ITS
Institut
Teknologi
Sepuluh Nopember

TUGAS AKHIR - TE 091399

PERANCANGAN DAN IMPLEMENTASI *DIRECT TORQUE CONTROL* UNTUK PENGATURAN KECEPATAN MOTOR INDUKSI 3 FASA MENGGUNAKAN KONTROLER FUZZY PI

Ika Sasmita Aji
NRP 2209105046

Dosen Pembimbing
Dr. Ir. Mochammad Rameli
Ir. Rusdhianto Effendi A.K, M.T.

JURUSAN TEKNIK ELEKTRO
Fakultas Teknologi Industri
Institut Teknologi Sepuluh Nopember
Surabaya 2011

PERNYATAAN KEASLIAN

PERNYATAAN KEASLIAN
TUGAS AKHIR

Dengan ini saya menyatakan bahwa isi sebagian maupun keseluruhan Tugas Akhir saya dengan Judul "Perancangan dan Implementasi *Direct Torque Control* Untuk pengaturan Kecepatan Motor Induksi 3 Fasa Menggunakan Kontroler *Fuzzy PI"* adalah benar benar hasil karya intelektual mandiri, diselesaikan tanpa menggunakan bahan-bahan yang tidak diijinkan dan bukan merupakan karya pihak lain yang saya akui sebagai karya sendiri.

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

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

Surabaya, 20 Juni 2011

Ika Sasmita Aji
NRP. 2209105046

LEMBAR PENGESAHAN

PERANCANGAN DAN IMPLEMENTASI *DIRECT TORQUE CONTROL* UNTUK PENGATURAN KECEPATAN MOTOR INDUKSI 3 FASA MENGGUNAKAN KONTROLER FUZZY PI

TUGAS AKHIR

Diajukan Guna Memenuhi Sebagian Persyaratan
Untuk Memperoleh Gelar Sarjana Teknik
Pada
Bidang Studi Teknik Sistem Pengaturan
Jurusan Teknik Elektro
Institut Teknologi Sepuluh Nopember

Menyetujui:

Dosen Pembimbing 1,	Dosen Pembimbing 2,
Dr.Ir.Mochammad Rameli	**Ir. Rusdhianto Effendi A.K, MT.**
NIP: 195412271981031002	**NIP: 195704241985021001**

SURABAYA
JUNI, 2011

ABSTRAK

PERANCANGAN DAN IMPLEMENTASI *DIRECT TORQUE CONTROL* UNTUK PENGATURAN KECEPATAN MOTOR INDUKSI 3 FASA MENGGUNAKAN KONTROLER FUZZYPI

Ika Sasmita Aji - 2209105046

Dr. Ir. Mochammad Rameli - 195412271981031002
Ir. Rusdhianto Effendi A.K, MT. - 195704241985021001

ABSTRAK

Motor induksi merupakan motor penggerak yang paling banyak digunakan dalam bidang industri. Keunggulan motor induksi diantaranya adalah konstruksinya yang sederhana, harganya yang lebih murah dibandingkan motor jenis lain, serta perawatannya yang mudah. Sedangkan, kekurangan dari motor induksi adalah teknik pengaturan kecepatannya yang relatif sulit dan membutuhkan arus *starting* yang tinggi sekitar enam kali arus nominal motor.Salah satu metode yang dikembangkan dalam pengaturan kecepatan motor induksi adalah *Direct Torque Control* (DTC).

Pada tugas akhir ini dicoba untuk melakukan perancangan dan implementasi *Direct Torque Control* (DTC) untuk mengatur kecepatan motor induksi 3 fasa dengan menggunakan kontroler Fuzzy PI agar kecepatan motor induksi dapat stabil sesuai dengan keinginan. Dari *setpoint*1400 rpm ternyata kecepatan motor yang terukur 1400 rpm. Namun pada kecepatan dibawah 1400 rpm terjadi *overshoot* sehingga perlu *tuning* nilai Ki dan pada batas Ki tertentu terjadi osilasi. Pada saat diberi *setpoint* 1000 rpm maka agar *overshoot* kecil Ki bernilai 5

Kata kunci: *Direct torque control, Kontroler Fuzzy PI,* Motor Induksi, *Inverter.*

ABSTRACT

DESIGN AND IMPLEMENTATION OF DIRECT TORQUE CONTROL (DTC) FOR MOTOR INDUCTION SPEED CONTROL USING PI FUZZY CONTROLLER

Ika Sasmita Aji - 2209105046

Dr. Ir. Mochammad Rameli - 195412271981031002
Ir. Rusdhianto Effendi A.K, MT. - 195704241985021001

ABSTRACT

Induction motor is the most widely used in industry. The advantages of induction motor such as simple construction, the price is cheaper than other types of motors, as well as easy maintenance. Meanwhile, the lack of induction motor speed regulation technique is relatively difficult and requires a high starting currents of about six times the motor nominal current. One method that was developed in the induction motor speed control is Direct Torque Control (DTC).

In this final project tried to perform design and implementation of Direct Torque Control (DTC) to control the speed of 3 phase induction motor using PI fuzzy controller for induction motor speed can be stabilized in accordance with the wishes.Setpoint of 1400 rpm was measured motor speed 1400 rpm. However, at speeds below 1400 rpm overshoot so it needs tuning value and at the boundary of Kisome oscillations occur. At 1000 rpm the setpoint given to a small overshoot Ki-value 5.

***Keyword** : Direct torque control, PI Fuzzy controller, Induction Motor, Inverter.*

KATA PENGANTAR

KATA PENGANTAR

Syukur Alhamdulillah, puja dan puji syukur penulis panjatkan kehadirat *Allah SWT* karena atas rahmat dan hidayat-Nya penulis dapat menyelesaikan tugas akhir ini dengan judul :

Perancangan Dan Implementasi *Direct Torque Control* untuk Pengaturan Kecepatan Motor Induksi 3 Fasa Menggunakan Kontroler Fuzzy PI

Dalam pengerjaan tugas akhir ucapan terima kasih yang sebesar-besarnya penulis sampaikan kepada :

1. Allah SWT Subhanallah, Alhamdulillah, Wa Laa ilaa ha Illallah, Allahu Akbar
2. Kedua orang tua tercinta yang telah memberikan segala bentuk dukungan, doa dan nasehat kepada penulis.
3. Bapak Dr. Ir. Mochammad Rameli dan Bapak Ir. Rusdhianto Effendi A.K, M.T selaku Dosen Pembimbing I dan Dosen Pembimbing II atas bimbingannya dalam pembuatan tugas akhir ini.
4. Seluruh keluarga besarku atas doa restunya padaku.
5. *Someone special* atas cinta dan kasih sayangnya, perhatian, semangat dan doanya yang tulus.
6. Teman-teman Lintas Jalur Gasal 2009, teman-teman SP yang sudah seperti keluarga sendiri selama di elektro.
7. Pihak lain yang tidak bisa disebutkan satu persatu yang telah membantu penyusunan tugas akhir ini.

Penulis sangat mengharapkan saran dan kritik. Penulis juga memohon maaf atas segala kekurangan yang ada dalam penulisan buku ini. Semoga tugas akhir ini dapat memberikan manfaat bagi kita semua.

Surabaya, Juni 2011

Penulis

DAFTAR ISI

DAFTAR ISI

Halaman

DAFTAR GAMBAR

DAFTAR GAMBAR

Halaman

DAFTAR TABEL

DAFTAR TABEL

Halaman

BAB I
PENDAHULUAN

BAB I
PENDAHULUAN

1.1. Latar Belakang

Motor induksi merupakan motor penggerak yang paling banyak digunakan dalam bidang industri. Dengan perkembangan dan kemajuan teknologi dibidang mikrokontroler, mikrokomputer, dan teori kontrol mempermudah operasi dan kinerja dari motor induksi sehingga dapat menggantikan peran motor DC sebagai penggerak elektrik. Keunggulan motor induksi diantaranya adalah konstruksinya yang sederhana seperti yang ditunjukkan pada Gambar 1.1, harganya yang lebih murah dibandingkan motor jenis lain, serta perawatannya yang mudah. Kekurangan dari motor induksi adalah teknik pengaturan kecepatannya yang relatif sulit dan membutuhkan arus *starting* yang tinggi sekitar enam kali arus nominal motor.

Gambar 1.1 Konstruksi dari motor induksi

Pengaturan kecepatan motor induksi dapat dilakukan dengan berbagai cara seperti kontrol tegangan/frekuensi (V/f) atau dikenal dengan kontrol skalar, serta kontrol vektor yang mengatur secara langsung arus stator motor. Metode kontrol vektor yang sekarang ini terus dikembangkan adalah metode *Direct Torque Control* (DTC). Yaitu suatu teknik kontrol yang mengarah pada pengaturan dengan nilai torsi yang berubah sesuai kebutuhan beban. Keunggulan dari penggunaan metode DTC adalah performansi motor yang baik, dan bisa dilakukan tanpa menggunakan sensor kecepatan.

Pada Tugas Akhir ini metode *Direct Torque Control* (DTC) diimplementasikan secara nyata pada motor induksi 3 fasa. Selain itu

ditambahkan juga kontroler *Fuzzy PI (Proportional Integral)* untuk mengatur kecepatan motor induksi agar sesuai dengan kecepatan yang diinginkan.

1.2. Permasalahan

Dalam membangun suatu sistem pengaturan torsi motor diperlukan perancangan dan implementasi secara nyata pada motor induksi. Sistem DTC yang akan dirancang memerlukan pemahaman mendalam tentang motor induksi dan prinsip kerja dari DTC. Pemilihan vektor tegangan yang tepat yang akan menentukan kondisi transistor *switching* pada inverter. Perubahan nilai inverter akan menentukan besarnya daya masukan yang akan digunakan motor induksi untuk berputar. Selain itu diperlukan pula kontroler *Fuzzy PI* agar diperoleh kecepatan putar motor induksi yang diinginkan

1.3. Batasan Masalah

Pada Tugas Akhir ini, pembahasan masalah dibatasi pada implementasi metode DTC dengan kontroler *Fuzzy PI* dan karakteristik dari motor induksi saja. *Plant* motor induksi dan parameter yang digunakan telah diketahui dan tersedia sehingga Tugas Akhir ini tidak membahas tentang pemodelan motor induksi.

1.4. Tujuan Penelitian

Perancangan dan implementasi DTC dengan kontroler *Fuzzy PI* ini bertujuan untuk mengetahui secara nyata penerapan DTC pada motor induksi. Kelebihan dan kekurangan DTC dikombinasikan dengan kontroler *Fuzzy* PI. Kontroler *Fuzzy* PI digunakan untuk mengatur kecepatan motor induksi. Diharapkan nantinya dapat diterapkan seara nyata pula dalam dunia industri.

1.5. Metodologi

Untuk dapat merealisasikan tujuan yang direncanakan, maka dalam pengerjaan Tugas Akhir ini dilakukan beberapa tahapan. Pertama, adalah mempelajari karakteristik dan prinsip kerja motor induksi. Selanjutnya adalah mempelajari metode DTC dan membangun implementasi DTC dengan menggunakan sistem *loop* terbuka. Kemudian menambahkan kontroler *Fuzzy PI.* Semua hasil yang diperoleh selanjutnya akan dilakukan analisis respon. Langkah terakhir adalah membuat suatu kesimpulan dari hasil perancangan dan hasil

analisis penelitian ini, sehingga dapat menjadi acuan untuk pengembangan penelitian selanjutnya.

1.6. Relevansi

Hasil yang diperoleh diharapkan menjadikan pengaturan motor induksi lebih mudah, dan dapat mengikuti nilai acuan yang diberikan. Hal terpenting, Tugas Akhir ini dapat menjadi acuan untuk penelitian selanjutnya tentang metode DTC dalam pengaturan motor induksi.

1.7. Sistematika Penulisan

Laporan Tugas Akhir ini memiliki sistematika penulisan sebagai berikut :

Bab I : PENDAHULUAN
Pada bab ini, dijelaskan tentang latar belakang, permasalahan, tujuan, batasan masalah, metodologi, relevansi, dan sistematika penulisan dari penelitian ini.

Bab II : TEORI PENUNJANG
Dasar pemikiran dan pengetahuan dari sistem DTC yang akan dirancang seperti teori motor induksi, konsep DTC, dan inverter serta kontroler Fuzzy PI

Bab III : PERANCANGAN SISTEM
Perancangan sistem terbagi menjadi perancangan perangkat keras (*hardware*) dan perancangan perangkat lunak (*software*).

Bab IV : PENGUJIAN DAN ANALISA
Merupakan hasil implementasi dan pengujian sistem secara keseluruhan.

Baba V : PENUTUP
Pada bab ini, diuraikan tentang kesimpulan akhir dari penelitian ini dan saran – saran mengenai kemungkinan pengembangan penelitian ini.

BAB II

TEORI PENUNJANG

BAB II
TEORI PENUNJANG

Dalam bab ini akan dijelaskan tentang teori-teori yang mendukung dalam pembuatan implementasi *Direct Torque Control* mulai dari motor induksi, teori DTC, *Inverter*, Mikrokontroler, dan rangkaian pendukung lainnya seperti : *Power supply* DC, DAC, dan Op-amp

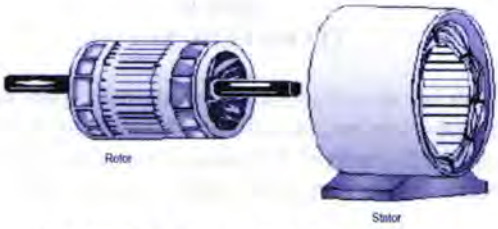
2.1 Motor Induksi

Motor induksi adalah motor listrik arus bolak-balik (AC) yang putaran rotornya tidak sama dengan putaran medan putar pada stator, dengan kata lain putaran rotor dengan putaran medan pada stator terdapat selisih putaran yang disebut *slip*.

Motor induksi adalah motor yang memiliki konstruksi yang baik, harganya lebih murah dan mudah dalam pengaturan kecepatannya, stabil ketika berbeban dan mempunyai efisiensi tinggi. Mesin induksi adalah mesin AC yang paling banyak digunakan dalam industri dengan skala besar atau kecil dan juga dalam rumah tangga. Alasannya adalah karena karakteristiknya hampir sesusai dengan kebutuhan dunia industri, pada umumnya dalam kaitannya dengan harga, pemeliharaan, dan kestabilan kecepatan. Motor induksi (asinkron) ini pada umumnya hanya memiliki satu suplai tenaga yang mengeksitasi belitan stator. Belitan rotornya tidak terhubung langsung dengan sumber tenaga listrik, melainkan belitan ini dieksitasi oleh induksi dari perubahan medan magnetik yang disebabkan oleh arus pada belitan stator.

2.1.1. Konstruksi Motor Induksi

Motor induksi adalah motor AC yang paling banyak dipergunakan, karena konstruksinya yang kuat dan karakteristik kerjanya yang baik. Secara umum motor induksi terdiri dari rotor dan stator. Rotor merupakan bagian yang bergerak, sedangkan stator bagian yang diam. Diantara stator dengan rotor ada celah udara yang jaraknya sangat kecil. Konstruksi motor induksi dapat diperlihatkan pada Gambar 2.1.



Gambar 2.1 Konstruksi motor induksi

2.1.2. Jenis Motor Induksi Tiga Fasa

Ada dua jenis motor induksi tiga fasa berdasarkan rotornya yaitu:

1. Motor induksi tiga fasa rotor sangkar tupai (*squirrel-cage motor*)
2. Motor induksi tiga fasa rotor belitan (*wound-rotor motor*)

Kedua motor ini bekerja pada prinsip yang sama dan mempunyai konstruksi stator yang sama tetapi berbeda dalam konstruksi rotor.

2.1.3. Klasifikasi motor induksi

Klasifikasi motor induksi dibagi ke dalam dua kelompok, yaitu motor induksi satu fasa dan motor induksi tiga fasa. Motor induksi satu fasa hanya memiliki satu gulungan *stator*, beroperasi dengan pasokan daya satu fasa, memiliki sebuah rotor sangkar, dan memerlukan sebuah alat untuk menghidupkan motornya. Motor ini merupakan jenis motor yang paling umum digunakan dalam peralatan rumah tangga, seperti *fan* angin, mesin cuci dan pengering pakaian. Penggunaan motor ini umumnya 3 sampai 4 HP.

Sedangkan pada motor induksi tiga fasa, medan magnet yang berputar dihasilkan oleh pasokan tiga fasa yang seimbang. Motor tiga fasa memiliki kemampuan daya yang tinggi dan dapat memiliki rotor sangkar ataupun rotor belitan (walaupun 90% memiliki rotor sangkar) serta penyalaan sendiri. Diperkirakan bahwa sekitar 70% motor di industri menggunakan jenis ini untuk pompa, kompresor, *belt conveyor*, jaringan listrik , dan *grinder*. Umumnya tersedia dalam ukuran 1/3 hingga ratusan HP.

2.1.4. Prinsip Kerja Motor Induksi Tiga Fasa[2]

Motor induksi adalah peralatan pengubah energi listrik ke bentuk energi mekanik. Pengubahan energi ini bergantung pada keberadaan phenomena alami magnetik, medan listrik, gaya mekanis dan gerak.

Jika pada belitan stator diberi tegangan tiga fasa, maka pada belitan stator akan mengalir arus tiga fasa, arus ini menghasilkan medan magnet yang berputar dengan kecepatan sinkron (n_s). Medan magnet ini akan memotong belitan rotor, sehingga pada belitan rotor akan diinduksikan tegangan. Arus yang mengalir pada belitan rotor berada dalam medan magnet yang dihasilkan stator, sehingga pada belitan rotor akan dihasilkan gaya (F). Gaya ini akan menghasilkan torsi (τ) dan jika torsi yang dihasilkan lebih besar dari torsi beban, maka rotor akan berputar dengan kecepatan n_r yang searah dengan medan putar stator.

Untuk memperjelas prinsip kerja motor induksi tiga fasa, maka dapat dijabarkan dalam langkah – langkah berikut:

1. Ketika tegangan tiga fasa yang seimbang diberikan pada belitan stator, maka belitan stator akan menghasilkan arus yang mengalir pada tiap – tiap fasanya.
2. Arus pada setiap fasa stator akan menghasilkan fluk yang berubah terhadap waktu.
3. Amplitudo fluk yang dihasilkan pada fasa stator berubah secara sinusoidal dan arahnya tegak lurus terhadap belitan.
4. Penjumlahan dari ketiga fluk pada belitan stator disebut medan putar yang berputar dengan kecepatan sinkron (n_s), besarnya nilai n_s ditentukan oleh jumlah kutub p dan frekuensi stator f yang dirumuskan dengan.

   $$n_s = \frac{120 \times f}{p} (\text{rpm}) \qquad (2.1)$$

5. Akibat fluk yang berputar tersebut maka timbul tegangan induksi pada belitan stator yang besarnya dapat dinyatakan dengan persamaan berikut.

   $$e_1 = -N_1 \frac{d\Psi}{dt} (\text{Volt}) \qquad (2.2)$$

   Atau

   $$E_1 = 4{,}44 f N_1 \Psi_{max} (\text{Volt}) \qquad (2.3)$$

6. Fluk yang berputar tersebut juga memotong belitan rotor. Akibatnya pada belitan rotor akan dihasilkan tegangan induksi

(ggl) sebesar E_2 yang besarnya dapat dinyatakan dengan persamaan berikut.

$$e_2 = -N_2 \frac{d\Psi}{dt} \text{(Volt)} \quad (2.4)$$

Atau

$$E_2 = 4{,}44 f N_2 \Psi_{max} \text{(Volt)} \quad (2.5)$$

di mana : E_2 = Tegangan induksi pada rotor saat rotor dalam keadaan diam (Volt).
N_2 = Jumlah lilitan kumparan rotor
Ψ_{max} = Fluk maksimum(Wb)

7. Karena kumparan rotor merupakan rangkaian tertutup, maka tegangan induksi tersebut akan menghasilkan arus I_2.
8. Arus I_2 ini berada pada medan magnet yang dihasilkan oleh stator, sehingga pada belitan rotor akan dihasilkan gaya (F).
9. Gaya (F) ini akan akan menghasilkan torsi (τ), jika torsi yang dihasilkan ini lebih besar dari torsi beban, maka rotor akan berputar dengan kecepatan n_r yang searah dengan medan putar stator.
10. Ada Perbedaan kecepatan medan putar pada stator (n_s) dengan kecepatan putaran rotor (n_r), perbedaan ini disebut *slip* (s) yang dapat dinyatakan dengan persamaan berikut

$$s = \frac{n_s - n_r}{n_s} \times 100\% \quad (2.6)$$

11. Setelah rotor dalam keadaan berputar, besarnya tegangan yang diinduksikan pada belitan rotor akan dipengaruhi atau tergantung terhadap *slip* (s). Tegangan induksi pada rotor dalam keadaan ini dapat dinyatakan dengan persamaan berikut.

$$E_2 = 4{,}44 f N_2 \Psi_{max} \text{(Volt)}$$
$$E_{2s} = s E_2 \text{(Volt)} \quad (2.7)$$

di mana E_{2s} = tegangan induksi pada rotor dalam keadaan berputar (Volt).
$f_2 = s \times f$ = frekuensi rotor (frekuensi tegangan induksi pada rotor dalam keadaan berputar) .

12. Akibat adanya *slip* (s), maka nilai frekuensi pada rotor (f_2) dan reaktansi rotor (x_2') akan dipengaruhi oleh *slip*, yang dapat dinyatakan dengan s f dan s x_2'.
13. Jika kecepatan putaran rotor (n_r) sama dengan kecepatan medan putar stator (n_s), maka *slip* bernilai nol, tidak ada fluk yang memotong belitan rotor sehingga pada belitan rotor tidak

diinduksikan tegangan, maka tidak ada arus yang mengalir pada belitan rotor, sehingga rotor tidak berputar, karena tidak ada gaya yang terjadi pada rotor.

2.1.5. *Slip*

Motor induksi tidak dapat berputar pada kecepatan sinkron. Seandainya hal ini terjadi, maka rotor akan tetap diam relatif terhadap fluk yang berputar. Maka tidak akan ada ggl yang diinduksikan dalam rotor, tidak ada arus yang mengalir pada rotor, dan karenanya tidak akan menghasilkan kopel. Kecepatan rotor sekalipun tanpa beban, harus lebih kecil sedikit dari kecepatan sinkron agar adanya tegangan induksi pada rotor, dan akan menghasilkan arus di rotor, arus induksi ini akan berinteraksi dengan fluk listrik sehingga menghasilkan kopel. Selisih antara kecepatan rotor dengan kecepatan sinkron disebut *slip* (s). *Slip* dapat dinyatakan dalam putaran setiap menit, tetapi lebih umum dinyatakan sebagai persen dari kecepatan sinkron. persamaan (2.6) di atas memberikan imformasi yaitu:

1. Saat s = 1 di mana $n_r = 0$, ini berati rotor masih dalam keadaan diam atau akan berputar.
2. S = 0 menyatakan bahwa $n_s = n_r$, ini berarti rotor berputar sampai kecepatan sinkron. Hal ini dapat terjadi jika ada arus dc yang diinjeksikan ke belitan rotor, atau rotor digerakkan secara mekanik.
3. $0 < s < 1$, ini berarti kecepatan rotor diantara keadaan diam dengan kecepatan sinkron. Kecepatan rotor dalam keadaan inilah dikatakan kecepatan tidak sinkron.

2.1.6. Frekuensi Rotor

Ketika rotor masih dalam keadaan diam, di mana frekuensi arus pada rotor sama seperti frekuensi masukan (sumber). Tetapi ketika rotor akan berputar, maka frekuensi rotor akan bergantung kepada kecepatan relatif atau bergantung terhadap besarnya *slip*.

Untuk besar *slip* tertentu, maka frekuensi rotor sebesar f' yaitu,

$$n_s - n_r = \frac{120 \times f'}{p}$$

$$n_s = \frac{120 \times f}{p} \qquad (2.8)$$

Dengan membagikan dengan salah satu, maka didapatkan :

$$\frac{f'}{f} = \frac{n_s - n_r}{n_s} = s \qquad (2.9)$$

Maka $f' = s \times f\ (Hz)$

Telah diketahui bahwa arus rotor bergantung terhadap frekuensi rotor $f' = s \times f$ dan ketika arus ini mengalir pada masing – masing fasa di belitan rotor, akan memberikan reaksi medan magnet. Biasanya medan magnet pada rotor akan menghasilkan medan magnet yang berputar yang besarnya bergantung atau relatif terhadap putaran rotor sebesar sn_s.

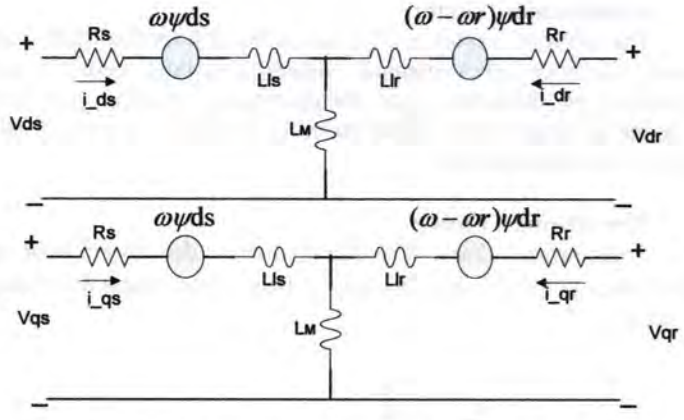
Pada keadaan tertentu, arus rotor dan arus stator menghasilkan distribusi medan magnet yang sinusoidal di mana medan magnet ini memiliki magnetudo yang konstan dan kecepatan medan putar n_s yang konstan. Kedua Hal ini merupakan medan magnetik yang berputar secara sinkron. Kenyataannya tidak seperti ini karena pada stator akan ada arus magnetisasi pada kumparannya.

2.1.7. Rangkaian Ekivalen Motor Induksi Tiga Fasa[3]

Dalam operasi riil ditemui permasalahan tegangan sumber yang tidak sinusoidal dan adanya perubahan beban. Oleh karena itu dibutuhkan model lain yang lebih fleksibel untuk mempermudah menganalisis motor induksi. Model motor induksi dalam koordinat dq digunakan untuk menganalisis motor dengan lebih komprehensif. Pertama ditinjau bahwa keadaan motor induksi dapat dianggap sebagai transformator di mana stator merupakan primer dan rotor sebagai rangkaian sekunder. Kemudian diasumsikan tegangan sumber adalah sinusoidal dan motor berada pada kondisi *steady state.* Sehingga didapatkan rangkaian ekivalen motor seperti terlihat pada Gambar 2.2.

Untuk mempermudah analisa dalam pengaturan posisi ataupun kecepatan motor induksi dan juga agar motor induksi 3 fasa dapat memiliki sifat atau perilaku yang menyerupai motor DC yang mudah dikontrol maka memerlukan suatu transformasi dari koordinat abc menjadi koordinat dq. Sehingga didapatkan persamaan tegangan stator dan rotor motor induksi dinyatakan dengan Persamaan 2.10.

$$
\begin{aligned}
V_{qs} &= r_s\, i_{qs} + \omega\, \Psi_{ds} + p\, \Psi_{qs} \\
V_{ds} &= r_s\, i_{ds} - \omega\, \Psi_{ds} + p\, \Psi_{qs} \\
V_{qr} &= r_r\, i_{qr} + (\omega-\omega_r)\, \Psi_{dr} + p\, \Psi_{qr} \\
V_{dr} &= r_r\, i_{dr} - (\omega-\omega_r)\, \Psi_{qr} + p\, \Psi_{dr}
\end{aligned}
\qquad (2.10)
$$



Gambar 2.2 Rangkaian ekivalen dq motor induksi 3 fasa

Sedangkan fluks yang tercakup dalam kumparan dinyatakan dengan,

$$\Psi_{qs} = Ll_s\, i_{qs} + LM\,(i_{qs} + i_{qr})$$
$$\Psi_{ds} = Ll_s\, i_{ds} + LM\,(i_{ds} + i_{dr})$$
$$\Psi_{qr} = Ll_s\, i_{qr} + LM\,(i_{qs} + i_{qr})$$
$$\Psi_{dr} = Ll_s\, i_{dr} + LM\,(i_{ds} + i_{dr}) \qquad (2.11)$$

Persamaan–persamaan diatas dapat dinyatakan dalam bentuk matrik berikut :

$$\begin{bmatrix} V_{qs} \\ V_{ds} \\ V_{qr} \\ V_{dr} \end{bmatrix} = \begin{bmatrix} r_s + pL_s & \omega L_s & pL_M & \omega L_M \\ -\omega L_s & r_s + pL_s & -\omega L_M & pL_M \\ pL_M & (\omega - \omega_r)L_M & r_r + pL_r & (\omega - \omega_r)L_r \\ -(\omega - \omega_r)L_M & pL_M & -(\omega - \omega_r)L_r & r_r + pL_r \end{bmatrix} \qquad (2.12)$$

dengan

$$L_s = Ll_s + LM$$
$$L_r = Ll_r + LM \qquad (2.13)$$

Dimana,

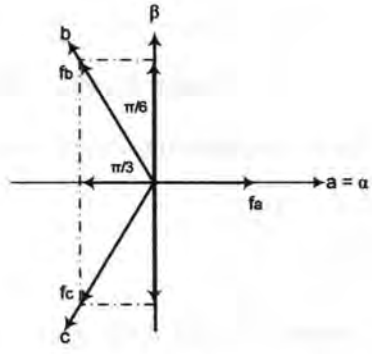
R_s = Resistansi Stator
R_r = Resistansi Rotor
L_s = Induktansi Stator
L_r = Induktansi Rotor
LM = Induktansi Mutual
p = Pasang Kutub Motor
ω = Kecepatan Motor
ω_r = Kecepatan Rotor

2.2. Transformasi Vektor

Transformasi vektor adalah transformasi koordinat dari sistem koordinat stasioner (abc) menjadi sistem koordinat berputar (dq). Transformasi ini dilakukan agar mempermudah analisis pengaturan motor induksi. Trasnformasi vektor memiliki 2 tahap, yaitu transformasi Clarke dan transformasi Park.

2.2.1. Transformasi *Clarke*

Transformasi *Clarke* adalah transformasi dari sistem koordinat tiga fasa (abc) ke dalam dua fasa ($\alpha\beta$) diam seperti yang ditunjukkan pada Gambar 2.3.



Gambar 2.3 Koordinat Transformasi Clarke

Dari Gambar 2.3, terlihat bahwa koordinat α sejajar dengan koordinat a pada 3-fasa. Sedangkan koordinat β tegak lurus dengan koordinat a atau dengan kata lain selisih 90^0 dengan koordinat α. Resultan vektor pada bidang α dan β dapat ditulis pada Persamaan 2.24 di mana f menyatakan fungsi yang ada pada motor induksi, baik arus, tegangan ataupun fluks.

$$[f_\alpha \quad f_\beta \quad f_o] = [f_a \quad f_b \quad f_c]\frac{2}{3}\begin{bmatrix} 1 & 0 & \frac{1}{2} \\ -\frac{1}{2} & \frac{\sqrt{3}}{2} & \frac{1}{2} \\ -\frac{1}{2} & -\frac{\sqrt{3}}{2} & \frac{1}{2} \end{bmatrix} \tag{2.24}$$

f_o merupakan pusat sumbu dengan nilai yang konstan yaitu 1. Dan 2/3 merupakan konstanta pada matrik transformasi Clarke.

Sedangkan *inverse* transformasi *Clarke* digunakan untuk mentransformasikan balik dari komponen αβ ke komponen abc melalui Persamaan 2.25.

$$i_a = i_\alpha$$
$$i_b = -\frac{1}{2}i_\alpha + \frac{\sqrt{3}}{2}i_\beta$$
$$i_c = -\frac{1}{2}i_\alpha - \frac{\sqrt{3}}{2}i_\beta \qquad (2.25)$$

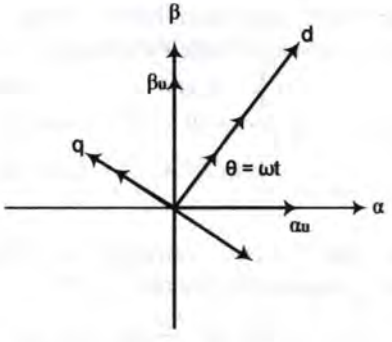
atau dalam bentuk matrik, persamaan diatas menjadi Persamaan 2.26 sebagai berikut:

$$\begin{pmatrix} i_a \\ i_b \\ i_c \end{pmatrix} = \begin{pmatrix} 1 & 0 \\ -1/2 & \sqrt{3}/2 \\ -1/2 & -\sqrt{3}/2 \end{pmatrix} \begin{pmatrix} i_\alpha \\ i_\beta \end{pmatrix} \qquad (2.26)$$

Persamaan 2.26 juga berlaku untuk perhitungan pada tegangan ataupun fluk motor.

2.2.2. Transformasi Park

Transformasi Park adalah transformasi dari sistem koordinat stationer (αβ) ke dalam sistem koordinat putar (dq) seperti yang ditunjukkan pada Gambar 2.4.



Gambar 2.4 Koordinat Transformasi Park

Gambar 2.4 menunjukkan karakteristik motor induksi yang mulanya berada pada sumbu stationer (αβ) lalu bekerja dan terjadi

perputaran rotor motor induksi sehingga fungsi arus, tegangan, dan fluk juga mengalami perubahan nilai. Dalam hal ini ditunjukkan dengan transformasi dari sumbu diam (αβ) ke sumbu putar (dq).

Transformasi dari sumbu αβ menjadi sumbu dq dapat ditulis dalam bentuk persamaan matrik sebagai berikut :

$$[f_d \quad f_q \quad f_o] = [f_\alpha \quad f_\beta \quad f_o] \begin{bmatrix} cos\,\theta & -sin\,\theta & 0 \\ sin\,\theta & cos\,\theta & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{2.27}$$

di mana f menyatakan fungsi yang ada pada motor induksi, baik arus, tegangan ataupun fluks.

Sedangkan *inverse* transformasi *Park* digunakan untuk mentransformasikan balik dari komponen dq ke komponen αβ melalui Persamaan 2.28 berikut.

$$i_\alpha = i_{ds}.\cos(\theta) - i_{qs}.\sin(\theta)$$
$$i_\beta = i_{ds}.\sin(\theta) + i_{qs}.\cos(\theta) \tag{2.28}$$

atau dalam bentuk matrik seperti pada Persamaan 2.2.6 berikut.

$$\begin{pmatrix} i_\alpha \\ i_\beta \end{pmatrix} = \begin{pmatrix} \cos(\theta) & -\sin(\theta) \\ \sin(\theta) & \cos(\theta) \end{pmatrix} \begin{pmatrix} i_{ds} \\ i_{qs} \end{pmatrix} \tag{2.29}$$

Persamaan 2.29 juga berlaku untuk perhitungan pada tegangan ataupun fluk motor.

Dari persamaan matrik transformasi Clarke dan Park yaitu Persamaan 2.24 dan 2.28, persamaan dalam bidang dq untuk motor induksi 3-fasa dapat dinyatakan dengan Persamaan 2.30.

$$[f_d \quad f_q \quad f_o] = \frac{2}{3}[f_a \quad f_b \quad f_c] \begin{bmatrix} cos\,\theta & -sin\,\theta & \frac{1}{2} \\ cos\,(\theta - \frac{2\pi}{3}) & -sin\,(\theta - \frac{2\pi}{3}) & \frac{1}{2} \\ cos\,(\theta + \frac{2\pi}{3}) & -sin\,(\theta + \frac{2\pi}{3}) & \frac{1}{2} \end{bmatrix} \tag{2.30}$$

Sedangkan dari *inverse* transformasi Clarke dan Park, persamaan motor induksi dalam bidang abc adalah:

$$\begin{pmatrix} f_a \\ f_b \\ f_c \end{pmatrix} = \begin{pmatrix} 1 & 0 \\ -1/2 & \sqrt{3}/2 \\ -1/2 & -\sqrt{3}/2 \end{pmatrix} \begin{pmatrix} \cos(\theta) & -\sin(\theta) \\ \sin(\theta) & \cos(\theta) \end{pmatrix} \begin{pmatrix} f_{ds} \\ f_{qs} \end{pmatrix}$$

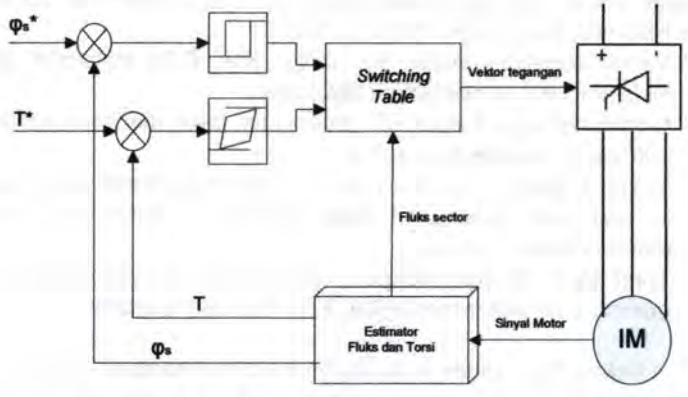
$$\begin{pmatrix} f_a \\ f_b \\ f_c \end{pmatrix} = \begin{pmatrix} cos(\theta) & -sin(\theta) \\ -0.5\,cos(\theta) + \sqrt{3}/2sin(\theta) & 0.5\,sin(\theta) + \sqrt{3}/2cos(\theta) \\ -0.5\,cos(\theta) - \sqrt{3}/2sin(\theta) & 0.5\,sin(\theta) - \sqrt{3}/2cos(\theta) \end{pmatrix} \begin{pmatrix} f_{ds} \\ f_{qs} \end{pmatrix} \quad (2.31)$$

di mana f menyatakan fungsi pada motor induksi, baik arus, tegangan ataupun fluk.

2.3 *Direct Torque Control (DTC)*

Suatu teknik kontrol yang lebih mengarah pada pengaturan dengan torsi yang berubah – ubah sesuai kebutuhan beban pada motor khususnya motor induksi dikenal dengan nama *Direct Torque Control* (DTC).

Beberapa keunggulan yang diperoleh bila menggunakan DTC dalam mengontrol torsi, yaitu performansi yang baik dan bisa dilakukan pengaturan motor tanpa menggunakan sensor kecepatan.



Gambar 2.5 Blok diagram DTC pada motor induksi

Gambar 2.5 adalah blok diagram dari sistem DTC untuk motor induksi. Konfigurasi DTC terdiri dari histeresis kontroler, *estimator* fluks dan torsi, dan sistem kontrol vektor tegangan sebagai masukan inverter. Transformasi koordinat antara frame stationer, frame sinkron, dan regulator PI tidak diperlukan pada sistem DTC. DTC juga tidak memerlukan *pulse width modulator* dan sensor posisi yang akan menimbulkan waktu tunda pada sistem. Konsep dasar dari DTC adalah

pengaturan langsung fluks stator (fluks magnetisasi) dan torsi elektromagnetik dengan pemilihan *switching* pada inverter secara tepat.

2.3.1 Konsep Dasar DTC

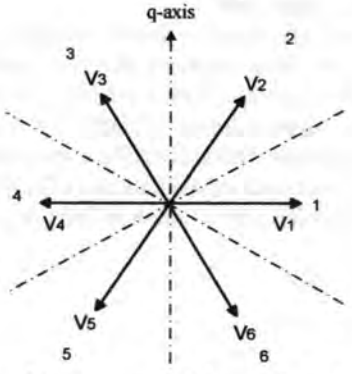
Dasar pemikiran dan prinsip kerja metode *Direct Torque Control* (DTC) pertama kali dikembangkan oleh Takahasi dan Noguchi pada tahun 1986, dapat diformulasikan seperti berikut ini **[4]**:

1. Nilai dari fluks stator merupakan perubahan atau penambahan nilai tegangan emf stator. Oleh karena itu nilai *magnitude* dari fluks stator sangat bergantung dari tegangan statornya.
2. Nilai torsi elektromagnetik berbanding lurus dengan nilai sinus dari sudut antara vektor fluks stator dan fluks rotor.
3. Perubahan nilai fluks rotor sangatlah kecil jika dibandingkan fluks stator.

Berdasarkan konsep dasar DTC diatas, nilai *magnitude* dari fluks stator dan torsi dapat dikontrol secara langsung dengan melakukan pemilihan vektor tegangan secara tepat, yaitu pemilihan state inverter secara berurutan dengan spesifikasi berikut :

1. Vektor tegangan bukan nol yang sudut fluks statornya tidak melebihi $\pm 90^0$ menyebabkan fluks naik.
2. Vektor tegangan bukan nol yang sudut fluks statornya melebihi $\pm 90^0$ menyebabkan fluks turun.
3. Vektor tegangan nol (V0 dan V7, Vektor tegangan saat *short - circuit*) tidak berpengaruh pada vektor fluks stator yang berarti motor berhenti bergeak.
4. Torsi dapat dikontrol dengan pemilihan *state inverter* yang tepat dimana fluks stator meningkat, berhenti, atau menurun.

Vektor fluks stator pada sumbu-dq dibagi menjadi enam sektor 60^0 (mulai sektor satu sampai enam). Sebuah vektor fluks stator dikatakan berhubungan dengan vektor tegangan ketika melewati sektor yang sama, yang berarti bahwa dari keenam vektor tegangan diorientasikan pada vektor tegangan yang terdekat. Sebagai contoh vektor fluks stator menjadi terkait dengan V_4 ketika berada atau melewati sektor ke-4. Begitu juga sebaliknya ketika fasa dari vektor yang sama adalah antara 150^0 sampai 210^0, maka vektor fluks stator tersebut berhubungan dengan tegangan V_4 seperti pada Gambar 2.6.



Gambar 2.6 Ruang vektor tegangan keluaran inverter dan bidang vektor dari enam sektor

Pengaruh tiap – tiap vektor tegangan terahadap fluks stator dan torsi dapat dilihat pada Tabel 2.1. Dampak dari vektor V_1 dan V_4 pada nilai torsi adalah ambigu, karena tergantung pada apakah vektor fluks mendahului atau tertinggal dari vektor tegangan merupakan tanda tanya. Vektor tegangan bernilai nol (V_0 dan V_7) tidak berpengaruh pada fluks stator tetapi menurunkan nilai torsi karena vektor dari peningkatan fluks rotor mengakibatkan fluks stator berhenti.

Tabel 2.1 Pengaruh tiap – tiap vektor tegangan terhadap nilai fluks stator dan torsi

	V_0	V_1	V_2	V_3	V_4	V_5	V_6	V_7
λ_s	-	↗↗	↗	↘	↘↘	↘	↗	-
T_e	↘	?	↗	↗	?	↘	↘	↘

2.3.2 Kontroler DTC

Pada dasarnya kontroler DTC berfungsi untuk menghasilkan sinyal *switching* transistor daya pada *inverter*. Sinyal *switching* yang dihasilkan adalah vektor tegangan yang aktif berdasarkan aturan pada *switching table*. Fluks stator dan torsi sebagai masukan pada *switching table* ditentukan oleh estimator dan komparator.

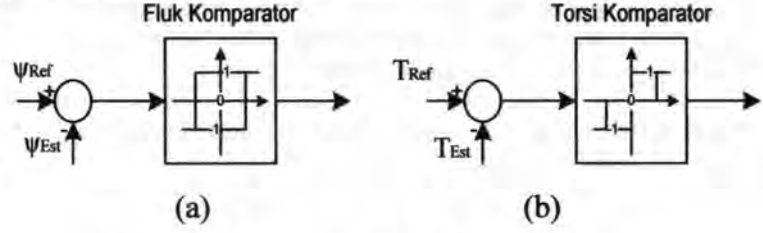
Komparator pada DTC

Pada metode DTC klasik, dalam mengatur kesalahan fluks dan torsi agar sesuai dengan fluks referensi dan torsi referensi berturut – turut menggunakan kontroler *two-level* dan *three-level hysteresis*. Nilai kesalahan fluks pada daerah histeresisnya didapatkan dengan menggunakan komparator. Pada kesalahan fluks dihasilkan dua nilai histeresis, yaitu -1 atau 1 dan pada kesalahan torsi dihasilkan tiga nilai histeresis yaitu -1, 0, 1.

Sehingga diperoleh nilai kesalahan fluks dan torsi :

$$\Delta T_e = T_{ref} - T_{est}$$
$$\Delta \psi^s = \psi^s{}_{ref} - \psi^s{}_{est} \qquad (2\text{-}32)$$

Nilai fluks akan dibandingkan dengan acuan fluks untuk mendapatkan fluks histeresis seperti pada gambar 2.7 (a). Sedangkan nilai elektromagnetik yang dihasilkan dari tahap sebelumnya yang kemudian dibandingkan dengan acuan torsi elektromagnetik diwakili oleh gambar 2.7 (b). Dengan cara ini, baik fluks dan torsi nilainya dapat meningkat, penurunan atau mempertahankan fluks atau torsi, tergantung pada keluaran dari komparator.



Gambar 2.7 Fluks dan torsi Komparator

Dari gambar 2.3, dapat disimpulkan bahwa :

$\psi = 1$, maka fluks naik ; $\psi = 1$, maka fluks naik ; dan
$T_e = -1$, maka torsi naik ; $T_e = 0$, maka torsi tetap
$T_e = -1$, maka torsi naik

Salah satu faktor penting dalam operasi ini adalah *band* histeresis dari fluks dan torsi komparator. Komparator memberikan bentuk gelombang fluks dan torsi yang lebih baik dan juga meningkatkan frekuensi *switching inverter*. Kombinasi dari keluaran

fluks dan torsi komparator dengan nilai sudut fluks stator kemudian diaplikasikan pada *rule* yang ada pada *switching table* yang akan memberikan vektor tegangan yang akan diaktifkan pada *inverter*.

2.3.3 *Estimator* Fluks, Torsi, dan Sektor Fluks Stator

Pada sumber tegangan inverter, berdasarkan fungsi *switching* didalamnya (yaitu Sa, Sb, dan Sc) dengan kemungkinan nilai 1 dan maka tegangan *inverter* menjadi :

$$V_s^{\ s} = \frac{2V_{dc}}{3}\left(S_a + e^{j\frac{2\pi}{3}} S_b + e^{j\frac{4\pi}{3}} S_c \right) \tag{2.33}$$

Pada pengaturan kecepatan putaran motor induksi metode DTC, *estimator* berfungsi sebagai pengganti sensor kecepatan dan posisi rotor. Nilai fluks dan torsi berdasarkan persamaan tegangan stator estimasi tidak memerlukan sinyal kecepatan atau posisi jika berada pada sumbu *stationer*. Persamaan tegangan dan arus stator berada pada sumb-dq berdasarkan transformasi Park. Pada metode DTC konvensional adapun persamaan fluks stator estimasi dalam fungsi tegangan dan arus stator sebagai berikut :

$$I_{qs} = \sqrt{\frac{2}{3}} I_{sa}$$

$$I_{ds} = \frac{1}{\sqrt{2}}(I_{sb} - I_{sc}) \tag{2.34}$$

$$V_{qs} = \sqrt{\frac{2}{3}} V_{dc}\left(S_a - \frac{1}{2}(S_b + S_c) \right)$$

$$V_{ds} = \frac{1}{\sqrt{2}} V_{dc}(S_b - S_c) \tag{2.35}$$

Persamaan stator fluks adalah sebagai berikut :

$$\psi_{qs} = \int (V_{qs} - i_{qs}.R_s)\, dt$$

$$\psi_{ds} = \int (V_{ds} - i_{ds}.R_s)\, dt \tag{2.36}$$

atau

$$\frac{d\psi_s}{dt} = V_s - i_s.R_s \tag{2.37}$$

Jika R_s nilainya sangat kecil, maka $R_s.\ i_s \approx 0$, sehingga :

$$\frac{d\psi\_s}{dt} = V_s \tag{2.38}$$

$$\Delta\psi_s = V_s \Delta t \tag{2.39}$$

Dimana Δt adalah periode sampling

Dengan nilai *magnitude* dan sudut fluks stator adalah sebagai berikut :

$$|\psi_s| = \sqrt{{\psi_{ds}}^2 + {\psi_{qs}}^2} \tag{2-40}$$

$$\alpha = \angle\psi_s = \tan^{-1}\frac{\psi_{qs}}{\psi_{ds}} \tag{2-41}$$

Nilai torsi estimasi didapatkan dari persamaan (2-1) diatas

$$T_e = \frac{3}{2} p\left(\lambda_{ds} i_{qs} - \lambda_{dq} i_{ds}\right) \tag{2-42}$$

Dimana : ψs = fluks stator (Wb)
Vs = tegangan stator (volt)
is = arus stator (A)
Rs = tahanan stator (ohm)
α = sudut fluks stator (derajat)

2.3.4 *Switching Table*

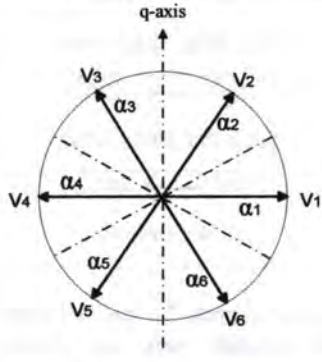
Sinyal masukan pada *switching table* didapat berdasarkan nilai dari fluks dan torsi histeresis, serta nilai dari sektor fluks stator yang dinyatakan dengan **α**. Algoritma DTC memilih vektor tegangan *inverter* untuk diterapkan pada motor induksi sebagai sumber tegangan. Adapun aturan pemilihan tegangan pada *switching table* seperti ditunjukkan pada tabel 2.2. Keluaran dari *switching table* mengatur perangkat *switching* yang aktif pada inverter. Gambar 2.8 menunjukkan hubungan antara vektor tegangan inverter dengan sektor dari fluks stator. Adapun nilai sektor fluks stator didapat dari persamaan (2.43)

$$\alpha = \angle\psi_s = \tan^{-1}\frac{\psi_{qs}}{\psi_{ds}} \tag{2-43}$$

Vektor tegangan aktif atau bernilai bukan nol terdiri dari $\vec{V}1$ (100); $\vec{V}2$ (110); $\vec{V}3$ (010); $\vec{V}4$ (011); $\vec{V}5$ (001); $\vec{V}6$ (101). Sedangkan vektor tegangan bernilai nol adadalah $\vec{V}0$ (000); $\vec{V}7$ (111).

Tabel 2.2 *Switching Table* dari vektor tegangan *inverter*

$d\psi$	dTe	α(1) sektor 1	α(2) sektor 2	α(3) sektor 3	α(4) sektor 4	α(1) sektor 5	α(1) sektor 6
-1	-1	$\bar{V}5$	$\bar{V}6$	$\bar{V}1$	$\bar{V}2$	$\bar{V}3$	$\bar{V}4$
	0	$\bar{V}0$	$\bar{V}7$	$\bar{V}0$	$\bar{V}7$	$\bar{V}0$	$\bar{V}7$
	1	$\bar{V}3$	$\bar{V}4$	$\bar{V}5$	$\bar{V}6$	$\bar{V}1$	$\bar{V}2$
1	-1	$\bar{V}6$	$\bar{V}1$	$\bar{V}2$	$\bar{V}3$	$\bar{V}4$	$\bar{V}5$
	0	$\bar{V}7$	$\bar{V}0$	$\bar{V}7$	$\bar{V}0$	$\bar{V}7$	$\bar{V}0$
	1	$\bar{V}2$	$\bar{V}3$	$\bar{V}4$	$\bar{V}5$	$\bar{V}6$	$\bar{V}1$



DTC konvensional :	DTC Modifikasi :
$-30^0 < \alpha(1) < 30^0$	$0^0 < \alpha(1) < 60^0$
$30^0 < \alpha(2) < 90^0$	$60^0 < \alpha(2) < 120^0$
$30^0 < \alpha(3) < 90^0$	$120^0 < \alpha(3) < 180^0$
$90^0 < \alpha(4) < 150^0$	$180^0 < \alpha(4) < -120^0$
$150^0 < \alpha(5) < 210^0$	$-120^0 < \alpha(5) < -60^0$
$210^0 < \alpha(6) < 270^0$	$-60^0 < \alpha(6) < 0^0$

Gambar 2.8 Vektor tegangan *inverter* dan sektor *switching* fluks stator pada DTC konvensional

Gambar 2.8 adalah vektor tegangan *inverter* yang berlaku pada metode DTC konvensional dan modifikasi. Dari Gambar 2.8 dapat dibuat tabel yang menyatakan hubungan dan perbandingan antara DTC konvensional dan DTC modifikasi pada Tabel 2.3.

Tabel 2.3 Perilaku dari tiap state dari DTC konvensional dan modifikasi

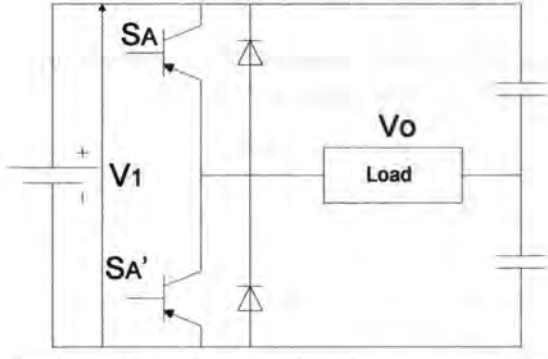
	DTC Konvensional $-30^0 \longrightarrow 30^0$	DTC Modifikasi $0^0 \longrightarrow 60^0$
V_1	$-30^0 \longrightarrow 30^0$ Torsi ambigu	$-60^0 \longrightarrow 0^0$ Torsi turun, fluks naik
V_2	$30^0 \longrightarrow 90^0$ Torsi dan fluks naik	$0^0 \longrightarrow 60^0$ Torsi dan fluks naik
V_3	$90^0 \longrightarrow 150^0$ Torsi dan fluks naik	$60^0 \longrightarrow 120^0$ Fluks ambigu
V_4	$150^0 \longrightarrow -150^0$ Torsi ambigu	$120^0 \longrightarrow 180^0$ Torsi naik, fluks turun
V_5	$-150^0 \longrightarrow -90^0$ Torsi turun, fluks turun	$-180^0 \longrightarrow -120^0$ Torsi dan fluks turun
V_6	$-90^0 \longrightarrow -30^0$ Torsi turun, fluks naik	$-120^0 \longrightarrow -60^0$ Fluks ambigu

Pada DTC konvensional, *state inverter* pada V_1 dan V_4 tidak digunakan karena nilai torsi tidak jelas apakah naik atau turun. Bergantung pada posisi sektornya apakah berada pada posisi 30^0 pertama atau yang kedua. Sedangkan pada DTC modifikasi state inverter yang tidak digunakan adalah V_3 dan V_6. Namun pada DTC modifikasi alas an tidak digunakan adalah karena pada V_3 dan V_6 nilai fluks stator yang ambigu, bukan torsi seperti pada DTC konvensional. Hal tersebut dianggap sebagai suatu keuntungan dalam mengontrol torsi menggunakan metode DTC modifikasi. Oleh karena itu, lebih baik kehilangan penggunaan dua nilai fluks yang ambigu daripada torsi.

2.4. *Inverter*

Rangkaian yang berfungsi untuk merubah tegangan DC (searah) menjadi tegangan AC (bolak-balik) adalah rangkaian *inverter* . Ada dua jenis *inverter* berdasarkan jenis sumber dayanya dan topologi dari rangkaian sumbernya, yaitu *voltage-source inverter* (VSI) dan *current-source inverter* (CSI). Pada Tugas Akhir ini *inverter* yang digunakan adalah tipe VSI. VSI yang paling sederhana, setengah rangkaian satu fasa ditunjukkan pada Gambar 2.9. Semua saklar tidak dapat hidup secara bersamaan karena akan terjadi hubungan pendek pada sumber daya. Fungsi saklar A didefiniskan sebagai :

$$a\begin{cases}0 \text{ jika SA } = \text{ hidup maka SA' } = \text{ mati} \\ 1 \text{ jika SA } = \text{ mati maka SA' } = \text{ hidup}\end{cases}$$



Gambar 2.9 Setengah rangkaian *inverter* satu fasa

Keluaran tegangan *inverter* dapat dituliskan pada persamaan 2.44.

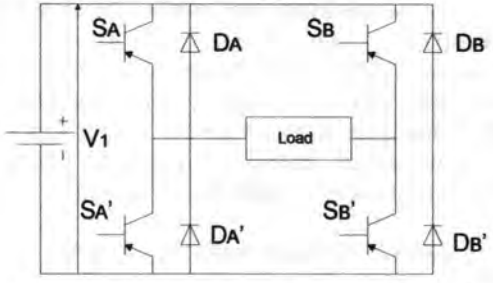
$$V_0 = V_i\left(a - \frac{1}{2}\right) \qquad (2.44)$$

Di mana *Vi* menunjukkan tegangan masukan DC. Nilai tegangan keluar (*Vo*) yang mungkin adalah *Vi*/2 atau *Vo*/2. Dari setengah rangkaian *inverter* diatas kita dapatkan rangkaian *inverter* satu fasa penuh seperti Gambar 2.9. Rangkaian ini mempunyai dua kaki (yaitu kaki A dan B) sebagai fungsi *switching*.Tegangan keluaran dari *inverter* ini ditulis sebagai fungsi *switching* A dan B seperti berikut ini :

$$Vo = Vi\,(\text{A-B}) \qquad (2.45)$$

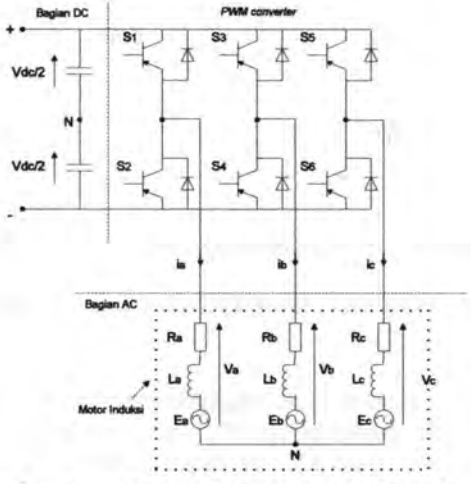
Maka tegangan keluaran dari *inverter* dapat diasumsikan mempunyai tiga kondisi nilai yang mungkin, yaitu $-Vi$, 0, atau Vi.

Penguatan tegangan maksimum *inverter* sebesar dua kali dari setengah rangkaian *inverter* satu fasa.



Gambar 2.10 Rangkaian penuh VSI satu fasa

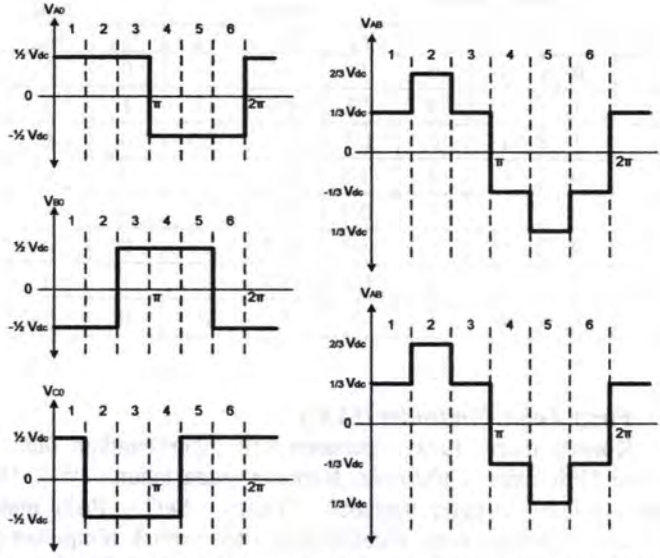
Topologi dasar dari *inverter* pada *two level* VSI tiga fasa terdiri dari enam *switch* aktif seperti pada Gambar 2.10.



Gambar 2.11 Topologi dasar dari VSI tiga fasa

VSI seperti yang telah diGambar kan pada Gambar 2.11 menghubungkan setiap fasa pada motor induksi dengan tegangan DC, sehingga dihasilkan sinyal tegangan yang bertingkat dan seakan – akan

merupakan sinyal sinusoidal. Tegangan keluaran yang diperoleh adalah tegangan fasa ke fasa.



Gambar 2.12 Sinyal tegangan keluaran dalam enam langkah

Tegangan keluaran fasa ke netral didapatkan dari tegangan per fasa berikut ini :

$$V_s^{\ s} = \frac{2V_{dc}}{3}\left(S_a + e^{j\frac{2\pi}{3}}S_b + e^{j\frac{4\pi}{3}}S_c\right) \tag{2.46}$$

atau dalam bentuk matrik yaitu :

$$\begin{bmatrix} V_{an} \\ V_{bn} \\ V_{cn} \end{bmatrix} = \begin{bmatrix} \frac{2}{3} & -\frac{1}{3} & -\frac{1}{3} \\ -\frac{1}{3} & \frac{2}{3} & -\frac{1}{3} \\ -\frac{1}{3} & -\frac{1}{3} & \frac{2}{3} \end{bmatrix} \begin{bmatrix} V_{ao} \\ V_{bo} \\ V_{co} \end{bmatrix} \tag{2.47}$$

Dari kondisi-kondisi perubahan saklar pada *inverter* , maka dapat disusun nilai *state* tegangan tiap –tiap vektor pada Tabel 2.4.

Tabel 2.1 Nilai tegangan tiap – tiap vektor pada *inverter*

Vektor Tegangan	Kondisi Saklar			Tegangan Line to Netral			Tegangan Line to Line		
	A	B	C	V_{an}	V_{bn}	V_{cn}	V_{ab}	V_{bc}	V_{ca}
V_0	0	0	0	0	0	0	0	0	0
V_1	1	0	0	2/3	-1/3	-1/3	1	0	-1
V_2	1	1	0	1/3	1/3	-2/3	0	1	-1
V_3	0	1	0	-1/3	2/3	-1/3	-1	1	0
V_4	0	1	1	-2/3	1/3	1/3	-1	0	1
V_5	0	0	1	-1/3	-1/3	2/3	0	-1	1
V_6	1	0	1	1/3	-2/3	1/3	1	-1	0
V_7	1	1	1	0	0	0	0	0	0

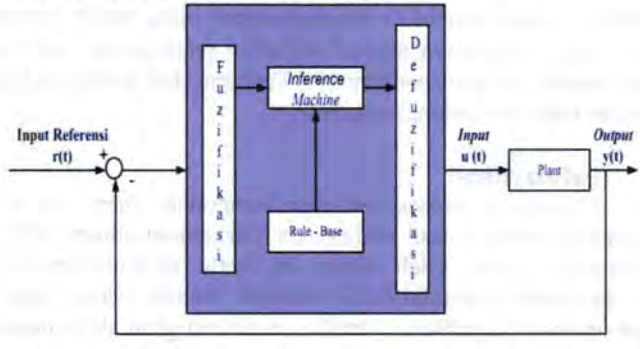
2.5 *Fuzzy Logic Controller* (FLC)

Konsep teori fuzzy pertama kali diperkenalkan oleh L.A. Zadeh dari Universitas California, Berkeley pada tahun 1965. Dalam makalah seminarnya yang berjudul "Fuzzy Set". Pada makalah tersebut ketidakpastian yang didefinisikan oleh sebuah himpunan yang mempunyai peranan penting dalam pemikiran manusia, khususnya dalam lingkup pengenalan pola, informasi komunikasi, dan lain-lain.
Ketidakpastian fuzzy bukan berasal dari pemilihan anggota himpunan yang acak, tetapi berasal dari konsep dan pemahaman alami manusia mengenai persoalan ketidakpastian dan ketidaktelitian. Teori fuzzy dikembangkan dari logika bolean, yaitu logika *True* (benar) atau *False* (salah). Perbedaan mendasar antara logika fuzzy dengan logika bolean terletak pada harga kebenaran. Pada logika fuzzy harga kebenaran diberikan dalam terminologi linguistik dengan menyertakan predikat kekaburan (*fuzzyness*) pada proposisinya. Harga kebenaran dan derajat kekaburan pada terminologi linguistik dapat dinyatakan dengan tolak ukur, misalnya agak, cukup, sangat dan sebagainya. Logika fuzzy memungkinkan nilai keanggotaan antara 0 dan 1, tingkat keabuan dan juga hitam dan putih, dan dalam bentuk linguistik, konsep tidak pasti seperti "sedikit", "lumayan", dan "sangat".

Fuzzy Logic Controller (FLC) merupakan kontroler yang terdiri dari aturan-aturan kontroler berbasis pengetahuan (*rules base*), pengamatan, dan pengenalan respon dari objek yang akan dikendalikan.

Aturan kontroler yang dipakai dalam *Fuzzy Logic Controller* (FLC) dinyatakan dalam variabel linguistik. *Fuzzy Logic Controller* (FLC) menggabungkan aspek pendefinisian himpunan fuzzy untuk memperoleh suatu kontroler yang dapat merpresentasikan cara kerja operator manusia. Skema *Fuzzy Logic Controller* (FLC) ditunjukkan pada Gambar 2.13. secara umum *Fuzzy Logic Controller* (FLC) memiliki empat bagian pokok yaitu :

1. Fuzzifikasi, berfungsi untuk mentransfer sinyal masukkan yang bersifat biner (*crisp*) kedalam konsep fuzzy. Konsep fuzzy tidak bekerja secara biner (himpunan *crisp*) tetapi berupa himpunan fuzzy di mana elemen-elemennya dinyatakan dengan pasangan elemen anggota himpunan x dan fungsi keanggotaannya.
2. Basis pengetahuan (*rule base*), berisi aturan dasar yang diperlukan untuk mencapai suatu pengontrolan.
3. Logika pengambilan kesimpulan (*Inference mechanism*), berisi bermacam operasi logika fuzzy untuk memperoleh aksi kontrol dari input yang diterima.
4. Defuzzifikasi, berfungsi mentransformasikan aksi kontrol yang bersifat fuzzy menjadi besaran *crisp*.



Gambar 2.13 Skema *Fuzzy Logic Controller (FLC)*

2.5.1. Fungsi Keanggotaan *Fuzzy* (Fuzzyfikasi)

Proses fuzzyfikasi adalah proses pengubahan nilai. Proses transformasi dilakukan dengan cara pemetaan ruang masukkan non fuzzy ke dalam masukkan variabel fuzzy dengan bantuan faktor penyekalaan (*scalling factor*)

Faktor penyekalaan diatur sedemikian rupa sehingga seluruh variabel *input* dapat terpetakan dalam himpunan semesta yang telah dirancang. Pada proses pengaturan faktor skala *gain*, perubahan kesalahan serta gain kesalahan diskala secara *heuristic*. Kemudian dilakukan pengolahan data dalam (Kontrol Logika Fuzzy) KLF berdasarkan teori himpunan *fuzzy* dengan menggunakan variabel *linguistic* yang bersifat *fuzzy*.

2.5.2 Aplikasi fungsi implikasi (*fuzzy rules*)

Fuzzy rules merupakan bagian penting dari *fuzzy* sistem yang berisi tentang dasar – dasar informasi dari *fuzzy* sistem. Aturan dasar (*rule based*) pada kontrol logika *fuzzy* merupakan suatu bentuk aturan relasi / implikasi "Jika – Maka" atau "*If – Then*".

2.5.3 Komposisi Aturan

Komposisi aturan merupakan proses implikasi dalam menalar nilai *input* guna penentuan nilai output sebagai bentuk pengambilan keputusan. Salah satu model penalaran yang banyak dipakai adalah penalaran *max min*. Pada penalaran max min proses pertama yang dilakukan adalah melakukan operasi *min* terhadap sinyal output lapisan fuzzyfikasi, yang diteruskan dengan operasi *max* untuk mencari nilai *output* yang selanjutnya akan dilanjutkan pada proses defuzzyfikasi sebagai bentuk *output* pengontrol. Terdapat dua jenis penalaran yang digunakan pada operasional *max min*.

2.5.4 Defuzzyfikasi

Merupakan proses pemetaan himpunan *fuzzy* ke himpunan tegas (*crips*). Sebagai hasil kesimpulan dari aturan-aturan *fuzzy* adalah sub himpunan *fuzzy*. Oleh sebab itu perlu dilakukan penerjemahan untuk mengubah nilai-nilai *fuzzy* menjadi bentuk *crisp* yang disebut dengan proses defuzzifikasi. Untuk menghubungkan aksi *output* dengan kondisi-kondisi input maka digunakan metoda dengan cara menarik kesimpulan dengan korelasi minimum dan *defuzzifikasi centroid*.

Terdapat 5 metode yang digunakan pada proses defuzzyfikasi dalam hal penentuan nilai *fuzzy* menjadi nilai *crips*, antara lain:

1. Metode maximum (max)
   Metode ini juga dikenal dengan metode puncak di mana nilai *output* dibatasi oleh sistem atau *plant* dari hasil fuzzyfikasi
2. Metode titik tengah (*Center of area*)

Metode ini juga disebut pusat area. Metode ini lazim dipakai pada proses defuzzyfikasi pada sistem

3. Metode rata – rata (*Average*)
   Metode ini digunakan untuk fungsi keanggotaan output yang simetris.
4. Metode penjumlahan titik tengah (*Summing of center area*)
   Metode ini digunakan untuk pengambilan data secara berkala dengan adanya penambahan nilai titik tengah di dalamnya
5. Metode titik tengah area terbesar
   Dalam metode ini *output* dipilih berdasarkan titik pusat area terbesar yang ada.

2.6 Mikrokontroler ATMEGA 16

Mikrokontroler adalah sistem mikroprosesor lengkap yang terkandung di dalam sebuah *chip*. Mikrokontroler berbeda dari mikroprosesor serba guna yang digunakan dalam sebuah *PC*, karena sebuah mikrokontroler umumnya telah berisi komponen pendukung sistem minimal mikroprosesor, yakni memori dan antarmuka I/O.

Mikrokontroler yang sering digunakan dalam instrumerntasi dan elektronika adalah mikrokontroler AVR ATMEGA 16.

2.6.1 Fitur ATMEGA 16

Beberapa fitur yang dimiliki ATMEGA 16 adalah sebagai berikut :

1. Mikrokontroler AVR 8 bit yang memiliki kemampuan tinggi, dengan daya rendah.
2. Arsitektur RISC dengan *throughput* mencapai 16 MIPS pada frekuensi 16 MHz.
3. Memiliki kapasitas *flash memory* 16 Kbyte, EEPROM 512 Byte dan SRAM 1 Kbyte.
4. Saluran I/O sebanyak 32 buah, yaitu *port* A, *port* B, *port* C, dan *port* B.
5. CPU yang terdiri atas 32 buah *register*.
6. Unit interupsi internal dan eksternal.
7. *Port* USART untuk komunikasi serial.
8. Fitur *peripheral*
   a) Tiga buah *timer/counter* dengan kemampuan pembandingan.
      - Dua buah *timer/counter* 8 bit dengan *prescaler* terpisah dan *compare mode.*

- Satu buah *timer/counter* 16 bit dengan *prescaler* terpisah, *mode compare, mode capture.*

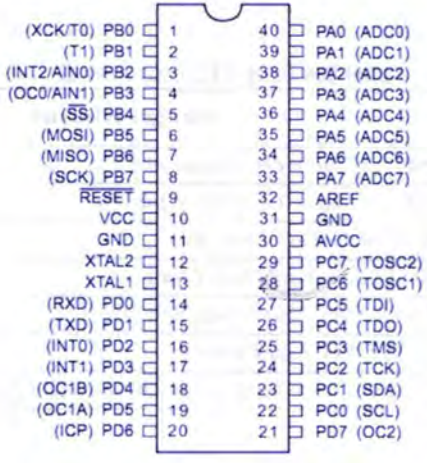
b) *Real time counter* dengan *oscillator* tersendiri.
c) *4 channel PWM.*
d) 8 *channel* , 10-bit ADC
   - 8 *single ended channel*
   - *7 differential channel* hanya pada kemasan TQFP
   - *2 differential channel* dengan *programmable gain 1x, 10x,* atau *20x.*
e) *Byte-oriented two wire serial interface*
f) *Programmable serial USART*
g) Antarmuka SPI
h) *Watchdog timer* dengan *oscillator internal.*
i) *One chip analog comparator.*

2.6.2 Konfigurasi pin ATMEGA 16

Konfigurasi pin ATMEGA 16 dengan kemasan 40 pin DIP (*Dual In-line Package*) dapat dilihat pada Gambar 2.14.

Dari Gambar 2.14 dapat dijelaskan fungsi dari masing-masing pin ATMEGA 16 sebagai berikut :

1. VCC menggunakan pin yang berfungsi sebagai masukan catu daya
2. GND merupakan pin *ground*
3. *Port* A (PA0...PA7) merupakan pin *input/output* dua arah dan pin masukan ADC
4. *Port B* (PB0...PB7) merupakan pin *input/output* dua arah dan pin fungsi khusus seperti dapat dilihat pada tabel 2.5.
5. *Port C* (PC0...PC7) merupakan pin *input/output* dua arah dan pin fungsi khusus seperti dapat dilihat pada tabel 2.6.
6. *Port D* (PD0...PD7) merupakan pin *input/output* dua arah dan pin fungsi khusus sepeti dapat dilihat pada tabel 2.7.
7. RESET merupakan pin yang digunakan untuk me-reset mikrokontroler.
8. XTAL1 dan XTAL2 merupakan pin masukan clock eksternal.
9. AVCC merupakan pin masukan tegangan untuk ADC.
10. AREF merupakan pin masukan tegangan referensi ADC.



Gambar 2.14 konfigurasi pin ATMEGA 16

Tabel 2.5 Fungsi Khusus *port* B

Pin	Fungsi Khusus
PB7	SCK (SPI *Bus Serial Clock*)
PB6	MISO (SPI *Bus Master Input/Slave Output*)
PB5	MOSI (SPI *Bus Master Output/Slave Output*)
PB4	SS (SPI *Slave Select Input*)
PB3	AIN1 (*Analog Comparator Negative Input*)
PB2	OC0 (*Timer/Counter0 Output Compare Match Output*)
	AIN0 (*Analog Comparator Positive Input*)
PB1	INT2 (*External Interrupt 2 Input*)
	T1 (*Timer/Counter 1 External Counter Input*)
PB0	T0 (*Timer/Counter 0 External Counter Input*)
	XCK (USART *External ClockInput/Output*)

Tabel 2.6 Fungsi Khusus *Port C*

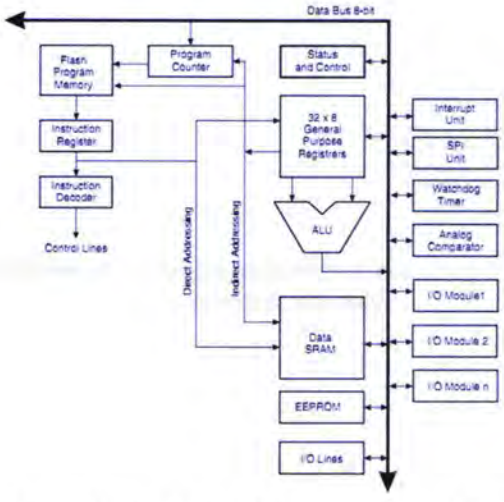
Pin	Fungsi Khusus
PC7	TOSC 2 (*Timer Oscillator pin 2*)
PC6	TOSC 1 (*Timer Oscillator pin 1*)
PC5	TDI (JTAG *Test Data In*)
PC4	TD0 (JTAG *Test Data Out*)
PC3	TMS (JTAG *Test Mode Select*)
PC2	TCK (JTAG *Test Clock*)
PC1	SDA (*Two wire Serial Bus Data Input Output Line*)

Tabel 2.7 Fungsi Khusus *Port D*

Pin	Fungsi Khusus
PD7	OC2 (*Timer/Counter 2 Output Compare Match Output*)
PD6	ICP (*Timer/Counter 1 Input Capture In*)
PD5	OC1A (*Timer/Counter 1 Output Compare A Match Output*)
PD4	OC1B (*Timer/Counter 1 Output Compare B Match Output)*
PD3	INT1 (*External Interrupt 1 Input*)
PD2	INT0 (*Ecternal Interrupt 0 Input*)
PD1	TXD (USART *Output pin)*

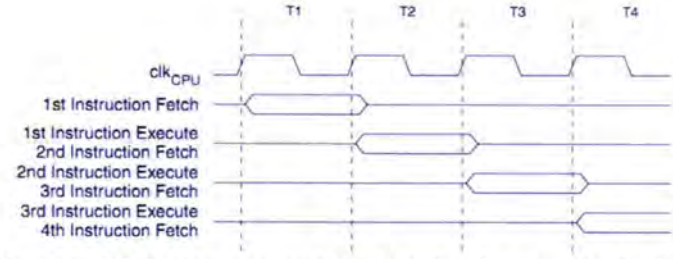
2.6.3 Arsitektur Mikrokontroler AVR RISC

Pada Gambar 2.15 adalah gambar arsitektur mikrokontroler AVR RISC untuk ATMEGA 16.

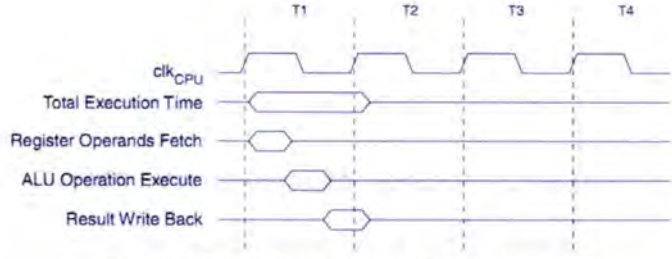


Gambar 2.15 Arsitektur Mikrokontroler AVR RISC

Dari Gambar 2.15, AVR menggunakan arsitektur Harvard dengan memisahkan antara memori dan bus untuk program dan data untuk memaksimalkan kemampuan dan kecepatan. Instruksi dalam memori program dieksekusi dengan *pipelining single level.* Di mana ketika satu instruksi dieksekusi, instruksi berikutnya diambil dari memori program. Konsep ini mengakibatkan instruksi dieksekusi setiap *clock cycle*. CPU terdiri dari 32x8-bit *general purpose register* yang dapat diakses dengan cepat dalam satu *clock cycle*, yang mengakibatkan operasi A*rithmetic Logic Unit (ALU)* dapat dilakukan dalam satu operasi dieksekusi dan hasilnya disimpan kembali pada *register* dalam satu *clock cycle*. Operasi aritmatik dan *logic* pada ALU akan mengubah bit-bit yang terdapat pada status register (SREG). Proses pengambilan instruksi dan pengeksekusian instruksi berjalan secara parallel, dapat dilihat pada Gambar 2.16.



Gambar 2.16 Proses pengambilan instruksi dan pengeksekusian instruksi secara parallel



Gambar 2.17 *Single Cycle ALU operation*

2.7 *Power Supply*

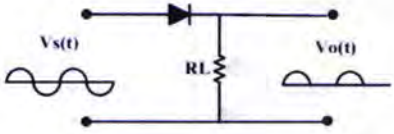
Power supply atau catu daya adalah sebuah rangkaian elektronika yang mengubah arus bolak balik (AC) menjadi arus searah (DC) yang kemudian digunakan untuk menyalurkan listrik atau bentuk energi jenis apapun yang dibutuhkan oleh rangkaian elektronika yang lainnya. Secara prinsip, rangkaian *power supply* terdiri atas tiga komponen, yaitu transformator, dioda, dan kapasitor. *Power supply* DC memiliki rangkaian utama yang disebut dengan *rectifier* (penyearah), yaitu rangkaian yang berfungsi untuk mengubah gelombang sinus AC menjadi deretan pulsa DC.

Untuk mengubah arus AC menjadi arus DC maka diperlukan suatu rangkaian penyearah. Ada dua jenis rangkaian penyearah yang sering digunakan, yaitu :

1. Penyearah setengah gelombang (*Half Wave Rectifier*).

Rangkaian penyearah setengah gelombang (*Half Wave Rectifier*) terdiri dari 1 buah dioda yang terpasang secara seri maju, yang

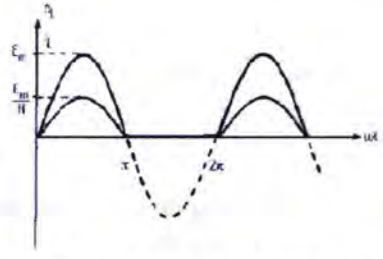
nantinya akan menghasilkan sinyal setengah gelombang, seperti pada Gambar 2.18.



Gambar 2.18 rangkaian penyearah setengah gelombang

Gelombang sinus pada penyearah setengah gelombang ini hanya dilewatkan separuh saja yaitu pada saat anoda dioda mendapat sinyal positif, namun jika sinyal yang diterima anoda negatif maka dioda akan *off*. Hal ini disebabkan oleh dioda yang hanya akan aktif jika anodanya lebih positif dari katoda.

Pada Gambar 2.19 dapat dilihat pada sinyal keluaran dari dioda saat 0-π adalah sinyal positif yang telah dilewatkan oleh dioda sedangkan saat π-2 π sinyal tersebut berada pada 0.



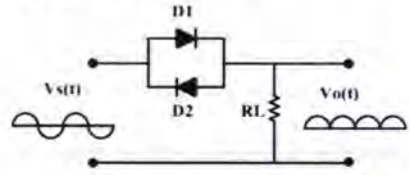
Gambar 2.19 bentuk gelombang sinus pada penyearah setengah gelombang

2. Penyerah gelombang penuh (*Full Wave Rectifier*)

Pada rangkaian penyearah setengah gelombang terdapat kelemahan, yaitu arus listrik yang mengalir ke beban hanya separuh dari setiap satu *cycle*. Hal ini akan menyulitkan dalam proses *filtering* (penghalusan). Untuk mengatasi kelemahan ini adalah penyearah gelombang penuh.

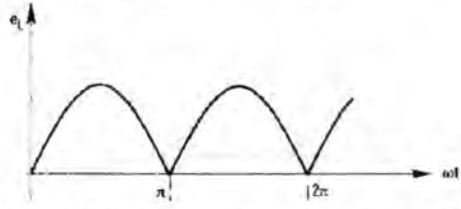
Rangkaian penyearah gelombang penuh tersusun dari dua dioda seperti yang terlihat pada Gambar 2.20, yang keduanya menghasilkan

penyearah gelombang penuh agar lebih mudah memberikan sinyal DC yang rata. Gelombang keluarannya lebih sempurna namun tentu saja masih terdapat riak (*ripple*).



Gambar 2.20 Rangkaian penyearah gelombang penuh dengan 2 dioda

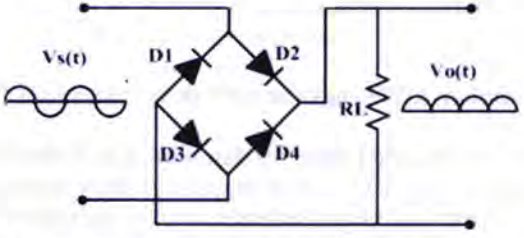
Prinsip kerja rangkaian penyearah gelombang penuh yang tersusun dari dua dioda adalah pada setengah siklus positif pertama dioda 1 (D1) akan akif dan melewatkan sinyal tersebut, sedangkan dioda 2 (D2) akan aktif pada saat setengah sinyal kedua, sehingga dioda bekerja secara bergantian. Pada Gambar 2.21 setengah gelombang dari 0-π dilewatkan oleh D1 sedangkan D2 bekerja pada saat π-2π.



Gambar 2.21 Bentuk gelombang sinus pada penyearah gelombang penuh

Sedangkan untuk rangkaian penyearah gelombang penuh yang terdiri dari dioda *bridge* dengan memakai 4 buah dioda, prinsip kerjanya adalah ketika priode sinyal positif AC (*sine wave*), dioda 1 (D1) dan dioda 2 (D2) mengalami konduksi. Arus listrik mengalir dari ujung lilitan bawah sekunder (transformator) melalui rangkaian beban, kemudian ke D1 dan D2, dan kembali ke lilitan bawah sekunder. Setengah periode berikutnya, polaritas *sine wave* berganti (periode sinyal negatif). Ujung lilitan atas sekunder sekarang menjadi negatif, ujung lilitan bawah menjadi positif. Dioda 3 (D3) dan dioda 4 (D4) mengalami konduksi. Pada kedudukan ini arus listrik mengalir dari ujung lilitan atas sekunder melalui rangkaian beban, kemudian ke D3

dan D4 dan kembali lilitan bawah sekunder. Arus listrik yang mengalir melalui rangkaian beban selalu dalam arah yang sama. Rangkaian penyearah gelombang penuh yang menggunakan dioda *bridge* dapat dilihat pada Gambar 2.22.

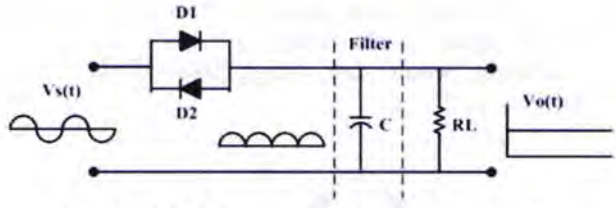


Gambar 2.22 rangkaian penyearah gelombang penuh dengan 4 dioda (*dioda bridge*)

Sinyal keluaran dari rangkaian penyearah (*rectifier*) belum dapat dimasukkan langsung ke rangkaian beban karena masih berupa deretan terdapat *ripple* tegangan. Untuk menghilangkan *ripple* tersebut dengan menggunakan kapasitor yang dipasang secara paralel pada rangkaian beban.

Pada saat anoda dioda 1 (D1) mendapat pulsa positif, D1 langsung konduksi dan kapacitor mulai mengisi. Ketika kapasitor telah mencapai tegangan puncak, D1 menyumbat karena katodanya lebih positif daripada anodanya. kapasitor harus membuang (*discharge*) muatannya melalui beban yang mempunyai resistan tertentu. Oleh karenanya waktu *discharge* kapasitor lebih lama dibanding waktu yang dibutuhkan AC untuk melakukan satu periode (*cycle*). Akibatnya sebelum kapasitor mencapai 0 volt diisi kembali oleh pulsa berikutnya. Tegangan input rata-rata sebesar 115 volt dan tegangan puncak sebesar 162 volt.

IC regulator yang sering digunakan pada *power supply* adalah jenis IC 78xx (untuk keluaran positif) dan IC 79xx (untuk keluaran negatif). Tanda 'xx' menunjukkan besar nilai yang dikeluarkan oleh regulator. Misalkan, dalam pembuatan Tugas akhir ini menggunakan *power supply* +12 Volt, -12 volt, dan +5 volt, maka digunakan IC 7812, 7912, dan 7805.



Gambar 2.23 Rangkaian *filter* dengan kapasitor

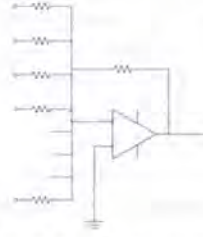
2.8 *Digital to Analog Converter* dan *Analog to Digital Converter*

Digital to Analog Converter digunakan untuk mengubah sinyal masukan *digital* menjadi sinyal keluaran *analog* sedangkan *Analog to Digital Converter* digunakan untuk tujuan sebaliknya yaitu mengubah sinyal masukan *analog* menjadi sinyal keluaran *digital*.

2.8.1 *Digital to Analog Converter (DAC)*

DAC adalah perangkat untuk mengkonversi sinyal masukan dalam bentuk *digital* menjadi sinyal keluaran dalam bentuk *analog* (tegangan). Tegangan keluaran yang dihasilkan DAC sebanding dengan nilai *digital* yang masuk ke dalam DAC.

Gambar 2.24 menunjukkan contoh rangkaian sederhana dari sebuah DAC.



Gambar 2.24 contoh sederhana rangkaian DAC

Tegangan keluaran dari gambar 2.18 adalah sebagai berikut :

$$V_0 = -R_f\left(\frac{b_1 V_{REF}}{R} + \frac{b_2 V_{REF}}{2R} + \cdots + \frac{b_N V_{REF}}{2^{N-1}R}\right)$$

$$= -2\frac{R_f}{R}V_{REF}\left(\frac{b_1}{2} + \frac{b_2}{4} + \cdots + \frac{b_N}{2^N}\right) \qquad (2.48)$$

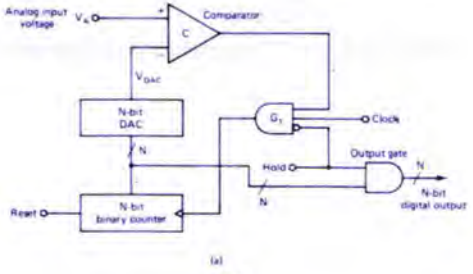
Dengan b_1 sebagai MSB, b_N sebagai LSN, dan VREF adalah tegangan sinyal digital.

2.8.2 *Analog to Digital Converter (ADC)*

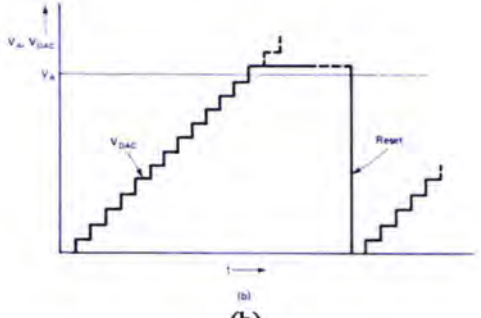
ADC adalah kebalikan dari DAC yaitu perangkat untuk mengkonversi sinyal masukan dalam bentuk *analog* menjadi sinyal keluaran dalam bentuk *digital.* Tegangan yang dihasilkan ADC sebanding juga dengan nilai *analog* yang masuk ke dalam ADC.

Secara umum ada tiga bentuk ADC yaitu :

1. *Counting type*
2. *Successive Approximation Register*
3. *Parallel Comparator (Brute Force)*



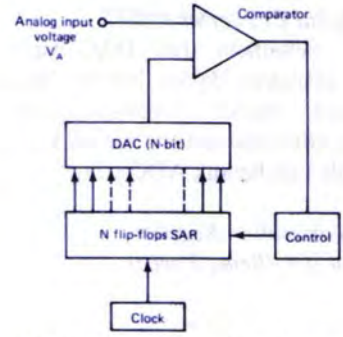
(a)



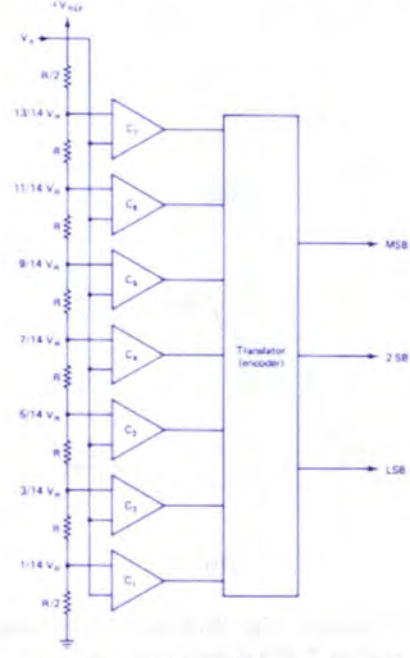
(b)

(a) blok diagram *counting ADC* (b) bentuk gelombang *output* DAC

Gambar 2.25 Gambar *counting* ADC



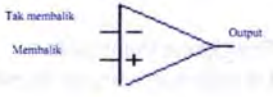
Gambar 2.26 Blok diagram *Successive Approximation Register*



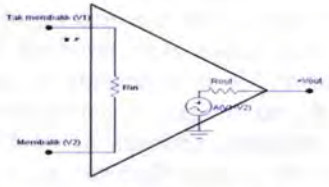
Gambar 2.27 *Parallel Comparator ADC*

2.9 Penguat Operasional (*Operational amplifier – Opamp)*

Penguat operasional merupakan rangkaian terpadu linier dasar (atau lebih tepat analog), yang sering diproduksi dalam satu sampai empat unit serupa dalam satu kemasan. Diagram Op Amp ditunjukkan pada gambar 2.28



(a)



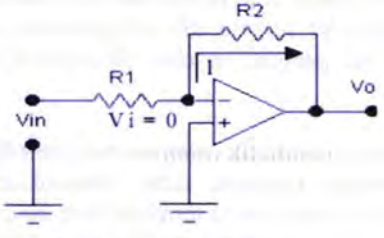
(b)

(a) Simbol penguat operasional (b) Rangkaian pengganti

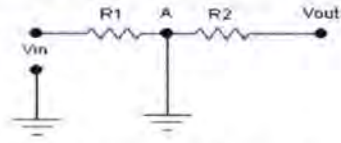
Gambar 2.28 Penguat Operasional

2.9.1 Penguat Membalik (*inverting amplifier*)

Salah satu penggunaan Op Amp adalah sebagai penguat membalik (*inverting*), yaitu penguat yang keluarannya mempunyai tanda tegangan yang terbalik dibandingkan dengan tanda tegangan masukan.



(a) Rangkaian Penguat



(b) Rangkaian Pengganti

Gambar 2.29 Rangkaian penguat membalik

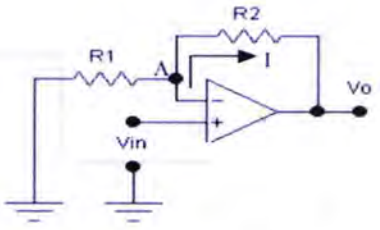
Salah satu sifat ideal Op Amp adalah bahwa impedansi masuk tak-hingga besar. Akibatnya tidak ada arus masuk ke kedua terminak masuk. Dan semua arus hanya akan melewati R1 dan R2, seperti ditunjukkan pada gambar 2.35-b. Disamping itu juga dikatakan bahwa perolehan tegangan Av tak hingga. Tegangan keluaran Vo = -Av.Vi terhingga (Vo < tak terhingga), sehingga Av tak hingga berarti Vi = 0. Sehingga tegangan di titik A dapat dikatakan nol (yang dinamakan bumi semu atau virtual ground) gambar 2.2-b menunjukkan rangkaian ganti yang jelas menunjukkan bahwa:

$$A_v = \frac{V_0}{V_{in}} = -\frac{R_2}{R_1} \qquad (2.49)$$

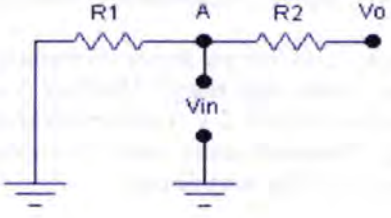
Persamaan diatas menunjukkan bahwa perolehan penguat tergantung pada perbandingan tahanan paralel (R2) dan tahanan seri (R1) dari penguat tersebut. Dari persamaan tersebut juga terlihat bahwa tanda tegangan keluar Vo terbalik dibandingkan dengan tanda tegangan masuk Vi Karena itu penguat tersebut dinamakan penguat membalik (inverting).

2.9.2 Penguat tak membalik (*noninverting amplifier*)

Jika tegangan masukan tidak dimasukkan lewat terminal inverting, tetapi dimasukkan lewat terminal non-inverting, yaitu sebesar V2, maka tegangan hasil penguatannya Vo akan tidak terbalik. Gambar 2.30 (a) menunjukkan gambar rangkaiannya dan gambar 2.30 (b) menunjukkan gambar rangkain pengganti dengan memahami bahwa *virtual ground* (Vi = 0), maka tegangan di titik A dianggap sama dengan V2, yakni VA = V2.



(a) Rangkaian Penguat



(b) Rangkaian Pengganti

Gambar 2.30 Rangkaian penguat tak membalik (*noninverting amplifier*)

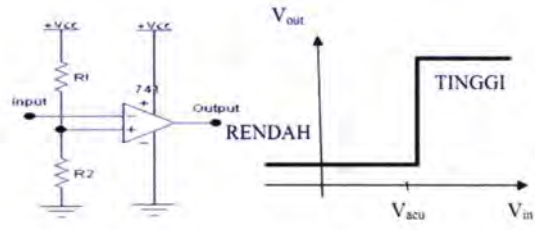
Dari gambar rangkaian pengganti dapat ditulis rumus berikut :

$$A_V = \frac{V_0}{V_{in}} = \frac{V_0}{V_A} = \frac{(R_2 + R_1)}{R_1} = 1 + \frac{R_2}{R_1} \qquad (2.50)$$

Persamaan tersebut selalu menghasilkan penguatan lebih besar dari satu dan tanda tegangan hasil penguatan tidak terbalik.

2.9.3 Pembanding (*comparator*)

Op amp dapat bekerja pada catu tunggal positif, seperti ditunjukkan pada gambar 2.31.



(a) (b)

Gambar 2.31 (a) Rangkaian Pembanding (b) Kurva karakteristik

Disini tegangan keluarnya hanya mempunyai satu polaritas, yaitu tengan positif rendah dan tinggi. Misalnya Vcc = +15,rentang keluaran berkisar antara 1V atau 2 V (keadaan rendah) sampai 13 atau 14 (keadaan tinggi). Tegangan acuan yang diterapkan pada masukan membalik berharga positif dan sama dengan:

$$V_{acu} = \frac{R_2}{R_1+R_2} V_{cc} \qquad (2.51)$$

Bila Vin lebih besar dari pada Vacu, keluarannya tinggi, seperti pada gambar 2.35-b. Bila Vin lebih kecil dari pada Vacu, keluarannya rendah. Dalam kedua kasus ini, keluarannya mempunyai polaritas yang positif.

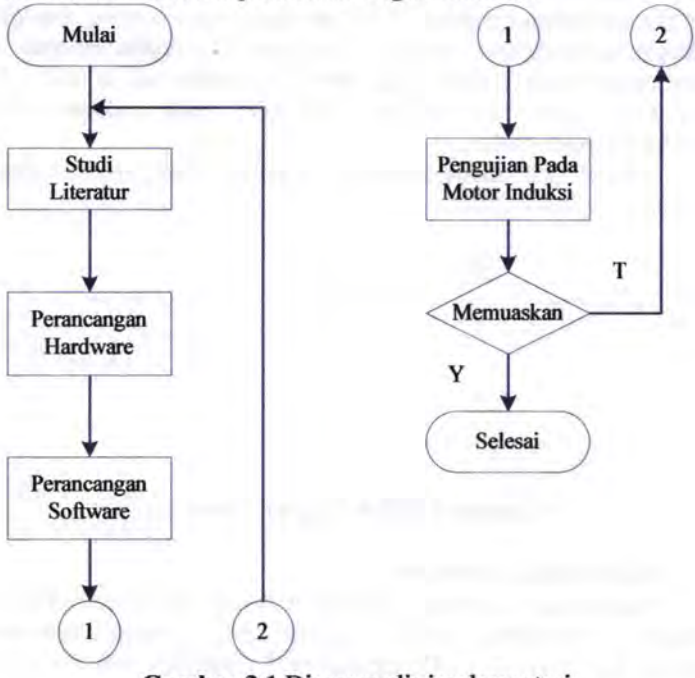
BAB III

PERANCANGAN SISTEM

III

BAB III
PERANCANGAN *DIRECT TORQUE CONTROL* UNTUK PENGATURAN KECEPATAN MOTOR INDUKSI 3 FASA MENGGUNAKAN KONTROLER FUZZY PI

Pada bab ini akan dibahas mengenai perancangan *Direct Torque Control* untuk mengatur kecepatan motor induksi 3 fasa menggunakan kontroler *fuzzy* PI. Gambar 3.1 menerangkan tentang yang harus dilakukan untuk implementasi Tugas Akhir ini.



Gambar 3.1 Diagram alir implementasi

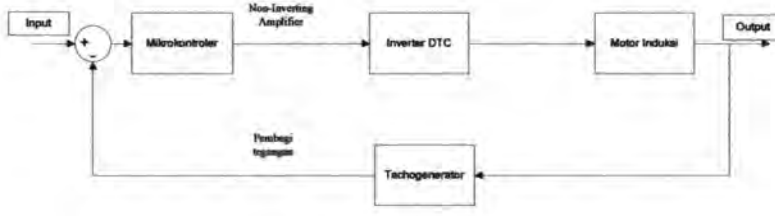
Dari gambar 3.1, maka dalam perancangan dan implementasi *Direct Torque Control* untuk pengaturan kecepatan motor induksi 3 fasa menggunakan kontroler PI Fuzzy terbagi dalam beberapa tahap.

Tahap pertama adalah mempelajari secara teori tentang sistem secara keseluruhan, kebutuhan sistem dan lainnya dari literatur yang

ada. Setelah itu dilanjutkan dengan merancang *hardware* yang dibutuhkan seperti Mikrokontroler, *power supply DC, Operational amplifier, Digital to Analog Converter* dan rangkaian pendukung lainnya. Langkah berikutnya adalah melakukan perancangan *software* untuk kontroler *Fuzzy* PI. Dan yang terakhir yaitu melakukan pengujian terhadap *hardware* maupun software.

Secara umum alur proses dari pengendalian kecepatan motor induksi 3 fasa adalah dengan menggunakan mikrokontroler yang telah diisi program berupa kontroler PI fuzzy yang akan digunakan untuk mengatur kecepatan motor induksi. Kemudian dari mikrokontroler dengan menggunakan rangkaian DAC dan juga *non-inverting amplifier* akan langsung menuju ke *inverter* 3 fasa yang kemudian inverter ini memutar motor induksi. Dari motor induksi kemudian ada umpan balik melalui *tacho generator* melalui *port* ADC pada mikrokontroler kembali ke mikrokontroler.

Gambar 3.2 memperlihatkan diagram blok sistem secara keseluruhan.



Gambar 3.2 Blok diagram sistem

3.1 Perancangan *hardware*

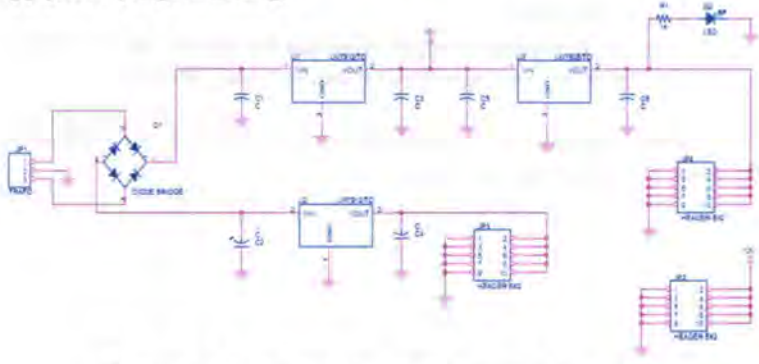
Perancangan *hardware* (perangkat keras) dari Tugas Akhir ini diantaranya merancang *power supply* DC, sistem minimum mikrokontroler, *keypad,* LCD, *Operational Amplifier,* dan *hardware* pendukung lainnya. Berikut akan dijelaskan mengenai *hardware* yang dirancang.

3.1.1 Perancangan *power supply DC*

Power supply DC diperlukan untuk mencatu rangkaian yang memerlukan tegangan DC seperti : mikrokontroler, DAC, *non-inverting amplifier.* Tegangan DC yang dirancang untuk sistem ini akan

menghasilkan tegangan DC 12 V, - 12 V dan 5 V. Untuk itu diperlukan regulator tegangan. Untuk menghasilkan 12 VDC menggunakan regulator 7812, untuk menghasilkan tegangan -12 VDC menggunakan regulator 7912, dan untuk menghasilkan tegangan 5 VDC menggunakan regulator 7805. Selain itu diperlukan juga trafo yang digunakan untuk mengubah tegangan AC menjadi DC. Trafo yang digunakan adalah trafo CT. Trafo dihubungkan dengan *diode bridge* (kiprok) kemudian dirangkai sedemikian rupa sehingga dapat menghasilkan tegangan DC sesuai dengan kebutuhan yang diinginkan.

Gambar 3.3 menggambarkan rangkaian skematik dari *power supply DC* yang dirancang.



Gambar 3.3 Skematik rangkaian *Power Supply DC*

3.1.2 Perancangan sistem mikrokontroler

Dalam praktiknya mikrokontroler berfungsi untuk mengatur kecepatan motor induksi 3 fasa. Mikrokontroler diisi program berupa kontroler *Fuzzy* PI yang kemudian dengan perantara *inverter* (sebagai alat untuk merubah arus DC menjadi AC 3 fasa) digunakan untuk mengatur kecepatan motor induksi 3 fasa.

Untuk dapat digunakan sebagai masukan *inverter* 3 fasa diperlukan rangkaian pendukung lain seperti *non-inverting amplifier*. Hal ini karena keluaran dari mikrokontroler berupa sinyal tegangan 5 Volt DC sehingga perlu *non-inverting amplifier* mengubahnya menjadi sinyal tegangan 10 Volt DC.

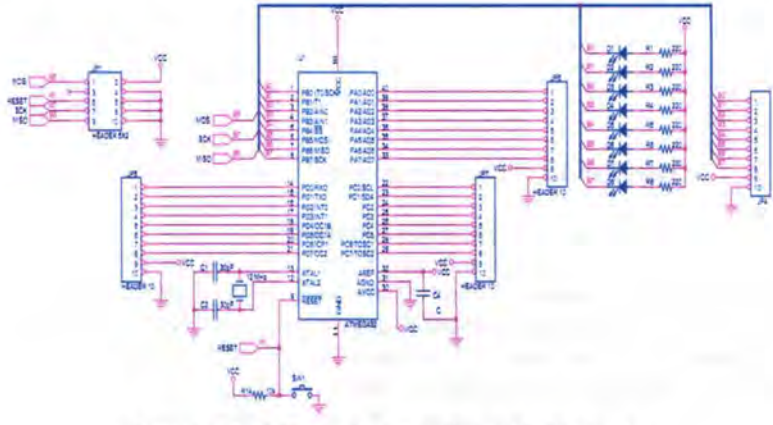
Mikrokontroler yang digunakan yaitu ATMEGA 16. Memiliki 4 buah *port* yang berfungsi sebagai *input/output* mikrokontroler. Untuk sistem ini menggunakan *port A* sebagai *Analog to Digital Converter*,

port B sebagai *output* penampil *LCD, port C* digunakan untuk *keypad* sebagai piranti untuk memasukkan parameter ke dalam mikrokontroler, dan *port D* sebagai masukan ke *inverter.*

Berikut adalah daftar *pin-pin* yang digunakan dalam Tugas Akhir ini :

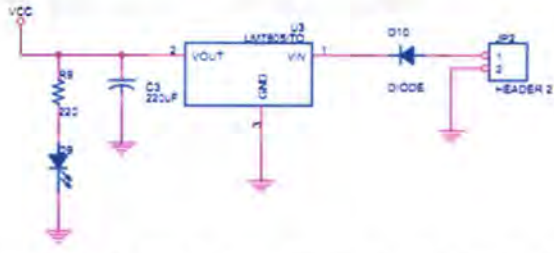
1. *Port A* digunakan sebagai masukan dari *tachogenerator. Port* ini adalah *port Analog to Digital Converter.*
2. *Port B* digunakan sebagai *port* untuk penampil LCD. Semua *port B* dipakai termasuk VCC dan GND sebagai catu LCD.
3. *Port C* digunakan sebagai masukan *keypad* untuk memasukkan parameter ke mikrokontroler. *Pin* yang digunakan yaitu *pin* 0, 1, 6, 7, dan GND.
4. *Port D* digunakan sebagai masukan ke dalam *inverter* untuk mengatur *inverter* agar dapat memutar motor induksi.

Pada Gambar 3.4 mengilustrasikan rangkaian skematik dari mikrokontroler beserta *pin-pin*-nya.



Gambar 3.4 Skematik rangkaian mikrokontroler

Selain itu, agar mikrokontroler dapat bekerja maka diperlukan catu daya. Catu daya yang dibutuhkan adalah 5 Volt DC. Catu daya ini digabung dalam satu rangkaian mikrokontroler. Gambar 3.5 Adalah gambar catu daya yang dipakai.

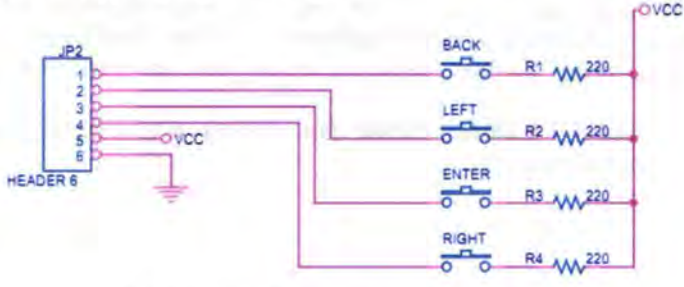


Gambar 3.5 Skematik rangkaian catu daya mikrokontroler

3.1.3 Perancangan *keypad*

Untuk dapat memasukkan besaran nilai berupa putaran, dan parameter lainnya ke dalam mikrokontroler maka dibutuhkan sebuah rangkaian *keypad*. *Keypad* dirancang dengan menggunakan tombol *push button* sebanyak 4 buah, resistor 220 Ohm 4 buah dan 6 *port* keluaran ke mikrokontroler. Tombol – tombol *keypad* diprogram dalam mikrokontroler agar berfungsi untuk memilih menu dan juga besaran nilai tertentu yang dibutuhkan sistem.

Gambar 3.6 menggambarkan rangkaian skematik dari rangkaian *keypad* yang dirancang.

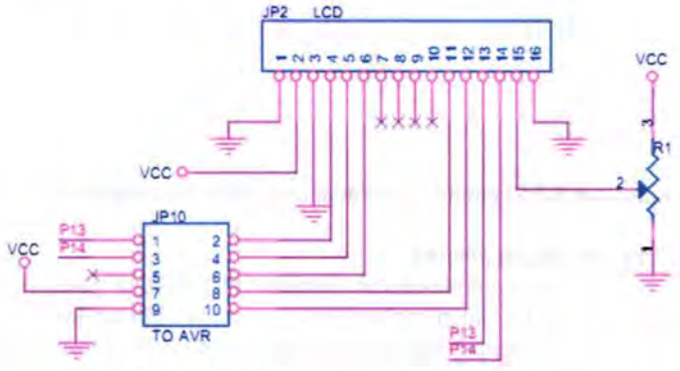


Gambar 3.6 Skematik rangkaian *keypad*

3.1.4 Perancangan LCD (*Liquid Crystal Display*)

Untuk menampilkan menu dan parameter masukan ke mikrokontroler diperlukan penampil LCD. LCD yang digunakan adalah LCD 16 x 2 karakter. LCD ini dihubungkan ke *port B* pada mikrokontroler.

Gambar 3.7 Adalah rangkaian skematik dari LCD yang dirancang untuk sistem ini.

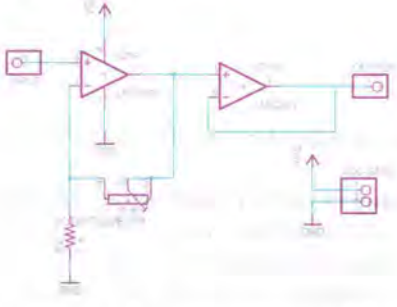


Gambar 3.7 Skematik rangkaian LCD

3.1.5 Perancangan *non-inverting amplifier*

Rangkaian lain yang diperlukan dalam sistem ini adalah *non-inverting amplifier*. Rangkaian ini diperlukan karena *output* tegangan dari mikrokontroler hanya 5 VDC padahal *input* tegangan yang dibutuhkan oleh inverter Toshiba adalah 0 – 10 VDC. Oleh karena itu, diperlukan suatu penguat tegangan dalam hal ini *non-inverting amplifier*.

Gambar 3.8 adalah gambar dari *non-inverting amplifier* yang dirancang untuk sistem ini.



Gambar 3.8 *Non inverting amplifier*

Dari Gambar 3.8 besar nilai resistor yang dipakai adalah 10 KΩ. Nilai ini didapat dari perhitungan rumus berikut :

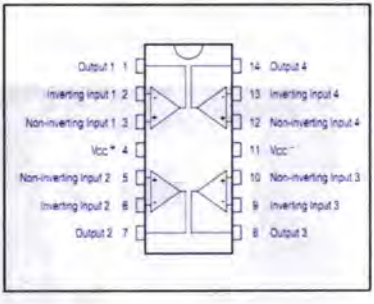
$$Vout = \left(1 + \frac{R_2}{R_1}\right) Vin \qquad (3.1)$$

Dimana :

V_{out} : Tegangan *Output* yang diinginkan
R_2 &R_1 : Nilai resistansi
V_{in} : Tegangan *input*

Setelah didapat nilai resistansi R1 dan R2 maka dirancang sebuah *non-inverting amplifier* seperti gambar 3.4. Rangkaian ini menggunakan IC LM 324. IC ini memiliki 4 *operational amplifier* yang independen. Selain itu juga dapat beroperasi dengan suplai daya tunggal melalui tegangan dengan *range* yang lebar, untuk tunggal dari 3 V sampai 30 V dan untuk suplai ganda ±1.5 V sampai ±15 V. Fitur lain yang dimiliki LM 324 adalah *gain bandwidth* sampai 1.3 MHz, *voltage gain* sampai 100dB.

Berikut adalah gambar konfigurasi pin dari LM 324 yang digunakan sebagai petunjuk perancangan *non-inverting amplifier*.



Gambar 3.9 Konfigurasi pin LM 324

3.1.6 *Inverter* Toshiba VF-S9

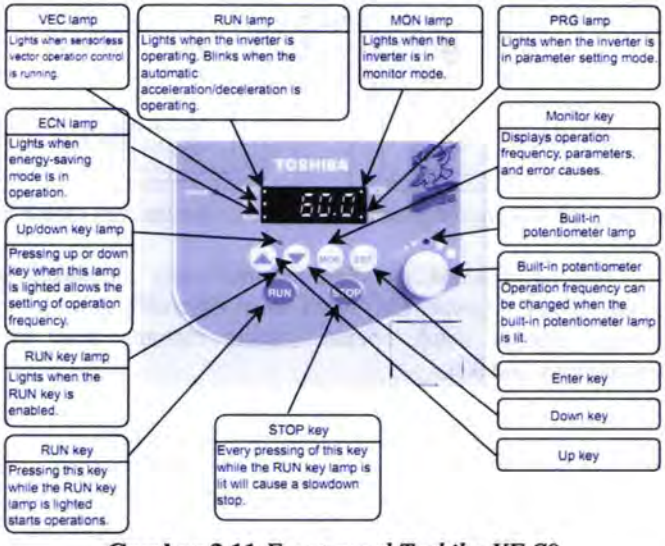
Inverter adalah rangkaian konverter dari DC ke AC, yang mempunyai fungsi mengubah tegangan *input* DC menjadi tegangan *output* AC simetri dengan besar dan frekuensi yang diinginkan.

Inverter yang sering digunakan dalam dunia industri diantaranya adalah *inverter* Toshiba VF-S9. *Inverter* ini adalah versi 101. Beberapa keunggulan dari *inverter* ini adalah :

1. *Built in noise filter*
2. *Simple operation*
    1. Fungsi otomatis (akselerasi/deselerasi torsi, fungsi pemrograman, environment programming).
    2. Adanya potensiometer pada panel depan memungkinkan kemudahan operasi
3. *Superior basic performance*
    1. Torsi dari frekuensi rendah sampai 150 % bahkan lebih.
    2. *Smooth operation*
    3. *Built-in current surge suppression circuit*
    4. *Maximum 400Hz high frequency output*
    5. *Maximum carrier frequency*
4. *Globally compatible*
    1. Kompatibel dengan suplai 240V dan 500 V
    2. *Sink/source switching of control input/output*
5. *Options allow use with a wide variety of applications*
    1. *Communication functions (RS485/RS232C)*
    2. *Extension panel/Parameter writer*
    3. *Other options are common to all models*

Gambar 3.10 *Inverter* Toshiba VF-S9

Gambar 3.10 Adalah gambar fisik dari *inverter* Toshiba VF-S9. Sedangkan untuk *front panel* dari *inverter* terlihat pada gambar 3.11.

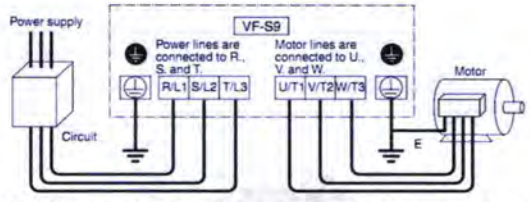


Gambar 3.11 *Front panel Toshiba VF-S9*

Dari Gambar 3.11 dapat dijelaskan mengenai fungsi tombol pada *front panel* sebagai berikut :

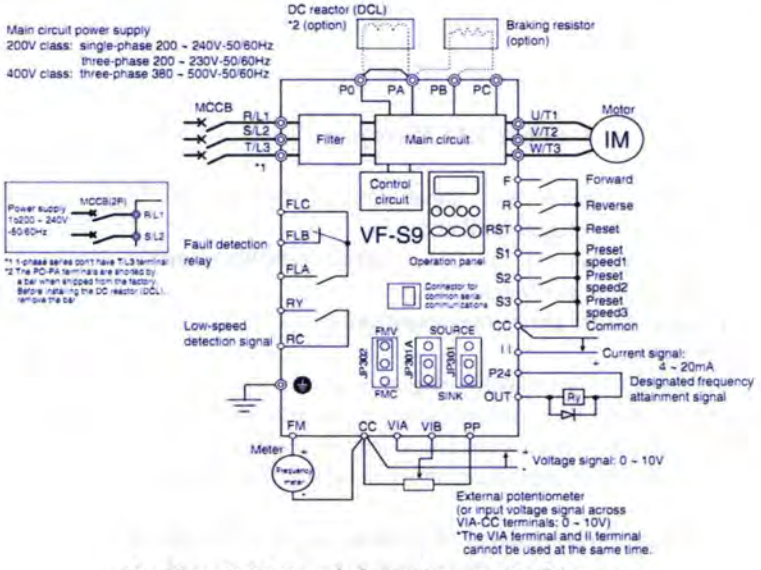
1. Tombol panah *Up & Down* untuk pemilihan menu dan untuk memasukkan nilai parameter.
2. Tombol MON untuk menampilkan menu, frekuensi operasi, error, parameter.
3. Tombol ENT digunakan sebagai *enter key*
4. *Built in potentiometer* digunakan untuk mengatur frekuensi operasi.
5. Tombol RUN digunakan untuk memulai operasi.
6. Tombol STOP digunakan untuk menghentikan operasi.

Agar *inverter* dapat digunakan untuk memutar atau mengatur kecepatan motor induksi maka motor perlu dihubungkan ke *inverter* dengan urutan *wiring* seperti pada gambar. Dimana *inverter* terlebih dulu terhubung ke sumber tegangan AC. Kemudian motor dihubungkan ke inverter seperti pada Gambar 3.12



Gambar 3.12 *Wiring diagram inverter* untuk pengaturan motor induksi

Inverter Toshiba memiliki terminal yang digunakan sebagai *input* dan *output inverter*. Dari *data sheet* diperoleh gambar *terminal* yang biasa dipakai untuk operasi standar memutar atau mengatur kecepatan motor induksi.



Gambar 3.13 Koneksi standar *terminal inverter*

Agar dapat digunakan untuk memutar motor induksi, maka *inverter* ini harus diatur terlebih dulu. Ada 2 cara pengaturan yang bisa dilakukan dengan *inverter* ini. Yaitu dengan menggunakan *default* dari *inverter* atau menggunakan pengaturan *inverter* dari perangkat tambahan luar *inverter*. Pada Tugas Akhir ini *inverter* akan dikendalikan oleh

mikrokontroler. Berikut adalah cara pengaturan *inverter* baik dengan inverter itu sendiri atau dengan dikendalikan dari perangkat tambahan :

1. Dengan *default inverter*
   Inverter VF-S9 ini bisa langsung digunakan memutar motor induksi dengan pengaturan terlebih dulu.
   a. Memilih menu dengan tombol MON pada panel.
   b. Dengan tombol UP dan DOWN pilih FNOD
   c. Tekan tombol ENT dengan menggunakan panah UP dan DOWN ubah nilai menjadi 2 kemudian tekan ENT
   d. Kemudian dengan UP DOWN dipilih menu CNOD dan ENT. Nilai CNOD di set 1 menggunakan tombol UP dan DOWN dilanjutkan dengan menekan ENT.

   Dengan pengaturan ini maka *inverter* telah dapat digunakan untuk memutar motor induksi. Dengan memutar potensio yang ada pada panel *inverter*.

2. Dengan perangkat luar
   Selain dapat langsung digunakan memutar motor induksi, *inverter* ini juga bisa dikontrol dari luar. Dalam Tugas Akhir in inverter ini akan diatur dengan mikrokontroler. Dengan memanfaatkan *terminal port* CC dan VI pada *inverter*. Seperti terlihat pada Gambar 3.13. *Port* ini membutuhkan tegangan masukan 0-10 VDC.
   Untuk itu diperlukan pengaturan pada *inverter* sebagai berikut :
   a. Dengan MON dipilih menu menggunakan tombol UP dan DOWN
   b. FNOD diubah nilainya menjadi 0 (nol)
   c. Kemudian dipilih menu F--- diisi menjadi F200 kemudian ENT. F200 diatur dengan nilai 5
   d. *Port CC* adalah negative dan *port VIA* adalah positif.

Dalam sistem ini dirancang pengendalian *inverter* dengan menggunakan mikrokontroler. Untuk itu *inverter* perlu diatur agar bisa digunakan untuk pengaturan dari luar.

3.1.7 Motor Induksi 3 fasa

Plant yang digunakan adalah motor induksi. Dan yang akan dipakai adalah motor induksi 3 fasa. Motor induksi ini dihubungkan

dengan *inverter* sebagai *driver*-nya. *Inverter* akan merubah arus DC dari mikrokontroler menjadi arus AC 3 fasa untuk memutar motor induksi.

Gambar 3.14 Motor induksi 3 fasa

Gambar 3.14 Adalah gambar motor induksi yang dipakai. Motor ini memiliki daya ≤ ¼ PK dan disuplai dengan sumber tegangan 3 fasa. Berikut adalah data lengkap motor induksi yang dipakai :

- Jenis motor (m) : 3 fasa
- Putaran nominal (n) : 1500 Rpm
- Frekwensi (f) : 50 Hz
- Jumlah alur (G) : 24 lobang (alur)
- Diameter dalam *stator* (D_1) : 48 mm = 4,8 cm
- Diameter luar *stator* (D_2) : 90 mm = 9 cm
- Panjang *stator* (L) : 51 mm = 5,1 cm
- Tebal gandar *stator* (Dy) : 21 mm = 2,1 cm
- Lebar gigi terkecil (W_{ts1}) : 4 mm = 0,4 cm
- Jenis gulungan : spiral / rata

Sedangkan setelah diukur dan dihitung, motor induksi yang dipakai memiliki data sebagai berikut :

- Putaran nominal (n) : 1490 rpm
- Arus / sanggul : 0.485 A
- Resistansi / sanggul : 28 Ω
- Induktansi / sanggul : 0.264 H

3.1.8 *Tacho generator*

Motor DC apabila diberi tegangan maka akan berfungsi sebagai motor dan bila mengeluarkan tegangan maka berfungsi sebagai generator. Oleh karena itu, motor DC dapat dipakai sebagai *tachogenerator* karena dapat menjadi generator DC yang menghasilkan tegangan. *Tacho generator* digunakan sebagai umpan balik ke mikrokontroler. Tegangan yang dihasilkan oleh *tacho* diumpanbalik ke mikrokontroler.

Tacho generator yang dipakai adalah berupa motor DC dengan spesifikasi seperti tabel berikut :

Tabel 3.1 Spesifikasi Motor DC

	Specification	Cable
Motor		
Input	+5 to +12 VDC	Red
	Ground	Blue
Speed Encoder		
Input	+5 VDC ± 5%	Orange
	Ground	Black
Output	Open collector	Yellow
	116 pulse/revolution	

Gambar 3.15 *Tachogenerator*

Dari Tabel 3.1 diketahui bahwa motor memerlukan catu data +5 sampai + 12 VDC. Selain itu juga motor DC ini dilengkapi dengan *encoder* yang menghasilkan pulsa sebanyak 116 pulsa setiap putaran motor. Karena motor difungsikan sebagai *tacho generator* maka motor DC dikopel dengan motor induksi. Sehingga pada saat motor induksi berputar maka dapat memutar motor DC pada saat yang bersamaan. Motor DC yang diputar oleh motor induksi akan berubah fungsi sebagai *generator* yang menghasilkan tegangan DC. Tegangan DC yang dihasilkan motor DC inilah yang digunakan sebagai umpan balik ke mikrokontroler melalui ADC pada mikrokontroler. Untuk selanjutnya

dapat digunakan sebagai acuan dalam pengaturan kecepatan motor induksi.

3.2 Perancangan *software*

Perancangan *software* yang dilakukan yaitu dengan memberikan masukan program kontroler *Fuzzy* PI ke dalam mikrokontroler. Selain itu juga diperlukan program untuk *keypad* dan LCD.

3.2.1 Perancangan Software mikrokontroler

Mikrokontroler perlu diisi program agar dapat digunakan untuk mengendalikan sesuatu sesuai dengan keinginan. Pada Tugas Akhir ini mikrokontroler akan digunakan untuk mengendalikan motor induksi dengan kontroler *Fuzzy* PI.

Kontroler *Fuzzy* PI digunakan dengan tujuan agar keluaran motor induksi sama atau mendekati dengan nilai *setpoint* yang diberikan. Dalam perancangan kontroler *Fuzzy* PI ini meliputi penghitungan *error* dan *delta error,* fuzzyfikasi, *rule base*, defuzzyfikasi dan tentunya penghitungan sinyal kontrol agar dapat menghasilkan putaran motor induksi yang sesuai dengan *setpoint*.

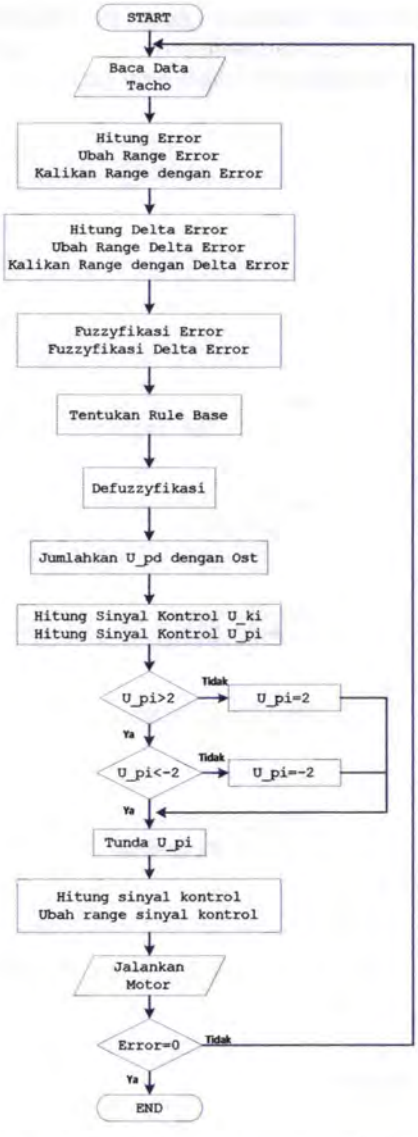
3.2.2 Perancangan Kontroler *Fuzzy* PI

Untuk mengendalikan *inverter* digunakan sinyal tegangan dari mikrokontroler. Untuk tujuan ini mikrokontroler perlu diisi program terlebih dulu (*download*). Program yang digunakan adalah program untuk implementasi kontroler logika *fuzzy* PI. Gambar 3.15 menampilkan *flowchart* untuk pembuatan kontroler *fuzzy* PI.

Perancangan kontroler *fuzzy* PI terdiri dari beberapa *subprogram* yang saling berhubungan menjadi satu kesatuan. Kontroler logika *fuzzy* adalah kontroler yang proses perhitungan sinyal kontrolnya dilakukan melalui operasi himpunan *fuzzy*. Kontroler *fuzzy* PI merupakan *fuzzy* statik karena memiliki himpunan pendukung *fuzzy* dan *rule base* tetap.

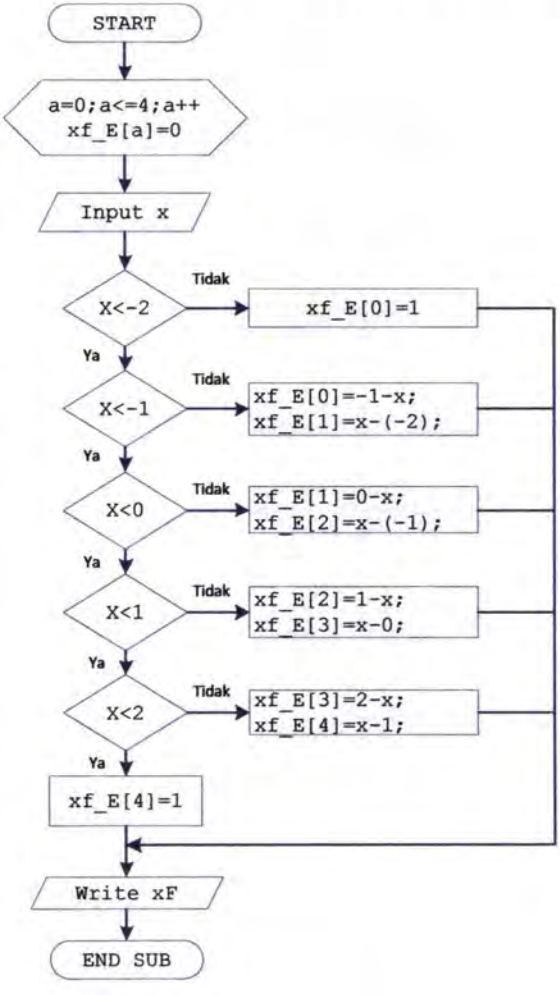
Kontroler logika *fuzzy* PI didapat dengan memodifikasi kontroler logika *fuzzy* PD yaitu dengan menambah nilai Ki pada keluaran kontroler logika *fuzzy* PD.

Subprogram yang dipakai dalam kontroler *fuzzy* PI meliputi *subprogram error fuzzy*, *delta erro fuzzy*, dan *rule base*.



Gambar 3.16 *Flowchart* kontroler *Fuzzy* PI

Dalam *flowchart* kontroler *fuzzy* PI terdapat *subprogram* fuzzyfikasi untuk *error fuzzy* dan *delta error fuzzy* dan juga *subprogram rule base*. Gambar 3.16 adalah *flowchart error fuzzy*.



Gambar 3.17 *Flowchart subprogram error fuzzy*

Dari Gambar 3.17 maka dibuat program untuk fuzzyfikasi error sebagai berikut :

```
void Fuzzification_Error(float x)
{
  int a;

  for (a=0;a<=4;a++)
  {
      xf_E[a]=0;
  }

  if (x<-2)
    xf_E[0]=1;
  else if (x<-1)
  {
    xf_E[0]=-1-x;
    xf_E[1]=x-(-2);
  }
  else if (x<0)
  {
    xf_E[1]=0-x;
    xf_E[2]=x-(-1);
  }
  else if (x<1)
  {
    xf_E[2]=1-x;
    xf_E[3]=x-0;
  }
  else if (x<2)
  {
    xf_E[3]=2-x;
    xf_E[4]=x-1;
  }
  else
  {
    xf_E[4]=1;
  }
}
```

Secara umum maksud dari *listing* program dari Gambar 3.16 adalah untuk fuzzifikasi *error*. Langkah-langkah yang dilakukan dalam fuzzifikasi error.

Langkah awal yang dilakukan yaitu dengan menentukan tipe data sebagai masukan yaitu menggunakan tipe data *integer* (int) dengan rentang nilai dari -32768 s/d 32767. Fuzzyfikasi menggunakan 5 *membership function* yaitu dari 0 – 4.

Selanjutnya dengan aturan *if-then* didefinisikan proses fuzzyfikasi *error*. Variabel x didefinisikan sebagai *input*. Berikut adalah *if-then rule* yang dipakai sebagai dasar pembuatan program fuzzyfikasi *error* :

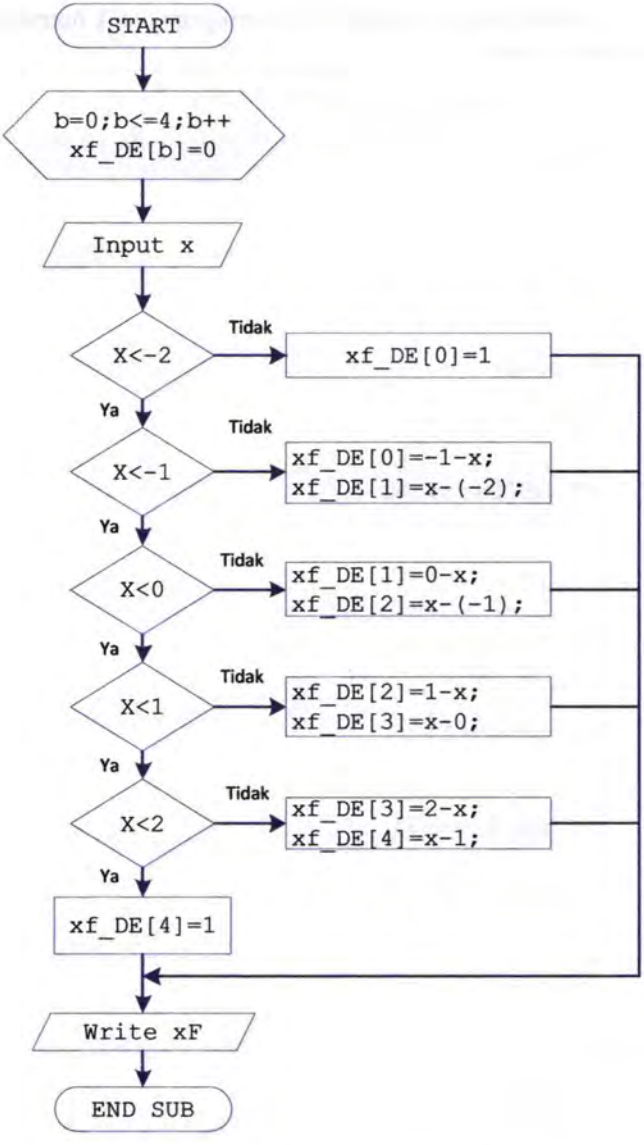
1. Jika $x < -2$ maka xf_E[0] = 1.
2. Jika $x < -1$ maka xf_E[0] = $-1 - x$; xf_E[1] = $x - (-2)$
3. Jika $x < 0$ maka xf_E[1] = $0 - x$; xf_E[2] = $x - (-1)$
4. Jika $x < 1$ maka xf_E[2] = $1 - x$; xf_E[3] = $x - 0$
5. Jika $x < 2$ maka xf_E[3] = $2 - x$; xf_E[4] = $x - 1$

Demikian halnya untuk penentuan fuzzyfikasi *delta error*, memiliki pola yang sama dengan fuzzyfikasi *error*. Menggunakan *membership function* sebanyak 5 mulai dari 0 – 4. Fuzzyfikasi *delta error* diwakili oleh variabel b dengan menggunakan tipe data *integer* (int) yang memiliki rentang nilai dari -32768 s/d 32767.

Selanjutnya dengan menggunakan aturan *if-then*, maka didapat fuzzyfikasi *delta error* sebagai berikut :

1. Jika $x < -2$ maka xf_DE[0] = 1.
2. Jika $x < -1$ maka xf_DE[0] = $-1 - x$; xf_DE[1] = $x - (-2)$
3. Jika $x < 0$ maka xf_DE[1] = $0 - x$; xf_DE[2] = $x - (-1)$
4. Jika $x < 1$ maka xf_DE[2] = $1 - x$; xf_DE[3] = $x - 0$
5. Jika $x < 2$ maka xf_DE[3] = $2 - x$; xf_DE[4] = $x - 1$

Dengan dasar itulah maka dibuat *listing* program dengan menggunakan bahasa C.



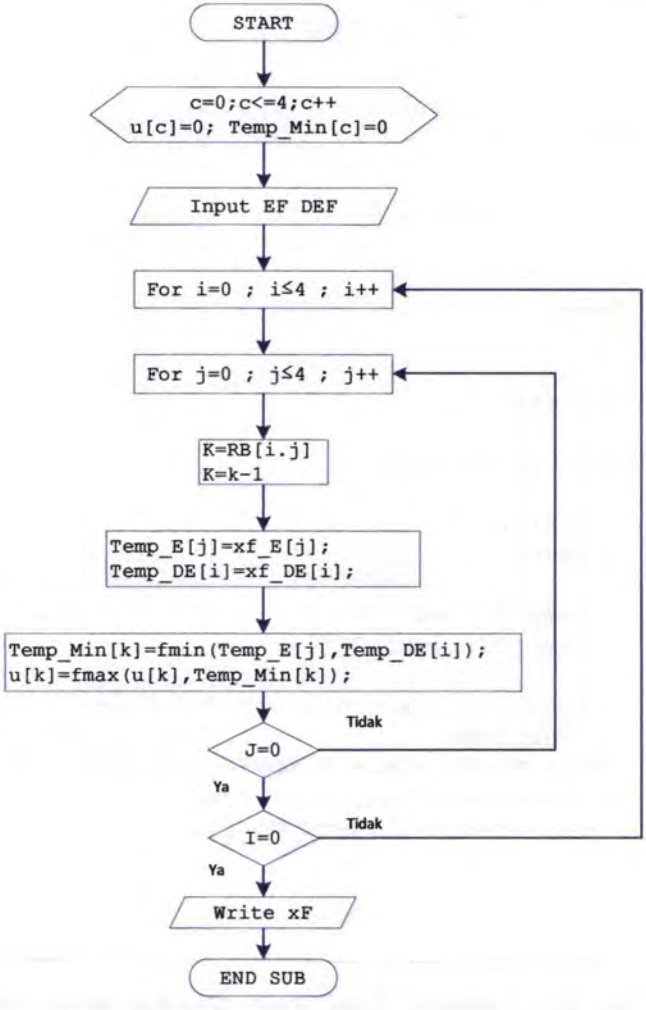
Gambar 3.18 *Flowchart subprogram delta error fuzzy*

Dari Gambar 3.18 maka dibuat program untuk fuzzyfikasi *delta error* sebagai berikut :

```
void Fuzzification_Delta_Error(float x)
{
  int b;

  for (b=0;b<=4;b++)
  {
      xf_DE[b]=0;
  }

  if (x<-2)
    xf_DE[0]=1;
  else if (x<-1)
  {
    xf_DE[0]=-1-x;
    xf_DE[1]=x-(-2);
  }
  else if (x<0)
  {
    xf_DE[1]=0-x;
    xf_DE[2]=x-(-1);
  }
  else if (x<1)
  {
    xf_DE[2]=1-x;
    xf_DE[3]=x-0;
  }
  else if (x<2)
  {
    xf_DE[3]=2-x;
    xf_DE[4]=x-1;
  }
  else
  {
    xf_DE[4]=1;
  }
}
```



Gambar 3.19 *Flowchart subprogram rule base fuzzy*

Dari gambar 3.19 maka dibuat program untuk *Rule Base Fuzzy* sebagai berikut :

```
void Rule_Base(void)
{

int i,j;
int temp;
int c;

  for (c=0;c<=4;c++)
  {
      u[c]=0;
      Temp_Min[c]=0;
  }

for(i=0;i<=4;i++)
{
    for(j=0;j<=4;j++)
    {
        k=r[i,j];
        k=k-1;

        Temp_E[j]=xf_E[j];
        Temp_DE[i]=xf_DE[i];

       //Mamdani (Generalize Modus Ponen/
         MaxofMin
       Temp_Min[k]=fmin(Temp_E[j],Temp_DE[i]);
       //temp=fmin(Temp_E[j],Temp_DE[i]);
       u[k]=fmax(u[k],Temp_Min[k]);

    }
}
```

Dari *listing* program diatas dapat dijelaskan bahwa untuk penentuan *rule base* menggunakan variabel c dengan tipe data integer (int) dengan rentang nilai dari -32768 s/d 32767. Dalam tabel *rule base* dikenal adanya *error* dan *delta error*. *Error (E)* merupakan kolom atau *j* dan *delta error (DE)* dalam baris atau *i*. Berikut adalah *rule base* yang digunakan untuk program.

Tabel 3.2 *Rule Base*

DE \ E	0	1	2	3	4
0	1	1	2	2	3
1	1	2	2	3	4
2	2	2	3	4	4
3	2	3	4	4	5
4	3	4	4	5	5

Dari Tabel 3.2 dibuatlah program *rule base* seperti pada *listing* program di atas. Kemudian dibuatlah *inference rule* dengan menggunakan metode mamdani dengan rumus :

```
Temp_Min[k]=fmin(Temp_E[j],Temp_DE[i]);
u[k]=fmax(u[k],Temp_Min[k]);
```

Sedangkan proses defuzzyfikasi berfungsi mentransformasikan aksi kontrol yang bersifat *fuzzy* menjadi besaran *crisp*. Hal ini dimaksudkan agar mendapatkan sinyal kontrol yang diinginkan.

Perancangan program defuzzyfikasi yang dilakukan adalah sebagai berikut :

```
void Deffuzification()
{

pemb=u[0]*c[0]+u[1]*c[1]+u[2]*c[2]+u[3]*c[3]+u
[4]*c[4];
peny=u[0]+u[1]+u[2]+u[3]+u[4];
U_pd=pemb/peny;
}
```

BAB IV
PENGUJIAN DAN ANALISIS

BAB IV
PENGUJIAN DAN ANALISA

Pada bab ini membahas tentang pengujian dan analisa dari perancangan dan implementasi *Direct Torque Control* untuk pengaturan kecepatan motor induksi 3 fasa menggunakan kontroler *Fuzzy* PI.

Pengujian dilakukan terhadap perangkat keras (*hardware*) dan perangkat lunak (*software*) sistem secara terpisah maupun setelah digabung menjadi satu.

4.1 Pengujian *Hardware* sistem

Pengujian *hardware* dilakukan bertujuan untuk mengetahui apakah *hardware* berfungsi dengan semestinya.

4.1.1 Pengujian *power supply DC*

Pengujian dilakukan untuk mengetahui daya yang dihasilkan oleh *power supply* yang dibuat. *Power supply* dirancang agar dapat menghasilkan tegangan +12 V DC, -12 V DC dan +5 V DC. Pengujian dilakukan dengan mengukur keluaran tegangan dengan menggunakan *multimeter digital*. Berikut adalah Tabel yang menampilkan hasil pengukuran.

Tabel 4.1 Hasil pengujian *power supply*

Tegangan seharusnya (Volt)	Tegangan terukur (Volt)
+ 12	+ 11.79
- 12	- 11.80
+ 5	+ 4.85

Dari Tabel 4.1 diketahui bahwa hasil pengukuran mendekati keluaran sesungguhnya tegangan DC yang diharapkan.

4.1.2 Pengujian Mikrokontroler

Mikrokontroler yang digunakan perlu diperiksa secara fisik yaitu menyangkut jalur-jalur PCB dan komponen yang dipakai pada rangkaian. Selanjutnya untuk menguji mikrokontroler yang digunakan

maka mikrokontroler harus diisi program terlebih dulu. Kemudian program yang telah diisikan ke dalam mikrokontroler dieksekusi. Program yang diisi harus dapat berjalan sesuai dengan yang diinginkan.

Pada tugas akhir kali ini, mikrokontroler akan diisi dengan program kontroler fuzzy PI. Program dibuat memakai *code vision AVR* yang kemudian di-*download* ke mikrokontroler untuk selanjutnya digunakan sebagai pengatur kecepatan motor induksi dengan bantuan *inverter.*

Hasil dari pengendalian oleh mikrokontroler ini akan ditampilkan pada sub-bab lain.

4.1.3 Pengujian *Operational Amplifier*

Operational amplifier adalah rangkaian yang berfungsi menguatkan tegangan. Besarnya nilai tegangan yang dihasilkan oleh penguatan ini tergantung pada perbandingan nilai resistansi yang dipakai. Nilai resistansi didapat dengan menggunakan perhitungan sebagai berikut :

$$Vout = \left(1 + \frac{R_2}{R_1}\right) Vin \qquad (4.1)$$

Dimana :

V_{out}	: Tegangan *Output* yang diinginkan
R_2 &R_1	: Nilai resistansi
V_{in}	: Tegangan *input*

Dari persamaan 4.1apabila diinginkan tegangan keluaran 10 volt sedangkan tegangan masukan yang diterima Op-Amp adalah 5 volt maka didapat nilai R_1 dan R_2 adalah sebesar 10 K Ω.

Setelah didapat nilai tersebut kemudian diimplementasikan ke dalam rangkaian yang kemudian diukur tegangan keluarannya. Tabel 4.2 menampilkan data hasil pengujian dari rangkaian Op-amp yang dipakai.

Tabel 4.2 Hasil pengujian *Op-Amp*

Tegangan *Input* (Volt)	Tegangan *Output* (Volt)
0	0
0.5	1.08
1.03	2.09
1.51	3.04
2.01	4.04
2.51	5.05
3.00	6.02
3.51	7.07
4.02	7.08
4.51	9.07
4.86	9.74

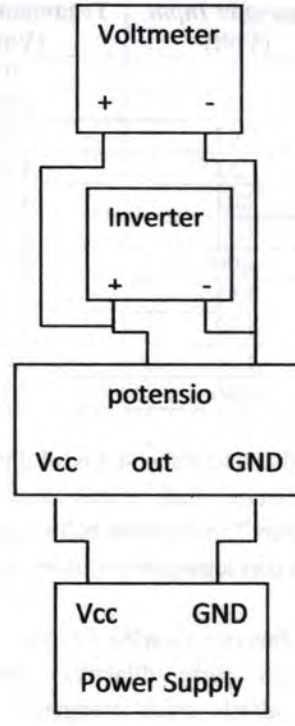
Seperti terlihat dari Tabel 4.2 hasil diperoleh dengan mengukur menggunakan *multimeter* baik untuk tegangan masukan maupun tegangan keluaran. Dari Tabel terlihat bahwa tegangan keluaran dari Op-Amp adalah dua kali dari tegangan masukan Op-Amp.

4.1.4 Pengujian *Inverter Toshiba VF-S9*

Inverter juga perlu dilakukan pengujian. Namun, pada pengujian kali ini adalah untuk mengetahui apakah *inverter* dapat dikendalikan dari perangkat lain. Pengujian dilakukan dengan cara sederhana menggunakan potensiometer yang dihubungkan ke sumber tegangan 12 volt.

Gambar 4.1 adalah diagram pengkabelan untuk pengujian *inverter* yang dikendalikan dari luar. Selain itu *inverter* juga diatur. Pengaturannya sebagai berikut :

1. Memilih menu dengan menekan MON
2. Dengan UP dan DOWN pilih FNOD kemudian ENT
3. FNOD diisi nilainya nol kemudian tekan ENT
4. Dengan UP dan DOWN pilih F- - - - tekan ENT hingga muncul F100. Dengan tombol UP dan DOWN buat hingga F200 kemudian tekan ENT



Gambar 4.1 *Wiring* pengujian *inverter*

5. Dengan UP dan DOWN pilih F- - - - tekan ENT hingga muncul F100. Dengan tombol UP dan DOWN buat hingga F200 kemudian tekan ENT
6. Nilai F200 diberi nilai 5 dengan menekan UP dan DOWN

Setelah diprogram, maka dengan menggunakan potensio *inverter* dapat dikendalikan untuk memutar motor induksi.

4.1.5 Pengujian Motor Induksi.

Pengujian dilakukan dengan menghubungkan motor induksi dengan sumber 3 fasa dari *inverter*. Lalu mengatur frekuensi pada *inverter* dan mengambil data Rpm motor dengan *Tachometer*.

Tabel 4.3 Data hasil pengukuran kecepatan motor

No	Frekuensi (Hz)	Putaran (Rpm)	No	Frekuensi (Hz)	Putaran (Rpm)
1	60	1456.7	14	0	0
2	55	1336.7	15	5	93.3
3	50	1216.7	16	10	306.7
4	45	1100	17	15	390
5	40	976.7	18	20	503.3
6	35	866.7	19	25	633.3
7	30	753.3	20	30	753.3
8	25	636.7	21	35	860
9	20	520	22	40	976.7
10	15	383.3	23	45	1096.7
11	10	300	24	50	1220
12	5	93.3	25	55	1340
13	0	0	26	60	1456.7

Tabel 4.3 menunjukan hubungan antara besarnya frekuensi berbanding dengan kecepatan motor. Pengujian dilakukan dengan menggunakan motor DC yang difungsikan sebagai *tachogenerator*. Motor DC yang dipakai memiliki nilai keluaran 30V/1000rpm. Dari perbandingan tersebut dapat diketahui kecepatan putar motor induksi dengan melakukan perhitungan. Sehingga didapat hasil penghitungan seperti pada Tabel 4.3.

4.1.6 Pengujian *Tachogenerator*

Pengujian dilakukan dengan mengkopel *tachogenerator* dengan motor induksi. Ketika motor induksi berputar maka akan memutar juga *tachogenerator*. Saat *tachogenerator* berputar akan mengeluarkan tegangan. Nilai tegangan ini diukur dengan *multimeter*. Tabel 4.4 menampilkan hasil pengukuran.

Tabel 4.4 Hasil pengukuran *tachogenerator*

Output Inverter (Hz)	*Output Tacho* (Volt)
60	9.32
50	7.66
40	6.08
30	4.52
20	3.02
10	1.58
0	0

Dari Tabel 4.4 diketahui bahwa apabila putaran motor induksi makin kencang (frekuensi makin besar) maka tegangan keluaran *tachogenerator* juga makin besar.

4.1.7 Pengujian pembagi tegangan

Rangkaian pembagi tegangan bertujuan membagi tegangan yang diterimanya. Diperlukan pembagi tegangan karena tegangan keluaran maksimal *tachogenerator* mencapai 10 volt sedangkan tegangan masukan mikrokontroler maksimal hanya boleh sebesar 5 volt. Masukan rangkaian pembagi tegangan dihubungkan dengan keluaran *tachogenerator*. Sedangkan keluaran rangkaian pembagi tegangan dihubungkan dengan mikrokontroler. Tabel 4.5 menampilkan hasil pengujian rangkaian ini.

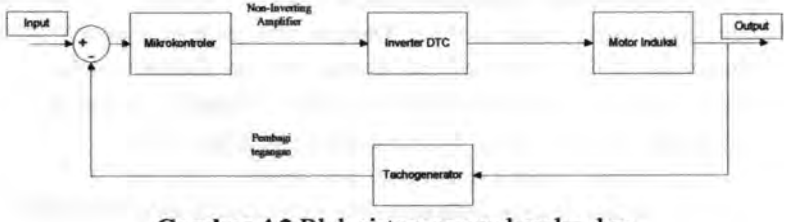
Tabel 4.5 Hasil pengujian rangkaian pembagi tegangan

***Output* Tegangan *Tacho* (Volt)**	***Output* Pembagi Tegangan (Volt)**
9.31	4.94
7.66	4.03
6.08	3.05
4.52	2.5
3.02	1.49
1.58	0.71
0	0

Dari Tabel 4.5 menunjukkan bahwa rangkaian pembagi tegangan ini bekerja dengan baik yaitu menurunkan tegangan yang diterimanya menjadi setengahnya sehingga aman untuk menuju mikrokontroler.

4.2 Pengujian *software* sistem

Setelah membuat program untuk kontroler *fuzzy* PI menggunakan *code vision AVR* selesai, maka program di-*download* ke dalam mikrokontroler. Selanjutnya mikrokontroler dirangkai seperti pada Gambar 4.2 yaitu Gambar blok sistem secara keseluruhan.



Gambar 4.2 Blok sistem secara keseluruhan

Dari Gambar 4.2 dapat dijelaskan bahwa mikrokontroler dihubungkan ke rangkaian *op-amp* melalui *port* D *pin* OC1A selain itu rangkaian LCD sebagai penampil dihubungkan dengan *port* B, sedangkan *keypad* sebagai masukan mikrokontroler dihubungkan dengan *port* C. Mikrokontroler diberi tegangan 12 VDC. Rangkaian *Op-*

amp diberi tegangan 12 VDC juga. Rangkaian *op-amp* dihubungkan pada terminal *inverter* tetapi sebelumnya *inverter* diatur pada *mode manual* terlebih dulu. Selanjutnya dari *inverter* dihubungkan ke motor induksi. Motor induksi dikopel dengan *tachogenerator*. Tegangan keluaran *tachogenerator* ini dihubungkan dengan *port* A pada mikrokontroler.

4.3 Hasil pengujian sistem keseluruhan

Setelah semua *hardware* rangkaian dirangkai sesuai dengan blok sistem yang dirancang dan juga *software* telah diisikan ke dalam mikrokontroler maka dilakukan pengujian sistem secara keseluruhan.

Hasil pengujian ini bertujuan mengetahui kondisi keluaran sistem telah sesuai dengan masukan sistem atau tidak. Pengujian dilakukan untuk mengetahui kinerja kontroler *fuzzy* PI terhadap pengaturan kecepatan motor induksi.

Dari hasil pengujian ini kemudian diambil datanya dan dilakukan analisa dengan menggunakan osiloskop untuk mengetahui respon kontroler terhadap pengaturan kecepatan motor.

4.4 Hasil pengamatan pada osiloskop

Osiloskop digunakan untuk melihat sinyal keluaran yang dihasilkan dari *tachogenerator*. Dengan alat ini bisa diketahui respon motor saat diberi masukan dari *keypad* melalui mikrokontroler. *Probe* dari osiloskop dihubungkan pada keluaran dari pembagi tegangan untuk mengetahui bentuk sinyal dari pengaturan kecepatan motor.

Selain dengan osiloskop pengamatan juga dapat dilakukan dengan melihat besaran putaran motor yang terukur oleh mikrokontroler melalui LCD. Besaran *setpoint, error* dan sinyal kontrol juga tertampil pada layar LCD.

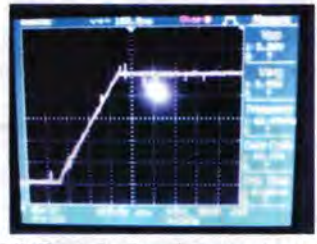
4.4.1 Bentuk sinyal pengaturan kecepatan motor

Sinyal diambil langsung dari keluaran rangkaian pembagi tegangan. Mikrokontroler diberi masukan *setpoint* menggunakan

keypad. Sinyal yang diperoleh untuk menganalisis kecepatan motor yang dikendalikan oleh kontroler *fuzzy* PI.

4.4.1.1 Bentuk sinyal saat *setpoint* 1400 rpm

Mikrokontroler diberi *setpoint* 1400 rpm kemudian nilai Ost diisi nol dan nilai Ki diisi 1.0 maka akan terlihat sinyal seperti Gambar 4.3.

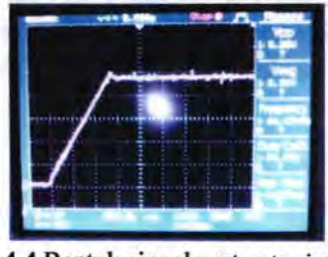


Gambar 4.3 Bentuk sinyal saat *setpoint* 1400 rpm

Dari Gambar 4.3 pada saat diberi *setpoint* 1400 tidak terjadi *overshoot*. Dengan tegangan keluaran *tacho* melalui pembagi tegangan adalah 5 V. dan *rise time* 6.5 detik. Dari tegangan yang dihasilkan bisa dihitung kecepatan motor dengan membandingkan tegangan terhadap kecepatan menjadi $\frac{5}{5} x 1400\ rpm = 1400\ rpm$.

4.4.1.2 Bentuk sinyal saat *setpoint* 1300 rpm

Mikrokontroler diberi *setpoint* 1300 rpm kemudian nilai Ost diisi nol dan nilai Ki diisi 1.0 maka akan terlihat sinyal seperti Gambar 4.4.

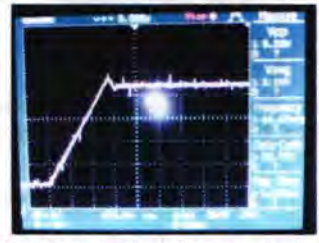


Gambar 4.4 Bentuk sinyal saat *setpoint* 1300rpm

Dari Gambar 4.4 didapat data bahwa terjadi *overshoot* sebesar 0.2 V = 4.2 %. *Rise time* 6.8 s dan tegangan keluaran *tacho* 4.72 V. Dari tegangan yang dihasilkan bisa dihitung kecepatan motor dengan membandingkan tegangan terhadap kecepatan menjadi $\frac{4.72}{5} x1400\ rpm = 1321.6\ rpm \approx 1322\ rpm.$

4.4.1.3 Bentuk sinyal saat *setpoint* 1200 rpm

Mikrokontroler diberi *setpoint* 1200 rpm kemudian nilai Ost diisi nol dan nilai Ki diisi 1.0 maka akan terlihat sinyal seperti Gambar 4.5.

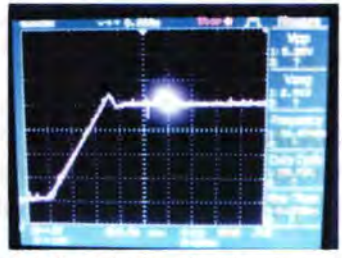


Gambar 4.5 Bentuk sinyal saat *setpoint* 1200 rpm

Dari Gambar 4.5 didapatkan tegangan keluaran *tacho* sebesar 4.36 V. *overshoot* sebesar 0.28 V = 6.4 % dan *rise time* 6.9 s. Dari tegangan yang dihasilkan bisa dihitung kecepatan motor dengan membandingkan tegangan terhadap kecepatan menjadi $\frac{4.36}{5} x1400\ rpm = 1220.8\ rpm \approx 1221\ rpm.$

4.4.1.4 Bentuk sinyal saat *setpoint* 1100 rpm

Mikrokontroler diberi *setpoint* 1100 rpm kemudian nilai Ost diisi nol dan nilai Ki diisi 1.0 maka akan terlihat sinyal seperti Gambar 4.6.

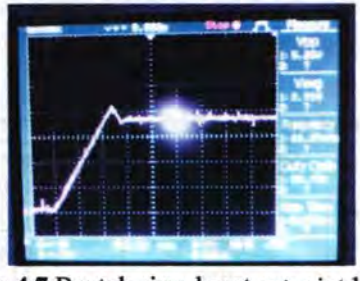


Gambar 4.6 Bentuk sinyal saat *setpoint* 1100 rpm

Dari Gambar 4.6 didapatkan tegangan keluaran *tacho* sebesar 4.00 V. *overshoot* sebesar 0.4 V = 10 % dan *rise time* 3.6 s. Dari tegangan yang dihasilkan bisa dihitung kecepatan motor dengan membandingkan tegangan terhadap kecepatan menjadi $\frac{4}{5}x1400\ rpm = 1120\ rpm$.

4.4.1.5 Bentuk sinyal saat *setpoint* 1000 rpm

Mikrokontroler diberi *setpoint* 1000 rpm kemudian nilai Ost diisi nol dan nilai Ki diisi 1.0 maka akan terlihat sinyal seperti Gambar 4.7.



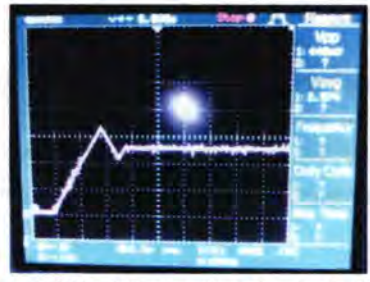
Gambar 4.7 Bentuk sinyal saat *setpoint* 1000 rpm

Dari Gambar 4.7 didapatkan tegangan keluaran *tacho* sebesar 3.64 V. *overshoot* sebesar 0.52 V = 14 % dan *rise time* 5.9 s. Dari tegangan yang dihasilkan bisa dihitung kecepatan motor dengan

membandingkan tegangan terhadap kecepatan menjadi $\frac{3.64}{5} x1400\ rpm = 1019.2\ rpm \approx 1019\ rpm.$

4.4.1.6 Bentuk sinyal saat *setpoint* 700 rpm

Mikrokontroler diberi *setpoint* 700 rpm kemudian nilai Ost diisi nol dan nilai Ki diisi 1.0 maka akan terlihat sinyal seperti Gambar 4.8.



Gambar 4.8 Bentuk sinyal saat *setpoint* 700 rpm

Dari Gambar 4.8 didapatkan tegangan keluaran *tacho* sebesar 2.6 V. *overshoot* sebesar 0.8 V = 30 % dan *rise time* 4.5 s. Dari tegangan yang dihasilkan bisa dihitung kecepatan motor dengan membandingkan tegangan terhadap kecepatan menjadi $\frac{2.6}{5} x1400\ rpm = 728\ rpm.$

Tabel 4.6 Tabel Perbandingan putaran motor

No	*Setpoint (Rpm)*	Terukur (Rpm)
1	1400	1400
2	1300	1322
3	1200	1221
4	1100	1120
5	1000	1019
6	700	728

Tabel 4.6 menunjukkan hasil perbandingan putaran motor antara setpoint yang diberikan dengan putaran motor yang terukur.

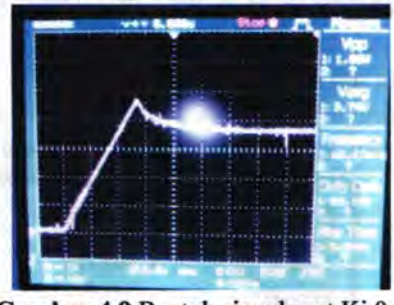
Dari berbagai percobaan diatas diketahui bahwa tegangan keluaran *tacho* melalui pembagi tegangan sebanding dengan nilai *setpoint* yang diberikan. Selain itu, makin pelan putaran motor maka akan terjadi *overshoot* sedangkan *rise time* akan makin cepat. Oleh karena makin besar *overshoot* apabila motor berputar makin pelan maka perlu dilakukan minimalisasi *overshoot* dengan pengubahan Ki pada masukan mikrokontroler.

4.4.2 Bentuk sinyal saat pengaturan Ki

Dari data kecepatan motor dapat dianalisa bahwa makin pelan putaran motor makan akan menyebabkan terjadinya *overshoot* makin besar. Untuk meminimalkan *overshoot* maka dilakukan *tuning* nilai Ki. Untuk analisa digunakan kecepatan tetap yaitu 1000 rpm kemudian nilai Ki diubah ubah mulai dari 0.1 sampai 5 dengan kenaikan setiap 0.5.

4.4.2.1 Bentuk sinyal saat Ki 0.1

Mikrokontroler diberi *setpoint* 1000 rpm, nilai Ost nol dan nilai Ki 0.1. Maka akan terlihat sinyal seperti Gambar 4.9.

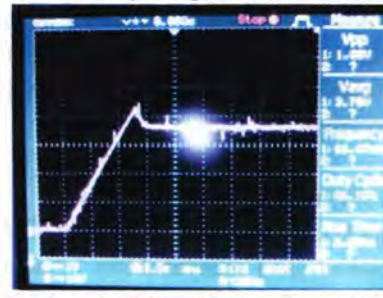


Gambar 4.9 Bentuk sinyal saat Ki 0.1

Dari Gambar 4.9 didapat data bahwa tegangan keluaran *tacho* melalui pembagi tegangan sebesar 3.64 V dan *overshoot* sebesar 1.04 V =28 % sedangkan untuk *rise time* sebesar 5 s. Dari tegangan yang dihasilkan bisa dihitung kecepatan motor dengan membandingkan tegangan terhadap kecepatan menjadi $\frac{3.64}{5} x1400\ rpm = 1019.2\ rpm.$

4.4.2.2 Bentuk sinyal saat Ki 0.5

Mikrokontroler diberi *setpoint* 1000 rpm, nilai Ost nol dan nilai Ki 0.5. Maka akan terlihat sinyal seperti Gambar 4.10.

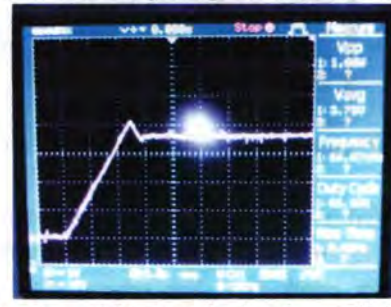


Gambar 4.10 Bentuk sinyal saat Ki 0.5

Dari Gambar 4.10 didapat data bahwa tegangan keluaran *tacho* melalui pembagi tegangan sebesar 3.64 V dan *overshoot* sebesar 0.72 V = 19.7 % sedangkan untuk *rise time* sebesar 5.6 s. Dari tegangan yang dihasilkan bisa dihitung kecepatan motor dengan membandingkan tegangan terhadap kecepatan menjadi $\frac{3.64}{5} x1400\ rpm = 1019.2\ rpm$.

4.4.2.3 Bentuk sinyal saat Ki 1.00

Mikrokontroler diberi *setpoint* 1000 rpm, nilai Ost nol dan nilai Ki 1.00. Maka akan terlihat sinyal seperti Gambar 4.11.

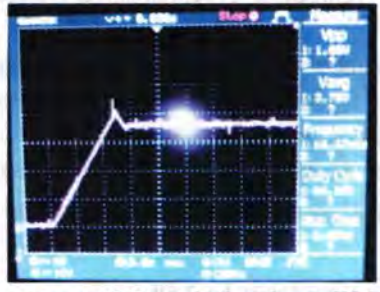


Gambar 4.11 Bentuk sinyal saat Ki 1.00

Dari Gambar 4.11 didapat data bahwa tegangan keluaran *tacho* melalui pembagi tegangan sebesar 3.64 V dan *overshoot* sebesar 0.52 V = 14 % sedangkan untuk *rise time* sebesar 5.3 s. Dari tegangan yang dihasilkan bisa dihitung kecepatan motor dengan membandingkan tegangan terhadap kecepatan menjadi $\frac{3.64}{5} x1400\ rpm = 1019.2\ rpm$.

4.4.2.4 Bentuk sinyal saat Ki 1.50

Mikrokontroler diberi *setpoint* 1000 rpm, nilai Ost nol dan nilai Ki 1.50. Maka akan terlihat sinyal seperti Gambar 4.12.

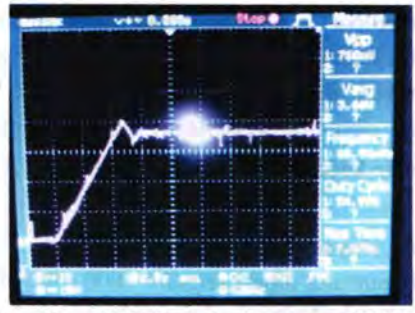


Gambar 4.12 Bentuk sinyal saat Ki 1.50

Dari Gambar 4.12 didapat data bahwa tegangan keluaran *tacho* melalui pembagi tegangan sebesar 3.64 V dan *overshoot* sebesar 0.40 V = 10.9 % sedangkan untuk *rise time* sebesar 5.2 s. Dari tegangan yang dihasilkan bisa dihitung kecepatan motor dengan membandingkan tegangan terhadap kecepatan menjadi $\frac{3.64}{5} x1400\ rpm = 1019.2\ rpm$.

4.4.2.5 Bentuk sinyal saat Ki 2.00

Mikrokontroler diberi *setpoint* 1000 rpm, nilai Ost nol dan nilai Ki 2.00. Maka akan terlihat sinyal seperti Gambar 4.13.



Gambar 4.13 Bentuk sinyal saat Ki 2.00

Dari Gambar 4.13 didapat data bahwa tegangan keluaran *tacho* melalui pembagi tegangan sebesar 3.64 V dan *overshoot* sebesar 0.32 V = 8.7 % sedangkan untuk *rise time* sebesar 5.5 s. Dari tegangan yang dihasilkan bisa dihitung kecepatan motor dengan membandingkan tegangan terhadap kecepatan menjadi $\frac{3.64}{5} x1400\ rpm = 1019.2\ rpm$.

4.4.2.6 Bentuk sinyal saat Ki 2.50

Mikrokontroler diberi *setpoint* 1000 rpm, nilai Ost nol dan nilai Ki 2.50. Maka akan terlihat sinyal seperti Gambar 4.14.

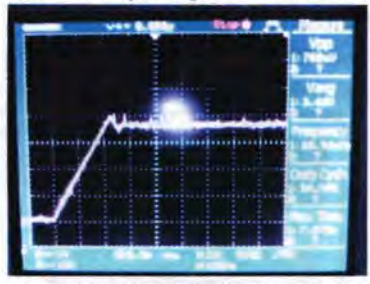
Gambar 4.14 Bentuk sinyal saat Ki 2.50

Dari Gambar 4.14 didapat data bahwa tegangan keluaran *tacho* melalui pembagi tegangan sebesar 3.64 V dan *overshoot* sebesar 0.32 V = 8.7 % sedangkan untuk *rise time* sebesar 5.4 s. Dari tegangan yang

dihasilkan bisa dihitung kecepatan motor dengan membandingkan tegangan terhadap kecepatan menjadi $\frac{3.64}{5} x1400\ rpm = 1019.2\ rpm$.

4.4.2.7 Bentuk sinyal saat Ki 3.00

Mikrokontroler diberi *setpoint* 1000 rpm, nilai Ost nol dan nilai Ki 3.00. Maka akan terlihat sinyal seperti Gambar 4.15.

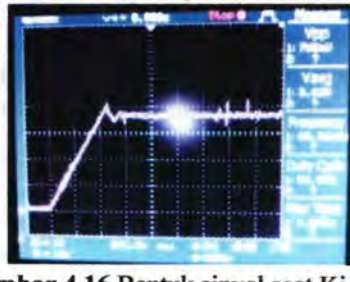


Gambar 4.15 Bentuk sinyal saat Ki 3.00

Dari Gambar 4.15 didapat data bahwa tegangan keluaran *tacho* melalui pembagi tegangan sebesar 3.64 V dan *overshoot* sebesar 0.28 V = 7.6 % sedangkan untuk *rise time* sebesar 5.2 s. Dari tegangan yang dihasilkan bisa dihitung kecepatan motor dengan membandingkan tegangan terhadap kecepatan menjadi $\frac{3.64}{5} x1400\ rpm = 1019.2\ rpm$.

4.4.2.8 Bentuk sinyal saat Ki 3.50

Mikrokontroler diberi *setpoint* 1000 rpm, nilai Ost nol dan nilai Ki 3.50. Maka akan terlihat sinyal seperti Gambar 4.16.

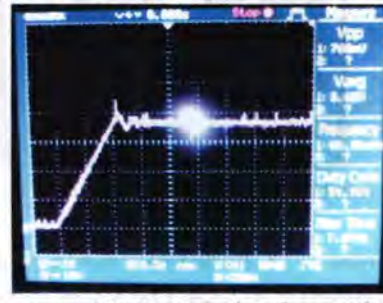


Gambar 4.16 Bentuk sinyal saat Ki 3.50

Dari Gambar 4.16 didapat data bahwa tegangan keluaran *tacho* melalui pembagi tegangan sebesar 3.64 V dan *overshoot* sebesar 0.28 V = 7.6 % sedangkan untuk *rise time* sebesar 5.2 s. Dari tegangan yang dihasilkan bisa dihitung kecepatan motor dengan membandingkan tegangan terhadap kecepatan menjadi $\frac{3.64}{5} x1400\ rpm = 1019.2\ rpm$.

4.4.2.9 Bentuk sinyal saat Ki 4.00

Mikrokontroler diberi *setpoint* 1000 rpm, nilai Ost nol dan nilai Ki 4.00. Maka akan terlihat sinyal seperti Gambar 4.17.

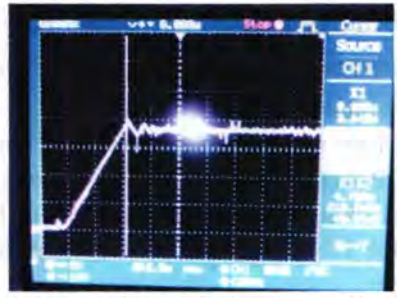


Gambar 4.17 Bentuk sinyal saat Ki 4.00

Dari Gambar 4.17 didapat data bahwa tegangan keluaran *tacho* melalui pembagi tegangan sebesar 3.64 V dan *overshoot* sebesar 0.26 V = 7.1 % sedangkan untuk *rise time* sebesar 5.3 s. Dari tegangan yang dihasilkan bisa dihitung kecepatan motor dengan membandingkan tegangan terhadap kecepatan menjadi $\frac{3.64}{5} x1400\ rpm = 1019.2\ rpm$.

4.4.2.10 Bentuk sinyal saat Ki 4.50

Mikrokontroler diberi *setpoint* 1000 rpm, nilai Ost nol dan nilai Ki 4.50. Maka akan terlihat sinyal seperti Gambar 4.18.

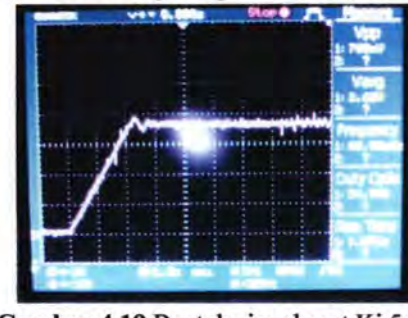


Gambar 4.18 Bentuk sinyal saat Ki 4.50

Dari Gambar 4.18 didapat data bahwa tegangan keluaran *tacho* melalui pembagi tegangan sebesar 3.64 V dan *overshoot* sebesar 0.32 V = 8.7 % sedangkan untuk *rise time* sebesar 5.2 s. Dari tegangan yang dihasilkan bisa dihitung kecepatan motor dengan membandingkan tegangan terhadap kecepatan menjadi $\frac{3.64}{5} x1400\ rpm = 1019.2\ rpm$.

4.4.2.11 Bentuk sinyal saat Ki 5.00

Mikrokontroler diberi *setpoint* 1000 rpm, nilai Ost nol dan nilai Ki 5.00. Maka akan terlihat sinyal seperti Gambar 4.19.



Gambar 4.19 Bentuk sinyal saat Ki 5.00

Dari Gambar 4.19 didapat data bahwa tegangan keluaran *tacho* melalui pembagi tegangan sebesar 3.64 V dan *overshoot* sebesar 0.24 V = 6.5 % sedangkan untuk *rise time* sebesar 5.1 s. Dari tegangan yang

dihasilkan bisa dihitung kecepatan motor dengan membandingkan tegangan terhadap kecepatan menjadi $\frac{3.64}{5} x1400\ rpm = 1019.2\ rpm$.

Dari perubahan Ki tersebut maka dapat disimpulkan bahwa pada kecepatan 1000 rpm untuk mendapatkan bentuk sinyal yang bagus, nilai Ki = 5.00. Pada saat Ki = 5.00 *overshoot* yang terjadi adalah 6.5 % dan bila Ki diisi lebih dari 5 maka akan terjadi osilasi sehingga putaran motor tidak stabil. Tabel 4.7 menampilkan perbandingan nilai Ki.

Tabel 4.7 Perbandingan nilai Ki

No	*Ki*	*Overshoot (%)*	*Rise Time (s)*
1	0.1	28	5
2	0.5	19.7	5.6
3	1.0	14	5.3
4	1.5	10.9	5.2
5	2.0	8.7	5.5
6	2.5	8.7	5.4
7	3.0	7.6	5.2
8	3.5	7.6	5.2
9	4.0	7.1	5.3
10	4.5	8.7	5.2
11	5.0	6.5	5.1

BAB V
PENUTUP

BAB V
KESIMPULAN DAN SARAN

5.1. Kesimpulan

Dari Tugas Akhir ini, ada beberapa hal yang dapat disimpulkan, yaitu :

1. Dalam merancang dan mengimplementasikan kontroler *fuzzy* PI untuk pengaturan kecepatan motor induksi meliputi fuzzyfikasi, *rule base*, dan defuzzyfikasi.
2. Dengan kontroler *fuzzy* PI nilai keluaran kecepatan motor induksi hampir sama atau mendekati dengan nilai *setpoint* yang diberikan.
3. Semakin rendah putaran motor maka akan semakin besar terjadinya *overshoot*.
4. Respon sistem dapat diperbaiki dengan mengubah-ubah nilai Ki pada mikrokontroler.
5. Pada kecepatan maksimal 1400 rpm dengan Ki bernilai 1 tidak terjadi *overshoot* dan kecepatan motor sama dengan *setpoint* yang diberikan.
6. Pada kecepatan mulai 1300 rpm dan lebih rendah lagi maka *overshoot* yang terjadi makin besar untuk itu diperlukan *tuning* nilai Ki.
7. Nilai K_Upd yang menghasilkan sinyal dengan *overshoot* kecil pada 1000 rpm adalah bernilai 5. Dan apabila lebih dari 5 maka akan terjadi osilasi.

5.2. Saran

Saran yang dapat diberikan untuk Tugas Akhir ini adalah :

1. Perlu penyempurnaan *rule base* agar bisa diperoleh sinyal kontrol yang lebih baik lagi.
2. Diharapkan ke depannya hasil implementasi ini dapat lebih disempurnakan lagi.

DAFTAR PUSTAKA

DAFTAR PUSTAKA

[1]. Surya Wiranata, Desain konstruksi motor induksi tiga fasa mini berdaya <1/4 PK, Tugas Akhir, 2006

[2]. David H. Sirait, Analisis Starting Motor Induksi Tiga Phasa Pada PT. Berlian Unggas Sakti TJ. Morawa, Tugas Akhir, Medan. 2008.

[3]. Zulfatman, Desain Pengendalian Kecepatan Motor Induksi 3 Phase Dengan PID Kontroller, GAMMA Volume 1, Nomor 2, Maret 2006.

[4]. Andrzej M. Trzynadlowski. *Control of Induction Motor.* San Diego, USA : Academic Press. 2001.

[5]. Pujol Arias Antoni, Thesis: *Improvement In Direct Torque Control of Induction Motor.*, Nopember, 2000.

[6]. R. Toufouti, S. Meziane dan H. Benalla, *Direct Torque Control For Induction Motor Using Intelligent Techniques*, Journal of Theoretical and Applied Information Technology, Algeria, 2007.

[7]. Rashid Muhammad H, *Power Electronics Handbook*, Academic Press, California USA, 2001.

[8]. Bejo agus, C & AVR rahasia kemudahan bahas C dalam mikrokontroler ATMEGA 8535, edisi pertama, Yogyakarta, Graha Ilmu, 2008.

[9]. Winoto Ardi, Mikrokontroler AVR ATMEGA 8/16/32/8535 dan pemrograman nya dengan bahasa C pada Win AVR, Informatika, Bandung, 2008.

[10]. Rubiantono Yoyok, Inverter 3 fasa 380 V AC untuk mengatur kecepatan motor induksi 3 fasa dengan metode pengaturan frekuensi yang berbasis mikrokontroler AT89C51, Tugas Akhir, Yogyakarta, 2008.

[Halaman ini sengaja dikosongkan]

RIWAYAT HIDUP

```
#include <mega16.h>
#include <stdio.h>
#include <delay.h>
#include <math.h>

#define PB_Back PINC.0
#define PB_Left PINC.1
#define PB_Enter PINC.6
#define PB_Right PINC.7
#define Motor OCR1A


#define FIRST_ADC_INPUT 0
#define LAST_ADC_INPUT 1
unsigned char adc_data[LAST_ADC_INPUT-FIRST_ADC_INPUT+1];
#define ADC_VREF_TYPE 0x20

// ADC interrupt service routine
// with auto input scanning
interrupt [ADC_INT] void adc_isr(void)
{
register static unsigned char input_index=0;
// Read the 8 most significant bits
// of the AD conversion result
adc_data[input_index]=ADCH;
// Select next ADC input
if (++input_index > (LAST_ADC_INPUT-FIRST_ADC_INPUT))
   input_index=0;
ADMUX=(FIRST_ADC_INPUT | (ADC_VREF_TYPE &
0xff))+input_index;
// Start the AD conversion
ADCSRA|=0x40;
}


eeprom int Sp_Eep=500, Kp_Eep=0, Ki_Eep=0, Speed_Eep=0;
eeprom float Ost_Eep=0, Kupd_Eep=1;

// Alphanumeric LCD Module functions
#asm
   .equ __lcd_port=0x18 ;PORTB
#endasm
```

```
#include <lcd.h>

float dat1=0.6,dat2=0.4,dat3,dat4;
float Temp_E[]={0,0,0,0,0};
float Temp_DE[]={0,0,0,0,0};
float Temp_Min[]={0,0,0,0,0};
float Temp_Max[]={0,0,0,0,0};
float u_Last[]={0,0,0,0,0};

int k;
float Temp_SP;
int Sp, Speed,menu,sinyal_kontrol,u_kontrol;
float
tacho_value,error=0,error_last=0,error_delta=0,pre_value=0,
pasca_value=0,measurement;
float error_fuzzy=0,error_delta_fuzzy=0;
float Ke,Kde,Mul_Ke,Mul_Kde,Ku=1,Ost=0,Kupd=1;
float pemb,peny,U_pd,U_ki,U_pi,U_pi_last;
char buffer[16],buffer2[16];
float xf_E[]={0,0,0,0,0};
float xf_DE[]={0,0,0,0,0};
float u[]={0,0,0,0,0};
int c[]={-2,-1,0,1,2};
int r[5][5]={{1,1,2,2,3},
             {1,2,2,3,4},
             {2,2,3,4,4},
             {2,3,4,4,5},
             {3,4,4,5,5}};

void read_tacho(void)
{
  tacho_value = ((float) adc_data[0]/255)*5;
  measurement = tacho_value * 280;
}

void reset_var(void)
{
      ;
}
```

```
void lcd_menus(void)
{
 Loop:

        Sp=Sp_Eep;
        Ost=Ost_Eep;
        Kupd=Kupd_Eep;

        lcd_clear();
        lcd_gotoxy(0,0);
        lcd_putsf("~~ Motor-Menu ~~");
        lcd_gotoxy(0,1);
        lcd_putsf("<< Run    Menu >>");
        delay_ms(100);

        if (PB_Enter==0)
        {
                menu=1;
                goto Mnu;
        }
        else if (PB_Back==0)
        {
                reset_var();
                goto exit;
        }
        else goto Loop;

//=========================================================
//>>>>>>>>>>          Daftar LCD Menu       <<<<<<<<<<<<<
//=========================================================
Mnu:
      delay_ms(50);
      switch (menu) {
      case 1 :
                lcd_clear();
                lcd_gotoxy(0,0);
                lcd_putsf(" 1.Set Setpoint");
                lcd_gotoxy(0,1);
                lcd_putsf("<<              >>");
                delay_ms(100);
                if (PB_Enter==0){delay_ms(100); goto
Set_Setpoint;}
                break;
```

```
case 2 :
          lcd_clear();
          lcd_gotoxy(0,0);
          lcd_putsf(" 2.Set Ost   ");
          lcd_gotoxy(0,1);
          lcd_putsf("<<            >>");
          delay_ms(100);
          if (PB_Enter==0){delay_ms(100); goto
              Set_Ost;}
          break;
case 3 :
          lcd_clear();
          lcd_gotoxy(0,0);
          lcd_putsf(" 3.Set Kupd  ");
          lcd_gotoxy(0,1);
          lcd_putsf("<<            >>");
          delay_ms(100);
          if (PB_Enter==0){delay_ms(100); goto
              Set_Kupd;}
          break;
default :
          lcd_clear();
          lcd_gotoxy(0,0);
          lcd_putsf(" Salah Mas  ");
          lcd_gotoxy(0,1);
          lcd_putsf("<<            >>");
          delay_ms(100);
}
if(PB_Back==0){delay_ms(100); goto exit;}
if(PB_Right==0){menu++;delay_ms(100);}
if(PB_Left==0){menu--;delay_ms(100);}
if (menu>3) menu=1;
if (menu<1) menu=3;
goto Mnu;
```

```
//================================================
//>>>>>>>>>>            Isi LCD Menu       <<<<<<<<<
//================================================
Set_Setpoint:
      if(PB_Right==0){Sp=Sp+50;delay_ms(1);}
       if(PB_Left==0){Sp=Sp-50;delay_ms(1);}
       if(Sp>1400)Sp=0;
       if(Sp<0)Sp=1400;
       lcd_clear();
       lcd_gotoxy(0,0);
       lcd_putsf("   SP value:     ");
       lcd_gotoxy(0,1);
       sprintf(buffer,"-)      %d     (+",Sp);
        lcd_puts(buffer);
        delay_ms(100);
        if (PB_Enter==0)
        {
              delay_ms(5);
              Sp_Eep=Sp;
              lcd_clear();
               lcd_gotoxy(0,0);
               lcd_putsf("  SP changed  ");
               reset_var();
               delay_ms(1300);
        }
        if(PB_Back==0){delay_ms(100); goto Mnu;}
        else goto Set_Setpoint;
Set_Ost:
      if(PB_Right==0){Ost=Ost+0.1;delay_ms(1);}
       if(PB_Left==0){Ost=Ost-0.1;delay_ms(1);}
       if(Ost>50)Ost=0;
       if(Ost<0)Ost=50;
       lcd_clear();
       lcd_gotoxy(0,0);
       lcd_putsf("   Ost value : ");
       lcd_gotoxy(0,1);
       sprintf(buffer,"-)     %.1f     (+",Ost);
      lcd_puts(buffer);
      delay_ms(100);
      if (PB_Enter==0)
       {
```

```
            delay_ms(5);
            Ost_Eep=Ost;
            lcd_clear();
             lcd_gotoxy(0,0);
             lcd_putsf(" Ost changed  ");
             reset_var();
             delay_ms(1300);
        }
        if(PB_Back==0){delay_ms(100); goto Mnu;}
        else goto Set_Ost;


Set_Kupd:
        if(PB_Right==0){Kupd=Kupd+0.1;delay_ms(1);}
      if(PB_Left==0){Kupd=Kupd-0.1;delay_ms(1);}
      if(Kupd>50)Kupd=0.1;
      if(Kupd<0.1)Kupd=50;
      lcd_clear();
      lcd_gotoxy(0,0);
      lcd_putsf("  Kupd value : ");
      lcd_gotoxy(0,1);
      sprintf(buffer,"-)     %.1f     (+",Kupd);
       lcd_puts(buffer);
       delay_ms(100);
       if (PB_Enter==0)
       {
              delay_ms(5);
              Kupd_Eep=Kupd;
              lcd_clear();
             lcd_gotoxy(0,0);
             lcd_putsf(" Kupd changed  ");
             reset_var();
             delay_ms(1300);
       }
       if(PB_Back==0){delay_ms(100); goto Mnu;}
       else goto Set_Kupd;


  exit:

}
```

```
void Fuzzification_Error(float x)
{
  int a;

  for (a=0;a<=4;a++)
  {
      xf_E[a]=0;
  }

  if (x<-2)
    xf_E[0]=1;
  else if (x<-1)
  {
    xf_E[0]=-1-x;
    xf_E[1]=x-(-2);
  }
  else if (x<0)
  {
    xf_E[1]=0-x;
    xf_E[2]=x-(-1);
  }
  else if (x<1)
  {
    xf_E[2]=1-x;
    xf_E[3]=x-0;
  }
  else if (x<2)
  {
    xf_E[3]=2-x;
    xf_E[4]=x-1;
  }
  else
  {
    xf_E[4]=1;
  }
}
```

```
void Fuzzification_Delta_Error(float x)
{
  int b;

  for (b=0;b<=4;b++)
  {
      xf_DE[b]=0;
  }

  if (x<-2)
    xf_DE[0]=1;
  else if (x<-1)
  {
    xf_DE[0]=-1-x;
    xf_DE[1]=x-(-2);
  }
  else if (x<0)
  {
    xf_DE[1]=0-x;
    xf_DE[2]=x-(-1);
  }
  else if (x<1)
  {
    xf_DE[2]=1-x;
    xf_DE[3]=x-0;
  }
  else if (x<2)
  {
    xf_DE[3]=2-x;
    xf_DE[4]=x-1;
  }
  else
  {
    xf_DE[4]=1;
  }
}
```

```
void Fuzzy_PI(void)
{
    read_tacho();
    error = Sp-measurement;
    // range dalam -1400 s/d 1400
    Ke=(float) 1/700;
    Mul_Ke = error * Ke;
    // range dalam -2 s/d 2
    error_delta = error-error_last;
    // range dalam -1400 s/d 1400
    Kde=(float) 1/700;
    Mul_Kde = error_delta * Kde;
    // range dalam -2 s/d 2

    Fuzzification_Error(Mul_Ke);
    Fuzzification_Delta_Error(Mul_Kde);

    Rule_Base();
    Deffuzification();

    U_pd = U_pd + Ost;
    U_ki = U_pd * Kupd;
    U_pi = U_ki + U_pi_last;

    if (U_pi>2)
    {
        U_pi = 2;
    }
    else if (U_pi<-2)
    {
        U_pi = -2;
    }
    U_pi_last=U_pi;
    // Di ubah ke range PWM dari -2 s/d 2 mjd -1000 s/d
    1000
    u_kontrol = U_pi * 700 / 1.4;
    // Di ubah ke range PWM dari  -1000 s/d 1000 mjd 0
    s/d 1000
    u_kontrol = u_kontrol + 1000;
    u_kontrol = u_kontrol / 2;
    // Di ubah ke range PWM mjd 0-1023
    u_kontrol = u_kontrol *1.023;
```

```
void Rule_Base(void)
{

int i,j;
int temp;
int c;

  for (c=0;c<=4;c++)
  {
      u[c]=0;
      Temp_Min[c]=0;
  }

for(i=0;i<=4;i++)
{
    for(j=0;j<=4;j++)
    {
        k=r[i,j];
        k=k-1;

        Temp_E[j]=xf_E[j];
        Temp_DE[i]=xf_DE[i];

        //Mamdani (Generalize Modus Ponen/MaxofMin
        Temp_Min[k]=fmin(Temp_E[j],Temp_DE[i]);
        //temp=fmin(Temp_E[j],Temp_DE[i]);
        u[k]=fmax(u[k],Temp_Min[k]);

    }
}


}
void Deffuzification()
{

pemb=u[0]*c[0]+u[1]*c[1]+u[2]*c[2]+u[3]*c[3]+u[4]*c[4];
peny=u[0]+u[1]+u[2]+u[3]+u[4];
U_pd=pemb/peny;
}
```

```
putar:
    sinyal_kontrol = (int)u_kontrol;
    error_last=error;
}

void init (void)
{
        PORTA=0x00;
        DDRA=0x00;

        PORTB=0x00;
        DDRB=0x00;

        PORTC=0xFF;
        DDRC=0x00;

        PORTD=0x00;
        DDRD=0xFF;

        Sp=Sp_Eep;
        Ost=Ost_Eep;
        Kupd=Kupd_Eep;

        // Timer/Counter 0 initialization
        // Clock source: System Clock
        // Clock value: Timer 0 Stopped
        // Mode: Normal top=FFh
        // OC0 output: Disconnected
        TCCR0=0x00;
        TCNT0=0x00;
        OCR0=0x00;
        // Timer/Counter 1 initialization
        // Clock source: System Clock
        // Clock value: 11059.200 kHz
        // Mode: Ph. correct PWM top=03FFh
        // OC1A output: Non-Inv.
        // OC1B output: Non-Inv.
        // Noise Canceler: Off
        // Input Capture on Falling Edge
        // Timer 1 Overflow Interrupt: Off
        // Input Capture Interrupt: Off
```

```
// Compare A Match Interrupt: Off
// Compare B Match Interrupt: Off
TCCR1A=0xA3;
TCCR1B=0x01;
TCNT1H=0x00;
TCNT1L=0x00;
ICR1H=0x00;
ICR1L=0x00;
OCR1AH=0x00;
OCR1AL=0x00;
OCR1BH=0x00;
OCR1BL=0x00;
// Timer/Counter 2 initialization
// Clock source: System Clock
// Clock value: Timer 2 Stopped
// Mode: Normal top=FFh
// OC2 output: Disconnected
ASSR=0x00;
TCCR2=0x00;
TCNT2=0x00;
OCR2=0x00;

// External Interrupt(s) initialization
// INT0: Off
// INT1: Off
// INT2: Off
MCUCR=0x00;
MCUCSR=0x00;

 // Timer(s)/Counter(s) Interrupt(s) initialization
TIMSK=0x00;

// Analog Comparator initialization
// Analog Comparator: Off
// Analog Comparator Input Capture by Timer/Counter
     1: Off
ACSR=0x80;
SFIOR=0x00;
```

```
        // ADC initialization
        // ADC Clock frequency: 345.600 kHz
        // ADC Voltage Reference: AREF pin
        // ADC Auto Trigger Source: None
        // Only the 8 most significant bits of
        // the AD conversion result are used
        ADMUX=FIRST_ADC_INPUT | (ADC_VREF_TYPE & 0xff);
        ADCSRA=0xCD;
        // LCD module initialization
        lcd_init(16);
        lcd_clear();
        lcd_gotoxy(0,0);
        lcd_putsf(">> Motor Test <<");
        lcd_gotoxy(0,1);
        lcd_putsf("~~ ********** ~~");
        delay_ms(1500);
        //Global enable interrupts
        #asm("sei")
}
void main(void)
{
init();
lcd_menus();
dat3=fmin(dat1,dat2);
dat4=fmax(dat1,dat2);
while (1)
      {
        Fuzzy_PI();
        Motor=sinyal_kontrol;
        lcd_clear();
        lcd_gotoxy(0,0);
        sprintf(buffer, "SP=%d
         Msr=%.1f",Sp,measurement);
        lcd_puts(buffer);
        lcd_gotoxy(0,1);
        sprintf(buffer2, "E=%.1f
         U=%d",error,sinyal_kontrol);
        lcd_puts(buffer2);
        delay_ms(100);
      };
}
```