

DAFTAR ISI

	Halaman
HALAMAN SAMBUNG.....	i
LEMBAR PENGESAHAN.....	ii
ABSTRAK	iii
KATA PENGANTAR	iv
DAFTAR ISI	vi
DAFTAR GAMBAR	ix
DAFTAR TABEL	xii
BAB I PENDAHULUAN	1
I.1 Latar Belakang	1
I.2 Rumusan Masalah	2
I.3 Maksud dan Tujuan Penelitian	2
I.4 Batasan Masalah.....	3
I.5 Metode Penelitian	3
I.6 Sistematika Penulisan.....	4
BAB II TEORI DASAR.....	6
II.1. <i>Programmable Logic Controller (PLC)</i>	6
II.1.1. <i>Hardware PLC-5 Allen Bradley</i>	11
II.1.1.1. Modul Analog <i>Input 1771-IFE Series C</i>	12
II.1.1.2. Modul Analog <i>Output 1771-OFE Series B</i>	14
II.1.2. <i>Software</i>	16
II.1.2.1. <i>RS Logix-5</i>	17

II.1.2.1.1. Instruksi <i>Bit</i>	18
II.1.2.1.2. Instruksi <i>Timer/Counter</i>	19
II.1.2.1.3. Instruksi Matematika	20
II.1.2.1.4. Instruksi <i>Compare</i>	22
II.1.2.1.5. Instruksi <i>Mov/ Logical</i>	24
II.1.2.1.6. Instruksi <i>Input/ Output</i>	24
II.1.2.1.7 Instruksi <i>Program Control</i>	26
II.1.2.2. <i>RS View32</i>	26
II.1.2.2.1 <i>Node window</i>	27
II.1.2.2.2 <i>Tag Database</i>	28
II.1.2.2.3 <i>Display Window</i>	29
II.2. Motor DC Servo Trainer	30
II.2.1. Pengendali Sudut Dengan Loop Tertutup	30
II.2.2. Kecepatan Motor dan Karakteristik Input	32
II.3. Rangkaian Pembagi Tegangan	33
BAB III PERANCANGAN SISTEM KENDALI	37
III.1 Perangkat Keras Kendalian	38
III.1.1 Rangkaian Pengendali Servo Posisi	39
III.1.2 Rangkaian Pengendali Kecepatan Motor.....	41
III. 2. Perancangan dan desain sistem antarmuka.....	43
III. 2. 1. Perancangan <i>Ladder diagram</i>	43
III.2.1.1 Program Pembacaan masukan dan keluaran analog	43
III.2.1.2 Pengontrolan Servo Posisi.....	45

III.2.1.3 Pengontrolan Kecepatan Motor	48
III. 2 . 2 Perancangan Antarmuka	55
III. 2. 2. 1. Pembuatan <i>Tag Database</i>	55
III. 2. 2. 2. Perancangan jendela <i>Main Menu</i>	57
III. 2. 2. 3. Perancangan jendela <i>Servo Position</i>	58
III. 2. 2. 4. Perancangan jendela <i>Motor Speed</i>	59
IV. PENGUJIAN DAN ANALISIS	63
IV.1 Pengujian tegangan masukan dan keluaran	63
IV.1.1 Pegaturan Servo Posisi	63
IV.1.2 Pengontrolan Kecepatan Motor.....	66
IV.2. Perbandingan Parameter Antarmuka dengan Alat Praktikum	68
IV.2.1 Pengaturan Sudut Posisi.....	69
IV.2.2 Pengontrolan Kecepatan Motor.....	70
V. PENUTUP	73
V.1 Kesimpulan	73
V.2 Saran.....	74
DAFTAR PUSTAKA	
LAMPIRAN A : Ladder Diagram Program	
LAMPIRAN B : <i>DC Servo Trainer</i> tipe ED-4400B	
LAMPIRAN C : Dokumentasi program	
LAMPIRAN D : Makalah seminar hasil	

DAFTAR GAMBAR

Gambar II.1. Sebuah <i>Programmable Logic Control</i>	7
Gambar II.2 Sistem PLC`	8
Gambar II.3. Sinyal: (a) diskrit, (b) digital, (c) analog	10
Gambar II.4. Pilihan plug pada modul analog masukan	13
Gambar II.5. Komunikasi antara prosesor dan modul masukan analog	14
Gambar II.6. Komunikasi antara prosesor dan modul	16
Gambar II.7. Tampilan awal <i>RSLogix5</i>	17
Gambar II.8. Simbol XIO.....	18
Gambar II.9. Simbol XIC	18
Gambar II.10. Simbol OTE	19
Gambar II.11 Simbol ONS.....	19
Gambar II.12 Tampilan instruksi TON.....	20
Gambar II.13 Tampilan instruksi ADD	20
Gambar II.14 Tampilan instruksi SUB	21
Gambar II.15 Tampilan Instruksi MUL	21
Gambar II.16. Tampilan Instruksi DIV	22
Gambar II.17. Tampilan instruksi EQU	22
Gambar II.18. Tampilan instruksi GEQ	23
Gambar II.19. Tampilan instruksi LEQ	23
Gambar II.20. Tampilan instruksi MOV	24
Gambar II.21. Tampilan instruksi BTR	25
Gambar II.22. Tampilan instruksi BTW	25

Gambar II.23. Tampilan instruksi MCR	26
Gambar II.24. Tampilan jendela <i>Node</i>	27
Gambar II.25 Tampilan jendela <i>Tag Database</i>	28
Gambar II.26. Tampilan jendela <i>Display</i>	30
Gambar II.27. Sebuah loop tertutup pengontrol posisi servo.....	31
Gambar II.28. Hubungan antara kecepatan motor dan tegangan masuk	32
Gambar II. 29 Rangkaian ekivalen pengontrolan motor.....	33
Gambar II. 30. Rangkaian pembagi tegangan	34
Gambar II. 31. Resistor sederhana pembagi tegangan.....	35
Gambar III.1 Diagram bagan kotak pengendali motor	37
Gambar III.2 Flowchart Rangkaian pengendali sudut posisi	39
Gambar III.3 Rangkaian Pengendali Servo Posisi.....	40
Gambar III.4 Flowchart Rangkaian pengendali kecepatan motor	41
Gambar III.5. Rangkaian pengendali kecepatan motor	42
Gambar III.6 Program pembacaan analog	43
Gambar III.7 Flowchart untuk servo posisi.....	45
Gambar III.8 Program untuk servo posisi	46
Gambar III.9 Flowchart untuk servo posisi.....	48
Gambar III.10 Program pengontrolan kecepatan motor bagian 1	50
Gambar III.11 Program pengontrolan kecepatan motor bagian 2	51
Gambar III.12 Program pengontrolan kecepatan motor bagian 3	52
Gambar III.13 Program pengontrolan kecepatan motor bagian 4	54
Gambar III.14 Database label yang dipakai	56

Gambar III.15 Jendela <i>Main Menu</i>	57
Gambar III.16 Jendela <i>Servo Position</i>	58
Gambar III.17 Jendela <i>Motor Speed</i>	60
Gambar IV.1. Grafik perbandingan tegangan masukan servo posisi	66
Gambar IV.2. Grafik perbandingan tegangan keluaran servo posisi	66
Gambar IV.3. Perbandingan Tegangan Masukan Kecepatan Motor	68
Gambar IV.4 Grafik perbandingan sudut pengontrolan sudut posisi	69
Gambar IV.5 Grafik perbandingan RPM pengontrolan kecepatan motor ...	70

DAFTAR TABEL

Tabel II.1 Pilihan Jarak Masukan	13
Tabel II.2 Pilihan Jarak Keluaran	15
Tabel III.1 Berkas label <i>Position</i>	56
Tabel III.2 Berkas label <i>Speed</i>	57
Tabel IV.1 Tegangan masukan servo posisi.....	64
Tabel IV.2 Tegangan Keluaran Servo Posisi	65
Tabel IV.3 Tegangan masukan Kecepatan Motor	67
Tabel IV.4 Derajat sudut pengontrolan posisi sudut.....	69
Tabel IV.5 Perbandingan RPM pada pengontrolan kecepatan motor	71

BAB I

PENDAHULUAN

I.1. Latar Belakang Masalah

Kemajuan teknologi telah membuat segala sesuatu menjadi lebih praktis, demikian juga dalam kegiatan sehari-hari, kebutuhan industri menginginkan hal yang demikian. Berbagai inovasi telah dibuat untuk mempermudah pekerjaan manusia.

Sistem kontrol saat ini kebanyakan masih menggunakan kontrol lokal, dimana seorang operator harus kelapangan untuk mengoperasikan mesin agar dapat bekerja. Begitu pula jika operator tersebut ingin memonitor status dari mesin tersebut, seorang operator pun harus ke lapangan untuk melakukan hal tersebut. Sekarang ini hal tersebut kurang efisien karena memerlukan waktu yang lama dan tidak bisa terus menerus. Dan bahkan juga hal tersebut tidak diperbolehkan jika operator tersebut meninjau ke lapangan dengan pertimbangan *safety* dari seorang operator tersebut dimana kondisi lingkungan yang panas dan bising membuat kenyamanan untuk bekerja menjadi kurang bahkan keselamatan diri menjadi sangat berbahaya.

Dengan melihat pertimbangan tersebut, dalam dunia industri dapat memanfaatkan teknologi pengendali yang ada untuk meningkatkan keselamatan kerja dan juga efisiensi satu pekerjaan. Banyak sistem manual tersebut dibuat menjadi otomatis untuk membuat pekerjaan tersebut lebih praktis. Seiring dengan perkembangan teknologi sistem kendali didunia industri, sistem pengontrolan dan monitoring mulai diambil alih oleh alat kendali untuk menggantikan pekerjaan

manual yang penuh resiko tersebut. Salah satunya adalah sistem pengendali dengan menggunakan *Programmable Logic Control (PLC)*.

PLC ini dengan segala fasilitas didalamnya mampu menggantikan peran manusia untuk mengoperasikan kendalian dari jarak jauh dengan sistem otomatis. Dalam [3] dijelaskan bahwa PLC dapat memantau masukan-masukan maupun keluaran-keluaran sesuai dengan instruksi didalam program dan melaksanakan aturan kontrol yang telah diprogram. Hal ini membuat pekerjaan lebih efisien dan efektif dari beberapa hal seperti ekonomi, *safety* dan tenaga kerja.

Berangkat dari hal tersebut maka kami membuat tugas akhir dengan judul **“Sistem Kendali Servo Posisi dan Kecepatan dengan *Programmable Logic Controller (PLC)*”**.

I.2 Rumusan Masalah

Permasalahan yang akan dibahas dalam tugas akhir ini yaitu, bagaimana merancang suatu sistem kendali berupa motor yang dapat di kontrol melalui suatu antarmuka dengan menggunakan PLC sehingga dapat di kontrol dan di monitoring setiap waktu.

I.3 Maksud dan Tujuan Penulisan

Berangkat dari latar belakang permasalahan yang telah diuraikan di atas, kami bermaksud untuk mengembangkan sistem kendali posisi dan kecepatan motor dengan menggunakan PLC, perancangan *hardware* dan *software*, yang dapat digunakan untuk mengendalikan putaran motor tersebut.

Adapun tujuan dari penulisan tugas akhir ini adalah :

1. Memahami prinsip kerja PLC dan aplikasinya untuk mengendalikan sebuah motor dalam perangkat sistem kendali terkhusus untuk analog masukan dan analog keluaran.
2. Memahami cara pengelolaan parameter analog berupa sinyal listrik yang masuk ke PLC dan memanfaatkannya untuk membaca parameter yang ada.
3. Mengolah parameter yang masuk ke PLC untuk membuat sebuah antarmuka yang berfungsi sebagai pusat pengendali dan monitoring motor.
4. Untuk menyelesaikan studi pada Jurusan Elektro Fakultas teknik Universitas Hasanuddin dan mendapatkan gelar Sarjana Teknik.

I.4. Batasan Masalah

Untuk kemudahan dan lebih terperinci pembahasan penulisan, permasalahan yang dibahas dalam tugas akhir ini dibatasi pada :

1. Untuk kendalian yang digunakan adalah sebuah Motor dari *DC Servo Trainer* tipe ED-4400B buatan *ED Co., Ltd.*
2. Sistem pengontrol yang digunakan adalah *Programmable Logic Control* jenis Allen Bradley tipe *PLC-5*.
3. *Software* yang digunakan adalah *RSLogix5* untuk pemograman dan *RSView32* untuk membuat suatu antarmuka.

I.5. Metode Penelitian

Dalam penyusunan tugas akhir ini ada beberapa metode yang akan kami gunakan yaitu:

1. Studi Literatur (*Library Research*). Yakni membaca dan mempelajari bahan kuliah, literatur-literatur, data sheet, dan tulisan-tulisan yang berkaitan dengan tugas akhir ini.
2. Merancang dan Membuat sistem secara *hardware*.
3. Merancang diagram ladder dengan menggunakan *RSLogix5* kemudian pembuatan antarmuka menggunakan *RSView 32*.
4. Menguji dan mengambil data dari perancangan.
5. Menganalisis hasil dan membuat kesimpulan.

I.6. Sistematika Penulisan

Sistematika penulisan tugas akhir ini terbagi dalam lima bab dengan harapan maksud dan tujuan dari penulisan ini dapat terangkum seluruhnya. Pembagian bab tersebut adalah sebagai berikut :

BAB I PENDAHULUAN

Bab ini berisi tentang penguraian secara singkat latar belakang, tujuan, perumusan masalah, batasan masalah, batasan masalah, metodologi penulisan, dan sistematika penulisan.

BAB II LANDASAN TEORI

Pada bab ini akan dijelaskan tentang teori penunjang yang digunakan dalam pembuatan proyek akhir ini. Teori tersebut antara lain mengenai PLC, sensor, dan motor.

BAB III PERANCANGAN SISTEM KENDALI KECEPATAN DAN POSISI

MOTOR

Pembahasan mengenai perancangan dan prinsip kerja sistem yang akan dibuat, meliputi hardware dan software.

BAB IV PENGUJIAN DAN ANALISIS

Pembahasan mengenai implementasi sistem dan analisa hasil yang diperoleh.

BAB V PENUTUP

Bab ini berisi tentang kesimpulan dari pembahasan permasalahan dan saran-saran untuk perbaikan dan penyempurnaan proyek akhir ini.

BAB II

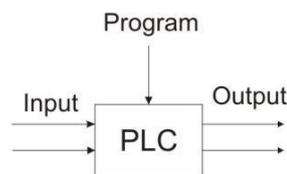
TEORI DASAR

Bab ini akan diawali dengan pembahasan tentang teori dasar sistem kendali berbasis *Programmable Logic controller* (PLC). Baik perangkat keras yang digunakan maupun perangkat lunak dari *PLC-5* dan teori dasar *Motor DC Servo Trainer* sebagai modul yang digunakan dalam proyek ini, serta teori dasar rangkaian pembagi tegangan sebagai perangkat yang modul dengan *PLC-5*.

II.1. Programmable Logic Controller (PLC)

Programmable Logic Control (singkatnya PLC) merupakan suatu bentuk khusus pengontrol berbasis mikroprosesor yang memanfaatkan memori yang dapat diprogram untuk menyimpan instruksi-instruksi dan untuk mengimplementasikan fungsi-fungsi semisal logika, *sequencing*, pemwaktuan (*timing*), pencacah (*counting*) dan aritmatika guna mengontrol mesin-mesin dan proses-proses (Gambar II.1) dan dirancang untuk dioperasikan oleh para insinyur yang hanya memiliki sedikit pengetahuan mengenai komputer dan pemrograman. Piranti ini dirancang sedemikian rupa agar tidak hanya programmer komputer saja yang dapat membuat atau mengubah program-programnya. Oleh karena itu, para perancang PLC telah menempatkan sebuah program awal didalam piranti ini (*pre-program*) yang memungkinkan program-program kontrol dimasukkan dengan menggunakan suatu bentuk bahasa pemrograman yang sederhana dan intuitif. Istilah logika (*logic*) dipergunakan karena pemrograman yang harus dilakukan sebagian besar berkaitan dengan pengimplementasikan operasi-operasi logika dan penyambungan (*switching*), misalnya jika A atau B terjadi maka sambungkan

(atau hidupkan) C, jika A dan B terjadi maka sambungkan D. Perangkat-perangkat masukan, yaitu , sensor-sensor semisal saklar, dan perangkat-perangkat keluaran didalam sistenm yang dikontrol, misalnya, motor, katub, dsb., disambungkan ke PLC. Sang operator kemudian memasukkan serangkai instruksi, yaitu, sebuah program, kedalam memori PLC. Perangkat pengontrol tersebut kemudian memantau masukan-masukan dan keluaran-keluaran sesuai dengan instruksi-instruksi didalam program dan melaksanakan aturan-aturan kontrol yang telah diprogram.



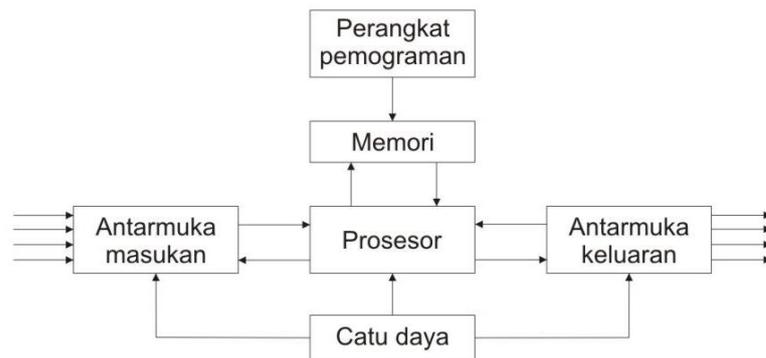
Gambar II.1. Sebuah *Programmable Logic Control*

PLC memiliki keunggulan yang signifikan, karena sebuah perangkat pengontrol yang sama dapat digunakan didalam beraneka ragam sistem kontrol. Untuk memodifikasi sebuah sistem kontrol dan aturan-aturan pengontrolan yang dijalankan, yang harus dilakukan oleh sorang operator hanyalah memasukkan seperangkat instruksi yang berbeda dari yang digunakan sebelumnya. Penggantian rangkaian kontrol tidak perlu dilakukan. Hasilnya adalah sebuah perangkat yang fleksibel dan hemat-biaya yang dapat dipergunakan didalam sistem-sistem kontrol yang sifat dan kompleksitasnya sangat beragam.

PLC serupa dengan komputer, bedanya : komputer dioptimalkan untuk tugas-tugas perhitungan dan penyajian data, sedangkan PLC dioptimalkan untuk

tugas-tugas pengontrolan dan pengoprasian didalam lingkungan industri. Dengan demikian PLC memiliki karakteristik:

1. Kokoh dan dirancang untuk tahan terhadap getaran, suhu, kelembaban dan kebisingan.
2. Antarmuka untuk masukan dan keluaran telah tersedia secara *built-in* didalamnya.
3. Mudah diprogram dan menggunakan sebuah bahasa pemograman yang mudah dipahami, yang sebagian besar berkaitan dengan operasi-operasi logika dan penyambungan.



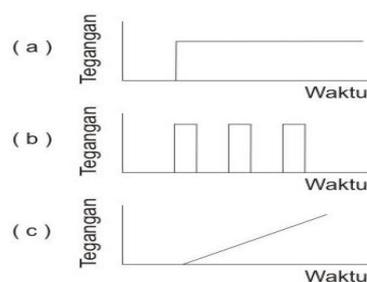
Gambar II.2 Sistem PLC

Perangkat PLC pertama dikembangkan pada tahun 1969. Dewasa ini PLC secara luas digunakan dan telah dikembangkan dari unit-unit kecil yang berdiri sendiri (*self-contained*) yang hanya mampu menangani sekitar 20 masukan/keluaran menjadi sistem-sistem modular yang dapat menangani masukan/keluaran dalam jumlah besar, menangani masukan/keluaran analog maupun digital, dan melaksanakan mode-mode kontrol proposional-integral-derivatif.

Umumnya, sebuah sistem PLC memiliki lima komponen dasar. Komponen-komponen ini adalah unit prosesor, memori, unit catu daya, bagian antarmuka masukan/keluaran, dan perangkat pemrograman. Gambar II.2 menampilkan konfigurasi dasarnya.

1. *Unit prosesor* atau *central processing unit* (unit pengolahan pusat)(CPU) adalah unit yang berisi mikroprosesor yang menginterpretasikan sinyal-sinyal masukan dan melaksanakan tindakan-tindakan pengontrolan, sesuai dengan program yang tersimpan dalam memori, lalu mengkomunikasikan keputusan-keputusan yang diambilnya sebagai sinyal-sinyal kontrol ke antarmuka keluaran.
2. *Unit catu daya* diperlukan untuk mengkonversikan tegangan AC sumber menjadi tegangan rendah DC (5 Volt) yang dibutuhkan oleh prosesor dan rangkaian-rangkaian didalam modul-modul antarmuka masukan dan keluaran.
3. *Perangkat pemrograman* dipergunakan untuk memasukkan program yang dibutuhkan didalam memori. Program tersebut dibuat dengan menggunakan perangkat ini dan kemudian dipindahkan kedalam unit memori PLC.
4. *Unit memori* adalah tempat dimana program yang digunakan untuk melaksanakan tindakan-tindakan pengontrolan oleh mikroprosesor disimpan.
5. *Bagian masukan dan keluaran* adalah antarmuka dimana prosesor menerima informasi dari dan mengkomunikasikan informasi kontrol ke

perangkat-perangkat eksternal. Sinyal-sinyal masukan, oleh karenanya, dapat berasal dari saklar-saklar pada kasus mesin bor otomatis, atau sensor-sensor lain, seperti misalnya sel-sel fotoelektris pada mekanisme perhitungan, sensor suhu atau sensor aliran cairan, dsb. Sinyal-sinyal keluaran mungkin diberikan pada kumparan-kumparan *starter* motor, katup-katup selenoida, dll. Perangkat-perangkat masukan dan keluaran dapat digolongkan menjadi perangkat-perangkat yang menghasilkan sinyal diskrit atau digital, dan yang menghasilkan sinyal-sinyal analog (Gambar II.3). Perangkat-perangkat yang menghasilkan sinyal-sinyal digital adalah perangkat-perangkat yang hanya mengindikasikan kondisi ‘mati’ (*off*) atau ‘hidup’(*on*). Sehingga, saklar adalah sebuah perangkat yang menghasilkan sebuah sinyal diskrit, yaitu, ada tegangan atau tidak ada tegangan. Perangkat-perangkat digital pada dasarnya dapat dipandang sebagai perangkat-perangkat diskrit yang menghasilkan serangkaian sinyal ‘mati;-‘hidup’. Perangkat-perangkat analog menghasilkan sinyal-sinyal yang amplitudonya sebanding dengan nilai variable yang dipantau. Sebagai contoh, sensor suhu akan menghasilkan tegangan yang nilainya sebanding dengan suhu.



Gambar II.3. Sinyal: (a) diskrit, (b) digital, (c) analog

PLC yang digunakan dalam tugas akhir ini adalah *PLC-5 Allen Bradley* yang penjelasannya dapat di bagi menjadi dua bagian yaitu perangkat keras (*hardware*) dan perangkat lunak (*software*).

II.1.1. Hardware PLC-5 Allen Bradley

Bagian –bagian perangkat keras dari *PLC-5 Allen Bradley* :

- *Chasis I/O*, adalah tempat atau *chasing* dimana modul *input* (masukan) dan modul *output* (keluaran) berada.
- Prosesor *PLC-5/80E*, komponen ini merupakan segmen pengolah dari keseluruhan bagian dan unsur-unsur yang terlibat dalam sistem pengontrolan.
- Modul *I/O*, yaitu piranti masukan dan keluaran (modul masukan dan keluaran) yang berfungsi untuk menghubungkan prosesor dengan masukan-masukan yang diperoleh dari peralatan kontrol seperti saklar otomatis, saklar manual, maupun masukan analog, dan lain-lain. Kemudian oleh modul keluaran hasil olahan prosesor diterima dan dihubungkan dengan peralatan keluaran agar pengontrolan yang diinginkan dapat dilakukan sesuai dengan isi pemrograman.
- *Power Supply*, yaitu sumber yang digunakan untuk mencatu daya keseluruhan rangkaian PLC mulai dari prosesor sampai ke modul masukan dan keluaran.
- *Programming terminal*, yaitu terminal program atau tempat penyimpanan sementara program yang akan dikirim ke prosesor.
- *Communication Card*.

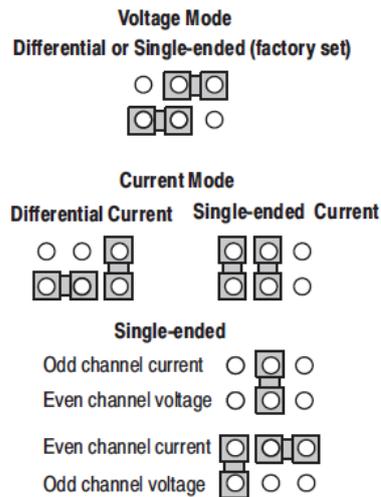
- Kabel *Programming terminal*.
- Kabel *Ethernet*.

Prosesor *PLC-5* adalah modul slot tunggal yang ditempatkan pada bagian paling kiri slot *1771 I/O chassis*. Prosesor *PLC-5* memiliki ruang yang besar untuk memori *I/O* dan kapabilitas yang tinggi dalam komunikasi.

II.1.1.1. Modul Analog Input 1771-IFE Series C

Modul analog *input* (masukan) adalah modul yang menghubungkan antara masukan sinyal analog dengan sinyal prosesor *PLC-5 Allen Bradley*. Modul analog *input* merupakan *single-slot* modul dan tidak memerlukan *power supply* tambahan. Modul menerima *power supply* pada *blackplane* dari *1771 I/O* sebesar 500mA. Arus ini menyuplai seluruh modul yang terdapat pada *I/O chassis*. Setelah sinyal analog masuk pada modul, data *input* dikonversikan ke bentuk format digital untuk dikirim ke prosesor.

Modul analog masukan terdiri dari 16 *channel single ended* atau 8 *channel differential* (pada simulasi ini digunakan 8 *channel differential*) dan mengubah sinyal masukan ke bentuk nilai integer. Setiap masukan dikonfigurasi sebagai arus atau tegangan masukan, adapun pilihan jarak masukannya dapat dilihat pada Tabel II.1. Modul ini (1771-IFE/C) mengkonfigurasi tegangan dan arus ke dalam jumper pada modul tegangan sinyal masukannya. Bisa dikonfigurasi menggunakan *single ended* atau *differential*.



Gambar II.4. Pilihan plug pada modul analog masukan

Tabel II.1 Pilihan Jarak Masukan

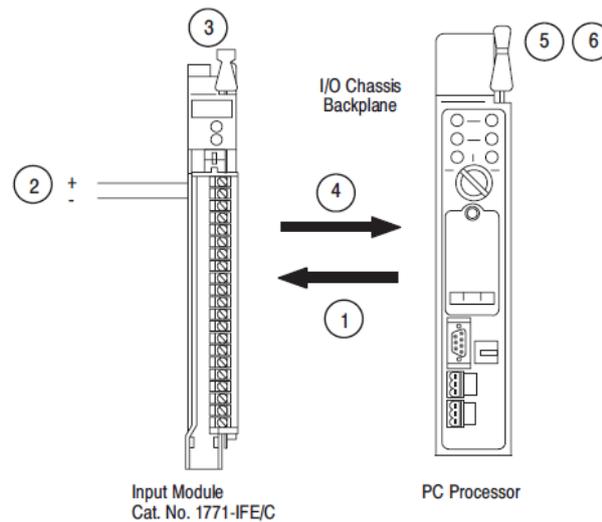
Tegangan (Vdc)	Arus (mA)
1 - 5	4 - 20
0 - 5	0 - 20
-5 - (+5)	-20 - (+20)
-10 - (+10)	

Namun pada proyek ini yang digunakan adalah jarak keluaran berupa tegangan antara 0-5 Volt.

Pemindahan data dari modul ke prosesor menggunakan instruksi BTW dan BTR yang terdapat pada *ladder diagram*. Cara kerja instruksi ini adalah :

1. Menggunakan peralatan yang menghasilkan sinyal analog untuk dihubungkan pada *channel* modul.
2. Modul mengubah sinyal analog ke bilangan integer dan menyimpan nilai-nilai tersebut hingga prosesor memintanya.
3. Ketika diinstruksikan oleh ladder, prosesor membentuk BTR untuk membaca nilai-nilai masukan yang terdapat pada modul.

4. Instruksi BTW digunakan untuk menulis nilai-nilai masukan dari modul ke memori prosesor.
5. Pada proses pemindahan ini dipastikan sudah tidak ada lagi *error*.



Gambar II.5. Komunikasi antara prosesor dan modul masukan analog

II.1.1.2. Modul Analog Output 1771-OFE Series B

Modul analog output (keluaran) adalah sebuah modul *intelligent block transfer* yang mengubah besaran biner atau nilai empat digit BCD (yang berasal dari prosesor) ke sinyal analog pada ke empat keluaran modul. Modul akan mencapai dengan transfer data dengan pemograman *block transfer*.

Pemograman *Block Transfer Write* (BTW) memindahkan 13 *words* data dari prosesor ke modul untuk konversi digital ke analog (D/A) pada sekali *program scan*. Informasi ini dikonversi ke sinyal analog dan akan dikirim ke saluran keluaran yang tepat. Sebuah *Block Transfer Read* (BTR) memindahkan lima *words* data dari modul ke data tabel prosesor, jika diperlukan, untuk tujuan pemeriksaan .

Modul ini mempunyai fitur *scaling* yang mengubah data yang dikirim ke modul dalam unit modul ke sinyal analog yang tepat. Ini bisa dihubungkan dengan empat alat keluaran analog, seperti *valve positioners*, kontrol kecepatan motor, konversi sinyal atau perekam, ke empat *channel* modul keluaran analog. Semua alat *input* dari keluaran analog harus dipastikan apakah yang digunakan adalah arus atau tegangan pada setiap *channel* modul keluaran.

Ada tiga versi dari modul keluaran analog :

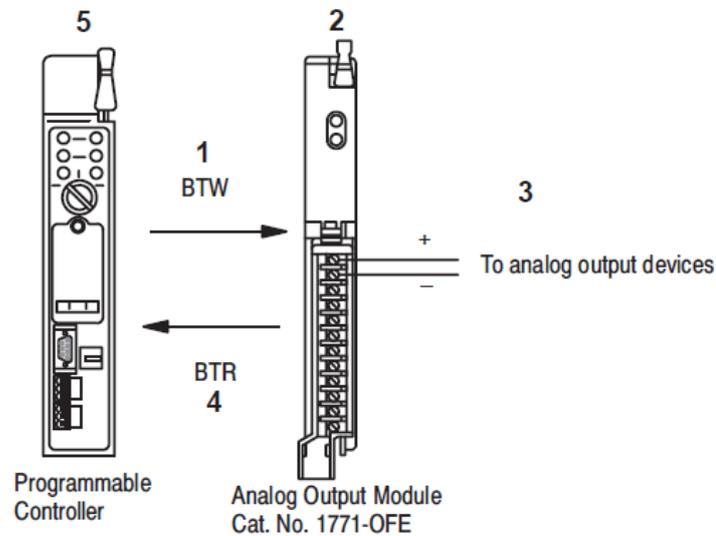
Tabel II.2 Pilihan Jarak Keluaran

Nomor Katalog	Modul keluaran	Jarak keluaran	Keterangan
1771-OFE1	Tegangan	1-5 V dc 0-10 V dc ± 10 V dc	Dipilih dengan konfigurasi jumper
1771-OFE2	Arus	4-20 mA	Pengaturan pabrik
1771-OFE3	Arus	0-50 mA	Pengaturan pabrik

Namun pada proyek ini yang digunakan adalah 1771-OFE2 dengan jarak keluaran berupa arus antara 4-20mA.

Komunikasi Modul Analog

Transfer data dari prosesor ke modul (*Block Transfer Write*) dan dari modul (*Block Transfer Read*) menggunakan instruksi BTW dan BTR pada perangkat lunak ladder diagram. Instruksi ini membuat prosesor mengirimkan nilai keluaran ke modul, membangun pilihan operasi modul (lihat ilustrsi dibawah) dan menerima status informasi dari modul.



Gambar II.6. Komunikasi antara prosesor dan modul

1. Prosesor mentransfer konfigurasi dan data output ke modul melalui instruksi *Block Transfer Write*.
2. Modul mengubah data menjadi proporsi keluaran tegangan atau arus.
3. Modul keluaran ini menuju alat eksternal analog.
4. Ketika selesai instruksi dari program ladder, prosesor akan membaca *block transfer* dari nilai keluaran dan sttus modul.
5. Prosesor dan modul akan menentukan transfer yang dibuat tanpa kesalahan.
6. Program ladder diagram bisa digunakan dan atau pindah (jika valid) sebelum ditulis dengan memindahkan data baru pada subsequent transfer.

II.1.2. Software

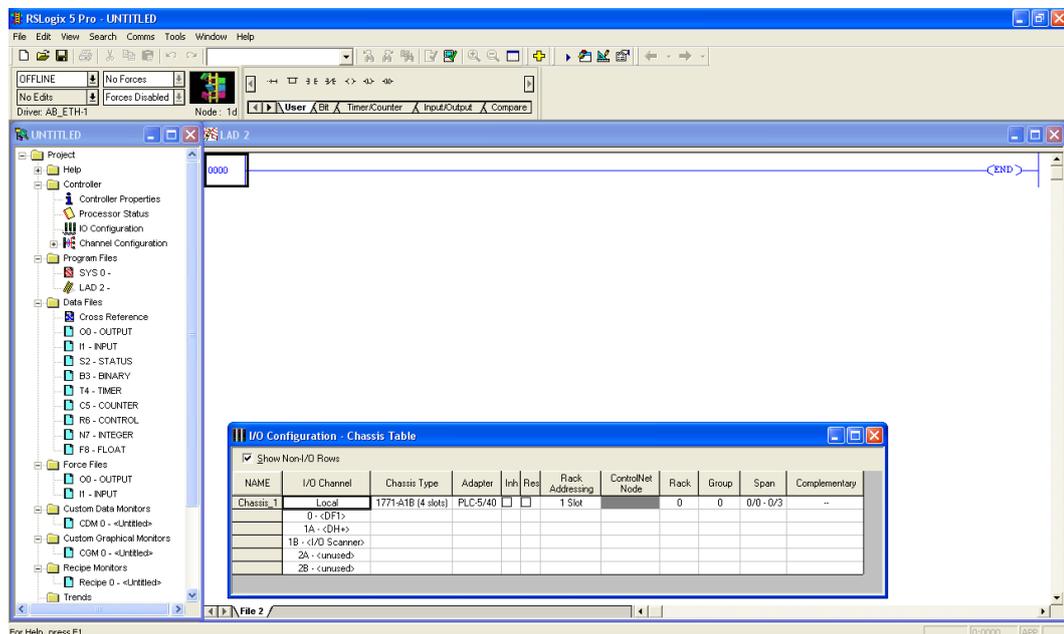
Software atau perangkat lunak menjadi bagian penting dalam merencanakan sebuah sistem kendali berbasis PLC, dimana setiap instruksi kerja yang hendak dilaksanakan terlebih dahulu harus melalui perancangan secara

perangkat lunak, sebelum dimasukkan ke dalam peralatan itu sendiri. Dalam hal ini terdapat berbagai macam tipe perangkat lunak yang digunakan dalam pemrogramannya. Untuk sistem berbasis *PLC -5 Allen Bradley* ini menggunakan *RSLogix-5*.

II.1.2.1. RS Logix-5

RSLogix-5 adalah perangkat lunak berbasis pada grafis sehingga memudahkan setiap pengguna dalam merencanakan program PLC yang diinginkan dengan menggunakan teknologi *Human Machine Interface*, sehingga pengguna dapat mengakses instruksi hanya dengan metode *klik* dan *drag*.

Perangkat lunak ini juga terdiri dari beberapa bagian seperti *menu bar*, *worksheet*, *sub menu*, serta *toolbar*.



Gambar II.7. Tampilan awal RSLogic5

II.1.2.1.1. Instruksi *Bit*

Instruksi mengoperasikan satuan *bit* per data. Selama pengoperasian, prosesor men-*set* atau me-*reset bit* berdasarkan logika kontinuitas dari *rung ladder*. Adapun instruksi *bit* ini terdiri dari instruksi-instruksi antara lain :

- ***Examine if Open (XIO)***

Operasi dari instruksi XIO memiliki alamat *file* data masukan. Ketika ada masukan dari divais eksternal pada suatu rangkaian, keadaan “*off*” menunjukkan bahwa terminal masukan terhubung pada divais.



Gambar II.8. Simbol XIO

- ***Examine if Closed (XIC)***

Operasi dari instruksi XIC memiliki alamat *file* data masukan. Ketika ada masukan dari divais eksternal pada suatu rangkaian, keadaan “*on*” menunjukkan bahwa terminal masukan terhubung pada divais.



Gambar II.9. Simbol XIC

- ***Output Energize (OTE)***

Operasi dari instruksi OTE memiliki alamat *file* data keluaran, keadaan dari terminal keluaran ditampilkan pada *file* data keluaran pada alamat *bit* tertentu.

Ketika prosesor menemukan jalur logika yang *true* pada *rung* yang memuat instruksi OTE, maka *bit set* (1). Perubahan ini membuat terminal

keluaran “on” dan mengakibatkan divais keluaran terhubung pada terminal. Kemudian ketika jalur *true* tidak lagi ditemukan, maka prosesor akan secara otomatis me-*reset* (0) atau membuat terminal “off” dan tidak lagi membangkitkan terminal keluaran.



Gambar II.10. Simbol OTE

- **One-Shot (ONS)**

Instruksi ONS adalah suatu instruksi masukan yang membuat ladder bernilai “on” untuk satu program *scan* pada transisi dari *false-to-true* dengan kondisi sebelum instruksi ONS di *rung* yang sama.

Biasanya instruksi ONS untuk memulai peristiwa yang dipicu oleh tombol tekan, seperti penghentian secara cepat nilai apa yang ditampilkan oleh LED.



Gambar II.11 Simbol ONS

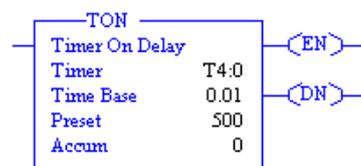
II.1.2.1.2. Instruksi *Timer/Counter*

- **Timer on Delay (TON)**

Gunakan instruksi TON untuk membuat keluaran “ON” atau “OFF” setelah *timer* telah selesai dengan waktu intervalnya. Instruksi keluaran ini dimulai dengan waktu (antara interval satu detik atau seperseratus detik)

ketika *rung* berubah menjadi “*true*” dan berlanjut hingga salah satu dibawah ini terjadi :

- Akumulasi nilai sama dengan waktu intervalnya
- *Rung* berubah menjadi “*false*”
- Instruksi reset menyetel ulang *timer*
- Prosesor telah menyetel ulang nilai akumulasi ketika *rung* dalam kondisi *false*, walaupun *timer* telah selesai ataupun tidak

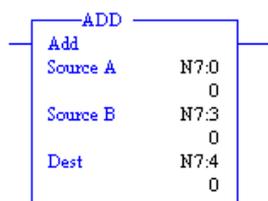


Gambar II.12 Tampilan instruksi TON

II.1.2.1.3. Instruksi Matematika

- **Add (ADD)**

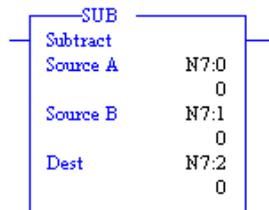
Simbol instruksi ini menjumlahkan antara suatu nilai (*source A*) dengan nilai lainnya (*source B*) dan menempatkan hasilnya pada alamat yang dituliskan pada *Dest (Destination)*. Baik nilai *source A* maupun *source B* akan menunjuk alamat yang mempunyai data pengganti nilai *source A* dan nilai *source B*.



Gambar II.13 Tampilan instruksi ADD

- **Subtract (SUB)**

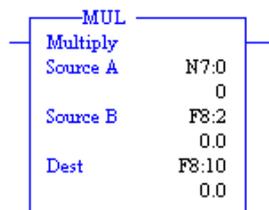
Simbol instruksi ini mengurangkan antara suatu nilai (*source A*) dengan nilai lainnya (*source B*) dan menempatkan hasilnya pada alamat yang dituliskan pada *Dest (Destination)*. Baik nilai *source A* maupun *source B* akan menunjuk alamat yang mempunyai data pengganti nilai *source A* dan nilai *source B*.



Gambar II.14 Tampilan instruksi SUB

- **Multiply (MUL)**

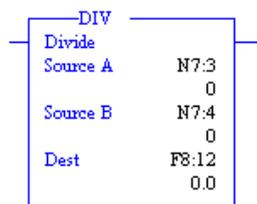
Simbol instruksi ini mengalikan antara suatu nilai (*source A*) dengan nilai lainnya (*source B*) dan menempatkan hasilnya pada alamat yang dituliskan pada *Dest (Destination)*. Baik nilai *source A* maupun *source B* akan menunjuk alamat yang mempunyai data pengganti nilai *source A* dan nilai *source B*.



Gambar II.15 Tampilan Instruksi MUL

- **Divide (DIV)**

Instruksi DIV membagi (*Source A*) dengan (*Source B*) dan menyimpan hasilnya pada *Dest*. (*Source A*) dan (*Source B*) bisa berisikan alamat ataupun konstanta. Instruksi ini akan mengeksekusi selama *rung* bernilai “*true*”, untuk mengeksekusi sekali, kondisikan *rung* dengan instruksi ONS.

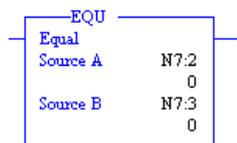


Gambar II.16. Tampilan Instruksi DIV

II.1.2.1.4. Instruksi *Compare*

Simbol ini mengoperasikan satuan *word* per data. Selama pengoperasian, prosesor men-*scan bit* berdasarkan logika *value* dari instruksi dan membaca memori sebagai referensi tindakan. Adapun *word* ini terdiri dari instruksi-instruksi antara lain :

- **Equal (EQU)**



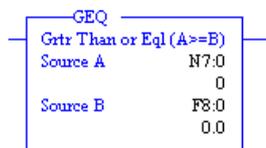
Gambar II.17. Tampilan instruksi EQU

Instruksi masukan ini membandingkan dua alamat dengan nilai yang spesifik. Jika nilainya sama, maka *rung* akan melanjutkannya. *Rung* akan

bernilai “true” dan keluaran akan *energized* (tergantung tidak ada instruksi lain pada *rung*)

- **Greater Than or Equal (GEQ)**

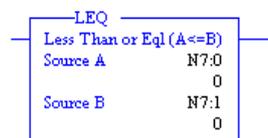
Masukan instruksi ini akan membandingkan dua nilai yang telah ditentukan. Jika nilai yang tersimpan dalam (*Source A*) lebih besar dari atau sama dengan nilai yang disimpan dalam (*Source B*), kondisi *rung* akan berlanjut. *Ladder* akan bernilai "ON" dan keluaran akan aktif (selama tidak ada instruksi lain yang mempengaruhi status *ladder*).



Gambar II.18. Tampilan instruksi GEQ

- **Less Than or Equal (LEQ)**

Masukan instruksi ini akan membandingkan dua nilai yang telah ditentukan. Jika nilai yang tersimpan dalam (*Source A*) kurang dari atau sama dengan nilai yang disimpan dalam (*Source B*), kondisi *rung* akan berlanjut. *Ladder* akan bernilai "ON" dan keluaran akan aktif (selama tidak ada instruksi lain yang mempengaruhi status *ladder*)



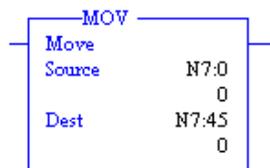
Gambar II.19. Tampilan instruksi LEQ

II.1.2.1.5. Instruksi *Mov/ Logical*

- **Move (MOV)**

Instruksi MOV merupakan instruksi *output* yang akan menyalin nilai dari alamat sumber (*Source*) ke alamat tujuan (*Destination*). Selama tidak ada instruksi lain yang mempengaruhi status *ladder*, instruksi data bergerak setiap *scan*.

Instruksi ini membuat salinan asli dan menempatkan hasil salinan di lokasi baru (*Destination*). Nilai yang asli tetap utuh dan tidak berubah di lokasi sumber.



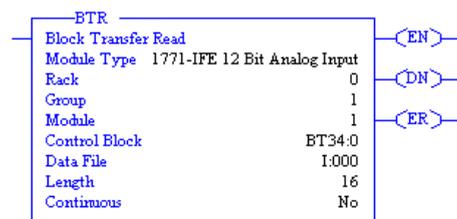
Gambar II.20. Tampilan instruksi MOV

II.1.2.1.6. Instruksi *Input/ Output*

- **Block Transfer Read (BTR)**

Simbol instruksi ini bertujuan untuk mentransfer maksimal 64 *word* pada waktu yang bersamaan dari sebuah *block transfer module* pada satu *chasis I/O*, dimana pemilihan jenis modul yang digunakan ditempatkan pada *module type*, pemilihan nomor *rack* yang digunakan ditempatkan pada *rack no*, pemilihan nomor grup *I/O* yang men-spesifikasikan posisi dari modul yang digunakan di dalam *chasis* ditempatkan pada *group*, dan pemilihan nomor slot dalam *group* yang digunakan ditempatkan pada *module*.

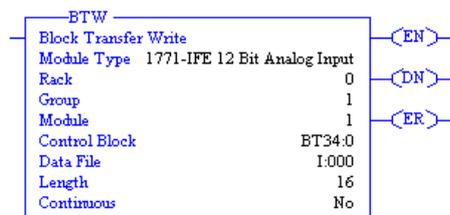
Adapun *control block* merupakan tempat untuk menuliskan tipe data *file control* yang digunakan, disarankan untuk menggunakan *file data block transfer* (BT), sedangkan *data file* merupakan tempat memasukkan alamat *input, output, status, integer, float, biner* atau *file data ASCII* darimana prosesor menulis atau ke mana prosesor membaca dan mentransfer data. Parameter lainnya yakni *length* merupakan tempat untuk memasukkan jumlah *word* yang akan ditransfer, yang mana panjangnya akan terkonfigurasi secara otomatis ketika *intelligent I/O* dikonfigurasi.



Gambar II.21. Tampilan instruksi BTR

- **Block Transfer Write (BTW)**

Instruksi ini memiliki kesamaan yang sangat banyak dengan instruksi BTR. Perbedaannya adalah instruksi BTW digunakan untuk mentransfer data *output* dari prosesor ke modul *output*, sedangkan BTR digunakan untuk mentransfer data *input* dari modul *input* ke prosesor



Gambar II.22. Tampilan instruksi BTW

II.1.2.1.7 Instruksi Program Control

- **Master Control Reset (MCR)**

Instruksi keluaran ini (kadang dikenal sebagai “Zone Control”) digunakan untuk mengatur area atau “zones” program *ladder* dimana semua keluaran yang ada didalamnya akan tidak aktif pada saat waktu yang sama dengan jarak waktunya. Ini dipakai berpasangan, satu MCR digunakan pada awal area *ladder* yang akan berpengaruh dan satu MCR digunakan pada akhir area tersebut.

Instruksi masukan untuk memprogram *rung* pada MCR awal ke logika kontrol *rung* selanjutnya. Ketika *rung* berubah menjadi “false”, semua keluaran yang termasuk dalam zona yang dikontrol akan tidak aktif. Ketika *rung* berubah menjadi “true”, semua *rung* akan diperiksa termasuk ke kondisi normal *rung* (diluar dari zona kntrol instruksi)



Gambar II.23. Tampilan instruksi MCR

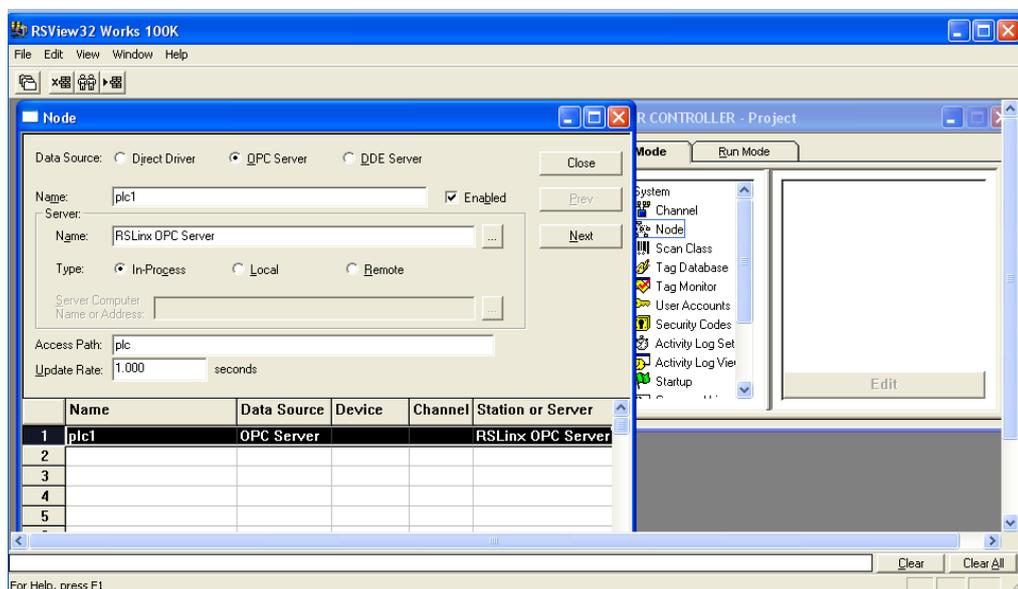
II.1.2.2. RS View32

RSView32 adalah sebuah perangkat lunak yang berfungsi untuk membuat sebuah antarmuka agar seluruh sistem pada PLC dapat dikendalikan dan dimonitor oleh antarmuka tersebut. Perangkat lunak ini dapat berkomunikasi dengan perangkat lunak pemogramman dari *Allen Bradley* (dalam hal ini *RSLogix 5*) sehingga setiap parameter dan kondisi yang berubah pada PLC dapat diketahui oleh antarmuka yang dibuat.

Penggunaannya pun sangat sederhana dan mudah dimengerti untuk membuat sebuah antarmuka. Karena didalamnya terdapat beberapa fitur untuk membuat sebuah *Node* yang bertujuan untuk merujuk ke sebuah PLC yang ingin dikendalikan, *Tag* yang berisikan sebuah alamat agar lebih mudah dikenali dan *Display* untuk pembuatan antarmuka sesuai yang kita inginkan.

II.1.2.2.1 Node window

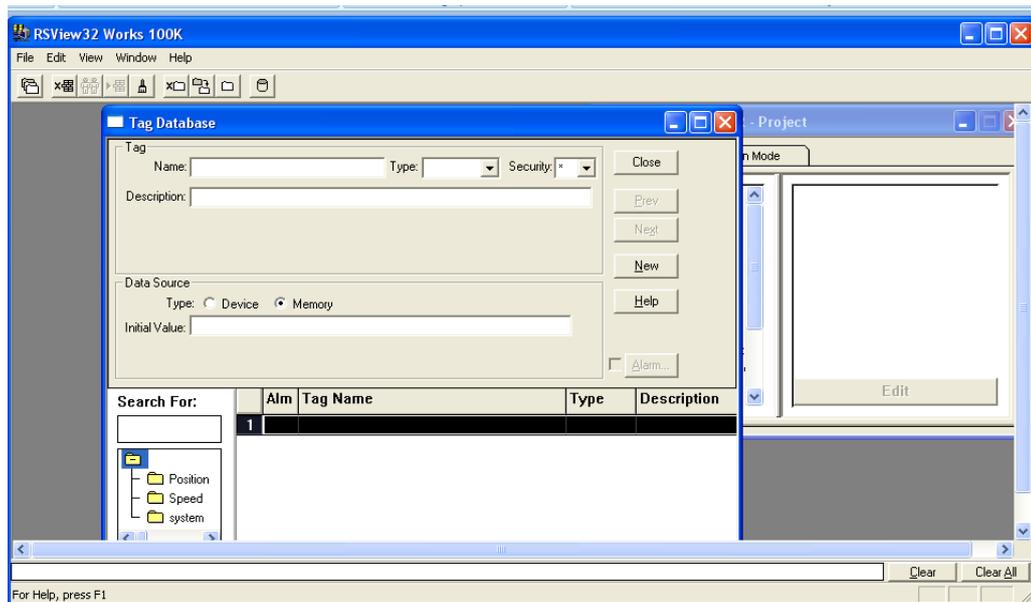
Tujuan untuk mengkonfigurasi awal node ini agar PLC yang akan dipakai selama proses pengontrolan akan dikenali. Ini sangat penting karena PLC selama bekerja akan selalu berkomunikasi dengan perangkat lunak ini melalui antarmuka yang telah dibuat. Maka perangkat lunak ini akan merujuk ke PLC tersebut dan akan terjadi proses interaksi apabila ada perubahan pada antarmuka maupun pada PLC.



Gambar II.24. Tampilan jendela *Node*

II.1.2.2.2 Tag Database

Tag atau pelabelan instruksi merupakan inti dari perangkat lunak antarmuka ini, karena kesalahan penggunaan *tagname* dapat mengakibatkan operasi program menjadi tidak normal dan bahkan mengalami kesalahan (*error*). Pada saat program dijalankan (*Runtime*), seluruh database pada objek grafis berisikan nilai-nilai saat antarmuka belum dijalankan, untuk itulah *tagname* digunakan agar informasi dari seluruh variabel yang telah dirancang dapat dikenali antarmuka sesuai dengan posisi kerjanya. Misalnya untuk *tagname* tipe masukan analog, antarmuka memerlukan informasi yang konkrit berapa integer pada PLC dan akan menampilkannya pada antarmuka yang telah dibuat.



Gambar II.25 Tampilan jendela *Tag Database*

Beberapa tipe label yang dapat dibuat adalah sebagai berikut ;

- *Digital tags*

Pelabelan digital akan menyimpan nilai numerik antara 0 atau 1. Label ini digunakan untuk mewakili alat yang mengondisikan dua keadaan: “ON”

atau “OFF” atau keadaan “*false*” atau “*true*”, seperti *switches*, kontak atau relay.

- *Analog tags*

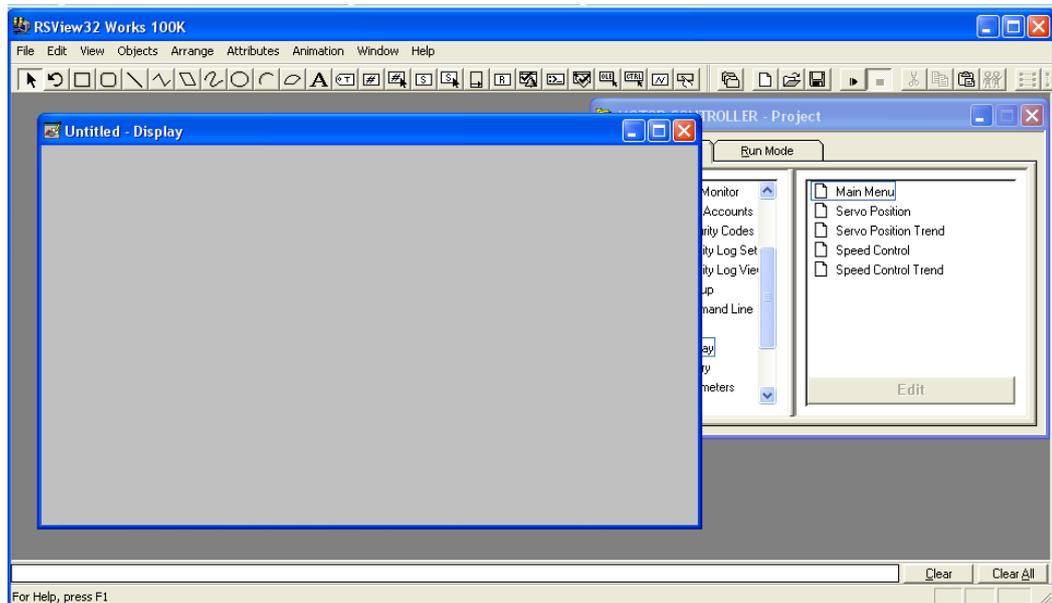
Pelabelan analog akan menyimpan nilai numerik dari jarak nilai yang didefinisikan oleh label tersebut. Biasanya digunakan untuk mewakili alat yang mempunyai jarak nilai seperti temperatur, tekanan, *flow*, atau posisi kendali rotasi.

- *String tags*

Pelabelan string akan menyimpan ASCII string yang berupa karakter string atau keseluruhan kata. Maksimum string yang diperbolehkan yaitu 255 karakter.

II.1.2.2.3 Display Window

Pada jendela *Display* akan diberikan keleluasan kepada programmer agar membuat antarmuka sesuai yang diperlukan. Disini juga terdapat beberapa *tools* untuk pembuatan antarmuka tersebut dari pembuatan teks sampai ke pembuatan animasi. Objek tersebut akan berisikan label (*Tags*) yang telah dibuat di *Tag Database* sehingga setiap perubahan parameter akan terjadi pula di antarmuka tersebut. Setelah pembuatan antarmuka selesai, pada jendela ini juga dapat kita jalankan (*Run*) untuk melihat apakah animasi yang dibuat sesuai dengan perubahan parameter yang terjadi pada PLC atau tidak.



Gambar II.26. Tampilan jendela *Display*

II.2. Motor DC Servo Trainer

DC Servo Trainer ED-4400B dari ED Co., LTD. adalah sistem loop tertutup servo DC yang didesain untuk modul praktikum. Inti konsep pembuatan sistem trainer ED-4400B adalah untuk memberikan pengetahuan praktik kerja pada sistem loop tertutup servo DC kepada pengguna dengan mengintegrasikan dasar teori dan langkah demi langkah percobaan dengan satu subjek.

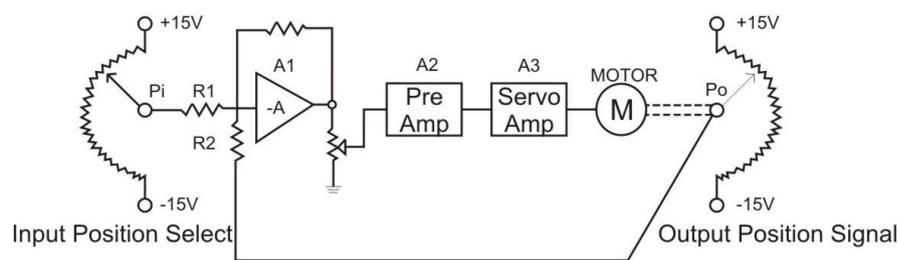
Modul praktikum pada ED-4400B sangat sederhana dan mudah untuk diuji cobakan pada sistem. Yang hanya dibutuhkan adalah untuk menghubungkan antara modul seperti yang diinstruksikan pada gambar rangkaian pada setiap seksi dengan menggunakan *jumper* yang telah disediakan.

II.2.1. Pengendali Sudut Dengan Loop Tertutup

Dalam sistem pengendali servo posisi, informasi posisi dari potensiometer yang dihubungkan pada motor menjadi umpan balik pada penguat kontrol (*control amplifier*). Kemudian, pengaturan posisi masukan dari masukan potensiometer

dikombinasikan dengan sinyal balik pada *input amplifier* yang menjalankan motor pada bagian yang berbeda antara dua sinyal. Ketika kedua posisi teridentifikasi, keluaran pada amplifier akan menjadi nol.

Sebuah diagram sederhana dari sistem pengontrolan posisi putaran dapat dilihat pada gambar II.26.



Gambar II.27. Sebuah loop tertutup pengontrol posisi servo

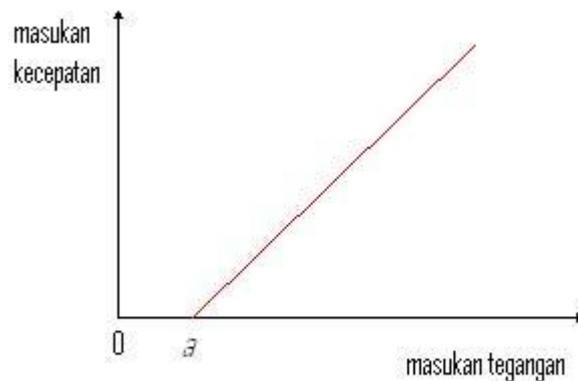
Terdapat tiga penguat pada gambar. A1 adalah sinyal dari generator, A2 adalah sinyal masukan penguat, dan A3 adalah *driver* untuk motor M. P1 berlawanan arah dengan P0. Perbedaan antara dua tegangan potensiometer menjadi masukan ke A1. Sinyal terus berlanjut melewati A2 dan A3, kemudian menjalankan motor dan tegangan pada P1 dan P0 saling mengurangi. Keadaan ini terus berlanjut sampai keluaran pada A1 sama dengan nol. Dalam hal ini, tegangan pada P1 dan P0 adalah sama tetapi dengan polaritas yang berbeda. Misalnya $P1 = +3V$ dan $P0 = -3V$, inilah membuat keluaran pada A1 menjadi nol.

Posisi akhir dari P1 dan P0 sangat relatif, bergantung pada ada tidaknya penambahan tegangan pada penguat. Untuk penambahan yang besar, posisi P0 hampir sama dengan P1, tetapi ketika penambahan tidak cukup maka posisi antara

keduanya tidak akan sama. Hal ini disebut sebagai kerugian pada pengontrolan posisi.

II.2.2. Kecepatan Motor dan Karakteristik Input

Secara umum motor merupakan sebuah mesin yang mengkonversi energi listrik menjadi energi gerak. Elemen kunci dari motor DC adalah medan (*field*) dan armatur. Aliran arus listrik yang melewati medan akan menimbulkan torka (tenaga putaran) pada armatur. Pada motor ED-4400B ini, efek medan dibangkitkan oleh magnet permanen. Magnet permanen ini menciptakan fluks magnet yang nilainya konstan. Sehingga kecepatan motor hanya bergantung pada tegangan masukan yang sampai pada armatur. Hubungan antara kecepatan motor dengan tegangan masukan dapat dilihat pada Gambar II.27.

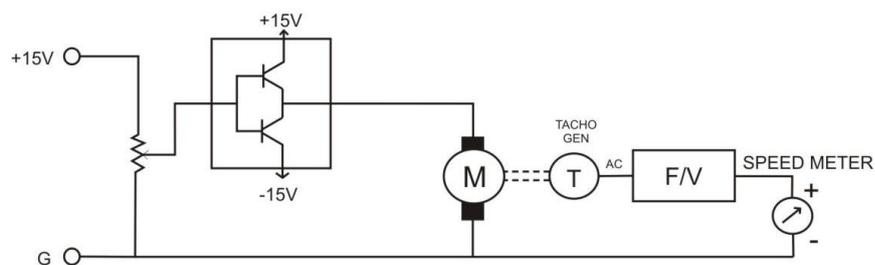


Gambar II.28. Hubungan antara kecepatan motor dan tegangan masuk

Pada Gambar II.23, *point a* terjadi karena motor membutuhkan tegangan minimum untuk mengatasi gesekan mesin dengan sikat-sikat, sampai akhirnya motor dapat berputar. Ketika tegangan masukan telah melebihi tegangan minimum (yang dibutuhkan), maka kecepatan motor akan naik secara linier sesuai

dengan bertambahnya tegangan masukan. Meskipun demikian, karakteristik linier akan berhenti pada saat terjadi saturasi. Hal ini disebabkan karena adanya batasan kapasitas bahan pada gulungan armatur, sehingga pada saat tegangan dinaikkan terus menerus maka arus yang dihasilkan pada armatur tidak akan bertambah (*saturation*).

Motor ED-4400B *system* dijalankan dengan *motor driver amplifier U-154* dan *Attenuator U-151* sebagai pengontrol tegangan. Kecepatan motor dapat di deteksi dan dilihat langsung pada *output tacho U-159*.



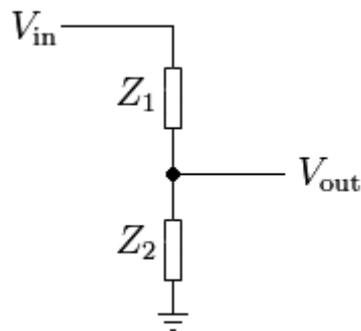
Gambar II. 29 Rangkaian ekivalen pengontrolan motor

II.3. Rangkaian Pembagi Tegangan

Pada elektronika, rangkaian pembagi tegangan (juga dikenal sebagai pembagi potensial) adalah rangkaian linear sederhana yang menghasilkan tegangan *output* (V_{out}) yang berasal dari tegangan input (V_{in}). Pembagi tegangan merujuk pada pembagian dari tegangan tergantung dari komponen yang pembaginya.

Rumus pembagi tegangan sama dengan rumus pembagi arus, namun rasio pembagi tegangan berasal dari impedansi tegangan yang ingin di cari, tidak seperti pembagi arus yang berasal dari komponen seberang yang akan dicari.

Contoh sederhana untuk rangkaian pembagi tegangan adalah terbagi atas dua resistor atau potensiometer yang dipasang seri. Pada umumnya ini digunakan untuk membuat tegangan referensi, atau untuk mendapatkan proporsi sinyal tegangan yang rendah ke tegangan yang akan terukur dan bisa juga digunakan sebagai sinyal attenuator dari frekuensi rendah.



Gambar II.30. Rangkaian pembagi tegangan

Rangkaian pembagi tegangan dapat dibuat dengan merangkai seri dua komponen impedansi listrik seperti Gambar II.29. Tegangan yang masuk dapat diterapkan bersebrangan impedansi seri dari Z_1 dan Z_2 dan tegangan yang keluar berasal dari Z_2 . Z_1 and Z_2 dapat dirangka dengan setiap kombinasi komponen seperti resistor, induktor dan kapasitor.

Dengan menerapkan hukum Ohm, hubungan antara tegangan masukan (V_{in}) dan tegangan keluaran (V_{out}) dapat dilihat sebagai berikut :

$$V_{out} = \frac{Z_2}{Z_1 + Z_2} \cdot V_{in} \dots\dots\dots (II.1)$$

Bukti rumus :

$$V_{in} = I \cdot (Z_1 + Z_2)$$

$$V_{out} = I \cdot Z_2$$

$$I = \frac{V_{in}}{Z_1 + Z_2}$$

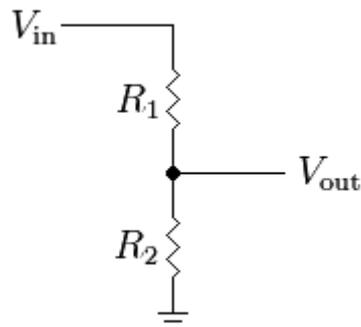
$$V_{out} = V_{in} \cdot \frac{Z_2}{Z_1 + Z_2}$$

Fungsi transfer (atau juga dikenal sebagai pembagi rasio tegangan) dari rangkaian ini dapat disederhanakan menjadi :

$$H = \frac{V_{out}}{V_{in}} = \frac{Z_2}{Z_1 + Z_2} \dots\dots\dots(II.2)$$

Pada umumnya fungsi transfer ini adalah kompleksifitas, fungsi rasio dari frekuensi.

Resistive divider



Gambar II.31. Resistor sederhana pembagi tegangan

Pembagi resistif adalah kasus khusus dimana antara impedansi, Z_1 and Z_2 , adalah murni resistor (Gambar II.30).

Dengan mengubah $Z_1 = R_1$ dan $Z_2 = R_2$ dengan rumus sebelumnya hingga menjadi :

$$V_{out} = \frac{R_2}{R_1 + R_2} \cdot V_{in} \dots\dots\dots(II.3)$$

Pada kasus yang lebih umum, R_1 dan R_2 mungkin bisa dari beberapa kombinasi dari resistor dengan rangkaian seri atau paralel.

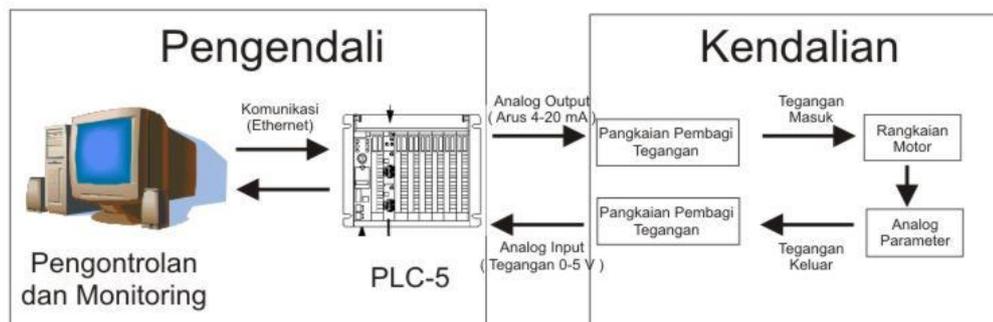
BAB III

PERANCANGAN SISTEM KENDALI KECEPATAN DAN POSISI MOTOR

Dalam bab ini akan dibahas cara kerja dan perencanaan komponen pengendali posisi dan kecepatan menggunakan *Programmable Logic Control* PLC-5 Allen Bradley. Perencanaan sistem ini dibagi menjadi dua bagian yaitu :

1. Perangkat Keras Kendalian
2. Perangkat Lunak Sistem Kendali

Diagram blok dari sistem ini ditunjukkan pada Gambar III.1. Prinsip kerja secara umum sistem yang akan direncanakan adalah sebagai berikut :



Gambar III.1 Diagram bagan kotak pengendali motor

Dalam tugas akhir ini dibuat sistem untuk mengendalikan motor dengan menggunakan PLC. Sebagai pengendali diperlukan sebuah komputer dan PLC. Didalam komputer terdapat antarmuka yang berfungsi sebagai pengendali dan monitoring. Komputer selalu berkomunikasi dengan PLC agar setiap perubahan nilai parameter pada PLC akan dapat dilihat diantarmuka.

Pada modul analog keluaran PLC, terdapat keluaran berupa arus sebesar 4-20mA. Ini sesuai dengan nilai minimum dan maksimum Integer yang diatur sebelumnya. Keluaran arus ini diubah menjadi tegangan dengan menggunakan Rangkaian Pembagi Tegangan. Ini dikarenakan untuk mengendalikan sudut posisi dan kecepatan motor diperlukan sumber tegangan. Tegangan tersebut memutar motor sehingga dapat dikendalikan. Besar tegangan yang diperlukan harus disesuaikan dengan keluaran dari modul analog keluaran PLC. Untuk menyesuaikannya maka nilai Integer yang dikelola oleh PLC menjadi parameter untuk mengendalikan motor tersebut. Setiap perubahan nilai Integer pada PLC membuat tegangan yang masuk ke motor berubah sesuai dengan pengaturan.

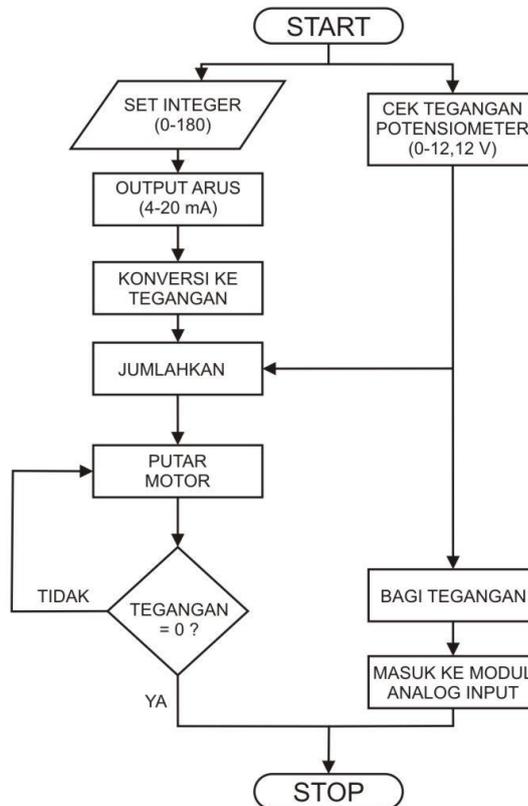
Sebagai umpan balik maka rangkaian motor menghasilkan sebuah parameter analog (potensiometer pada sudut posisi dan *tachometer* pada kecepatan motor). Parameter analog ini berupa tegangan sehingga dapat dimasukkan kedalam modul analog masukan. Nilai tegangan yang masuk menjadi referensi PLC (Integer) agar dapat dikelola dan dilihat pada antarmuka. Setiap perubahan parameter pada modul analog masukan akan terlihat perubahannya pada antarmuka.

III.1 Perangkat Keras Kendalian

Untuk mengendalikan motor maka diperlukan dua bagian rangkaian perangkat keras sesuai yang diperlukan yaitu :

1. Rangkaian pengendali servo posisi
2. Rangkaian pengendali kecepatan motor

III.1.2 Rangkaian Pengendali Servo Posisi



Gambar III.2 Flowchart Rangkaian pengendali sudut posisi

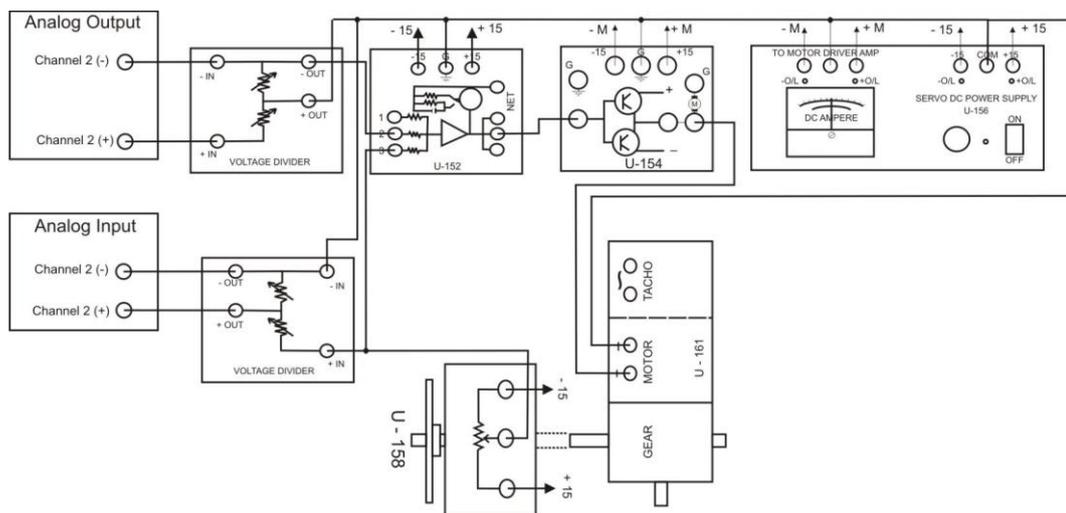
Untuk menggerakkan Motor pada pengendali sudut posisi adalah penjumlahan tegangan antara potensiometer dengan masukan dari modul analog keluaran.

Nilai integer yang diatur pada PLC menghasilkan nilai arus yang keluar pada modul analog keluaran. Nilai arus tersebut dikonversi menjadi tegangan dengan memakai rangkaian pembagi tegangan. Tegangan kedua berasal dari modul potensiometer motor. Kedua tegangan tersebut dijumlahkan pada modul praktikum *Sum Amplifier* U-153. Setelah dijumlahkan ada perbedaan tegangan, maka motor berputar sampai tidak ada lagi perbedaan tegangan.

Rangkaian pembagi tegangan diperlukan untuk merubah arus menjadi tegangan. Untuk mengetahui berapa tahanan yang dipakai, terlebih dahulu diukur secara manual (menggunakan Multimeter) tegangan maksimal dari motor. Setelah diukur sudut potensiometer antara 0-180° bekerja pada tegangan 0-12,12 Volt. Setelah diketahui tegangan maksimal dari potensiometer dapat diketahui nilai tahanan yang diperlukan dengan menggunakan rumus sebagai berikut :

$$V = I * R \dots\dots\dots (III.1)$$

Dengan $V = 12,32$ Volt dan $I = 20$ mA, maka kita memperoleh $R = 606 \Omega$



Gambar III.3 Rangkaian Pengendali Servo Posisi

Untuk membaca parameter sudut yang dihasilkan oleh potensiometer, maka jumlah tegangan potensiometer juga dimasukkan kedalam modul masukan analog. Karena tegangan modul masukan analog hanya berjarak antara 0 – 5 Volt, maka diperlukan rangkaian pembagi tegangan agar perubahan tegangan potensiometer antara 0 – 12,12 Volt sama dengan 0 – 5 Volt. Maka diperlukan nilai resistansi dari kedua tahanan pada rangkaian pembagi tegangan yang dipakai.

Untuk nilai tahanan yang diperlukan pada rangkaian pembagi tegangan dapat menggunakan Rumus II.3.

Dengan $V_{out} = 5$ Volt, $R_1 = 500 \Omega$ dan $V_{in} = 12,12$ Volt, maka yang didapat adalah $R_2 = 354 \Omega$.

III.1.2 Rangkaian Pengendali Kecepatan Motor

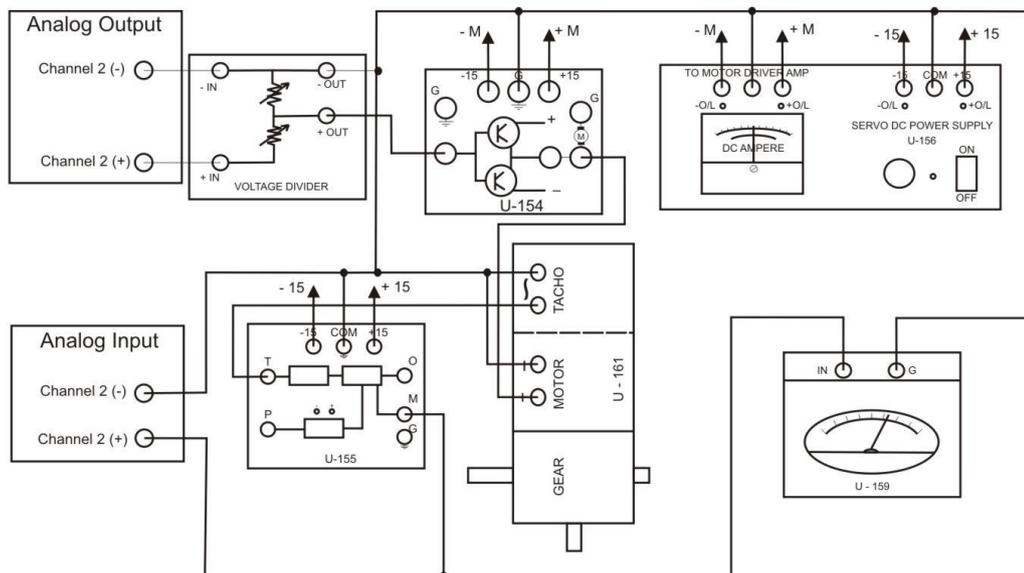


Gambar III.4 Flowchart Rangkaian pengendali kecepatan motor

Perubahan nilai integer berpengaruh pada meningkatnya kecepatan motor. Modul analog keluaran menghasilkan arus sesuai dengan nilai Integer yang diatur. Kemudian diperlukan rangkaian pembagi tegangan untuk merubah dari arus menjadi tegangan. Tegangan tersebut akan memutar motor dan menghasilkan nilai *Round per Minute* (RPM) sesuai kecepatan motor.

Nilai RPM ini dikonversi menjadi tegangan dengan menggunakan modul praktikum *F/V Converter* U-155. Nilai tegangan yang dihasilkan masuk ke dalam modul analog masukan dan akan menjadi parameter pada antarmuka.

Rangkaian pembagi tegangan diperlukan untuk mengubah nilai arus pada modul analog keluaran menjadi tegangan. Diperlukan pengukuran secara manual (menggunakan Multimeter) untuk mengetahui tegangan maksimum yang diperlukan motor. Setelah diukur didapatkan perubahan kecepatan motor antar 0-4000 RPM diperlukan tegangan antara 0-9,31 Volt. Dengan nilai tegangan $V = 9.31$ Volt dan nilai arus sebesar $I = 20$ mA maka dengan menggunakan Rumus III.1 nilai tahanan dipakai pada rangkaian pembagi tegangan sebesar $R = 465 \Omega$.



Gambar III.5. Rangkaian pengendali kecepatan motor

III. 2. Perancangan dan desain sistem antarmuka

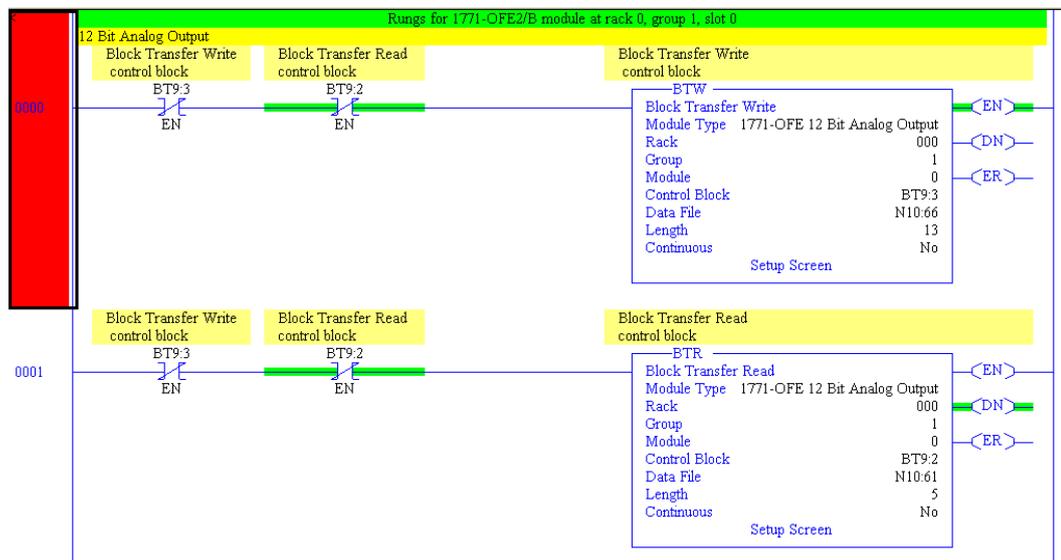
Untuk mengontrol dan memonitor motor sesuai dengan yang kita inginkan maka diperlukan sebuah antarmuka. Pembuatan antarmuka inipun dibagi atas dua

yaitu pembuatan program dengan menggunakan *RSLogix 5*. Program tersebut yang akan kita unggah ke dalam prosesor agar I/O dapat menjalankan logika dari program yang telah kita buat. Kemudian setelah programnya selesai, maka dibutuhkan sebuah antarmuka agar lebih memudahkan pengguna untuk mengontrol motor tersebut dengan software *RSView32*.

III. 2. 1. Perancangan *Ladder diagram*

Ladder diagram adalah barisan instruksi dalam program yang akan menjalankan masukan dan keluaran sesuai dengan apa yang dibuat. Untuk lebih memahami program yang telah dibuat, maka program ini dibagi atas beberapa bagian.

III.2.1.1 Program Pembacaan masukan dan keluaran analog



Gambar III.6 Program pembacaan analog

Gambar III.4 menunjukkan instruksi *Scanning* untuk Analog keluaran. Namun untuk Analog masukan mempunyai pengaturan yang tidak jauh berbeda

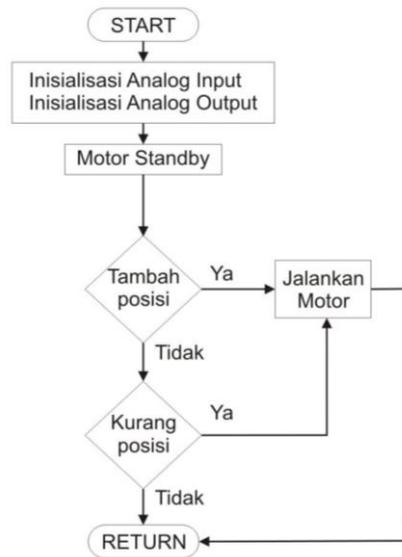
daripada instruksi tersebut. Instruksi ini terdiri atas dua bagian yaitu BTR dan BTW. Pada Instruksi BTR berfungsi untuk mentransfer maksimal 64 *word* pada waktu yang bersamaan dari sebuah *block transfer module* pada suatu *chasis* I/O, dimana pemilihan jenis modul yang digunakan ditempatkan pada module tipe, pemilihan nomor *rack* yang digunakan pada *rack no*, pemilihan nomor *group* I/O yang menspesifikasikan posisi dari modul yang digunakan didalam *chasis* ditempatkan pada *group* dan pemilihan nomor *slot* dalam grup yang digunakan ditempatkan pada modul.

Adapun *control block* merupakan tempat untuk menuliskan tipe data file control yang digunakan, disarankan untuk menggunakan data file *block transfer* (BT) sedangkan data file merupakan tempat memasukkan alamat input, output, status, integer, float, biner atau file data ASCII dari prosesor (menulis) atau ke prosesor (membaca) dan mentransfer data. Parameter lainnya yakni *length* merupakan tempat untuk memasukkan jumlah *word* yang akan ditransfer, yang mana panjangnya akan terkonfigurasi secara otomatis ketika intelligent I/O dikonfigurasi.

Sedangkan pada instruksi BTW memiliki kesamaan dengan BTR. Perbedaanya adalah instruksi BTW digunakan untuk *transfer data output* dari prosesor ke modul keluaran, sedangkan BTR digunakan untuk mentransfer data masukan dari modul ke prosesor.

III.2.1.2 Pengontrolan Servo Posisi

Sistem untuk kendalian Servo Posisi dapat dilihat dari flowchart berikut :



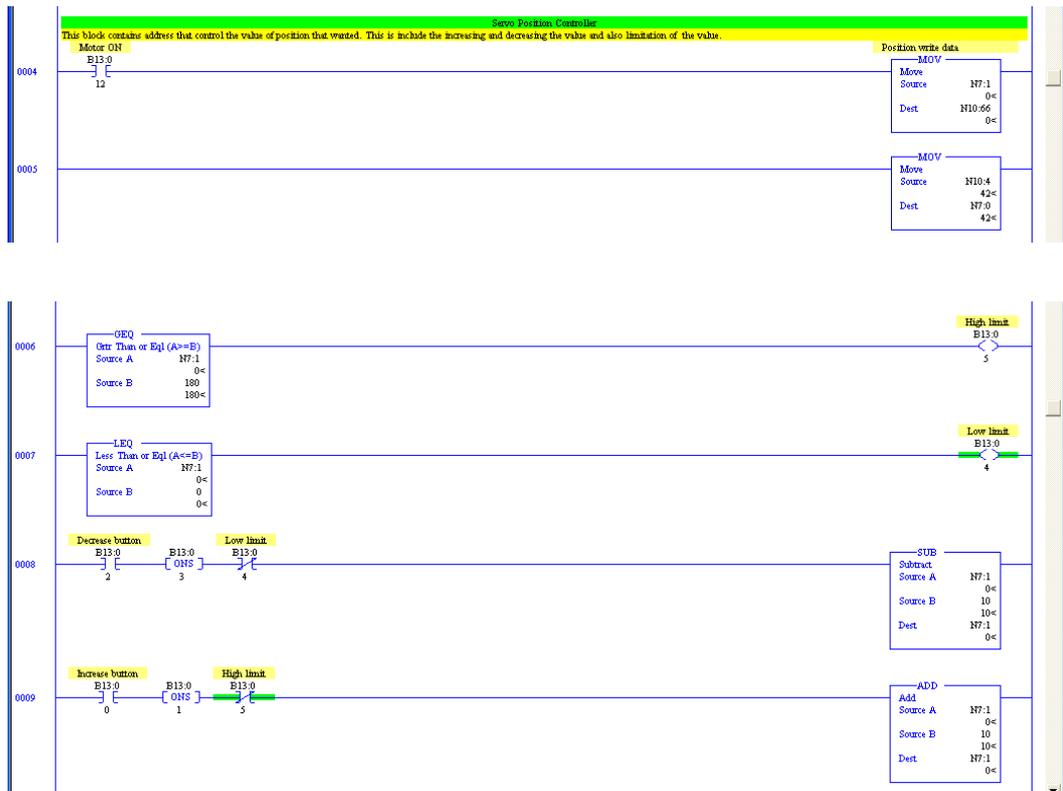
Gambar III.7 Flowchart untuk servo posisi

Ketika Motor *Standby*, maka kita dapat mengatur posisi potensiometer sesuai yang diinginkan. Pengendaliannya berupa kenaikan dan menurunnya nilai integer sehingga terjadi perubahan sudut setiap perubahan nilai Integer.

Instruksi ini adalah instruksi pengontrolan posisi motor yang berisikan alamat-alamat yang akan dipakai oleh antarmuka nantinya. Berikut penjelasan tiap *rung* :

- **Rung 0004**

Instruksi pada *rung* ini dimaksudkan untuk memindahkan data dari alamat memori yang akan menjadi parameter yaitu N7:1 ke pembacaan analog keluaran yaitu N10:66. Alamat memori N7:1 nantinya akan dipakai oleh antarmuka agar lebih mudah dikenali dan digunakan.



Gambar III.8 Program untuk servo posisi

- **Rung 0005**

Sama halnya dengan instruksi *rung* 0004 yang berfungsi untuk memindahkan nilai *word* dari alamat satu ke alamat lain. Namun perbedaannya yaitu alamat memori baru yang menjadi tujuan perpindahan alamat karena alamat yang menjadi sumber nilai adalah alamat modul analog input yang menerima data dari luar.

- **Rung 0006**

Kita memerlukan sebuah batasan agar nilai operasional analog output tidak melebihi dari jarak yang telah ditentukan. Maka digunakan instruksi GEQ yang membatasi batas maksimal dari nilai integer yang diatur yaitu 180.

- **Rung 0007**

Sama hanya dengan *rung* 0006 namun ini membuat batasan minimal dari jarak yang ditentukan. Diperlukan instruksi LEQ untuk membuat batasan minimal yaitu 0.

- **Rung 0008**

Fungsi dari rung ini adalah mengurangi nilai dari alamat N7:1 yang akan disesuaikan dengan antarmuka nantinya. Dengan adanya instruksi SUB jika alamat B13:0/2 aktif maka alamat N7:1 akan berkurang. Instruksi ONS sesuai dengan fungsinya hanya menaikkan satu kali saja ke instruksi SUB karena pada kenyataan walaupun alamat B13:0/2 aktif namun dengan siklus pembacaan PLC yang cepat maka akan terbaca beberapa kali. Sedangkan alamat B13:0/4 berisikan instruksi batasan minimal pada *rung* 0007.

- **Rung 0009**

Sama halnya dengan *rung* 0008 namun pada rung ini adalah penambahan nilai dari alamat N7:1 dengan instruksi ADD. Terdapat juga instruksi ONS untuk memberikan nilai satukali saja untuk dilewati dan instruksi batasan maksimal dari *rung* 0006.

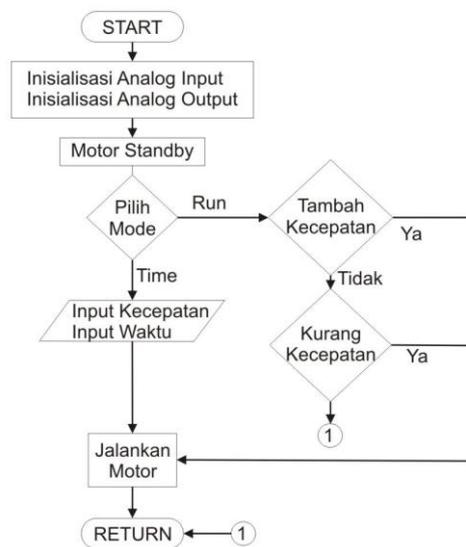
Untuk pengaturan posisi motor, nilai yang kita olah adalah N10:66 sebagai arus keluaran sesuai pengaturan yaitu 0-180. Ini dimaksudkan sebagai arus keluaran akan sama nilainya dengan derajat. Kemudian untuk masukan analog maka alamat yang dipakai adalah N10:4. Alamat tersebut adalah tegangan keluaran 0 sampai 5 Volt yang sama dengan derajat keluaran yaitu 0 sampai 180.

Ini dipakai agar mempermudah Integer yang ditampilkan untuk pengelolaan nilai tersebut menjadi sebuah antarmuka.

Untuk mengelola nilai yang akan masuk ke dalam motor, maka pengelolaan alamat N10:66 dibutuhkan. Maka untuk mengubah nilai pada alamat tersebut dibutuhkan dua buah alamat lain. Kedua alamat yang kita pakai adalah B13:0/0 untuk menaikkan nilai dan B13:0/2 untuk menurunkan nilai analog yang masuk ke motor.

III.2.1.2 Pengontrolan Kecepatan Motor

Sistem untuk kendalian kecepatan Motor dapat dilihat dari flowchart berikut :



Gambar III.9 Flowchart untuk servo posisi

Untuk pengontrolan kecepatan motor agak berbeda dengan pengaturan sudut posisi karena kita menambahkan program pengaturan kecepatan motor dan waktu berputar motor tersebut. Dapat dikatakan bahwa secara umum program

pengaturan kecepatan motor ini terdiri atas dua program, yaitu program *Motor Run* dan *Motor Time*. Prinsip kerja untuk mode *Motor Run* adalah menaikkan atau menurunkan kecepatan motor sesuai dengan nilai Integernya. Sedangkan untuk mode *Motor Time* diperlukan pengaturan berapa kecepatan yang diinginkan dan berapa waktu putaran motor tersebut. Setelah dijalankan maka motor akan berputar sesuai dengan kecepatan dan waktu yang kita atur. Kemudian setelah selesai maka motor kembali dalam keadaan *standby*.

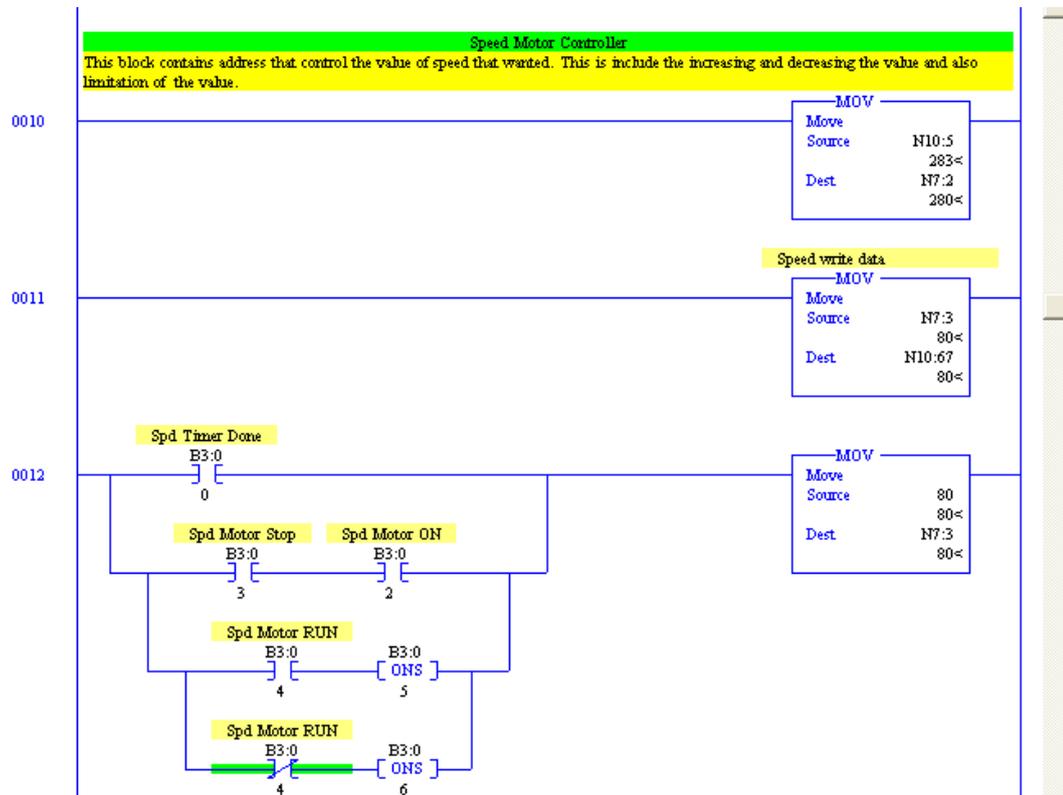
Berikut adalah penjelasan dari tiap-tiap *rung* :

- ***Rung 0010***

Rung ini berfungsi untuk membaca nilai tegangan yang masuk ke modul analog masukan dan akan dipindahkan ke alamat memori baru agar dapat diolah. Alamat analog masukannya yaitu N10:5 yang berasal dari pengaturan modul analog input yang akan ditransfer ke alamat tujuan memori baru yaitu N7:2. Alamat baru ini nantinya akan diolah oleh program antarmuka agar dapat dilihat kondisi sebenarnya dari keadaan motor.

- ***Rung 0011***

Seperti fungsi dari *rung* 0010 namun sumber dan tujuan alamat terbalik. Ini dikarenakan fungsi pada *rung* ini adalah alamat modul analog keluaran yaitu N10:67 akan menjadi tujuan dari alamat memori baru yaitu N7:3. Alamat N7:3 nantinya akan dirubah-rubah sesuai dari pengaturan kecepatan yang diinginkan.



Gambar III.10 Program pengontrolan kecepatan motor bagian 1

- **Rung 0012**

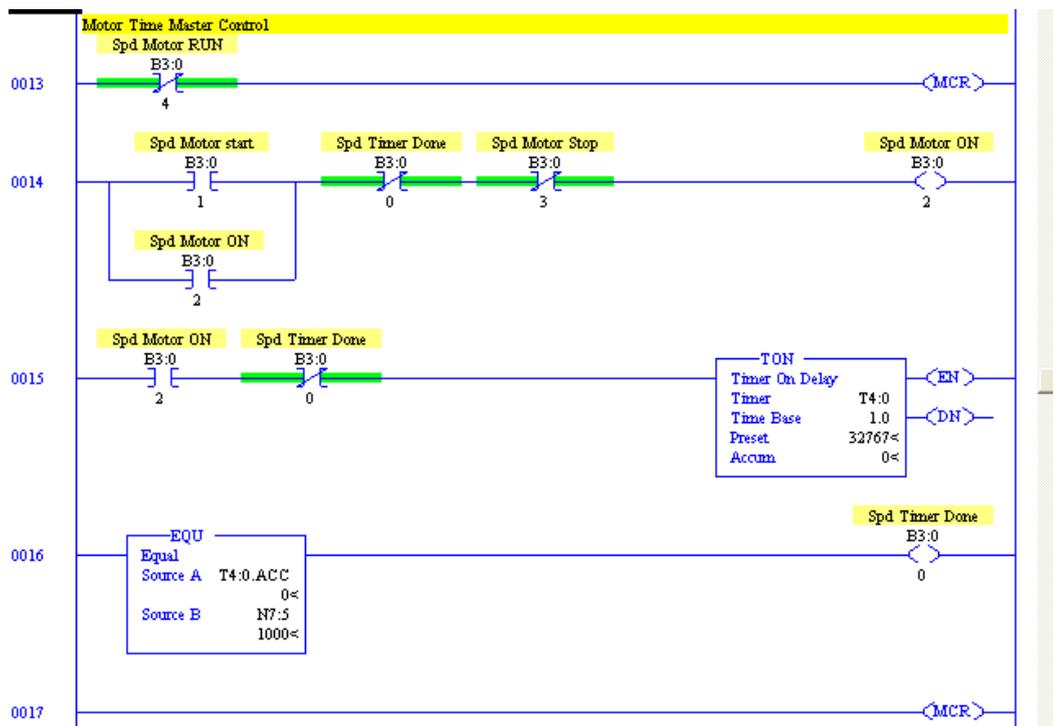
Rung ini berfungsi membuat kondisi motor menjadi keadaan *stand-by*. Dapat dilihat jika salah satu pemicunya memicu instruksi MOV maka nilai integer akan menjadi 80, dimana nilai tersebut adalah kondisi dimana motor *stand-by*. Pemicu untuk membuat keadaan tersebut adalah jika motor selesai berputar berdasar waktu yang diatur, tombol *stop* ditekan pada saat motor berjalan dan perubahan mode dari *Motor Run* menjadi *Motor Time* atau sebaliknya.

- **Rung 0013**

Pada *rung* ini terdapat instruksi *Master Control* menandakan awal dari program *Motor Time* pada pengontrolan kecepatan motor. Hal yang memicu untuk menggunakan mode ini adalah alamat B3:0/4 pada kondisi 'off'.

- **Rung 0014**

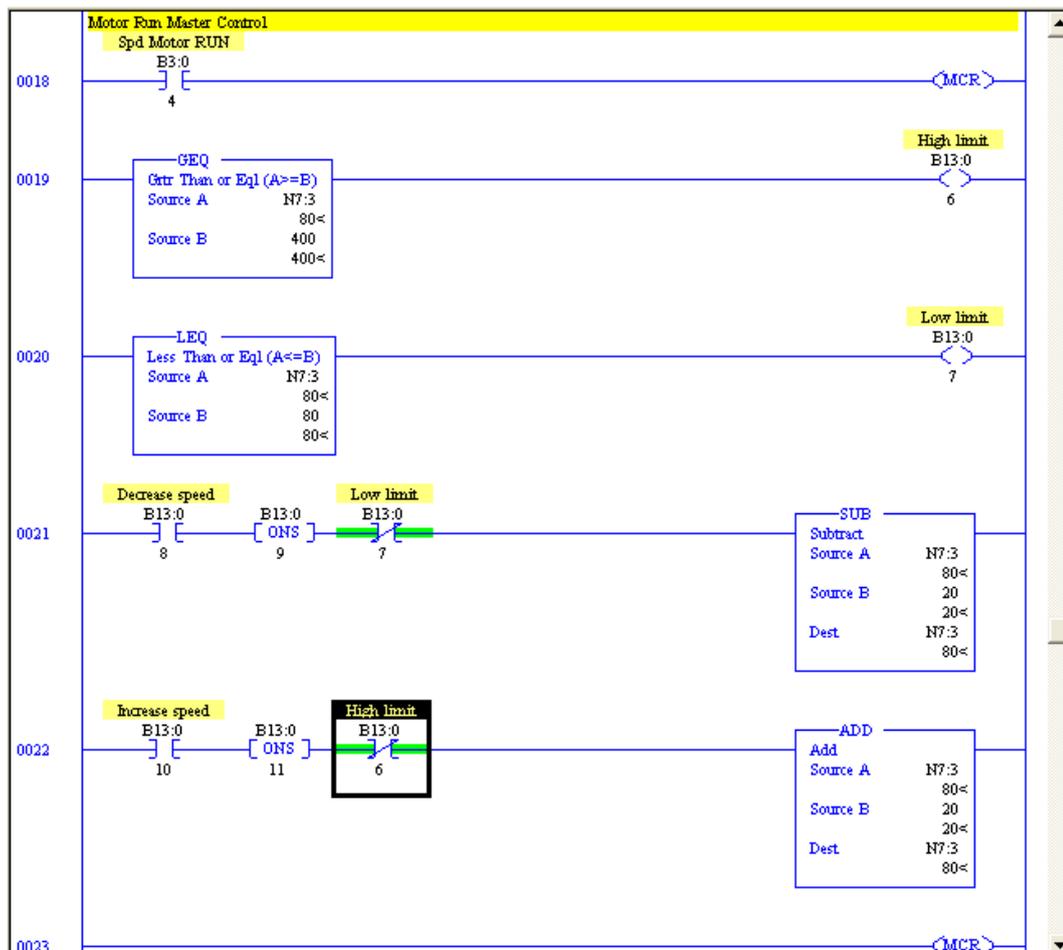
Rung ini berfungsi untuk operasi keadaan motor bekerja berdasarkan pengaturan kecepatan dan waktu yang telah diatur. Motor akan bekerja jikatombol *start* ditekan dan akan berhenti jika waktu yang diatur selesai atau tombol *stop* ditekan.



Gambar III.11 Program pengontrolan kecepatan motor bagian 2

- **Rung 0015**

Pada *rung* ini terdapat instruksi TON sebagai acuan waktu yang dikendalikan oleh PLC. Instruksi ini diatur agar akumulasi dengan nilai maksimal yaitu 32767 karena program tidak membatasi berapa waktu yang akan diatur. *Time base* yang diambil adalah 1.0 yang menandakan bahwa jumlah detik yang dapat diatur adalah sama dengan jumlah maksimal dari instruksi ini. Instruksi ini akan dimulai jika motor mulai dijalankan dan akan berhenti jika waktu yang diatur telah selesai.



Gambar III.12 Program pengontrolan kecepatan motor bagian 3

- **Rung 0016**

Instruksi didalamnya yaitu EQU membandingkan antara waktu akumulasi dan yang telah diatur sama. Jika demikian, maka akan menandakan bahwa waktu tersebut telah selesai.

- **Rung 0017**

Rung ini menandakan bahwa akhir dari mode *Motor Time*.

- **Rung 0018**

Pada rung ini terdapat instruksi *Master Control* menandakan awal dari program *Motor Run* pada pengontrolan kecepatan motor. Hal yang memicu untuk menggunakan mode ini adalah alamat B3:0/4 pada kondisi 'on'.

- **Rung 0019**

Rung ini berfungsi untuk membatasi alamat N7:3 sesuai dari jarak maksimal dalam pengaturan integer keluaran, yaitu 400.

- **Rung 0020**

Rung ini berfungsi untuk membatasi alamat N7:3 sesuai dari jarak minimal dalam pengaturan integer kuluaran, yaitu 80.

- **Rung 0021**

Fungsi dari *rung* ini adalah mengurangi nilai dari alamat N7:3 yang akan disesuaikan dengan antarmuka nantinya. Dengan adanya instruksi SUB jika alamat B13:0/8 aktif maka alamat N7:3 akan berkurang. Instruksi ONS sesuai dengan fungsinya hanya menaikkan satu kali saja ke instruksi SUB karena pada kenyataan walaupun alamat B13:0/8 aktif namun dengan

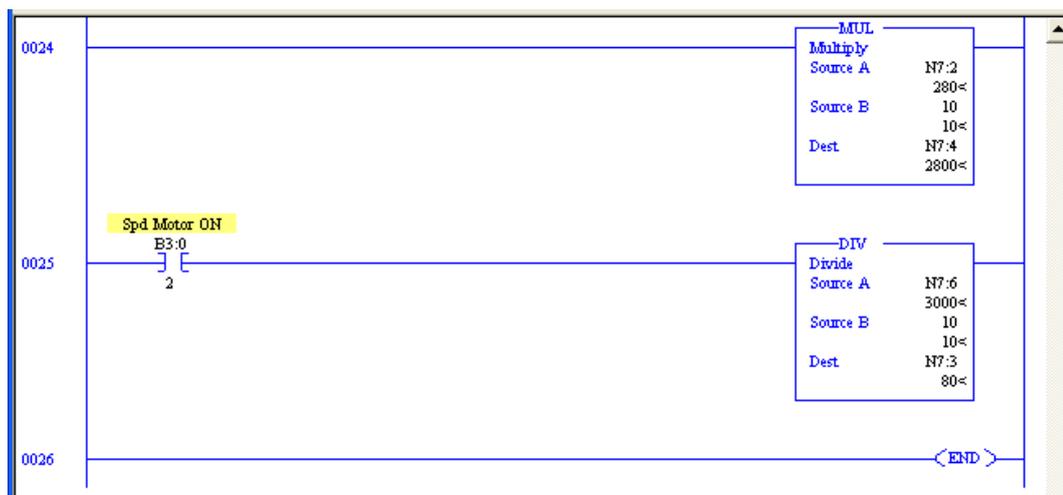
siklus pembacaan PLC yang cepat maka akan terbaca beberapa kali. Sedangkan alamat B13:0/7 berisikan instruksi batasan minimal pada *rung* 0020.

- **Rung 0022**

Sama halnya dengan *rung* 014 namun pada *rung* ini adalah penambahan nilai dari alamat N7:3 dengan instruksi ADD. Terdapat juga instruksi ONS untuk memberikan nilai satu kali saja untuk dilewati dan instruksi batasan maksimal dari *rung* 0019.

- **Rung 0023**

Rung ini menandakan untuk mengakhiri dari mode *Motor Run*.



Gambar III.13 Program pengontrolan kecepatan motor bagian 4

- **Rung 0024**

Instruksi MUL berfungsi untuk mengalikan nilai pada alamat N7:2 dengan nilai yang kita pasang yaitu 10. Ini bertujuan agar nilai integer pada alamat tersebut sesuai dengan nilai integer kecepatan putaran motor.

- **Rung 0025**

Instruksi DIV pada rung ini berfungsi agar nilai pada alamat N7:6 dibagi 10 dan akan menjadi nilai integer pada alamat N7:3. Alamat N7:6 adalah alamat dimana kita mengatur berapa putaran motor yang diinginkan pada mode *Motor Time*.

- **Rung 0026**

Rung ini adalah akhir dari program sistem kendali motor.

III. 2 . 2 Perancangan Antarmuka

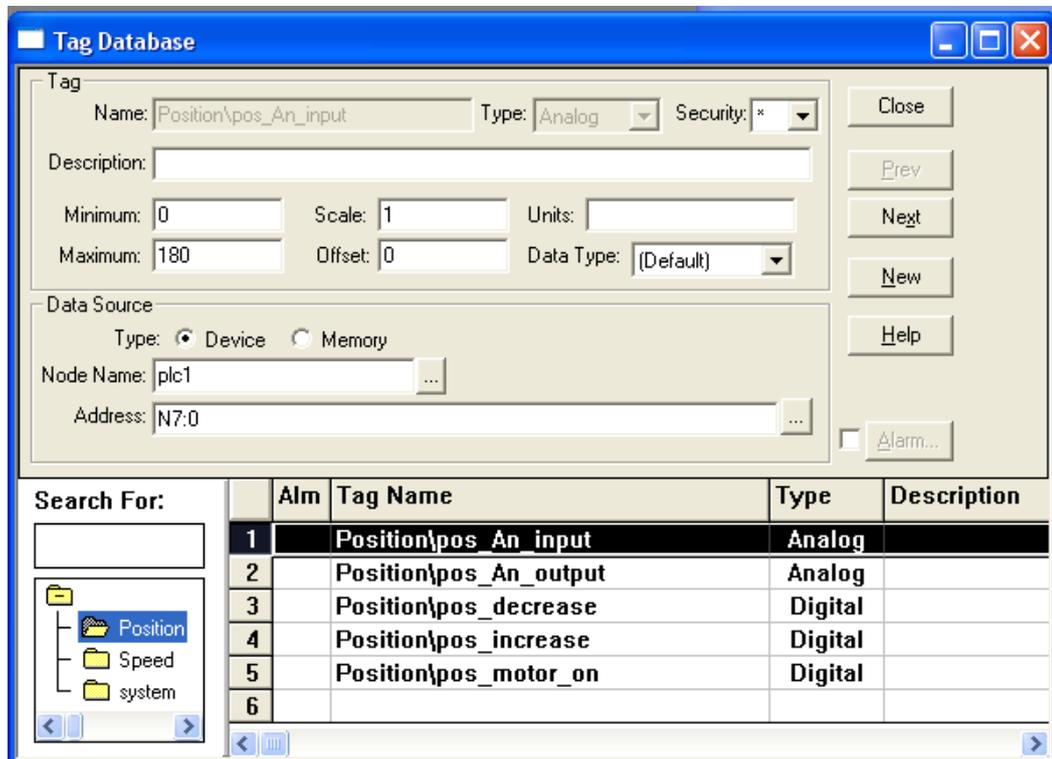
Motor yang dikendalikan diperlukan sebuah antarmuka agar pengontrolannya tidak dilakukan lagi secara manual. Pengontrolan yang dilakukan cukup dengan komputer dan dapat diketahui kondisi dari motor yang dikendalikan. Untuk itu pada perancangan ini akan dibahas pembuatan antarmuka dengan menggunakan perangkat lunak *RSView32*.

III. 2. 2. 1. Pembuatan *Tag Database*

Tag Database adalah sekumpulan label yang berisi keterangan yang berisi data yang ingin diinginkan untuk memantau. Didalmnya dapat mengatur label dalam kelompok-kelompok dengan menggunakan berkas.

Sebagai contoh pada label diatas dengan nama *Position\pos_An_input* dengan tipe analog. Pada label tersebut sumber data merujuk pada PLC dengan nama *Node plc1* dan dengan alamat N7:0. Dapat diambil kesimpulan bahwa label ini nantinya akan berhubungan langsung dengan alamat PLC yang dirujuk. Setiap

perubahan pada alamat N7:0 akan sama dengan nilai label ini sesuai dengan fungsi yang kita pasang.



Gambar III.14 Database label yang dipakai

Pada perancangan *Tag Database* pada sistem ini terdapat dua berkas yaitu berkas *Position* dan berkas *Speed*. Berkas *Position* mempunyai beberapa label yaitu dapat dilihat dari tabel berikut :

Tabel II.1 Berkas label *Position*

No	Nama label	Tipe	Alamat
1.	<i>Position\pos_An_input</i>	Analog	N7:0
2.	<i>Position\pos_An_output</i>	Analog	N7:1
3.	<i>Position\pos_decrease</i>	Digital	B13:0/2
4.	<i>Position\pos_increase</i>	Digital	B13:0/0
5.	<i>Position\pos_motor_on</i>	Digital	B13:0/12

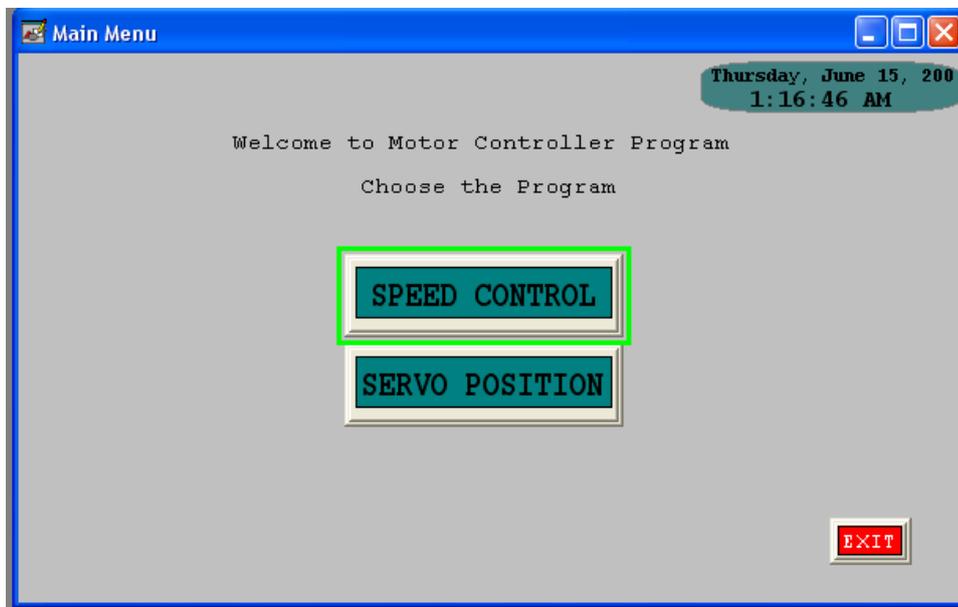
Pada berkas Speed mempunyai beberapa label yaitu dapat dilihat dari tabel

berikut :

Tabel 4.2 Berkas label *Speed*

No	Nama label	Tipe	Alamat
1.	<i>Speed\spd_An_input</i>	Analog	N7:2
2.	<i>Speed\spd_An_output</i>	Analog	N7:3
3.	<i>Speed\spd_decrease</i>	Digital	B13:0/8
4.	<i>Speed\spd_increase</i>	Digital	B13:0/10
5.	<i>Speed\spd_run</i>	Digital	B3:0/4
6.	<i>Speed\spd_motor_start</i>	Digital	B3:0/1
7.	<i>Speed\spd_motor_stop</i>	Digital	B3:0/3
8.	<i>Speed\spd_parameter</i>	Analog	N7:4
9.	<i>Speed\spd_timer</i>	Analog	N7:5

III. 2. 2. 2. Perancangan jendela *Main Menu*



Gambar III.15 Jendela *Main Menu*

Position\pos_increase. Maka setiap penekanan tombol ini mengakibatkan sudut posisi akan bertambah.

- **Tombol *Decrease***

Tombol ini sama halnya dengan tombol *Increase* namun perubahannya yaitu sudut posisi akan berkurang. Label yang digunakan adalah *Position\pos_Decrease*.

- ***Numeric Position Angle***

Untuk keakuratan sudut posisi maka dibutuhkan angka numerik dari sudut posisi yang ditunjuk. Dengan menggunakan label *Position\pos_An_input* maka setiap perubahan pada nilai analog masukan akan ditunjukkan oleh fitur ini.

- ***Animasi Position Angle***

Jarum pada animasi ini akan berubah mengikuti perubahan nilai pada label *Position\pos_An_input*. Fitur ini membuat tampilan menjadi menarik dan mendekati kenyataan akan perubahan sudut posisi dari motor.

- ***Tombol Main Menu***

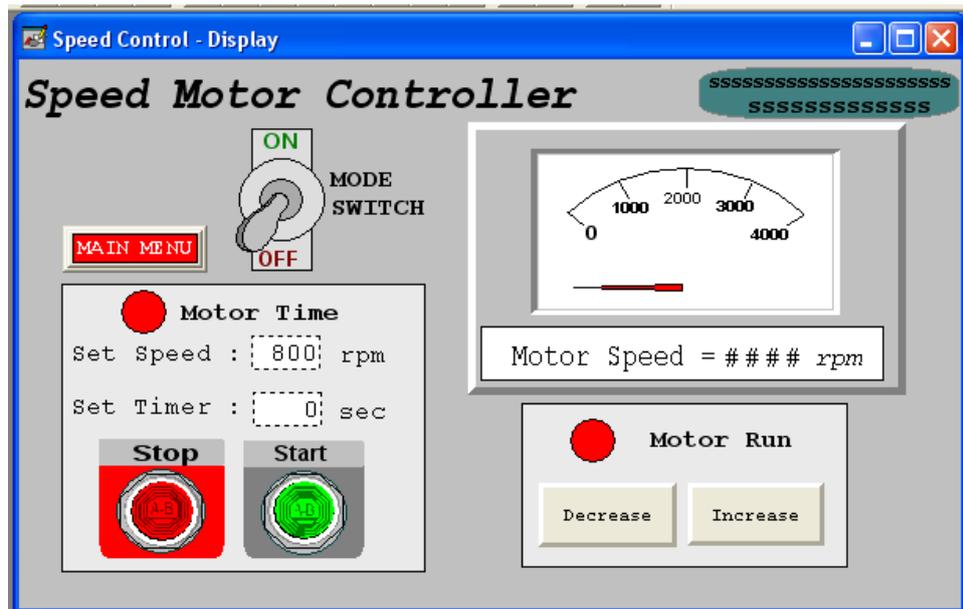
Tombol ini akan membuat antarmuka memasuki jendela *Main Menu*.

III. 2. 2. 4. Perancangan jendela *Motor Speed*

Jendela *Speed Motor* adalah layar utama untuk mengendalikan motor dengan fungsi kecepatan putaran motor. Didalamnya terdapat beberapa fitur utama dan fitur tambahan. Penjelasan tiap-tiap fitur dapat dilihat sebagai berikut :

- **Mode Switch**

Disini dapat diatur apakah yang diinginkan adalah mode *Motor Run* atau *Motor Time*. Jika *Motor Run* dipilih maka *Motor Time* tidak dapat digunakan, begitu pula sebaliknya.



Gambar III.17 Jendela *Motor Speed*

- **Lampu Motor Time**

Fitur ini hanya menunjukkan keadaan mode *Motor Time*. Jika berwarna hijau maka Mode ini dapat bekerja namun jika berwarna merah maka mode ini tidak dapat digunakan.

- **Set Speed**

Pengaturan kecepatan motor untuk mode *Motor Time* dapat diatur disini. Jarak yang diperbolehkan adalah 800-4000 rpm sesuai kecepatan minimal dan maksimal motor. Dalam keadaan *stand-by* maka nilai kecepatan motor adalah 800 rpm.

- ***Set Timer***

Untuk mengatur berapa lama motor berputar dapat diatur disini. Satuan yang digunakan adalah *second* (detik).

- ***Tombol Start***

Setelah diatur berapa kecepatan motor dan lama waktu, maka tombol ini akan memutar motor sesuai pengaturan tersebut.

- ***Tombol Stop***

Motor akan berubah keadaan *stand-by* jika tombol ini ditekan selama motor berputar.

- ***Lampu Motor Run***

Fitur ini hanya menunjukkan keadaan mode *Motor Run*. Jika berwarna hijau maka Mode ini dapat bekerja namun jika berwarna merah maka mode ini tidak dapat digunakan.

- ***Tombol Increase***

Pada tombol ini nilai analog keluaran akan terintegrasi jika kita menekan tombol ini. Ini dikarenakan label yang digunakan tombol ini adalah label *Speed\spd_increase*. Maka setiap penekanan tombol ini mengakibatkan sudut posisi akan bertambah.

- ***Tombol Decrease***

Tombol ini sama halnya dengan tombol Increase namun perubahannya yaitu sudut posisi akan berkurang. Label yang digunakan adalah *Speed\spd_decrease*.

- ***Numeric Speed Motor***

Untuk keakuratan kecepatan putaran motor maka dibutuhkan angka numerik dari kecepatan putaran yang ada. Dengan menggunakan label *Speed\spd_An_input* maka setiap perubahan pada nilai analog masukan akan ditunjukkan oleh fitur ini.

- ***Animasi Motor Speed***

Jarum pada animasi ini akan berubah mengikuti perubahan nilai pada label *Speed\spd_An_input*. Fitur ini membuat tampilan menjadi menarik dan mendekati kenyataan akan perubahan kecepatan putaran dari motor.

- ***Tombol Main Menu***

Tombol ini akan membuat antarmuka memasuki jendela *Main Menu*.

BAB IV

PENGUJIAN DAN ANALISIS

Pada bab ini akan dibahas apakah program antarmuka berjalan sesuai yang diinginkan dan melihat berapa parameter analog yang keluar dari PLC untuk menggerakkan motor dan berapa parameter analog yang masuk ke PLC berasal dari potensiometer untuk pengaturan sudut dan *tachometer* untuk pengaturan kecepatan motor sebagai *feedback* dari kendalian. Kemudian kita menganalisis parameter yang muncul di antarmuka dan membandingkannya dengan kenyataan agar bisa dianalisis berapa persentase kesalahan dari parameter tersebut.

IV.1 Pengujian tegangan masukan dan keluaran

Tujuan dari pengujian ini untuk membandingkan antara tegangan yang dapat dihitung secara teori dan tegangan yang diukur secara praktek. Hasil perbandingan dari kedua data tersebut kemudian dilihat dalam persentasi kesalahan. Rumus untuk melihat persentase kesalahan adalah sebagai berikut :

$$PK = \left| \left| \frac{\text{teori} - \text{praktek}}{\text{teori}} \times 100\% \right| \right| \dots\dots\dots(\text{IV.1})$$

Pengujian ini dibagi atas dua yaitu pengujian pada bagian servo posisi dan pengujian pada bagian pengontrolan kecepatan motor.

IV.1.1 Pengendali Servo Posisi

Pada bagian ini terdiri atas dua bagian yaitu bagian tegangan masukan dan keluaran. Untuk bagian tegangan masukan, nilai arus keluaran PLC antara 4-20 mA sama dengan jarak Integer antara 0 – 180. Tegangan masukan secara teori

dihitung dengan menggunakan Rumus III.1 dan memakai nilai tahanan $R = 606.5$

Ω . Hasilnya dapat dilihat sebagai berikut :

Tabel IV.1 Tegangan masukan servo posisi

Integer	Arus (mA)	Vin teori (Volt)	Vin praktek (Volt)	Presentase Kesalahan
0	4	2.43	2.44	0.58%
10	4.89	2.97	2.99	0.82%
20	5.78	3.51	3.53	0.70%
30	6.67	4.05	4.06	0.36%
40	7.56	4.59	4.6	0.32%
50	8.45	5.12	5.15	0.49%
60	9.34	5.66	5.68	0.27%
70	10.23	6.20	6.22	0.25%
80	11.12	6.74	6.75	0.08%
90	12.01	7.28	7.29	0.08%
100	12.9	7.82	7.8	0.30%
110	13.79	8.36	8.29	0.88%
120	14.68	8.90	8.86	0.49%
130	15.57	9.44	9.39	0.56%
140	16.46	9.98	9.92	0.63%
150	17.35	10.52	10.45	0.69%
160	18.24	11.06	11	0.57%
170	19.13	11.60	11.53	0.62%
180	20	12.13	12.07	0.49%

Pada Tabel IV.1 dapat dianalisis bahwa setiap kenaikan integer berpengaruh pada besarnya nilai arus yang keluar dari PLC. Maka setiap kenaikan arus maka tegangan akan bertambah. Dan juga dapat dilihat bahwa antara tegangan masukan pengaturan servo posisi tidak jauh berbeda antara teori dan praktek. Selisih perbedaan tegangan dapat dilihat dari presentase kesalahan yang berkisar antara 0.08% sampai dengan 0.82%.

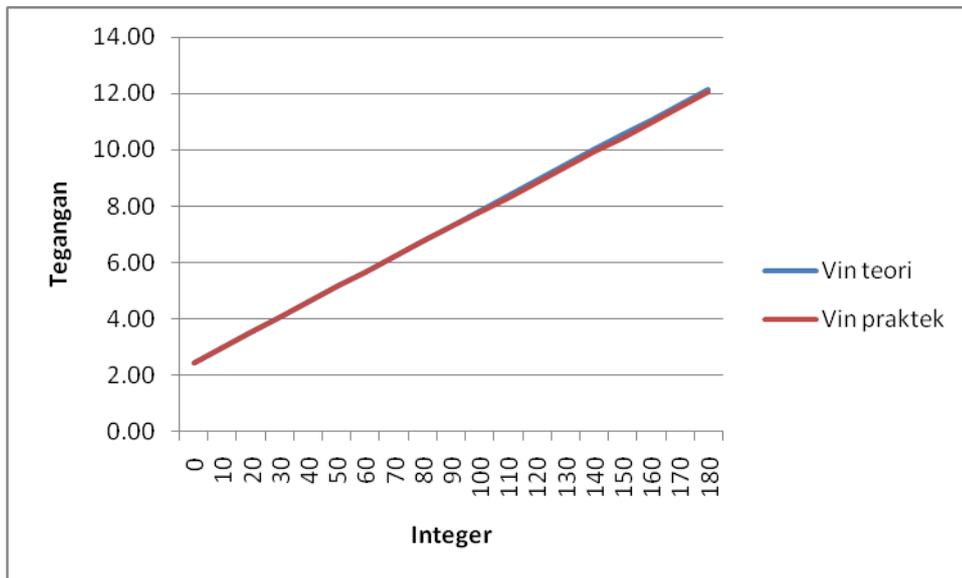
Untuk Tegangan keluaran pada pengaturan servo posisi maka dibandingkan pula tegangan yang masuk ke dalam PLC. Tegangan tersebut

dihitung secara teoritis menggunakan Rumus II.3 setelah dikonversi dengan menggunakan rangkaian pembagi tegangan. Untuk tegangan sumbernya berasal dari tegangan keluaran dari potensiometer, tahanan pertama $R_1 = 500 \Omega$ dan tahanan kedua $R_2 = 354.1 \Omega$.

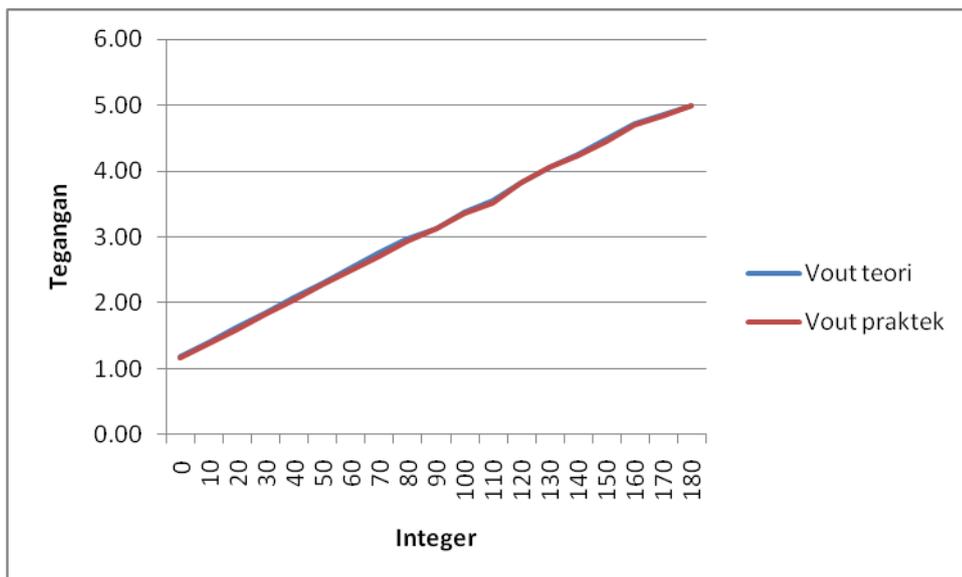
Tabel IV.2 Tegangan Keluaran Servo Posisi

Integer	Tegangan Sumber (Volt)	Vout teori (Volt)	Vout praktek (Volt)	Presentase Kesalahan
0	2.85	1.18	1.17	0.98%
10	3.38	1.40	1.38	1.52%
20	3.92	1.63	1.6	1.55%
30	4.46	1.85	1.83	1.03%
40	4.99	2.07	2.04	1.39%
50	5.53	2.29	2.27	0.99%
60	6.09	2.52	2.49	1.38%
70	6.64	2.75	2.71	1.56%
80	7.15	2.96	2.94	0.82%
90	7.57	3.14	3.13	0.27%
100	8.18	3.39	3.37	0.63%
110	8.57	3.55	3.52	0.93%
120	9.26	3.84	3.82	0.50%
130	9.8	4.06	4.05	0.32%
140	10.27	4.26	4.23	0.65%
150	10.86	4.50	4.45	1.17%
160	11.4	4.73	4.7	0.56%
170	11.73	4.86	4.84	0.48%
180	12.06	5.00	4.99	0.20%

Pada Tabel IV.2 dapat dianalisis bahwa untuk setiap kenaikan Integer mempengaruhi kenaikan tegangan yang dihasilkan oleh potensiometer. Dan juga dapat dilihat bahwa antara tegangan keluaran pengaturan servo posisi tidak jauh berbeda antara teori dan praktek. Selisih perbedaan tegangan dapat dilihat dari presentase kesalahan yang berkisar antara 0.20% sampai dengan 1.55%.



Gambar IV.1. Grafik perbandingan tegangan masukan servo posisi



Gambar IV.2. Grafik perbandingan tegangan keluaran servo posisi

IV.1.2 Pengontrolan Kecepatan Motor

Perbandingan tegangan secara teori dan praktik hanya digunakan pada tegangan masukan. Ini mengingat pada tegangan kaluaran tidak memakai rangkaian pembagi tegangan. PLC mengeluarkan arus keluaran antara 4-20 mA

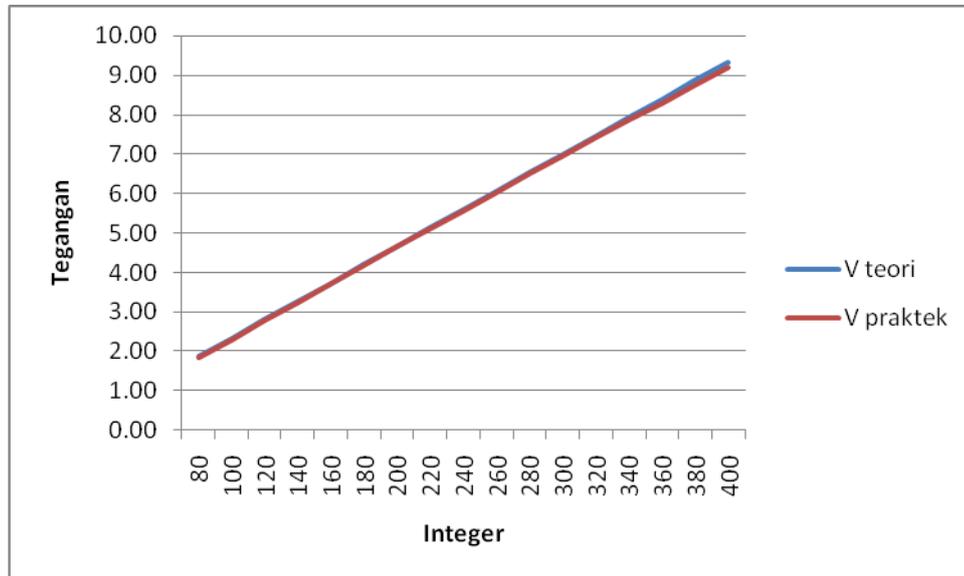
sehingga diperlukan rangkaian tegangan untuk mengubah arus tersebut menjadi tegangan masukan ke motor. Integer yang mewakili perubahan kecepatan motor adalah 80-400. Pada rangkaian pembagi tegangan dengan menggunakan Rumus II. 1 nilai tahanan yang dipakai adalah $R = 465 \Omega$. Hasil yang didapat adalah sebagai berikut :

Tabel IV.3 Tegangan masukan Kecepatan Motor

Integer	Arus (mA)	Vin teori	Vin praktek	Presentase Kesalahan
80	4	1.86	1.84	1.22%
100	5	2.33	2.31	0.79%
120	6	2.79	2.78	0.50%
140	7	3.26	3.25	0.30%
160	8	3.73	3.72	0.14%
180	9	4.19	4.19	0.02%
200	10	4.66	4.65	0.14%
220	11	5.12	5.12	0.04%
240	12	5.59	5.58	0.14%
260	13	6.05	6.03	0.39%
280	14	6.52	6.5	0.30%
300	15	6.98	6.95	0.50%
320	16	7.45	7.41	0.54%
340	17	7.92	7.87	0.58%
360	18	8.38	8.3	0.98%
380	19	8.85	8.76	0.99%
400	20	9.31	9.2	1.22%

Pada Tabel IV.3 dapat dianalisis bahwa setiap kenaikan integer berpengaruh pada besarnya nilai arus yang keluar dari PLC. Maka setiap kenaikan arus maka tegangan bertambah. Dan dapat juga dilihat bahwa antara tegangan keluaran pengontrolan kecepatan motor tidak jauh berbeda antara teori dan

praktek. Selisih perbedaan tegangan dapat dilihat dari presentase kesalahan yang berkisar antara 0.04% sampai dengan 1.22%.



Gambar IV.3. Perbandingan Tegangan Masukan Kecepatan Motor

IV.2. Perbandingan Parameter Antarmuka dengan Alat Praktikum

Selain pengujian tegangan pada alat, sistem ini juga dibandingkan antara antarmuka yang dibuat dan dengan menggunakan modul praktikum ED-4400 *Servo System*. Alat praktikum ini dilengkapi potensiometer untuk servo posisi dan tachometer untuk kecepatan motor sehingga dengan mudah bisa membandingkan hasilnya dengan sistem ini. Untuk mendapatkan hasil yang diinginkan maka perubahan kenaikan integer adalah sebagai parameter yang diambil sehingga bisa menghasilkan hasil yang mendekati nilai pembacaan dari alat praktikum tersebut.

Hasil yang didapat dari sistem ini kemudian dibandingkan dengan alat ukur praktikum. Kemudian dibandingkan dengan antarmuka sehingga presentase

kesalahan dapat menggunakan Rumus IV.1. Perbandingannya pun terbagi atas dua sistem yaitu pengendali sudut motor dan kecepatan putaran motor.

IV.2.1 Pengaturan Sudut Posisi

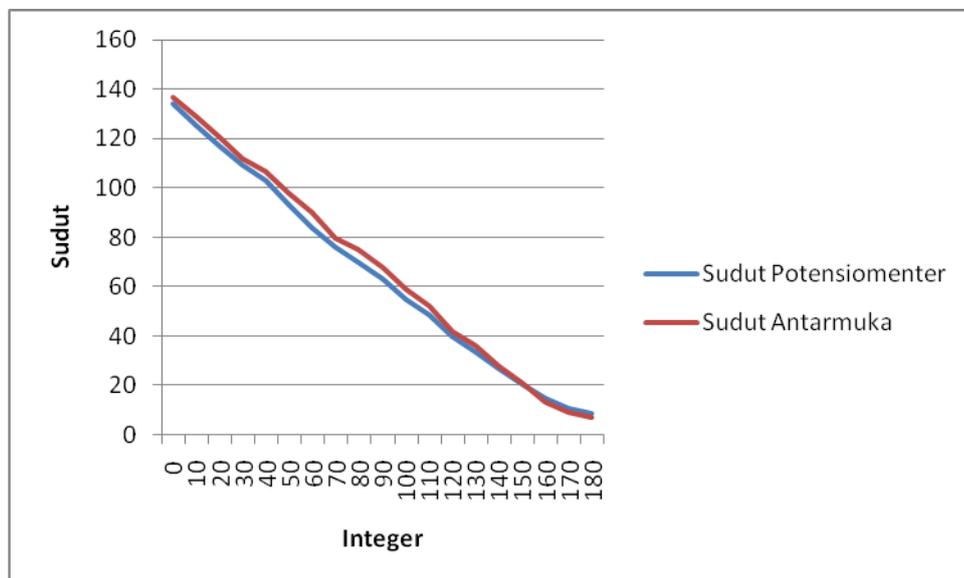
Disini akan dibandingkan antara sudut yang ditunjukkan oleh antarmuka dan sudut dari potensiometer modul praktikum. Hasil yang didapat dapat dilihat dari tabel berikut :

Tabel IV.4 Derajat sudut pengontrolan posisi sudut

Integer	Sudut Potensiometer ($^{\circ}$)	Sudut Antarmuka ($^{\circ}$)	Presentase Kesalahan
0	134	137	2.24%
10	125	129	3.20%
20	117	121	3.42%
30	109	112	2.75%
40	103	107	3.88%
50	93	98	5.38%
60	84	90	7.14%
70	76	80	5.26%
80	70	75	7.14%
90	63	68	7.94%
100	55	59	7.27%
110	49	52	6.12%
120	40	42	5.00%
130	34	36	5.88%
140	27	28	3.70%
150	21	21	0.00%
160	15	13	13.33%
170	11	9	18.18%
180	9	7	22.22%

Dari data diatas bahwa presentase kesalahan terbesar berada pada awal posisi yaitu pada integer 180 sampai 160 yaitu diatas 10%. Ini disebabkan karena perbedaan tegangan yang masuk dan pembacaan pada integer. Kemudian penyebab besarnya perbedaan sudut karena kurang akuratnya potensiometer

dalam memberikan tegangan yang masuk ke dalam masukan analog menyebabkan PLC kurang tepat membaca sudut yang ditunjukkan oleh potensiometer. Dan juga masih adanya selisih perbedaan tegangan yang masuk ke dalam modul praktikum *Sum Amplifier* U-153. Selisih tegangan tersebut sangat kecil sehingga menyebabkan motor tidak berputar karena tidak cukup tegangan untuk memutar motor. Selisih perbedaan sudut posisi dapat dilihat dari presentase kesalahan yang berkisar antara 2.24% sampai dengan 22.22%.



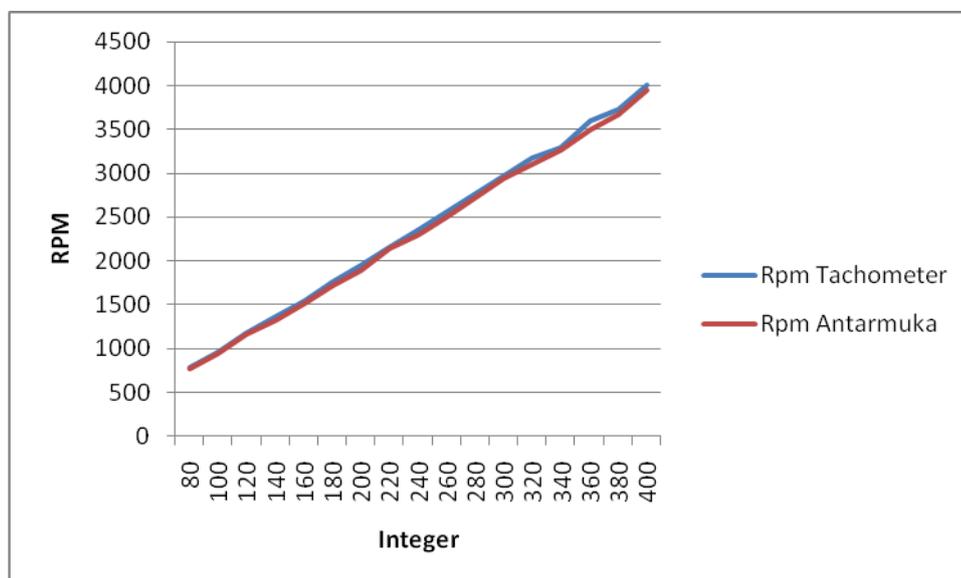
Gambar IV.4 Grafik perbandingan sudut pengontrolan sudut posisi

IV.2.2 Pengontrolan Kecepatan Motor

Kecepatan putaran motor akan dibaca dengan melihat berapa *Round per minute* (RPM) yang dihasilkan motor. Untuk melihat RPM dari motor dengan menggunakan *tachometer* pada alat praktikum. Hasil yang diperoleh adalah sebagai berikut :

Tabel IV.5 Perbandingan RPM pada pengontrolan kecepatan motor

Integer	Rpm Tachometer	Rpm Antarmuka	Presentase Kesalahan
80	790	770	2.53%
100	970	950	2.06%
120	1180	1160	1.69%
140	1370	1330	2.92%
160	1540	1510	1.95%
180	1770	1720	2.82%
200	1950	1900	2.56%
220	2160	2140	0.93%
240	2360	2300	2.54%
260	2560	2510	1.95%
280	2770	2720	1.81%
300	2970	2950	0.67%
320	3170	3100	2.21%
340	3290	3260	0.91%
360	3590	3500	2.51%
380	3720	3680	1.08%
400	4000	3950	1.25%



Gamabr IV.5 Grafik perbandingan RPM pengontrolan kecepatan motor

Pada tabel IV.5 dilihat bahwa setiap perubahan nilai Integer akan menyebabkan naiknya nilai RPM pada motor. Dan juga dapat dilihat bahwa RPM

dari pengontrolan kecepatan motor tidak jauh berbeda antara *tachometer* dan antarmuka yang ada. Selisih perbedaan kecepatan motor dapat dilihat dari presentase kesalahan yang berkisar antara 0.67% sampai dengan 2.92%.

BAB V

PENUTUP

V.1 Kesimpulan

1. Untuk pengujian tegangan masukan dan tegangan keluaran pada kendalian tidak jauh berbeda antara teori dan praktek. Persentase kesalahannya berkisar antara 0.04% sampai dengan 1.55%.
2. Besar sudut untuk kendalian servo posisi antara antarmuka dan potensiometer mendekati kenyataan. Untuk presentase kesalahan dari kendalian ini berkisar antara 2.24% sampai dengan 22.22%.
3. Besar sudut untuk kendalian servo posisi antara antarmuka dan potensiometer mendekati kenyataan. Untuk presentase kesalahan dari kendalian berkisar antara 0.67% sampai dengan 2.92%.
4. Pengukuran alat praktikum yaitu potensiometer pada pengontrolan servo posisi dan *tachometer* pada pengaturan kecepatan motor menghasilkan tegangan. Tegangan ini bisa dimasukkan dalam modul analog masukan PLC kemudian dikelola oleh antarmuka yang dibuat.

V.2 Saran

1. Kebutuhan PLC sebagai sistem kendali pada industri dewasa ini sehingga sangat bagus dipelajari. Namun karena tidak adanya mata kuliah tentang topik ini sehingga disarankan untuk menyeleraskan mata kuliah yang ada dengan topik ini.

2. Untuk modul keluaran analog, lebih baik diganti dengan model *Allen Bradley 1771 - OFE1B* yang keluarannya berupa tegangan sehingga tidak perlu lagi dikonversi menggunakan rangkaian pembagi tegangan.
3. Antarmuka dan program *ladder diagram* dibuat lebih banyak fitur yang memanfaatkan kecepatan dan sudut posisi motor contohnya kenaikan kecepatan motor berdasarkan waktu atau pengaturan posisi secara otomatis sehingga aplikasinya lebih berguna.