bahasa pemrograman untuk pengendalian sistem dimana pemrograman tersebut berbasis *fuzzy logic controller*. Komputer digunakan sebagai penerima respon dari sensor *posisi* yang kemudian respon tersebut akan diubah menjadi bentuk grafik pada tampilan program VB. Selain sebagai penerima respon dari sensor, komputer juga digunakan untuk mengirimkan data 24 bit *digital* untuk diproses kembali, dimana data 24 bit *digital* tersebut akan dikirim untuk melakukan proses pembuatan sinyal PWM (*Pulse Width Modulation*). Proses penerimaan dan pengiriman data ini melalui *port paralel* (LPT) yang mempunyai alamat 0x378 H atau 888. Dimana pada komputer menggunakan slot LPT DB-25 *female* yang terdapat di belakang komputer. Adapun konfigurasi slot LPT DB-25 *female* adalah sebagai berikut:



Gambar 3.2. Konfigurasi slot LPT DB-25.

Port LPT 1 merupakan salah satu *port* yang dimiliki oleh komputer. Biasanya *port* ini dipakai untuk menghubungkan PC dengan *printer*. Keunggulan dari *port* ini terletak pada kemampuannya untuk menyampaikan data lebih cepat, karena *port* ini menggunakan komunikasi secara paralel. Meskipun memerlukan banyak kabel, namun memilih *port* ini sebagai *interface* sangat tepat karena jarak alat tidak terlalu jauh dari komputer.

Fungsi dari masing-masing Pin dalam DB 25 dapat dilihat pada tabel 3.3 berikut ini :

Tabel 3.3. Daftar pin *printer port*.

Pin DB-25	Nama Pin	Keterangan	Arah input/output	Sifat
1	*Strobe*	*Printer Control* 0 (PC-0)	*Output*	*Inverting*
2 – 9	Data *Output*	Data *Port* (DP0 - DP9)	*Output*	*Normal*
10	*Acknowledge*	*Printer Status* 6 (PS-6)	*Input*	*Normal*
11	*Busy*	*Printer Status* 7 (PS-7)	*Input*	*Inverting*
12	*Paper End*	*Printer Status* 5 (PS-5)	*Input*	*Normal*
13	*Select*	*Printer Status* 4 (PS-4)	*Input*	*Normal*
14	*Autofeed*	*Printer Control* 1 (PC-1)	*Output*	*Inverting*
15	*Error*	*Printer Status* 3 (PS-3)	*Input*	*Normal*
16	*Init*	*Printer Control* 2 (PC-2)	*Output*	*Normal*
17	*Select IN*	*Printer Control* 3 (PC-3)	*Output*	*Inverting*
18-25	*Ground*			

Port LPT 1 akan dapat diakses apabila alamat *port* tersebut pada komputer diketahui. Cara mengakses *port* ini tentunya dengan *software*. Adapun alamat dari LPT 1 pada PC sebagaimana tercantum dalam tabel 3.3.

Tabel 3.4. Alamat Port LPT 1.

Nama	Alamat
Data *Port* (DP)	378H (888)
Printer Status (PS)	379H (889)
Printer Control (PC)	37AH (890)

Umumnya LPT 1 pada PC digunakan untuk mengirimkan data dari PC ke *printer*. Hal ini berarti bahwa LPT 1 hanya dapat dipakai untuk *output* data saja. Oleh karena itu, diperlukan teknik tertentu untuk menjadikan LPT 1 sebagai *input* dan *output* data.

3.3. Perancangan Software

Perancangan sistem ini hanya memanfaatkan *software*, dimana pada sistem ini software yang digunakan yaitu *Visual Basic* (VB).

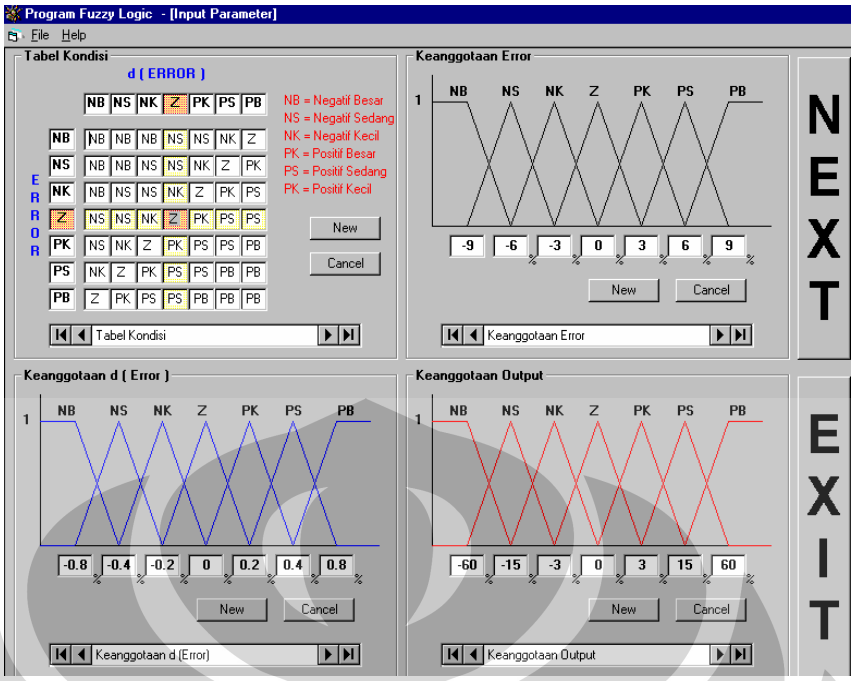
3.3.1. Visual Basic (VB)

Pada pemrograman dengan menggunakan *Visual Basic* (VB) ini penulis menggunakan metode *fuzzy logic* untuk menganalisa sistem pengendalian single variable. Pengendali logika *fuzzy* yang digunakan memiliki 7 buah fungsi keanggotaan *error*, 7 buah fungsi keanggotaan *d(error)* dan 7 buah fungsi keanggotaan *output*.

Ketujuh fungsi keanggotaan tersebut adalah :

1. Fungsi keanggotaan Negatif Besar (NB).
2. Fungsi keanggotaan Negatif Sedang (NS).
3. Fungsi keanggotaan Negatif Kecil (NK).
4. Fungsi keanggotaan *Zero*.
5. Fungsi keanggotaan Positif Kecil (PK).
6. Fungsi keanggotaan Positif Sedang (PS).
7. Fungsi keanggotaan Positif Besar (PB).

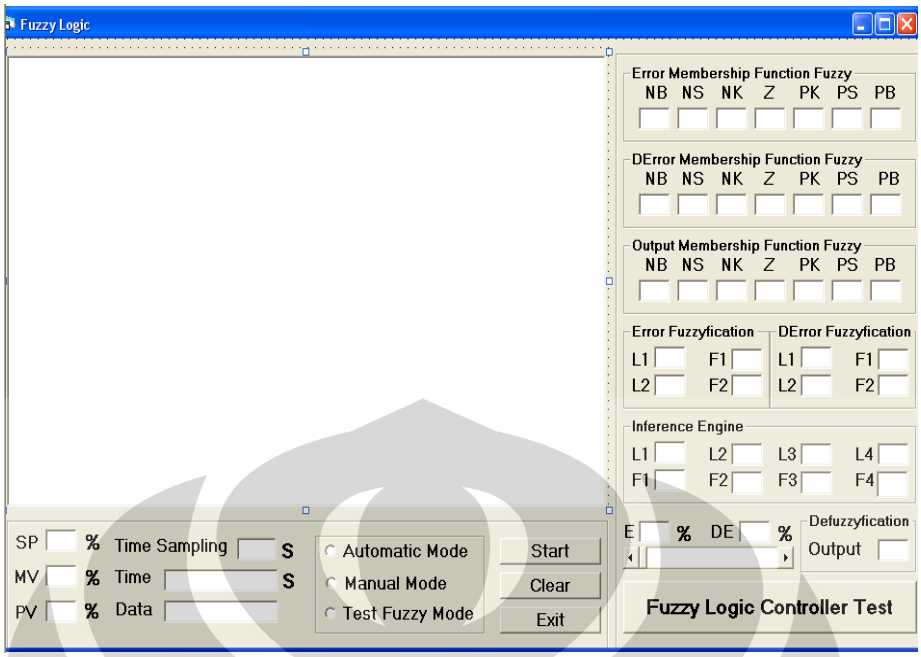
Fungsi keanggotaan untuk Positif Besar (PB) dan Negatif Besar (NB) memiliki bentuk trapesium dan yang lainnya berbentuk segitiga. Untuk lebih jelasnya dapat dilihat pada gambar *form* sebagai berikut :



Gambar 3.3. *Input* parameter pengendali logika *fuzzy*.

Fungsi keanggotaan setiap grup dapat diubah dengan memasukkan nilai-nilai keanggotaannya atau memilih dari *database* yang sudah ada dengan cara menekan tombol panah kiri atau kanan pada masing-masing grup. Untuk membuat fungsi keanggotaan yang baru dilakukan dengan cara menekan tombol "*New*" yang kemudian berubah menjadi tombol "*Save*" yang diikuti dengan hilangnya semua angka pada kotak yang tersedia, kemudian angka baru dimasukkan. Data tersebut akan disimpan bila tombol "*Save*" ditekan.

Pada gambar 3.3 tersebut, terdapat tabel kondisi yang berfungsi sebagai basis aturan pada pengendali logika *fuzzy* ini. Basis aturan dapat diganti dengan memilih basis aturan yang sudah tersimpan pada *file database*. Untuk memilihnya dengan menekan tombol panah kanan atau panah kiri pada tabel kondisi. Untuk membuat baru basis aturan adalah dengan tombol "*New*" yang berubah menjadi tombol "*Save*" yang diikuti seluruh basis aturan yang ada akan hilang, selanjutnya semua kotak kosong tersebut harus diisi yang diakhiri dengan menekan tombol "*Save*". Setelah selesai, tombol "*Next*" ditekan, lalu akan muncul gambar:



Gambar 3.4 Tampilan grafik pengendali logika fuzzy

Setelah tombol "*Next*" ditekan, maka tampilan komputer akan berubah menjadi gambar 3.15 diatas. Pada tampilan ini, nilai *set point* untuk masing-masing pengendali *fuzzy* ditentukan dengan memasukkan nilai dari 0% hingga 100%. Setelah menentukan nilai *set point*, harus ditentukan pula nilai-nilai parameter pada pengendali *fuzzy*. Pengendali *fuzzy* ini terdapat 3 buah *mode* pemilihan yaitu *automatic mode*, *manual mode* serta *test fuzzy mode*. Setelah semua parameter pengendali *fuzzy* telah ditentukan, tombol "*Start*" ditekan yang akan mengakibatkan tombol tersebut berubah menjadi tombol "*Stop*" dan akan timbul grafik PV, MV serta SP dari masing-masing pengendali pada masing-masing grafik. Apabila grafik telah mencapai batas kanan akan terhapus yang kemudian akan muncul kembali pada batas kiri. Untuk mengakhiri program tombol "*Stop*" ditekan, apabila ingin keluar dari program tombol "*Exit*" ditekan. Sedangkan tombol "*Clear*" ditekan untuk menghapus grafik. Pada bagian kanan *form* program terdapat beberapa *text box* yang berisi nilai-nilai parameter *input*, *output* fungsi keanggotaan *error*, *d(error)* dari pengendali *fuzzy*. Terdapat tombol "*Fuzzy Logic Controller Test*" untuk mengetes apakah sistem serta program dapat berjalan atau tidak dengan menggunakan pengendalian *fuzzy*.

Untuk pengambilan data basis aturan dilakukan secara mendatar. Karena basis aturan pada layar monitor berupa matriks 7 x 7, maka dilakukan 7 *subroutine* untuk mengambil seluruh basis data. Ketujuh *subroutine* tersebut adalah :

1. *sub txtErrNB_Change(Index As Integer)*
2. *sub txtErrNS_Change(Index As Integer)*
3. *sub txtErrNK_Change(Index As Integer)*
4. *sub txtErrZ_Change(Index As Integer)*
5. *sub txtErrPB_Change(Index As Integer)*
6. *sub txtErrPS_Change(Index As Integer)*
7. *sub txtErrPK_Change(Index As Integer)*

Serta terdapat tiga buah *subroutine* untuk mengambil fungsi keanggotaan *error*, *d(error)* dan *output* adalah :

1. *txtError_Change(Index As Integer)*
2. *txtDErr_Change(Index As Integer)*
3. *txtOut_Change(Index As Integer)*

Keseluruhan data disimpan dalam bentuk *array*. Program untuk mengambil data proses untuk *error* dan *d(error)* adalah :

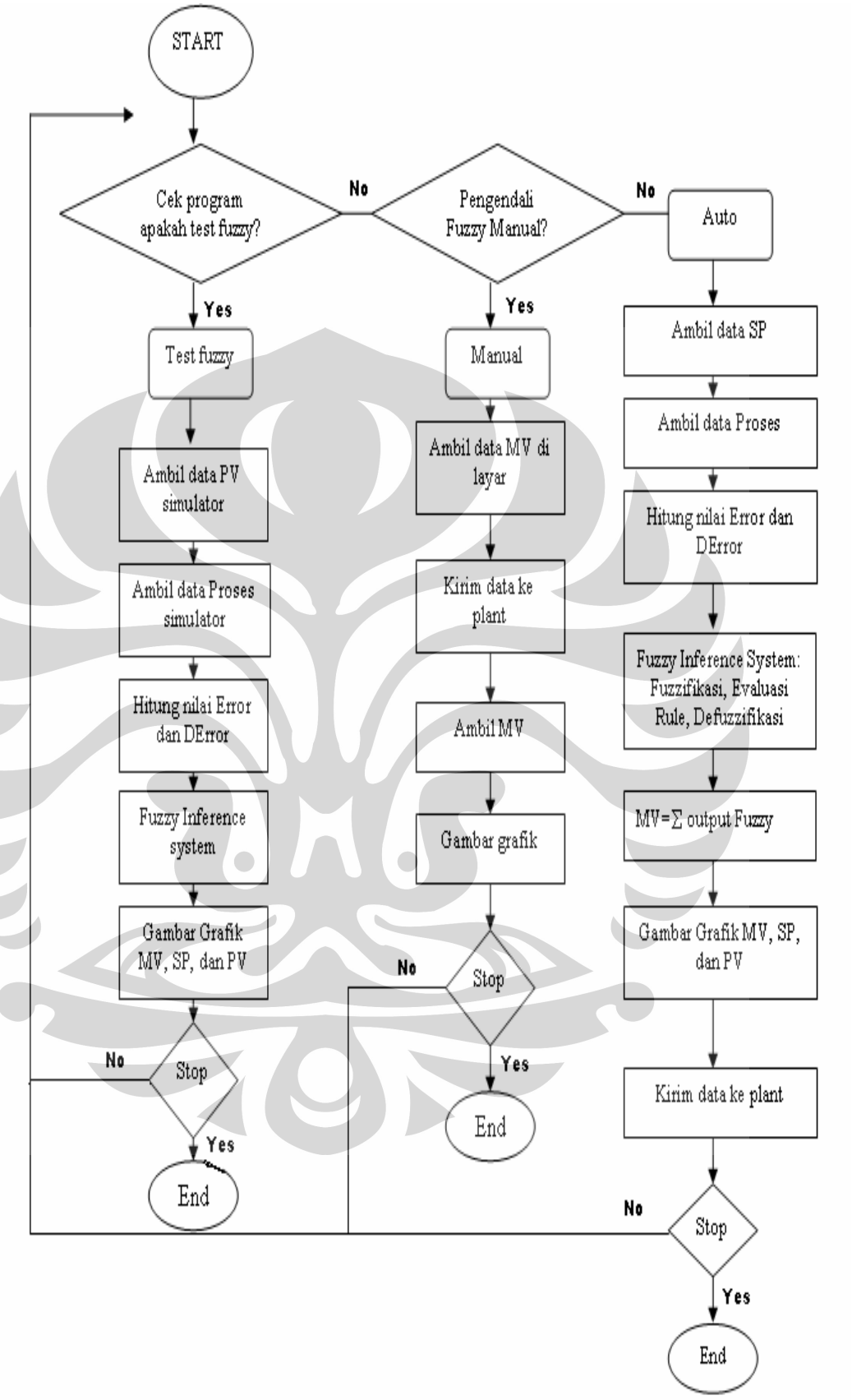
```
Public Sub Data_Masuk1()
    sglErr_11 = sglErr_21
    Call TerimaData1
    bytePV1 = Cacahan_Pulsa

    If Not bytePV1 = 0 Then
        sglDtPV1 = (bytePV1 / 1544)
    Else
        sglDtPV1 = 0
    End If
    byteSP1 = Val(TxtSP1.Text)
    sglErr_21 = byteSP1 - sglDtPV1
    sglDError1 = sglErr_21 - sglErr_11
    TxtError1.Text = Format(sglErr_21, "#0.00")
    TxtDError1.Text = Format(sglDError1, "#0.00")
    End Sub
```

Pada program ini, data *error* yang sudah ada disimpan dalam variabel *sglErr_11* untuk mencari *d(error)*. Kemudian dilanjutkan dengan pengambilan data dari *port printer* dan nilainya akan dikonversi ke nilai persen. Selanjutnya dilakukan proses *error* dan *d(error)* yang hasilnya disimpan dalam variabel *sglErr_21* dan *sglDErr_1*. Setelah didapat nilai *error* dan *d(error)*, maka nilai tersebut akan dimasukkan ke *subroutine* fuzzifikasi. Lalu hasil defuzzifikasi akan disimapn dalam variabel *sgloutput1*.

3.3.2. Flowchart Program

Proses kendali Logika Fuzzy dilakukan oleh program yang dibuat dengan bahasa visual basic 6.0. Pada progam ini kita dapat memilih apakah program test fuzzy atau pengendali fuzzy. Jika kita memilih pengendali fuzzy maka kita mempunyai dua pilihan yaitu apakah program manual atau auto. Pada program manual akan malakukan pembacaan data dari proses lalu akan dibandingkan dengan nilai manipulated variable (MV) untuk mengendalikan kecepatan motor. Kemudian plant akan mengirim data dalam bentuk grafik. Grafik pada program manual hanya terdiri dari MV. Pada grafik, MV ditunjukkan dengan dua warna yaitu hijau dan merah. Hal ini dikarenakan arah putaran motor maju dan mundur, sehingga nilai MV untuk putaran maju adalah hijau, sedangkan nilai MV untuk putaran mundur adalah merah.

Jika kita memilih program auto, maka program ini melakukan pembacaan data dari proses yang mempresentasikan kecepatan motor yang kemudian dibandingkan dengan nilai set point (SP) dan melakukan proses fuzzy inference yang meliputi fuzzifikasi, evaluasi rule, dan defuzzifikasi. Hasil fuzzy inference dioutputkan ke manipulated variable (MV) untuk mengendalikan kecepatan motor. Respon dari sistem ditampilkan dalam bentuk grafik antara SP, PV, dan MV.



Gambar 3.5. Flowchart Program Pengendali Logika Fuzzy

Nilai proses variabel pada grafik ditunjukkan dengan warna hitam. Nilai PV pada proses berkisar dari 0% sampai 100% dimana nilai PV diperoleh dari:

E = SP-PV

PV = E-SP, dimana E = error

Nilai PV diperoleh dari grafik putaran dengan pulsa dan jarak perpindahan yang akan diukur:

1 putaran = 192 pulsa

Jarak perpindahan yang akan diukur = 800mm

1 putaran = 2mm perpindahan, maka jarak yang akan diukur:

800mm/2mm = 400 mm

$$\% = \frac{100*PV}{192*400}$$

PV (100%) = 768

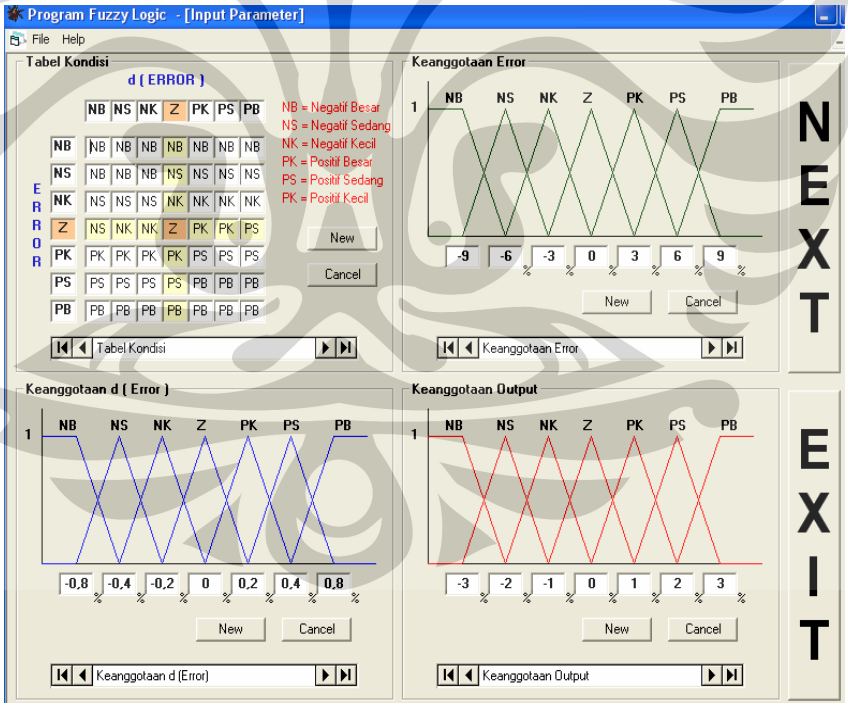
BAB 4

HASIL DAN ANALISA

Setelah dilakukan pengerjaan keseluruhan sistem, maka perlu dilakukan pengujian alat serta penganalisaan terhadap alat, apakah sistem sudah bekerja dengan baik atau tidak.

4.1. Pengujian Sistem dengan Logika Fuzzy

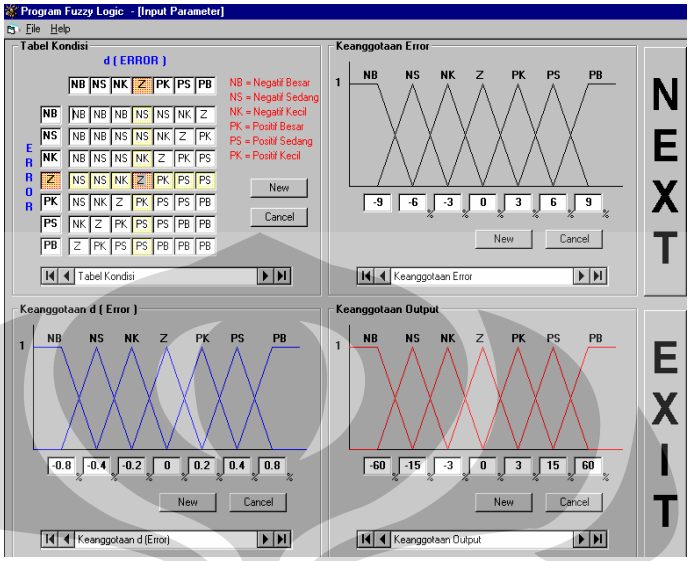
Pengujian sistem dilakukan dengan menggunakan program VB yang menggunakan pengendalian logika *fuzzy* dengan pengaturan SP dan menggunakan *automatic mode*, pengujian dilakukan dengan cara mengubah rulebase dan nilai-nilai keanggotan fuzzy. Berikut merupakan tampilan penentu fungsi keanggotaaan pada fuzzy.



Gambar 4.1 Penentuan Fungsi Keanggotaan Pada Fuzzy

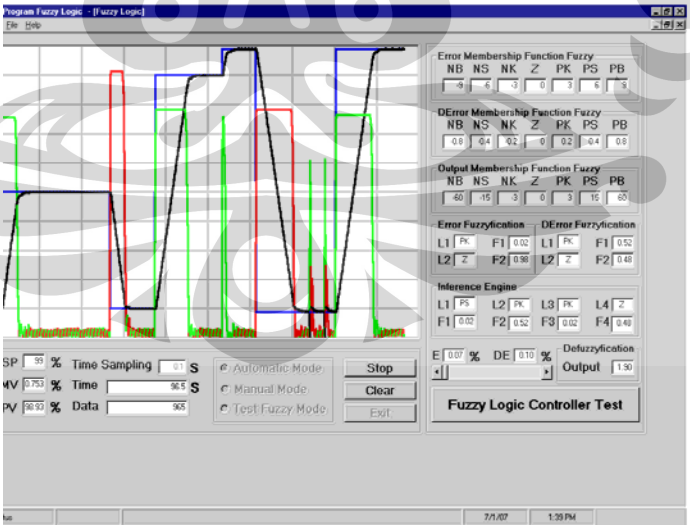
Pada penentuan fungsi keanggotaan ini penulis menetapkan nilai-nilai dari fungsi keanggotaan *error*, d*Error* dan *output*. Nilai-nilai tersebut dapat terlihat pada gambar 4.1

Kemudian penulis menentukan basis rule (basis aturan) yang terdapat pada tabel rule yang dibuat dan menentukan keanggotaan *error*, d*Error* dan *output* . Basis aturan yang menurut penulis ideal adalah seperti pada gambar 4.2 :



Gambar 4.2 Tabel Basis Aturan ideal

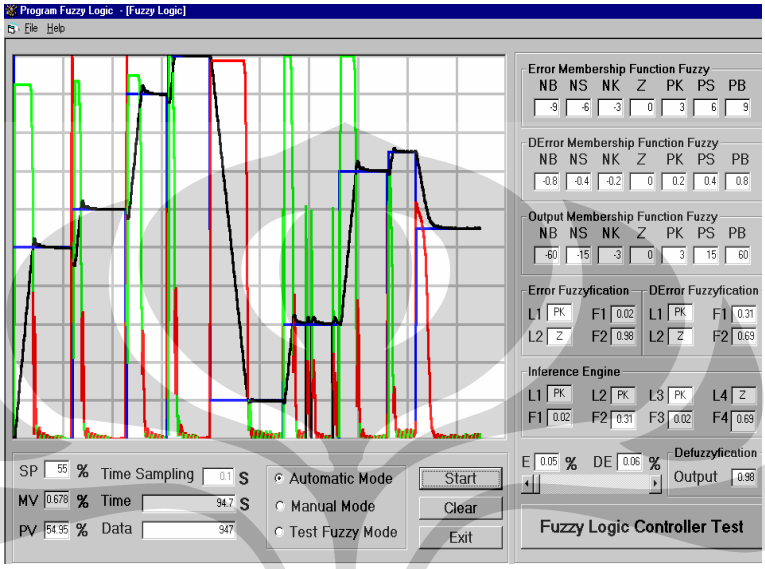
Pada pemrograman visual basic untuk menentukan basis rule yang ideal seperti diatas penulis melakukan percobaan program visual basic. Penulis hanya mengubah-ubah tabel rule dan keanggotaan *error*, d*Error* dan *output* yang ada. Berikut merupakan respon dari pengolahan data pada tabel aturan gambar 4.2.



Gambar 4.3 Grafik respon dengan basis aturan ideal

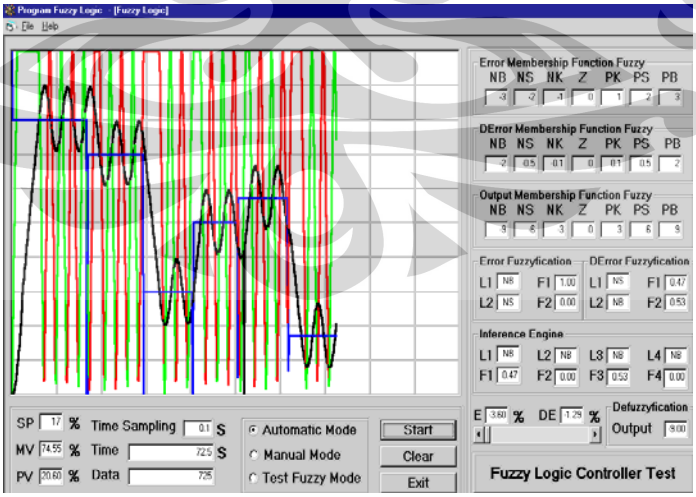
Dari grafik dapat kita lihat dengan set point yang berubah-ubah nilai PV akan mengikuti SP yang menandakan sistem stabil, dimana error pada grafik ini

sangat kecil dan mendekati nol. Hal ini menandakan bahwa sistem memiliki pengendalian yang baik. Dari grafik dapat kita lihat dengan set point yang berubah-ubah nilai PV akan mengikuti SP yang menandakan sistem stabil, dimana error pada grafik ini sangat kecil dan mendekati nol. Hal ini menandakan bahwa sistem memiliki pengendalian yang baik.



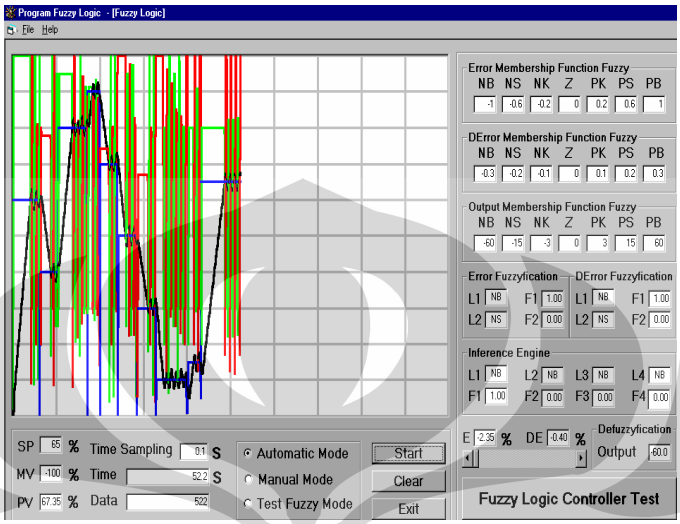
Gambar 4.4 Respon dengan mengubah rulebase, keanggotaan *error,Eerror*,dan *output* tetap

Dari grafik tersebut terjadi peningkatan nilai PV yang mendekati SP sehingga sistem dikatakan belum stabil atau mendekati stabil.



Gambar 4.5 Respon dengan mengubah rulebase dan keanggotaan kecil

Dari grafik dapat kita lihat bahwa dengan perubahan rulebase dan keanggotaan fuzzy perubahan PV sangat cepat sehingga error yang diperoleh juga besar. Sistem dengan grafik seperti ini belum stabil.



Gambar 4.6 Respon dengan keanggotaan *error, dError* kecil dan *output* besar

Dari grafik tersebut dikatakan bahwa dengan keanggotaan output yang besar dan keanggotaan *error* dan *dError* yang kecil sistem masih belum stabil. Dimana PV masih melebihi SP.

Pada keempat respon grafik diatas dapat diketahui bahwa sistem tetap cenderung stabil. Pada grafik respon dengan rulebase(basis aturan) yang ideal dikatakan bahwa sistem memiliki pengendalian yang baik. Hal ini dibuktikan dengan respon grafik yang cenderung stabil walaupun nilai SP berubah-ubah. Pada grafik respon dengan fungsi keanggotaan pengendali logika *fuzzy* yang kecil, sistem pengendalian terjadi osilasi, hal ini seharusnya tidak boleh terjadi.

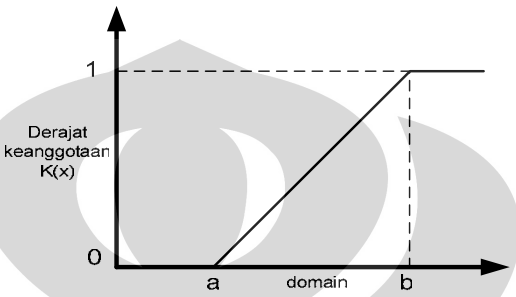
Berikut merupakan cara perhitungan yang digunakan pada logika fuzzy:

1. Fuzzification

   Pada proses fuzzifikasi digunakan 2 representasi dalam logika fuzzy yang pertama adalah representasi linier dan yang kedua adalah representasi kurva segitiga. Berikut merupakan penjelasannya :

   a. Representasi linier

Pada representasi linier, pemetaan input ke derajat keanggotaannya digambarkan sebagai suatu garis lurus. Ada 2 keadaaan himpunan fuzzy yang linier. Pertama, kenaikan himpunan dimulai pada nilai domain yang memiliki derajat keanggotaan nol (0) bergerak ke kanan menuju ke nilai domain yang memiliki derajat keanggotaan lebih tinggi ini disebut juga dengan representasi linear naik. Terlihat pada gambar 4.7.

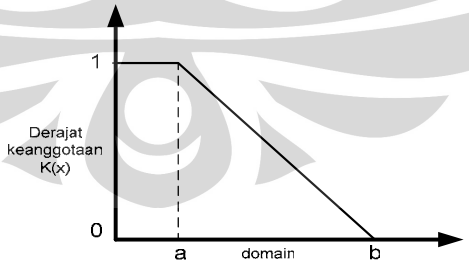


Gambar 4.7 Representasi Liniear Naik

Dari representasi linier naik memiliki fungsi keanggotaan :

$$K(x) = \begin{cases} 0; & x \le a \\ (a - x) / (a - b); & a \le x \le b \\ 1; & x \ge b \end{cases}$$

Kemudian yang kedua, merupakan kebalikan dari yang pertama. Nilai domain yang memiliki derajat keanggotaan tertinggi berada pada sisi kiri, kemudian bergerak menurun ke nilai domain yang memiliki derajat keanggotaan lebih rendah. Terlihat pada gambar 4.8.



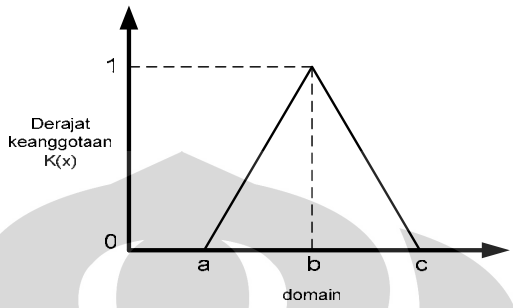
Gambar 4.8 Representasi Liniear Turun

Dari representasi linier turun memiliki fungsi keanggotaan :

$$K(x) = \begin{cases} 1; & x \le a \\ (b - x) / (b - a); & a \le x \le b \\ 0; & x \ge b \end{cases}$$

b. Representasi kurva segitiga

Kurva segitiga pada dasarnya merupakan gabungan antara 2 garis (linear) seperti terlihat pada gambar 4.9.



Gambar 4.9 Representasi Kurva Segitiga

Dari representasi kurva segitiga memiliki fungsi keanggotaan :

$$K(x) = \begin{cases} 0; & x \le a \text{ atau } x \ge c \\ (a - x) / (a - b); & a \le x \le b \\ (c - x) / (c - b); & b \le x \le c \\ 1; & x = b \end{cases}$$

2. Rule Evaluation

Pada prose rule evaluation pengolahan data menggunakan metode min-max atau yang biasa disebut dengan metode mamdani. Pada metode ini, fungsi implifikasi yang digunakan adalah pengambilan data nilai minimum.

3. Defuzzification

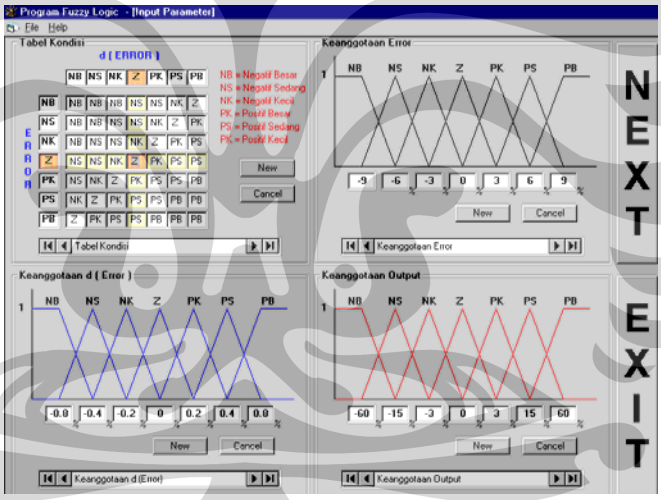
Pada proses defuzzification digunakan metode *Mean of Maximum* (MOM). Dimana pengolahan data dari hasil rule evaluation diambil nilai maksimum dan dirata-ratakan.

BAB 5
KESIMPULAN DAN SARAN

5.1. Kesimpulan

Setelah menyelesaikan perancangan sistem serta pengujian terhadap sistem tersebut, maka penulis dapat mengambil suatu kesimpulan bahwa :

1. Sistem pada *plant* telah stabil walaupun nilai set poin berubah-ubah, sehingga dapat dikatakan bahwa pengendalian yang telah dilakukan bekerja dengan baik.
2. Respon yang paling baik ditunjukkan pada rulebase (basis aturan) dengan nilai fungsi keanggotaan pada gambar di bawah ini, dimana respon pada sistem menunjukkan stabil.



3. Pengendali logika fuzzy relatif mudah untuk diimplementasikan karena tidak membutuhkan model matematika tetapi bekerja berdasarkan rule yang dapat diekstrak dari pengalaman dan keahlian seorang operator.

5.2. Saran

1. Diharapkan pada pembuatan software menggunakan windows XP agar hasil respon yang didapatkan lebih cepat dan lebih baik.

2. Pembuatan sistem *pengendalian posisi* ini harus disesuaikan dengan keadaan lingkungan serta sistem, terutama pembuatan *plant*, yang disesuaikan dengan maksud dan tujuannya.

DAFTAR ACUAN

- Klir, J.R., Bo Yuan, 1999, "Fuzzy Sets and Fuzzy Logic Theory and Aplications". New Jersey: Prentice Hall.
- Center for Emerging Computer Technologies, "Fuzzy Logic Education Program", Motorola Inc, 1994.
- Yan, J., Ryan M, dan Power, J. 1994. "Using Fuzzy Logic Towards Intelligent System". New York: Prentice Hall.
- Kusumadewi, Sri, Hari Purnomo. 2004. "Aplikasi Logika Fuzzy untuk Pendukung Keputusan". Yogyakarta: Graha Ilmu.
- Prasetia, Ratna, Catur Edi Widodo. "Interfacing Port Parallel dan Port Serial Komputer dengan Visual Basic 6.0". Yogyakarta: Andi.

LAMPIRAN

- **Form Fuzzy**

```
Option Explicit
    'Deklarasi Alamat input dan output
     Const OutData = &H378
     Const InData = &H379
     Const Control = &H37A
     Const AlamatData1 = &H4
     Const AlamatData2 = &H0
     Const AlamatData3 = &H5
     Const AlamatData4 = &H1
     Const AlamatData5 = &H6
     Const AlamatData6 = &H2
     Const AlamatMotor = &HB

    'Deklarasi Variabel input dan output
     Dim PortData As Integer
     Dim Data1, Data2, Data3, Data4, Data5, Data6 As Integer
     Dim Cacahan_Pulsa, bytePV1, Temp As Double
     Dim Mulai, Ulang As Boolean


    Dim ExcelApp As Excel.Application   'inisialisasi excell
    Dim ExcelWkb As Excel.Workbook      'inisialisasi workbook
    Dim ExcelSht As Excel.Worksheet     'inisialisasi worksheet
    Dim MyExcel As Boolean              'inisialisasi file excell
    Dim i As Double
    Dim A, B As Double

Private Sub CmdClear_Click()
     Call GarisGrafik1
     intX1 = 0
     Kounter = 0
End Sub

Private Sub cmdClose_Click()
     Unload Me
     frmInputParameter.Hide
     mdiFuzzy.Hide
     End
End Sub
Private Sub TerimaData1()
    '=======================================
    'Sub untuk menerima data dari LPT
    '=======================================
     Dim Data_Temp As Byte
    If OptAuto.Value = True Or OptManual.Value = True Then
       Port_Out Control, AlamatData1
```

```
        PortData = Port_In(InData)
        Data1 = (PortData And 120) / 8
        Port_Out Control, AlamatData2
        PortData = Port_In(InData)
        Data2 = (PortData And 120) / 8
        Port_Out Control, AlamatData3
        PortData = Port_In(InData)
        Data3 = (PortData And 120) / 8
        Port_Out Control, AlamatData4
        PortData = Port_In(InData)
        Data4 = (PortData And 120) / 8
        Port_Out Control, AlamatData5
        PortData = Port_In(InData)
        Data5 = (PortData And 120) / 8
        Port_Out Control, AlamatData6
        PortData = Port_In(InData)
        Data6 = (PortData And 120) / 8
        Cacahan_Pulsa = Data1 + 16 * Data2 + 256 * Data3 + 4096 *
Data4 + 65536 * Data5 + 1048576 * Data6
    Else
        Temp = PV1Scroll.Value
        Cacahan_Pulsa = Temp * 1536

    End If
    If Cacahan_Pulsa > 200000 Then Cacahan_Pulsa = 0
 End Sub

Private Sub cmdStart_Click()

     If cmdStart.Caption = "Start" Then
          PctGrafik1.Cls
          Call GarisGrafik1

          'fuzzy 1
          bytePV1 = 0: sglErr_11 = 0: sglErr_21 = 0
          sglDtPV1 = 0
          intX1 = 0: intPoint1_11 = PctGrafik1.Height
          intPoint1_21 = PctGrafik1.Height
          intPoint2_11 = PctGrafik1.Height
          intPoint2_21 = PctGrafik1.Height
          sglOutput1 = 0
          Kounter = 0
          Ulang = True
          Mulai = True
          OptAuto.Enabled = False
          OptManual.Enabled = False
          OptTest.Enabled = False
          txtSampling.Enabled = False
          cmdClose.Enabled = False
          cmdStart.Caption = "Stop"
          Call Pilih_Option
     Else
          Port_Out OutData, 0
          Port_Out Control, AlamatMotor
          Ulang = False
          Mulai = False
          cmdStart.Caption = "Start"
          txtSampling.Enabled = True
          cmdClose.Enabled = True
```

```
          OptAuto.Enabled = True
          OptManual.Enabled = True
          OptTest.Enabled = True
     End If

End Sub
Private Sub Pilih_Option()
    If OptAuto.Value = True Then
       Call Program_Auto
    ElseIf OptManual.Value = True Then
       TxtMV1.Text = ""
       TxtPV1.Text = ""
       TxtSP1.Text = 0
       Call Program_Manual
    Else
       TxtSP1.Text = 50
       PV1Scroll.Value = 50
    End If
End Sub

Private Sub Form_Load()
     frmFuzzy.WindowState = 2
     Call GarisGrafik1
     txtSampling.Text = 0.1
     TxtSP1.Text = 50
     TxtErr1NB = sglErr1(0)
     TxtErr1NS = sglErr1(1)
     TxtErr1NK = sglErr1(2)
     TxtErr1Z = sglErr1(3)
     TxtErr1PK = sglErr1(4)
     TxtErr1PS = sglErr1(5)
     TxtErr1PB = sglErr1(6)

     TxtDErr1NB = sglDErr1(0)
     TxtDErr1NS = sglDErr1(1)
     TxtDErr1NK = sglDErr1(2)
     TxtDErr1Z = sglDErr1(3)
     TxtDErr1PK = sglDErr1(4)
     TxtDErr1PS = sglDErr1(5)
     TxtDErr1PB = sglDErr1(6)

     TxtOut1NB = sglOut1(0)
     TxtOut1NS = sglOut1(1)
     TxtOut1NK = sglOut1(2)
     TxtOut1Z = sglOut1(3)
     TxtOut1PK = sglOut1(4)
     TxtOut1PS = sglOut1(5)
     TxtOut1PB = sglOut1(6)
     Ulang = False
     OptAuto.Value = True
     OptManual.Value = False
     OptTest.Value = False

     Randomize
     On Error Resume Next
     Err.Clear
     Set ExcelApp = CreateObject("Excel.Application")
     ExcelApp.Visible = False
End Sub
```

```
Public Sub GarisGrafik1()
     '============================
     'Membuat Grid Line Pada Grafik1
     '============================
     Dim i As Integer

     PctGrafik1.Cls
     'Membuat Garis Horisontal
     For i = 0 To PctGrafik1.Height Step (PctGrafik1.Height / 10)
          PctGrafik1.Line (0, i)-(PctGrafik1.Width, i)
     Next
     'Membuat Garis Vertikal
     For i = 0 To PctGrafik1.Width Step (PctGrafik1.Width / 10)
          PctGrafik1.Line (i, 0)-(i, PctGrafik1.Height)
     Next
End Sub

Private Sub PctGrafik1_Resize()
     Call GarisGrafik1
End Sub
Private Sub KirimData1()
Dim DataKeluar As Integer

If sglOutput1 >= 0 Then
   DataKeluar = (sglOutput1) * 1.26
Else
   DataKeluar = 128 + (-1 * sglOutput1) * 1.26
End If

Port_Out OutData, DataKeluar
Port_Out Control, AlamatMotor

End Sub
Private Sub Program_Auto()
     AwalData
     i = 0
     Kounter = 0
     Do While Ulang = True
        Kounter = Kounter + 1
        Waktu = Kounter * Val(txtSampling.Text)

        Call Data_Masuk1
        Call Proses_Error1
        Call Proses_DError1
        Call Proses_Data1
        Call Data_Keluar1
        Call KirimData1
        Call Gambar_Grafik1

        TxtTime.Text = Waktu
        TxtData.Text = Kounter
        TxtMV1.Text = sglOutput1
        TxtPV1.Text = sglDtPV1
        Tunda Val(txtSampling.Text) * 1000
        SimpanData
     Loop
     SaveFile
End Sub
Private Sub CmdTest_Click()
```

```
   If OptTest.Value = True And Mulai = True Then
      Call Data_Masuk1
      Call Proses_Error1
      Call Proses_DError1
      Call Proses_Data1
      Call Data_Keluar1

      If sglOutput1 >= 100 Then
         sglOutput1 = 100
      ElseIf sglOutput1 < -100 Then
         sglOutput1 = -100
      End If

      TxtMV1.Text = Format(sglOutput1, "#0.00")
      TxtPV1.Text = Format(sglDtPV1, "#0.00")
      Call Gambar_Grafik1
   End If
End Sub

Public Sub Data_Masuk1()

     sglErr_11 = sglErr_21
     Call TerimaData1
     bytePV1 = Cacahan_Pulsa

     If Not bytePV1 = 0 Then
          sglDtPV1 = (bytePV1 / 1536)
     Else
          sglDtPV1 = 0
     End If
     byteSP1 = Val(TxtSP1.Text)
     sglErr_21 = byteSP1 - sglDtPV1
     sglDError1 = sglErr_21 - sglErr_11
     TxtError1.Text = Format(sglErr_21, "#0.00")
     TxtDError1.Text = Format(sglDError1, "#0.00")

End Sub
Public Sub Proses_Error1()
     Dim i As Integer

     i = 0
     Do While i <= 5
          If sglErr_21 <= sglErr1(0) Then
               sglErrOut1(0) = 1:    strErr_21 = "NS"
               sglErrOut1(1) = 0:    strErr_11 = "NB"
               Exit Do
          ElseIf sglErr_21 > sglErr1(i) And sglErr_21 <= sglErr1(i
+ 1) Then
               sglErrOut1(0) = (sglErr_21 - sglErr1(i)) /
(sglErr1(i + 1) - sglErr1(i))
               sglErrOut1(1) = (sglErr1(i + 1) - sglErr_21) /
(sglErr1(i + 1) - sglErr1(i))
                         Select Case i
                              Case 0
                                   strErr_11 = "NS": strErr_21 =
"NB"
                              Case 1
```

```
                                    strErr_11 = "NK": strErr_21 =
"NS"
                               Case 2
                                    strErr_11 = "Z": strErr_21 =
"NK"
                               Case 3
                                    strErr_11 = "PK": strErr_21 =
"Z"
                               Case 4
                                    strErr_11 = "PS": strErr_21 =
"PK"
                               Case 5
                                    strErr_11 = "PB": strErr_21 =
"PS"
                          End Select
                    Exit Do
               ElseIf sglErr_21 > sglErr1(6) Then
                    sglErrOut1(0) = 1
                    sglErrOut1(1) = 0
                         strErr_11 = "PB": strErr_21 = "PS"
                    Exit Do
               End If
               i = i + 1
          Loop
          TxtLE11 = strErr_11: TxtFE11 = Format(sglErrOut1(0), "#0.00")
          TxtLE12 = strErr_21: TxtFE12 = Format(sglErrOut1(1), "#0.00")
End Sub
Public Sub Proses_DError1()
          Dim i As Integer

          i = 0
          Do While i <= 5
               If sglDError1 <= sglDErr1(0) Then
                    sglDErrOut1(0) = 1
                    sglDErrOut1(1) = 0
                         strDErr_11 = "NB": strDErr_21 = "NS"
                    Exit Do
               ElseIf sglDError1 > sglDErr1(i) And sglDError1 <=
sglDErr1(i + 1) Then
                    sglDErrOut1(0) = (sglDError1 - sglDErr1(i)) /
(sglDErr1(i + 1) - sglDErr1(i))
                    sglDErrOut1(1) = (sglDErr1(i + 1) - sglDError1) /
(sglDErr1(i + 1) - sglDErr1(i))
                          Select Case i
                               Case 0
                                    strDErr_11 = "NS": strDErr_21 =
"NB"
                               Case 1
                                    strDErr_11 = "NK": strDErr_21 =
"NS"
                               Case 2
                                    strDErr_11 = "Z": strDErr_21 =
"NK"
                               Case 3
                                    strDErr_11 = "PK": strDErr_21 =
"Z"
                               Case 4
                                    strDErr_11 = "PS": strDErr_21 =
"PK"
```

```
                              Case 5
                                   strDErr_11 = "PB": strDErr_21 =
"PS"
                         End Select
               Exit Do
          ElseIf sglDError1 > sglDErr1(6) Then
               sglDErrOut1(0) = 1
               sglDErrOut1(1) = 0
                    strDErr_11 = "PB": strDErr_21 = "PS"
               Exit Do
          End If
          i = i + 1
     Loop
     TxtLDE11 = strDErr_11: TxtFDE11 = Format(sglDErrOut1(0),
"#0.00")
     TxtLDE12 = strDErr_21: TxtFDE12 = Format(sglDErrOut1(1),
"#0.00")
End Sub
Public Sub Proses_Data1()
     Dim i, j, n, klm As Integer
     Dim strError1(1) As String
     Dim strError2(1) As String

     strError1(0) = strErr_11
     strError1(1) = strErr_21

     klm = 0:  j = 0:    n = 0

     With frmInputParameter
          Do While klm <= 6
               If .lblDErr(klm).Caption = strDErr_11 Then Exit Do
               klm = klm + 1
          Loop

          Do While n <= 1
               For i = 0 To 1
                    Select Case strError1(i)
                         Case "NB"
                              strHasil1(i + j) = .txtErrNB(klm)
                         Case "NS"
                              strHasil1(i + j) = .txtErrNS(klm)
                         Case "NK"
                              strHasil1(i + j) = .txtErrNK(klm)
                         Case "Z"
                              strHasil1(i + j) = .txtErrZ(klm)
                         Case "PK"
                              strHasil1(i + j) = .txtErrPK(klm)
                         Case "PS"
                              strHasil1(i + j) = .txtErrPS(klm)
                         Case "PB"
                              strHasil1(i + j) = .txtErrPB(klm)
                    End Select

                    If sglDErrOut1(n) > sglErrOut1(i) Then
                         sglDataOut1(i + j) = sglErrOut1(i)
                    Else
                         sglDataOut1(i + j) = sglDErrOut1(n)
                    End If
```

```
                Next

                klm = 0
                Do While klm <= 6
                     If .lblDErr(klm).Caption = strDErr_21 Then
Exit Do
                     klm = klm + 1
                Loop

                j = 2
                n = n + 1
           Loop
      End With

      TxtLI11 = strHasil1(0): TxtFI11 = Format(sglDataOut1(0),
"#0.00")
      TxtLI12 = strHasil1(1): TxtFI12 = Format(sglDataOut1(1),
"#0.00")
      TxtLI13 = strHasil1(2): TxtFI13 = Format(sglDataOut1(2),
"#0.00")
      TxtLI14 = strHasil1(3): TxtFI14 = Format(sglDataOut1(3),
"#0.00")

End Sub
Public Sub Data_Keluar1()
      Dim i, j, n As Integer
      'variable fuzzy 1
      '----------------------
      Dim sglTotal1, sglTotalOut1, sglTotData1 As Single
      Dim sglDataKeluar1(3) As Single
      Dim intIndikator1(3) As Boolean
      Dim intKeluaran1 As Integer

      sglTotalOut1 = 0: sglTotData1 = 0: sglTotal1 = 0

     For i = 0 To 3
         Select Case strHasil1(i)
                Case "NB"
                     sglTotalOut1 = sglTotalOut1 + (sglDataOut1(i)
* sglOut1(0))
                Case "NS"
                     sglTotalOut1 = sglTotalOut1 + (sglDataOut1(i)
* sglOut1(1))
                Case "NK"
                     sglTotalOut1 = sglTotalOut1 + (sglDataOut1(i)
* sglOut1(2))
                Case "Z"
                     sglTotalOut1 = sglTotalOut1 + (sglDataOut1(i)
* sglOut1(3))
                Case "PK"
                     sglTotalOut1 = sglTotalOut1 + (sglDataOut1(i)
* sglOut1(4))
                Case "PS"
                     sglTotalOut1 = sglTotalOut1 + (sglDataOut1(i)
* sglOut1(5))
                Case "PB"
                     sglTotalOut1 = sglTotalOut1 + (sglDataOut1(i)
* sglOut1(6))
         End Select
```

```
        sglTotData1 = sglTotData1 + sglDataOut1(i)
    Next

    TxtOut1 = Format((sglTotalOut1 / sglTotData1), "#0.00")
    sglOutput1 = sglOutput1 + (sglTotalOut1 / sglTotData1)
    If sglOutput1 >= 100 Then
        sglOutput1 = 100
    ElseIf sglOutput1 < -100 Then
        sglOutput1 = -100
    End If
End Sub

Public Sub Gambar_Grafik1()
     intPoint1_11 = intPoint1_21
     intPoint2_11 = intPoint2_21
     intPoint3_11 = intPoint3_21

     intPoint2_21 = PctGrafik1.Height - ((sglDtPV1 / 100) *
PctGrafik1.Height)
     If sglOutput1 >= 0 Then
        intPoint1_21 = PctGrafik1.Height - ((sglOutput1 / 100) *
PctGrafik1.Height)
     Else
        intPoint1_21 = PctGrafik1.Height - ((-1 * sglOutput1 /
100) * PctGrafik1.Height)
     End If
     intPoint3_21 = PctGrafik1.Height - ((Val(TxtSP1.Text) / 100)
* PctGrafik1.Height)

     If OptTest.Value = True Then
         PctGrafik1.Line (intX1, intPoint3_11)-(intX1 +
PctGrafik1.Width / 100, intPoint3_21), vbBlue
         If sglOutput1 >= 0 Then
            PctGrafik1.Line (intX1, intPoint1_11)-(intX1 +
PctGrafik1.Width / 100, intPoint1_21), vbGreen
         Else
            PctGrafik1.Line (intX1, intPoint1_11)-(intX1 +
PctGrafik1.Width / 100, intPoint1_21), vbRed
         End If
         PctGrafik1.Line (intX1, intPoint2_11)-(intX1 +
PctGrafik1.Width / 100, intPoint2_21), vbBlack
         intX1 = intX1 + PctGrafik1.Width / 100
     Else
         PctGrafik1.Line (intX1, intPoint3_11)-(intX1 +
PctGrafik1.Width / 1000, intPoint3_21), vbBlue
         If sglOutput1 >= 0 Then
            PctGrafik1.Line (intX1, intPoint1_11)-(intX1 +
PctGrafik1.Width / 1000, intPoint1_21), vbGreen
         Else
            PctGrafik1.Line (intX1, intPoint1_11)-(intX1 +
PctGrafik1.Width / 1000, intPoint1_21), vbRed
         End If
         PctGrafik1.Line (intX1, intPoint2_11)-(intX1 +
PctGrafik1.Width / 1000, intPoint2_21), vbBlack
         intX1 = intX1 + PctGrafik1.Width / 1000
     End If

     If intX1 >= (PctGrafik1.Width) Then
          intX1 = 0
```

```
          PctGrafik1_Resize
     End If
End Sub
Private Sub Program_Manual()
        AwalData
        i = 0
        Do While Ulang = True
           Kounter = Kounter + 1
           Waktu = Kounter * Val(txtSampling.Text)

           Call TerimaData1
           sglOutput1 = Val(TxtMV1.Text)
           KirimData1
           Call Gambar_Grafik1
           TxtTime.Text = Waktu
           TxtData.Text = Kounter
           TxtPV1.Text = sglDtPV1
           Tunda Val(txtSampling.Text) * 1000
           SimpanData
         Loop
         SaveFile
End Sub

Private Sub txtSP1_Change()
     If Not IsNumeric(TxtSP1) Then
          MsgBox "Data Bukan Numerik !", vbCritical + vbOKOnly,
"Error"
          TxtSP1.SetFocus
          TxtSP1.SelStart = 0
          TxtSP1.SelLength = Len(TxtSP1)
          Exit Sub
     Else
          If Val(TxtSP1.Text) < 0 Then
               MsgBox "Minimum Set Point adalah 0 % !",
vbExclamation, "Error"
               TxtSP1.SetFocus
               TxtSP1.SelStart = 0
               TxtSP1.SelLength = Len(TxtSP1)
               Exit Sub
          End If
     End If
End Sub
Private Sub txtSP1_LostFocus()
     byteSP1 = Val(TxtSP1.Text)
End Sub
Private Sub PV1scroll_change()
    If OptTest.Value = True Then
        TxtPV1.Text = PV1Scroll.Value / 2.55
    End If

End Sub
Private Sub AwalData()

On Error Resume Next
Err.Clear

   A = 3
   B = 1
```

```
Set ExcelWkb = ExcelApp.Workbooks.Add           'aktifkan workbook
ecell
Set ExcelSht = ExcelWkb.Worksheets(1)           'aktifkan workseet
ecell
   MyExcel = True
With ExcelSht                                   'pengaturan nama
pada kolom ecell
        .Cells.ColumnWidth = 9
        .Cells(1, 1) = "Diambil pada :"
        .Cells(1, 2) = Date
        .Cells(1, 3) = Time()
        .Cells(2, 1) = "SP"
        .Cells(2, 2) = "MV"
        .Cells(2, 3) = "PV"
        .Cells(2, 4) = "Time Sampling"
        .Cells(2, 5) = "time"
        .Cells(2, 6) = "Data "
End With
End Sub

Private Sub SaveFile()
Dim c, d, f, G As String
Dim e As Integer
e = 1

If Len(Dir(App.Path & "\Data\data.xls")) <> 0 Then
On Error Resume Next
Err.Clear

G = "\Data\data" & e & ".xls"
 Do While Len(Dir(App.Path & G)) <> 0
  G = "\Data\data" & e & ".xls"
  If Len(Dir(App.Path & G)) = 0 Then
    Exit Do
  End If
  e = e + 1
 Loop
   ExcelWkb.SaveAs App.Path & G
Else
MkDir (App.Path & "\Data")
  ExcelWkb.SaveAs App.Path & "\Data\data.xls"
End If
ExcelWkb.Close True
If MyExcel Then
   ExcelApp.Quit
   End If
End Sub

Private Sub SimpanData()
'simpan data Proses KE excell

    ExcelSht.Cells(A, 1) = TxtSP1.Text
    ExcelSht.Cells(A, 2) = TxtMV1.Text
    ExcelSht.Cells(A, 3) = TxtPV1.Text
    ExcelSht.Cells(A, 4) = txtSampling.Text
    ExcelSht.Cells(A, 5) = TxtTime.Text
    ExcelSht.Cells(A, 6) = TxtData.Text
    A = A + 1
End Sub
```

- Form input parameter

```
Option Explicit


Private Sub cmdDErrCancel_Click()
     AdoDError.Refresh
     cmdDErrNew.Caption = "New"
End Sub

Private Sub cmdDErrNew_Click()
     Dim i As Integer

     With AdoDError.Recordset
          If cmdDErrNew.Caption = "New" Then
               .AddNew
               cmdDErrNew.Caption = "Save"
               txtDErr(0).SetFocus
          ElseIf cmdDErrNew.Caption = "Save" Then
               .Update
               cmdDErrNew.Caption = "New"
          End If
     End With
End Sub

Private Sub cmdErrCancel_Click()
     AdoError.Refresh
     cmdErrNew.Caption = "New"
End Sub

Private Sub cmdErrNew_Click()
     Dim i As Integer

     With AdoError.Recordset
          If cmdErrNew.Caption = "New" Then
               .AddNew
               cmdErrNew.Caption = "Save"
               txtError(0).SetFocus
          ElseIf cmdErrNew.Caption = "Save" Then
               .Update
               cmdErrNew.Caption = "New"
          End If
     End With
End Sub

Private Sub cmdExit_Click()
    Unload Me
```

```
    frmInputParameter.Hide
    End
End Sub

Private Sub cmdNext_Click()
     frmInputParameter.Hide
     frmFuzzy.Show
     Call frmFuzzy.GarisGrafik1

End Sub

Private Sub cmdOutCancel_Click()
     AdoOutput.Refresh
     cmdOutNew.Caption = "New"
End Sub

Private Sub cmdOutNew_Click()
     Dim i As Integer

     With AdoOutput.Recordset
          If cmdOutNew.Caption = "New" Then
               .AddNew
               cmdOutNew.Caption = "Save"
               txtOut(0).SetFocus
          ElseIf cmdOutNew.Caption = "Save" Then
               .Update
               cmdOutNew.Caption = "New"
          End If
     End With
End Sub

Private Sub cmdTblCancel_Click()
     AdoKebenaran.Refresh
     cmdTblNew.Caption = "New"
End Sub

Private Sub cmdTblNew_Click()
     Dim i As Integer

     With AdoKebenaran.Recordset
          If cmdTblNew.Caption = "New" Then
               .AddNew
               cmdTblNew.Caption = "Save"
               txtErrNB(0).SetFocus
          ElseIf cmdTblNew.Caption = "Save" Then
               .Update
               cmdTblNew.Caption = "New"
          End If
     End With
End Sub

Private Sub Form_Load()
     frmInputParameter.WindowState = 2

End Sub

Private Sub Form_Unload(Cancel As Integer)
     frmInputParameter.Hide
     frmFuzzy.Show
```

```
End Sub

Private Sub txtDErr_Change(Index As Integer)
     Dim i As Integer

     For i = 0 To 6
         sglDErr1(i) = Val(txtDErr(i))
     Next
End Sub

Private Sub txtErrNB_Change(Index As Integer)
     Dim i As Integer

     For i = 0 To 6
          txtErrNB(i) = UCase(txtErrNB(i))
          strNB1(i) = txtErrNB(i)
     Next
End Sub

Private Sub txtErrNK_Change(Index As Integer)
     Dim i As Integer

     For i = 0 To 6
          txtErrNK(i) = UCase(txtErrNK(i))
          strNK1(i) = txtErrNK(i)
     Next
End Sub

Private Sub txtErrNS_Change(Index As Integer)
     Dim i As Integer

     For i = 0 To 6
          txtErrNS(i) = UCase(txtErrNS(i))
          strNS1(i) = txtErrNS(i)
     Next
End Sub

Private Sub txtError_Change(Index As Integer)
     Dim i As Integer

     For i = 0 To 6
          sglErr1(i) = Val(txtError(i))
     Next
End Sub

Private Sub txtErrPB_Change(Index As Integer)
     Dim i As Integer

     For i = 0 To 6
          txtErrPB(i) = UCase(txtErrPB(i))
          strPB1(i) = txtErrPB(i)
     Next
End Sub

Private Sub txtErrPK_Change(Index As Integer)
     Dim i As Integer

     For i = 0 To 6
          txtErrPK(i) = UCase(txtErrPK(i))
```

```
          strPK1(i) = txtErrPK(i)
     Next
End Sub

Private Sub txtErrPS_Change(Index As Integer)
     Dim i As Integer

     For i = 0 To 6
          txtErrPS(i) = UCase(txtErrPS(i))
          strPS1(i) = txtErrPS(i)
     Next
End Sub

Private Sub txtErrZ_Change(Index As Integer)
     Dim i As Integer

     For i = 0 To 6
          txtErrZ(i) = UCase(txtErrZ(i))
          strZ1(i) = txtErrZ(i)
     Next
End Sub

Private Sub txtOut_Change(Index As Integer)
     Dim i As Integer

     For i = 0 To 6
          sglOut1(i) = Val(txtOut(i))
     Next
End Sub
```

- Mdi fuzzy

```
Option Explicit

Private Sub itmExit_Click()
     Unload frmInputParameter
     Unload frmFuzzy
     mdiFuzzy.Hide
     End
End Sub

Private Sub itmNew_Click()
     frmInputParameter.Show
     itmNew.Enabled = False
End Sub
```

- Module fuzzy

```
Option Explicit

'Deklarasi file DLL.
Public Declare Sub Tunda Lib "Port_IO.dll" (ByVal lama As Integer)
Public Declare Sub Port_Out Lib "Port_IO.dll" (ByVal nPort As
Integer, ByVal nData As Byte)
Public Declare Function Port_In Lib "Port_IO.dll" (ByVal nPort As
Integer) As Byte
```

```
Public Declare Function Shift_Kiri Lib "Port_IO.dll" (ByVal DATA
As Byte, ByVal kali As Byte) As Byte
Public Declare Function Shift_Kanan Lib "Port_IO.dll" (ByVal DATA
As Byte, ByVal kali As Byte) As Byte

'variable fuzzy 1
'-------------------------
     Public byteOptFuzzy As Byte

     Public strNB1(6), strNS1(6), strNK1(6), strZ1(6) As String
     Public strPK1(6), strPS1(6), strPB1(6) As String

     Public sglErr1(6), sglDErr1(6), sglOut1(6) As Single

     Public bytePV1 As Byte
     Public sglDError1, sglErr_11, sglErr_21 As Single
     Public sglDtPV1, byteSP1 As Single

     Public sglErrOut1(1) As Single
     Public strErr_11, strErr_21 As String
     Public sglDErrOut1(1) As Single
     Public strDErr_11, strDErr_21 As String
     Public strHasil1(3) As String
     Public sglDataOut1(3) As Single
     Public intDataOutput1 As Integer
     Public intSumbuX1, intX1, intPoint1_11, intPoint1_21,
intPoint2_11, intPoint2_21, intPoint3_11, intPoint3_21 As Integer
     Public sglOutput1 As Single

     Public Kounter, Waktu As Double
```